Capitolo 1

Approccio computazionale: alcune sorgenti di errore

1.1 Fonti di errore nella risoluzione di un problema mediante calcolatore

Uno dei problemi fondamentali della Matematica Numerica è quello di valutare l'accuratezza del risultato di un calcolo e dunque l'affidabilità[1] del risultato stesso.
Dati due numeri x e $\tilde{x}$ e supposto che $\tilde{x}$ sia un'approssimazione di x, il primo problema che si pone è quello di stimare la "bontà" di $\tilde{x}$ come approssimazione di x.
La maniera più intuitiva per fare ciò si basa sul concetto di *errore assoluto*.

Definizione 1.1. (Errore assoluto)
Dati un numero x ed una sua approssimazione $\tilde{x}$, si dice errore assoluto in $\tilde{x}$ la quantità

$$E = |x - \tilde{x}| .$$

Si può ritenere che $\tilde{x}$ sia una buona approssimazione di x se E è "sufficientemente piccolo". La difficoltà consiste ora nel chiarire che cosa si intenda per "sufficientemente piccolo".

♣ **Esempio 1.1.** Si considerino $x = 10.1294$ e $\tilde{x} = 10.1253$. Le loro parti intere e le loro prime due cifre decimali sono uguali e risulta $E = 0.0041 < 10^{-2}$.

♣

In generale vale la seguente:

[1] Per ora basta intuire il significato del termine *affidabile*, basandoci su considerazioni empiriche o sull'esperienza che ciascuno di noi ha.

Proprietà 1.1. (dell'errore assoluto)
Se due numeri x e $\tilde{x}$ hanno la stessa parte intera e le stesse prime m cifre decimali allora risulta

$$E = |x - \tilde{x}| < 10^{-m}\,.$$

Tale proprietà fornisce un'interpretazione quantitativa dell'errore assoluto e quindi chiarisce, in qualche modo, il significato dell'espressione "sufficientemente piccolo". [2]

L'errore assoluto non sempre basta a valutare la bontà di un'approssimazione.

♣ **Esempio 1.2.** Se si considerano $x_1 = 10.12$ e $\tilde{x}_1 = 10.05$, si ha $E_1 = |x_1 - \tilde{x}_1| = 0.07$, ma anche se si considerano $x_2 = 10000.14$ e $\tilde{x}_2 = 10000.07$ risulta $E_2 = |x_2 - \tilde{x}_2| = 0.07$. L'errore assoluto è dunque lo stesso in entrambi i casi, benché, intuitivamente, nel secondo caso l'approssimazione appaia essere migliore.
Analogamente, un errore di 50 mila euro nelle previsioni di spesa annua di un'industria con un fatturato annuo di 2500 milioni di euro è praticamente irrilevante, mentre lo stesso errore ha certamente effetti notevoli sul bilancio di una piccola impresa familiare, con un fatturato annuo di 200 mila euro.
♣

Infatti, per come è definito, l'errore assoluto non tiene conto della grandezza del valore da approssimare. Un modo per fare questo è rapportare l'errore assoluto a $|x|$, ovvero "scalare" l'errore considerando $|x|$ come unità di misura. A tal fine si introduce il concetto di *errore relativo.*

Definizione 1.2. (Errore relativo)
Dati un numero x ed una sua approssimazione $\tilde{x}$, si dice errore relativo in $\tilde{x}$ la quantità

$$E' = \frac{|x - \tilde{x}|}{|x|} \qquad (x \neq 0)\,.$$

♣ **Esempio 1.3.** Consideriamo nuovamente $x_1 = 10.12$, $\tilde{x}_1 = 10.05$, $x_2 = 10000.14$ e $\tilde{x}_2 = 10000.07$ e calcoliamo gli errori relativi corrispondenti. Otteniamo così $E'_1 \simeq 0.0069 = 0.69 \times 10^{-2}$ ed $E'_2 \simeq 0.0000069 = 0.69 \times 10^{-5}$, cioè, come era prevedibile, l'errore relativo è nel secondo caso molto più piccolo

[2] Si noti che tale proprietà non può essere invertita, nel senso che non è vero che se due numeri differiscono in modulo per meno di 10^{-m} allora hanno la stessa parte intera e le stesse prime m cifre decimali; un esempio significativo è costituito da $x = 3.000000$ e $\tilde{x} = 2.999999$. Questa restrizione, comunque, è dovuta unicamente al criterio di rappresentazione posizionale, cioè al significato che hanno le cifre della rappresentazione posizionale di un numero, e quindi è una restrizione formale. Pertanto, in seguito si assumerà che x e $\tilde{x}$ hanno le stesse cifre fino alla m-ma cifra decimale ogni qualvolta risulti $|x - \tilde{x}| < 10^{-m}$.

che nel primo. Si osservi che x_1 e $\tilde{x}_1$ hanno le prime due cifre significative[3] uguali ed è $E'_1 < 10^{-1}$, mentre x_2 e $\tilde{x}_2$ hanno le prime cinque cifre significative uguali ed è $E'_2 < 10^{-4}$.

♣

In generale vale la seguente:

Proprietà 1.2. (dell'errore relativo)
Se due numeri x e $\tilde{x}$ hanno le stesse prime m cifre significative, allora risulta

$$E' = \frac{|x - \tilde{x}|}{|x|} < 10^{-m+1}\,.$$

Pertanto l'errore relativo misura la distanza tra due numeri in termini di cifre significative corrette, piuttosto che di cifre decimali corrette. Si noti che la relazione

$$E' = \frac{|x - \tilde{x}|}{|x|} < 10^{-m+1} \tag{1.1}$$

non implica che x e $\tilde{x}$ abbiano le stesse prime m cifre significative.

♣ **Esempio 1.4.** Considerati $x = 57.27$ e $\tilde{x} = 57.3$, si ha:

$$E' = \frac{|57.27 - 57.3|}{57.27} \simeq 0.52 \times 10^{-3} < 10^{-3} = 10^{-4+1}\,,$$

ma x e $\tilde{x}$ non hanno le prime quattro cifre significative uguali, hanno uguali soltanto le prime due.

♣

La proprietà 1.2 non individua il massimo valore di m per cui vale la relazione (1.1) e non può pertanto essere invertita. Si può invece dimostrare che se

$$E' = \frac{|x - \tilde{x}|}{|x|} < 10^{-m}\,,$$

allora risulta che x e $\tilde{x}$ hanno almeno le prime m cifre significative uguali, a patto di trascurare la restrizione formale di cui si è parlato nella nota (2). In seguito tale restrizione sarà trascurata.

Non è possibile affermare che, in generale, l'errore relativo sia più efficace dell'errore assoluto; infatti la scelta dell'uno o dell'altro dipende dal particolare problema in

[3] Con l'espressione *cifre significative* si indicano tutte le cifre comprese tra la prima e l'ultima cifra non nulla del numero considerato, incluse la prima e l'ultima. Ad esempio, considerati $x_1 = 4000000$, $x_2 = 0.000000200$ e $x_3 = 0.04501010$, si ha che x_1 e x_2 hanno una sola cifra significativa (rispettimamente la cifra 4 e la cifra 2) e x_3 ha sei cifre significative (le cifre 450101).

esame. Per esempio, quando si progetta il controllo di un missile a lunga gittata, si desidera che l'obiettivo prefissato sia raggiunto con un errore indipendente dalla distanza dell'obiettivo stesso dal punto di lancio. In questo caso, infatti, interessa proprio che l'errore assoluto tra le coordinate del punto di impatto e quelle dell'obiettivo sia tale da consentire la distruzione dell'obiettivo stesso.

Si osservi infine che in molti casi non si ha interesse a conoscere esattamente il valore numerico dell'errore (assoluto o relativo), ma si cerca di avere informazioni sul suo *ordine di grandezza*[4].

In generale, nella risoluzione di un problema mediante calcolatore (*risoluzione computazionale*), non ci si aspetta di calcolare la soluzione "esatta", in quanto il processo di risoluzione comporta sempre l'introduzione di errori. Tali errori sono intrinsecamente legati a questo processo; ciò implica che essi non possono essere evitati, ma solo controllati e stimati, se possibile quantitativamente, per esempio attraverso maggiorazioni. A tale proposito è doveroso citare la frase del Prof. B.Parlett, professore emerito all'università della California, Berkeley, apparsa nel SIAM NEWS, vol. 36, N. 9, del Novembre 2003. B.Parlett, riferendosi ad alcune questioni di calcolo numerico a precisione finita, afferma che le difficoltà sono spesso subdole, piuttosto che profonde ("[...] My suggestion is that the difficulties in matrix computations may not be *deep* but they are *subtle*."). Si suddivida il processo di risoluzione computazionale di un problema nelle fasi fondamentali seguenti (Figura 1.1):

1. descrizione del problema mediante un *modello (problema) matematico*;
2. approssimazione del modello matematico mediante un *modello (problema) numerico*;
3. descrizione del modello numerico mediante un *algoritmo*;
4. *implementazione* dell'algoritmo in uno specifico ambiente di elaborazione (*software*).

[4]

Definizione 1.3. (Ordine di grandezza)
Un numero reale x ha **ordine di grandezza** β^m *se*

$$|x| = \mu \times \beta^m \quad \beta^{-1} \leq \mu < 1 \,, \tag{1.2}$$

dove β è la base di numerazione del sistema aritmetico considerato; in particolare, se $m = 0$, ovvero $\beta^m = 1$, si dirà che x è dell'ordine dell'unità. Se vale la (1.2) si dirà anche che x è di ordine m, sottintendendo la base di numerazione.

Con tale definizione, se si considera ad esempio $\beta = 10$, risulta che $r = 0.0000000000529$ (raggio di Bohr dell'atomo di idrogeno, in cm) ha ordine di grandezza 10^{-10}, ovvero è di ordine -10, e $c = 300000000$ (velocità della luce in m/sec) ha ordine di grandezza 10^9, ovvero è di ordine 9.

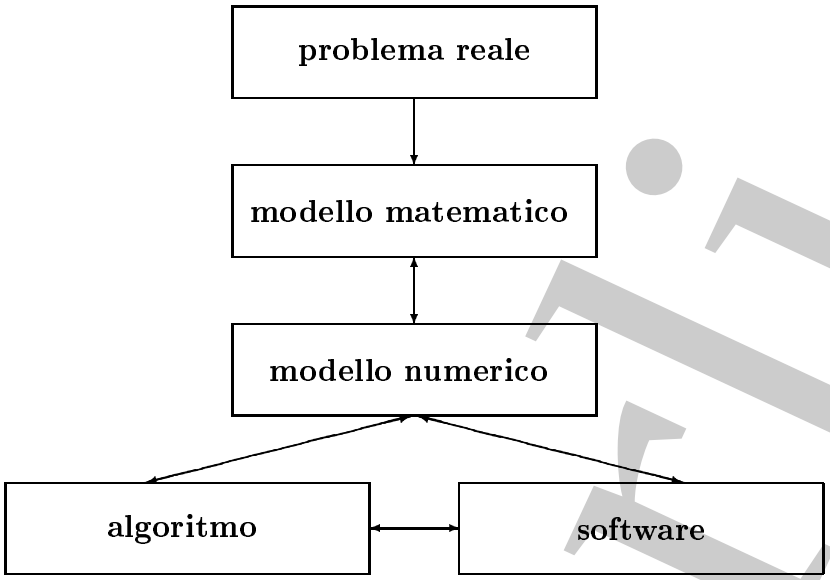


Figura 1.1: Schema del processo di risoluzione computazionale di un problema.

Tale processo presuppone che il problema in esame sia stato già formulato correttamente ed analizzato, individuando chiaramente le informazioni a disposizione per ottenerne la soluzione.

La fase 1 consiste nel passaggio dal problema reale (fisico, chimico, ingegneristico, economico, ...) ad un problema matematico, mediante l'individuazione di relazioni logico-matematiche (equazioni algebriche, equazioni differenziali, disequazioni, ...) che colleghino tra loro le informazioni note, relative al problema in esame, e la soluzione. Si noti che in questa fase devono essere individuate le quantità che si ritiene descrivano esaustivamente il problema (i *dati* del problema) e quelle che sono assunte come *soluzione* del problema stesso.

♣ **Esempio 1.5.** Si consideri una sbarra cilindrica in posizione orizzontale, con un'estremità fissa e l'altra libera. Si vuole determinare la forma assunta dalla sbarra quando all'estremità libera è applicata una forza (deflessione elastica della sbarra), avendo a disposizione le seguenti informazioni: lunghezza e sezione della sbarra, materiale di cui è costituita la sbarra, forza applicata all'estremità libera.

Si vuole costruire un modello matematico per tale problema. Se si suppone che la sezione della sbarra sia piccola rispetto alla sua lunghezza, ci si riconduce ad un problema bidimensionale. Scelto un sistema di riferimento cartesiano ortogonale Oxy, con l'origine coincidente con l'estremità fissa, l'asse x tale che la sbarra giaccia su di esso e l'asse y orientato nella direzione della forza applicata all'estremità libera (Figura 1.2), si ha che la deflessione elastica è la soluzione del seguente problema differenziale

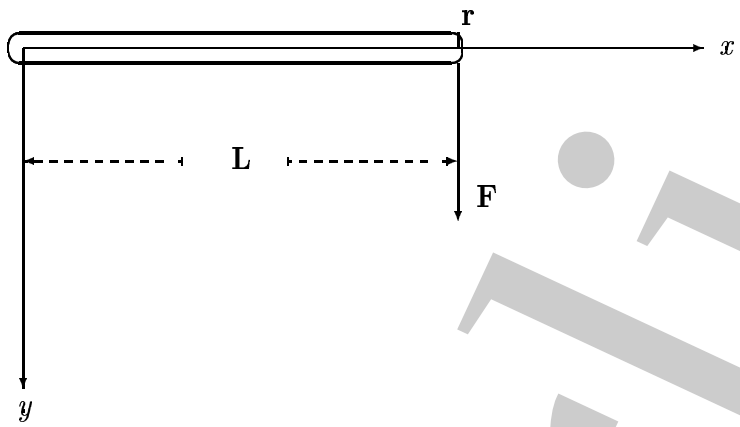


Figura 1.2: Deflessione elastica di una sbarra. Rappresentazione della sbarra prima della flessione, in un riferimento cartesiano.

(modello A):

$$\begin{aligned} &\frac{y''(x)}{(1+y'(x)^2)^{3/2}} = \frac{F(L-x)}{CI} \\ &y(0) = 0 \\ &y'(0) = 0 \end{aligned}$$

dove C è una costante dipendente dal materiale (modulo di trazione o di Young), I il momento di inerzia della sezione trasversale della sbarra, L la lunghezza della sbarra, F la forza applicata all'estremità libera. Le condizioni $y(0) = 0$ e $y'(0) = 0$ indicano che lo spostamento e la pendenza dell'estremità fissa della sbarra sono nulli. Si noti che C, I, L, F e x sono grandezze note e sono dette *dati di input* del modello matematico; la quantità y, da determinare, è la soluzione del modello ed è denominata *dato di output*.

Si osservi che nella costruzione di tale modello sono stati introdotti degli errori dovuti al fatto che si trascura non solo la sezione della sbarra, ma anche la torsione, la forza di gravità, etc. Un'ulteriore semplificazione del problema si può ottenere supponendo che il peso applicato sia "piccolo" e quindi la pendenza $y'(x)$ sia tale che $y'(x)^2 << 1$. In tal caso si ha il seguente problema differenziale (modello B):

$$\begin{aligned} &y''(x) = \frac{F(L-x)}{CI} \\ &y(0) = 0 \\ &y'(0) = 0\ . \end{aligned}$$

Quest'ultima semplificazione facilita il calcolo della soluzione, ma, ovviamente, introduce in essa ulteriori errori. In Figura 1.3 sono riportati i grafici delle soluzioni relative ai modelli A e B per una sbarra di acciaio ($C = 200GN/m^2 = 200 \times 10^6 kg/(mm \cdot s^2)$) [5] di lunghezza $L = 100$ mm e sezione di raggio $R = 1.5$ mm, al variare della forza F ($F = 10^4N, 5 \cdot 10^4N, 10 \cdot 10^4N, 15 \cdot 10^4N$)[6] Si noti che per

[5] $1GN = 10^9N = 10^9 kg \cdot m/s^2$

[6] Se la sezione trasversale della sbarra è circolare di raggio R, il centro di massa è al centro della circonferenza ed il momento di inerzia lungo ciascun asse che giace nella sezione trasversale e passante per il centro, è $I = \frac{\pi}{4}R^4$.

$F = 10^4 N$ i grafici relativi alle due soluzioni sono praticamente coincidenti, per $F = 5 \cdot 10^4 N$ essi differiscono leggermente, per $F = 10 \cdot 10^4 N$ tale differenza diventa sensibile (le posizioni dell'estremo libero della sbarra secondo il modello A ed il modello B distano circa 10 mm), per $F = 15 \cdot 10^4 N$ le due soluzioni si discostano significativamente.

♣

In generale, anche se si riesce a determinare un'espressione analitica della soluzione di un problema matematico, spesso non è possibile utilizzarla per calcolare il valore della soluzione in un insieme assegnato di punti. Ciò accade, ad esempio, per il modello A della deflessione elastica di una sbarra[7], o, più semplicemente, per la serie $\sum_{n=0}^{\infty}(1/n^3)$, la funzione integrale $\int_0^x (\sin t/t)dt$, la funzione $\log x$, etc. La fase 2 consiste nel passare dal modello matematico al *modello numerico*, cioè ad un particolare modello matematico che coinvolge un numero *finito* di numeri reali e relazioni matematiche tra questi, calcolabili con un numero *finito* di operazioni aritmetiche.

♣ **Esempio 1.6.** Si consideri lo sviluppo in serie di Mac Laurin della funzione e^x:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \,. \tag{1.3}$$

Se si sostituisce alla serie la somma dei primi n termini si ottiene un'espressione valutabile con un numero finito di operazioni aritmetiche. Naturalmente tale sostituzione comporta l'introduzione di un errore, detto *errore di troncamento analitico*, che dipende dalla scelta di n. Più precisamente, dato che

$$e^x = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + R_n(x) \,, \tag{1.4}$$

dove $R_n(x)$ è il resto di indice n della serie considerata, si ha che $R_n(x)$ rappresenta l'errore di troncamento analitico. A questo punto bisogna affrontare il problema di stimare l'errore di troncamento analitico. In questo caso si ha:

$$R_n(x) = e^{\xi} \frac{x^{n+1}}{(n+1)!} \,,$$

dove ξ appartiene ad un opportuno intervallo del tipo $[-a, a]$; dunque, una stima calcolabile dell'errore di troncamento analitico è la seguente:

$$|R_n(x)| \leq e^a \frac{|x|^{n+1}}{(n+1)!} \,. \tag{1.5}$$

[7] In questo caso la soluzione (e, quindi, la forma della sbarra) ha l'espressione parametrica:

$$\begin{aligned} x &= \sqrt{\frac{2IC}{F}}(\sqrt{\cos\theta_0} - \sqrt{\cos\theta_0 - \cos\theta}) \\ y &= \sqrt{\frac{IC}{2F}} \int_0^{\pi/2} \frac{\cos\theta d\theta}{\sqrt{\cos\theta_0 - \cos\theta}} \end{aligned}$$

dove $\theta = \theta(l)$ è l'angolo formato dalla tangente alla sbarra nel generico punto a distanza l dall'estremo fissato, e dall'asse y. All'estremo fissato ($l = 0$), $\theta(0) = \frac{\pi}{2}$, mentre all'estremo libero ($l = L$, lunghezza della sbarra) si pone $\theta(L) = \theta_0$.

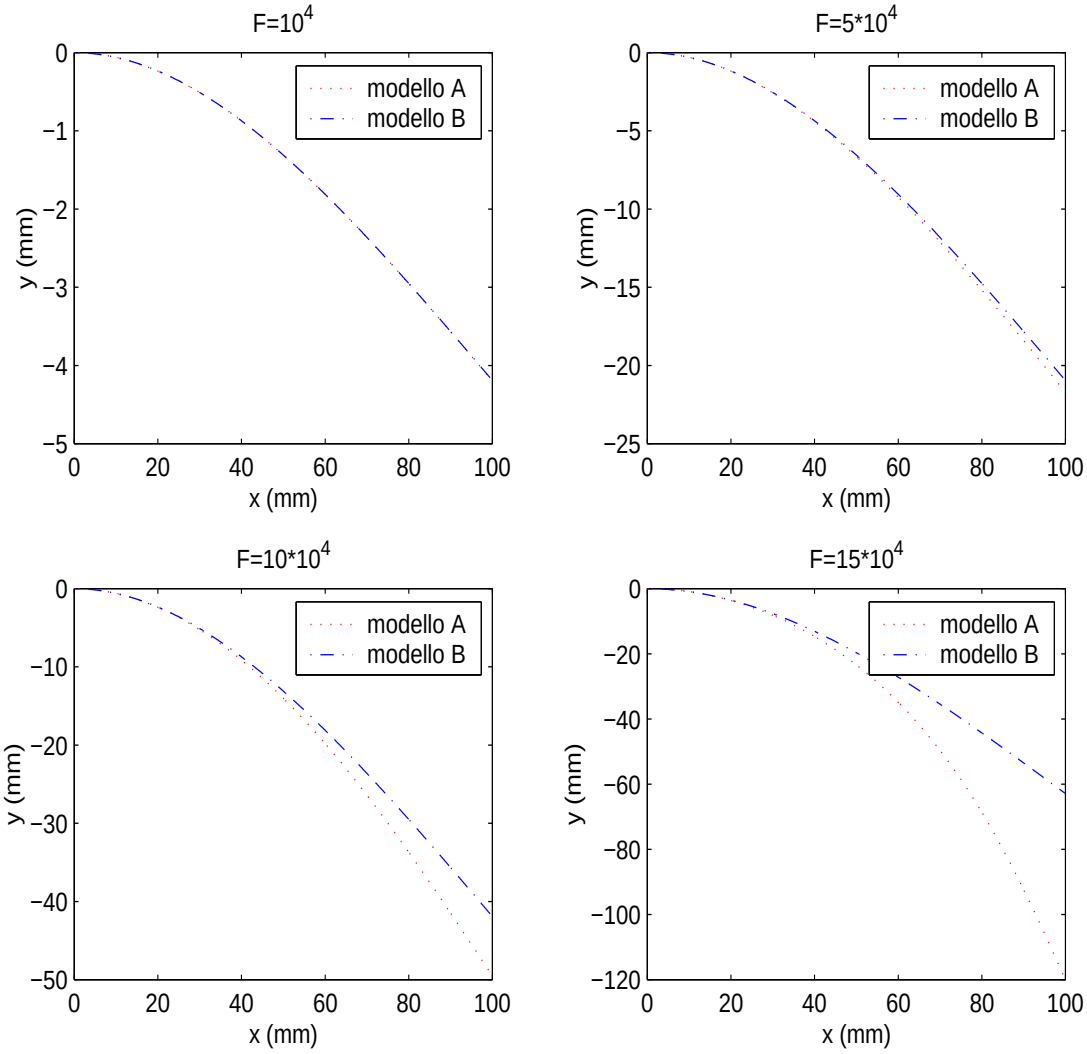


Figura 1.3: Deflessione elastica di una sbarra. Soluzioni relative ai modelli A e B.

Si osservi comunque che la maggiorazione (1.5) risolve solo apparentemente il problema della stima di $R_n(x)$; infatti il problema si sposta alla determinazione di a, o meglio di un valore di a per cui $e^a \frac{|x|^{n+1}}{(n+1)!}$ non sia "eccessivamente grande".

♣

♣ **Esempio 1.7.** Sia $f(x)$ una funzione reale di variabile reale, derivabile. Se si sostituisce alla derivata di $f(x)$ il rapporto incrementale:

$$\frac{df}{dx}(x) \simeq \frac{f(x+h) - f(x)}{h} ,$$

dove h è scelto opportunamente[8], si ottiene un'espressione valutabile con un numero finito di operazioni aritmetiche, supposto, ovviamente, che il calcolo di $f(x)$ in un punto possa essere effettuato con un numero finito di operazioni aritmetiche. Chiaramente l'errore di troncamento analitico, o *errore di discretizzazione* [9], dipende da h e, poiché

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = \frac{df}{dx}(x) ,$$

l'errore di troncamento analitico tende a 0 al tendere di h a 0.

♣

La fase 3 consiste nell'individuare un *algoritmo* risolutivo per il modello numerico, cioè una sequenza finita di operazioni aritmetiche che, se eseguite, consentono di calcolare, a partire dai dati di input, i dati di output.

Il concetto di algoritmo è indissolubilmente legato al suo esecutore, nel senso che le operazioni che vi compaiono devono essere chiaramente specificate ed eseguibili in relazione alle possibilità dell'esecutore. Nel nostro caso l'esecutore di algoritmi è un calcolatore con un *sistema aritmetico a precisione finita*, cioè che opera solo su numeri rappresentati mediante un numero finito di cifre significative.

Questa fase del processo risolutivo comporta l'introduzione di un errore, detto *errore di roundoff*, dovuto proprio all'utilizzo di un sistema aritmetico a precisione finita. E' a questo tipo di errore che sarà dedicato ampio spazio in seguito.

La fase 4 consiste nell'implementazione dell'algoritmo in uno specifico ambiente di calcolo (*software* o *programma*).

I dati di output del programma dipendono quindi non solo dall'algoritmo, ma anche dalle caratteristiche specifiche dell'ambiente di calcolo, come, ad esempio, i compilatori, il particolare sistema aritmetico, etc. I dati di output del programma sono assunti come la "soluzione" del problema reale nell'approccio computazionale.

[8] A tale proposito si veda l'esercizio 16, del §1.10.2

[9] Nella nostra accezione un problema matematico è caratterizzato dal concetto di infinito ed un problema numerico dalla finitezza; un problema matematico può essere inoltre continuo o numerabile. Talvolta il passaggio dal modello matematico al modello numerico richiede prima il passaggio da un processo continuo ad uno numerabile e poi da questo ad un processo finito; in tal caso, spesso l'errore associato al primo passaggio è detto di *discretizzazione* e quello associato al secondo è detto di *troncamento analitico*. Qui si userà la denominazione *troncamento analitico* per indicare l'errore complessivo dovuto alla sostituzione del modello matematico col modello numerico.

1.2 I sistemi aritmetici a precisione finita

Il termine *sistema aritmetico a precisione finita* si riferisce in generale ai criteri di rappresentazione in memoria dei dati di tipo numerico ed alle operazioni elementari (dell'unità aritmetica) definite su di essi.[10]
I tipi di dati[11] numerici elementari qui considerati sono il tipo *intero* ed il tipo *reale floating-point* (reale f.p.)[12], poiché, come si vedrà in seguito, sono quelli che possono generare situazioni computazionali che riducono o annullano l'affidabilità di un algoritmo numerico.

1.2.1 Il sistema aritmetico intero

Il criterio di rappresentazione dei dati di tipo intero è quello posizionale in base β ($\beta \in \mathcal{N}$, $\beta > 1$) [13]. Usualmente $\beta = 2$, cioè si utilizza la rappresentazione binaria; ad esempio:

$$\begin{aligned} 1_{10} &= 1_2 , \\ 3_{10} &= 11_2 , \\ 16_{10} &= 10000_2 , \\ 734_{10} &= 1011011110_2 , \end{aligned}$$

e la memorizzazione avviene ponendo in una locazione la stringa di bit, da destra verso sinistra. Il bit più a sinistra è riservato all'informazione (binaria) del segno.

La lunghezza finita l di una locazione di memoria comporta l'esistenza di un massimo (e di un minimo) intero rappresentabile. Si consideri, ad esempio, un sistema aritmetico intero con $l = 8$ [14]; si ha allora che il numero $64_{10} = 1000000_2$ è rappresentabile e viene memorizzato nel modo seguente:[15]

+	1	0	0	0	0	0	0

In tale sistema il più grande numero rappresentabile, che sarà detto $imax$, è 127_{10}, poichè la sua memorizzazione è

[10] Si prenderà in considerazione solo la rappresentazione dei numeri in memoria (rappresentazione interna). La rappresentazione dei numeri sui dispositivi di I/O (video, stampante, etc.) è la normale rappresentazione decimale; allo stesso modo si prenderanno in considerazione solo le operazioni aritmetiche eseguibili dall'unità aritmetica.

[11] Si definisce *tipo di dato* l'insieme dei valori che un dato può assumere.

[12] Il tipo di dato *complesso f.p.* è essenzialmente riconducibile a quello reale f.p., a cui appartengono la parte reale ed il coefficiente dell'immaginario.

[13] $\mathcal{N}$ indica l'insieme dei numeri naturali.

[14] Nella maggior parte degli calcolatori $l = 32, 64, 128$.

[15] Per maggiore chiarezza, per rappresentare il bit del segno sono usati i simboli + e − al posto delle cifre binarie 0 e 1.

+	1	1	1	1	1	1	1

mentre la rappresentazione binaria di $128_{10} = 10000000_2$ è costituita da 8 cifre e non può essere memorizzata. Si può concludere che un intero $n \in \mathcal{Z}$ [16] è un intero *rappresentabile* se e solo se

$$|n| \leq imax$$

e che l'insieme I degli interi rappresentabili è l'insieme finito

$$\{-imax, -imax + 1, ..., -1, 0, 1, ..., imax - 1, imax\}$$

dove

$$imax = 2^{l-1} - 1 \ .$$

Se $n \in \mathcal{Z}$, ma $n \notin I$, allora n non è rappresentabile ed un eventuale tentativo di memorizzazione provoca la situazione eccezionale detta *overflow* degli interi. In generale l'insieme I è individuato univocamente dalla sola lunghezza l. [17]

Sui dati di tipo intero sono definite le operazioni: *addizione, sottrazione, moltiplicazione* e *divisione*[18], denotate qui in modo diverso dalle corrispondenti operazioni in $\mathcal{Z}$: $\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, $\boxed{/}$. Il risultato r di un'operazione $\boxed{\#}$ (dove $\#$ sta per $+, -, \times, /$) si dice *definito* se $r \in I$, altrimenti si dice che esso è indefinito e che si è verificata una situazione eccezionale di overflow degli interi.

♣ **Esempio 1.8.** Siano $imax = 100$, $n = 50$, $m = 60$. Si ha:

$$|n + m| = 110 > imax$$

e quindi il risultato dell'operazione $50 \boxed{+} 60$ è indefinito. ♣

In generale, se n e m sono interi tali che

$$|n| < imax, \ |m| < imax \ ,$$

[16] $\mathcal{Z}$ indica l'insieme degli interi relativi.

[17] Oltre alla rappresentazione per segno e modulo, qui utilizzata, esistono altre rappresentazioni degli interi relativi. In particolare, si ricorda quella del *complemento a due* [7]: il primo bit di sinistra ha ancora il ruolo di bit di segno, ed i numeri non negativi hanno la stessa forma fornita dalla rappresentazione in segno e modulo, mentre il valore, in modulo, di un numero negativo rappresentato in complemento a due si ottiene invertendo il valore di ogni singolo bit (complemento a uno), valutando la rappresentazione come intero senza segno, ed aggiungendo uno al risultato; in essa pertanto, il tipo intero non è simmetrico rispetto allo zero, essendo $imin = -imax - 1$ ($imin$ = minimo intero rappresentabile). Ad esempio nella rappresentazione in *complemento a due* il numero $1010_2 = -6_{10}$, infatti: $1010 = -(0101 + 1) = -(5 + 1) = -6$.

[18] Il risultato della divisione intera è la parte intera del quoziente.

allora

$$n \boxed{\#} m = \begin{cases} n\#m & \text{se } |n\#m| \leq imax \\ \text{indefinito} & \text{se } |n\#m| > imax\,. \end{cases} \tag{1.6}$$

♣ **Esempio 1.9.** In un sistema aritmetico intero con $\beta = 2$ e $l = 32$ si vuole calcolare $n!$. Per $n > 13$ tale calcolo provoca overflow in quanto

$$13! > 2^{31} - 1 = imax$$

(si ricordi che uno dei 32 bit a disposizione per la rappresentazione del numero è riservato al segno).

♣

♣ **Esempio 1.10.** Siano $imax = 100$, $n = 60$, $m = 50$ e $p = -40$ e si voglia calcolare

$$n + m + p\,.$$

Il risultato dell'operazione

$$(n \boxed{+} m) \boxed{+} p = (60 \boxed{+} 50) \boxed{+} (-40)$$

è indefinito, essendo

$$|60 + 50| > 100 = imax\,,$$

mentre si ha che

$$n \boxed{+} (m \boxed{+} p) = 60 \boxed{+} (50 \boxed{+} (-40)) = 60 \boxed{+} 10 = 70\,.$$

Non vale dunque la proprietà associativa dell'addizione.

♣

In generale, la (1.6) comporta che alcune proprietà dell'aritmetica tradizionale (proprietà associativa dell'addizione e proprietà distributiva della moltiplicazione rispetto all'addizione) non siano più valide.

In conclusione, ricapitolando quanto detto precedentemente, un sistema aritmetico intero è costituito da:

- un insieme finito $I \subset \mathcal{Z}$, i cui elementi (gli elementi del tipo intero), sono detti gli *interi rappresentabili*;
- un criterio di rappresentazione degli elementi di I, su cui è basata la memorizzazione degli elementi del tipo;
- un insieme di operazioni, dette operazioni aritmetiche sul tipo intero, in cui sia gli operandi sia i risultati sono elementi del tipo;
- un insieme di algoritmi (cablati nell'unità aritmetica), che consentono l'esecuzione delle operazioni aritmetiche sul tipo intero.

1.2.2 Il sistema aritmetico floating-point

Il criterio di rappresentazione degli elementi di tipo reale è quello della rappresentazione *floating-point normalizzata* (f.p.n.) in base β [19]. Tale criterio si basa sull'idea di utilizzare solo le cifre significative di un numero reale, in modo che questo sia rappresentato come prodotto di un numero f, con $\beta^{-1} \leq |f| < 1$, per una potenza intera della base β. Per esempio, il numero $x = 22.334$ ha la rappresentazione f.p.n. in base 10:

$$x = 0.22334 \times 10^2\,, \quad \text{con } f = 0.22334,\ 10^{-1} \leq 0.22334 < 1\,.$$

Si noti che la limitazione $\beta^{-1} \leq |f| < 1$ (*normalizzazione*) implica che la prima cifra della parte frazionaria di f (nella base β) sia diversa da zero [20] e rende unica la rappresentazione. Qualsiasi numero reale $x \neq 0$ può essere rappresentato in forma f.p.n. e si ha

$$x = f \times \beta^e, \quad e \in \mathcal{Z}\,,\ \beta^{-1} \leq |f| < 1\,.$$

Detto m il numero intero formato dalle cifre della parte frazionaria di f ed avente lo stesso segno di f, il numero $x = f \times \beta^e$ è individuato dalla coppia (m, e); m è detto mantissa ed e è detto esponente della rappresentazione f.p.n.

♣ **Esempio 1.11.** Considerato $\beta = 10$, si ha:

$$\begin{array}{lcll}
x = 35.16904 & \overset{f.p.n.}{\Longrightarrow} & x = 0.3516904 \times 10^2 & \Longrightarrow \quad (m,e) = (+3516904, +2) \\
x = 0.00012 & \overset{f.p.n.}{\Longrightarrow} & x = 0.12 \times 10^{-3} & \Longrightarrow \quad (m,e) = (+12, -3) \\
x = -300000000 & \overset{f.p.n.}{\Longrightarrow} & x = -0.3 \times 10^9 & \Longrightarrow \quad (m,e) = (-3, +9) \\
x = -0.00000479 & \overset{f.p.n.}{\Longrightarrow} & x = -0.479 \times 10^{-5} & \Longrightarrow \quad (m,e) = (-479, -5)
\end{array}$$

♣

La memorizzazione di un numero reale rappresentato mediante il criterio f.p.n. richiede quindi la memorizzazione delle informazioni seguenti: il segno di m, le cifre di m, il segno di e e le cifre di e. [21] A causa della lunghezza finita di una locazione di memoria si ha un numero finito di cifre per la rappresentazione di m ed un numero finito di cifre per la rappresentazione di e.

Il criterio di rappresentazione f.p.n. è quindi univocamente caratterizzato dalle quantità:

[19] La scelta di β è fatta sulla base di considerazioni di efficienza e semplicità di implementazione hardware. L'unico requisito teorico è che β sia un numero intero maggiore dell'unità.

[20] Infatti si ha:

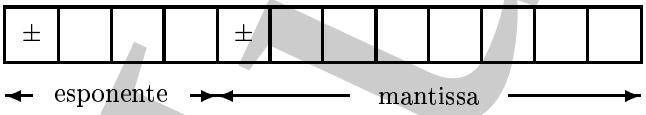
$$|f| = 0.d_1 d_2 \ldots\,, \qquad \begin{array}{l} 1 \leq d_1 \leq \beta - 1\,, \\ 0 \leq d_k \leq \beta - 1\,, \quad k > 1\,. \end{array}$$

[21] Si noti che se $\beta = 2$ la prima cifra di m è necessariamente 1 e quindi è inutile memorizzarla esplicitamente (memorizzazione con il primo *bit implicito*).

1. la base β;
2. il massimo numero t di cifre (nella base β) della mantissa m;
3. il massimo numero di cifre (nella base β) per l'esponente e, ovvero un minimo ed un massimo esponente rappresentabile.

Il numero t è detto *precisione* del sistema aritmetico f.p.n.. Si denoteranno inoltre con $emin$ ed $emax$ rispettivamente il minimo ed il massimo esponente rappresentabile.[22] In termini di mantissa ed esponente, l'unicità della rappresentazione $\pm m \times \beta^e$, risulta legata all'essere $emin \leq e \leq emax$ ed alla condizione di normalizzazione per la mantissa, $\beta^{t-1} \leq m \leq \beta^t - 1$. Negli esempi seguenti β sarà rappresentata in base 10, mentre m ed e saranno in genere rappresentati in base β.

♣ **Esempio 1.12.** Si consideri il criterio f.p.n. a precisione finita caratterizzato da $\beta = 2$, $t = 7$, $emin = -7$ ed $emax = 7$, a cui corrisponde una locazione di memoria (fissata di lunghezza 12) suddivisa come segue:



Si considerino i numeri, per semplicità già in forma binaria,

$$a = 1011.11\,, \quad b = -101.100111\,, \quad c = 101110101.1\,,$$

che in notazione f.p.n. sono rappresentati come segue:

$$\begin{array}{lllll} a = 0.101111 \times 2^{100} & \text{cioè} & m = +101111\,, & e = +100, \\ b = -0.101100111 \times 2^{11} & \text{cioè} & m = -101100111\,, & e = +11, \\ c = 0.1011101011 \times 2^{1001} & \text{cioè} & m = +1011101011\,, & e = +1001. \end{array}$$

Solo a può essere rappresentato secondo il criterio f.p.n. a precisione finita fissato e quindi memorizzato nella corrispondente locazione di memoria. Infatti, b ha l'esponente rappresentabile ($emin \leq 3 \leq emax$), ma il numero di cifre della sua mantissa è 9, mentre per c non sono rappresentabili nè l'esponente ($9 > emax$), nè la mantissa ($10 > t$).[23]

♣

[22] Si noti che, poiché lo zero ha mantissa nulla, la sua rappresentazione f.p.n. è anomala; si assumerà quindi

$$0 \overset{f.p.n.}{\Longrightarrow} 0.00\ldots \times \beta^{emin}\,.$$

[23] La rappresentazione dell'esponente e, qui considerata, è detta *rappresentazione segno/modulo*. Spesso, però, l'esponente e non è rappresentato in tale forma, ma nella forma $e + e_b$, dove e_b è un numero intero (detto *bias*) tale che $e + e_b \geq 0$. Ad esempio, se $\beta = 2$ e il numero di cifre per l'esponente (segno incluso) è 8, si considera solitamente $e_b = 127$ e $0 \leq e + e_b \leq 255$ (si noti che l'insieme degli esponenti rappresentabili non è simmetrico rispetto allo zero, in quanto $emin = -127$ ed $emax = 128$). Tale rappresentazione è detta *eccesso 127*.

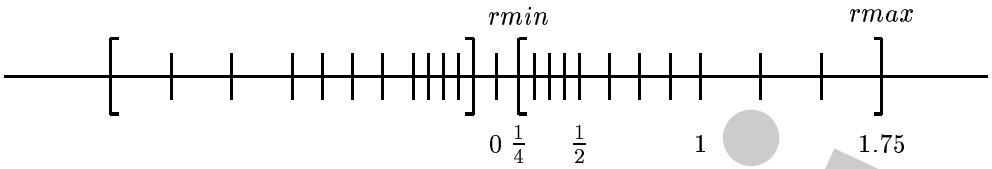


Figura 1.4: Rappresentazione grafica dell'insieme $F(\beta, t, emin, emax)$, e del corrispondente insieme di rappresentabilità (cfr. esempio 1.13)

Il fatto che le locazioni di memoria abbiano lunghezza finita comporta conseguenze notevoli:

1. esistono un massimo numero reale rappresentabile $rmax$ (come per il tipo intero) ed un minimo numero reale positivo rappresentabile $rmin$ (ciò discende dalla necessaria finitezza della rappresentazione di e);

2. non tutti i numeri reali appartenenti all'intervallo $[rmin, rmax]$ sono rappresentabili secondo il criterio f.p.n. a precisione finita (ciò discende dalla necessaria finitezza della rappresentazione di m).

Fissato un particolare criterio f.p.n. a precisione finita, consegue che risulta univocamente determinato l'insieme $F = F(\beta, t, emin, emax)$ degli elementi del tipo reale che gode delle seguenti proprietà:

- F ha un elemento di massimo modulo, $rmax$, ed un elemento di minimo modulo, $rmin$;

- F contiene lo zero;

- F è costituito da un numero finito di elementi;

- gli elementi di F sono i numeri reali che appartengono all'insieme

$$[-rmax, -rmin] \cup \{0\} \cup [rmin, rmax] \tag{1.7}$$

  ed in notazione f.p.n. hanno la mantissa con un numero di cifre minore o uguale a t.

Gli elementi di F si dicono *numeri rappresentabili esattamente* o *numeri macchina* e l'insieme (1.7) si dice *insieme di rappresentabilità* (Figura 1.4)

♣ **Esempio 1.13.** Si consideri $F(2,3,-1,1)$. F è il sottoinsieme finito di $\Re$:

$$\begin{aligned} F &= \{0,\ \pm 0.100\times 2^{-1},\ \pm 0.101\times 2^{-1},\ \ldots,\ \pm 0.110\times 2^{1},\ \pm 0.111\times 2^{1}\} \quad = \\ &= \{0,\ \pm 0.25_{10},\ \pm 0.3125_{10},\ \ldots,\ \pm 1.5_{10},\ \pm 1.75_{10}\}, \end{aligned}$$

costituito da 25 elementi (Figura 1.4). In questo caso $rmax = 0.111\times 2^{1} = 1.75_{10}$ e $rmin = 0.100\times 2^{-1} = 0.25_{10}$.

♣

In generale, dato $F(\beta,t,emin,emax)$, il numero di elementi di F è[24]

$$2(\beta-1)\beta^{t-1}(emax-emin+1)+1$$

e risulta

$$\begin{aligned} &rmax = 0.dd\ldots d\times\beta^{emax} = \beta^{emax}(1-\beta^{-t}) \quad (d=\beta-1);\\ &rmin = 0.10\ldots 0\times\beta^{emin} = \beta^{emin-1} \end{aligned}$$

Gli elementi di F non sono uniformemente spaziati fra loro, ma solo fra potenze della base successive. Dunque la spaziatura assoluta non è uniforme, ma quella relativa lo è; infatti ogni elemento di F differisce dagli adiacenti per una sola unità sull'ultima cifra della mantissa.

Per ogni numero reale $x = f\times\beta^{e}$ si verifica una ed una sola delle tre situazioni:

1. $x\in F$ ed è quindi esattamente rappresentabile;
2. x non è rappresentabile in F;
3. x è rappresentabile, ma non esattamente, in F.

Il caso 2 si presenta se $|x| < rmin$, ed allora si dice che si è verificato un *underflow*, oppure se $|x| > rmax$, ed allora si dice che si è verificato un *overflow* ($rmin$ e $rmax$ sono chiamati rispettivamente *soglia di underflow* e *soglia di overflow*); tale situazione in pratica si ha quando $e < emin$ oppure $e > emax$. Per la gestione delle cosiddette *situazioni eccezionali* si rimanda al paragrafo 1.7 sul sistema aritmetico standard IEEE.

♣ **Esempio 1.14.** Dato l'insieme $F(2,3,-1,1)$ (cfr. esempio 1.13), si ha underflow per ogni $x\in\Re$ tale che $x < 0.25$ ed overflow per ogni $x > 1.75$.

♣

Il caso 3 si verifica quando $emin \le e \le emax$ e il numero di cifre binarie di m è maggiore di t. Si pone dunque il problema di come rappresentare x mediante un elemento di F.

[24] Fissato un valore dell'esponente, la prima cifra della mantissa di un numero diverso da zero può assumere $\beta-1$ valori distinti e ciascuna delle rimanenti $t-1$ cifre può assumere β valori distinti, per un totale di $\beta^{t-1}(\beta-1)$ possibili combinazioni delle cifre della mantissa. Se si osserva che gli esponenti rappresentabili sono $emax-emin+1$ e che per ogni numero rappresentabile positivo il suo opposto è rappresentabile, e si considera anche lo zero, si trova che il numero di elementi di F è quello specificato.

♣ **Esempio 1.15.** Si consideri $F(10, 2, -9, 9)$. Un numero $\tilde{x} \in F$ è del tipo

$$\tilde{x} = \pm 0.d_1 d_2 \times 10^e\,, \quad e = -9, -8, \ldots, 8, 9\,.$$

Il numero $x = 1.26$ non appartiene a F, ma può essere rappresentato mediante un elemento di F. Infatti x è strettamente compreso tra due numeri di F, cioè

$$1.2 < x < 1.3$$

ed uno dei due si può scegliere come approssimazione di x mediante un elemento di F.

♣

Per approssimare un numero reale $x = 0.d_1 d_1 \ldots d_t d_{t+1} \ldots \times \beta^e$ appartenente all'insieme di rappresentabilità di F, ma non rappresentabile esattamente, con un numero $fl(x) \in F$, si può utilizzare una delle due tecniche seguenti:

- *troncamento*: $fl(x)$ si ottiene troncando le cifre della mantissa seguenti la t-ma, cioè

$$fl(x) = 0.d_1 d_2 \ldots d_t \times \beta^e\,;$$

- *arrotondamento*: $fl(x)$ si ottiene da x sommando $\frac{1}{2}\beta$ alla $(t+1)$-esima cifra della sua mantissa e poi troncando le cifre seguenti la t-ma, ovvero incrementando di 1 d_t se è $d_{t+1} \geq \frac{1}{2}\beta$ e poi troncando le cifre della mantissa successive alla t-ma, cioè:

$$fl(x) = \begin{cases} 0.d_1 d_2 \ldots d_t \times \beta^e & d_{t+1} < \frac{1}{2}\beta \\ 0.d_1 d_2 \ldots (d_t + 1) \times \beta^e & d_{t+1} \geq \frac{1}{2}\beta \end{cases}$$

In relazione all'esempio 1.15 si ha:

$$\begin{array}{ll} \text{troncamento :} & fl(1.26) = 1.2\,; \\ \text{arrotondamento :} & fl(1.26) = 1.3\,. \end{array}$$

Se invece di $x = 1.26$ si considera $x = 1.24$, si ha:

$$\begin{array}{ll} \text{troncamento :} & fl(1.24) = 1.2\,; \\ \text{arrotondamento :} & fl(1.24) = 1.2\,. \end{array}$$

In un sistema aritmetico reale f.p., le caratteristiche legate alla rappresentazione, cioè l'insieme F e le funzioni di troncamento ed arrotondamento, sono dette le *caratteristiche statiche del sistema*. Si definiscono *caratteristiche dinamiche* le proprietà legate alle operazioni f.p. definite in F ed agli algoritmi utilizzati dall'unità aritmetica per implementarle.

Le operazioni f.p. sono: addizione f.p. $\oplus$, sottrazione f.p. $\ominus$, moltiplicazione f.p. $\otimes$, divisione f.p. $\oslash$, qui denotate in modo diverso dalle corrispondenti usuali operazioni in $\Re$. [25]

[25] Non ci si occuperà dell'operazione di conversione dalla rappresentazione in base β alla rappresentazione decimale e da quella decimale a quella in base β, cioè dalla rappresentazione interna a quella esterna e viceversa.

Il risultato $\tilde{r} = x \oplebig y$ (dove $\#$ sta per $+, -, \times, /$), con $x, y \in F$, si dice *definito* se $r = x\#y$ appartiene all'insieme di rappresentabilità, altrimenti il risultato è indefinito e si dice che si è verificata una situazione eccezionale di *overflow* se $|r| > rmax$, o di *underflow* se $|r| < rmin$. E' opportuno sottolineare che anche nel caso di risultato r rappresentabile si ha in generale $\tilde{r} \neq r$, in quanto $\tilde{r}$ appartiene sempre a F, mentre r può non appartenere a F.

♣ **Esempio 1.16.** Si considerino $F(10, 2, -2, 2)$ e la moltiplicazione f.p. tra $x = 2.8$ e $y = 7.7$. E' immediato verificare che $x, y \in F$; infatti in rappresentazione f.p.n. si ha $x = 0.28 \times 10^1$ e $y = 0.77 \times 10^1$ e quindi x e y sono rappresentabili esattamente. Il risultato $r = x \times y = 21.56 = 0.2156 \times 10^2$ della moltiplicazione usuale in $\Re$ appartiene all'insieme di rappresentabilità, ma, avendo una mantissa di 4 cifre, non è rappresentabile esattamente, cioè non appartiene a F.

♣

Il risultato $\tilde{r}$ di una operazione f.p. dipende dall'algoritmo utilizzato dal sistema aritmetico f.p., cioè, in definitiva, dall'organizzazione e dal progetto (a livello hardware) dell'unità aritmetica. Si considererà ottimale, da un punto di vista dinamico, un sistema aritmetico f.p. per il quale si abbia

$$\tilde{r} = x \oplebig y = fl(r) = fl(x\#y)\,. \tag{1.8}$$

La (1.8) implica che il risultato $\tilde{r}$ dell'operazione f.p., che è un numero macchina, è esattamente la rappresentazione f.p.n. a precisione finita del risultato r della corrispondente operazione in $\Re$. Un tale sistema aritmetico garantisce dunque che il risultato di qualsiasi operazione f.p. differisca dal risultato dell'operazione in $\Re$ corrispondente (il risultato esatto), di una quantità che è il solo errore di rappresentazione di r, e viene perciò detto *sistema a massima accuratezza dinamica*.

Si consideri, ad esempio, l'addizione f.p.. La sua esecuzione consta essenzialmente di quattro fasi:

1. confronto tra gli esponenti della rappresentazione f.p.n. degli addendi per individuare l'addendo con esponente più piccolo;
2. "shift" delle cifre della mantissa di tale addendo in modo che il relativo esponente risulti uguale a quello dell'altro addendo;
3. addizione delle mantisse degli addendi;
4. arrotondamento e normalizzazione del risultato.

Usualmente l'unità aritmetica memorizza gli operandi f.p., ed anche i dati intermedi generati durante l'esecuzione dell'operazione f.p., in locazioni dette registri (fisicamente all'interno dell'unità) con precisione $t_{reg} > t$. Quindi i passi 2 e 3 sono eseguiti avendo a disposizione per la mantissa un numero di cifre maggiore della precisione del sistema aritmetico considerato.

Se si suppone che $\beta = 10$ e $t = 4$ e che i passi 2 e 3 siano eseguiti facendo uso di un registro a doppia precisione, il calcolo di $\tilde{c} = a \oplus b$, con $a = 0.9983 \times 10^2$ e $b = 0.4652 \times 10^{-1}$, avviene nel modo seguente:

1. confronto degli esponenti: ovviamente $2 > -1$;
2. shift della mantissa di b: $0.4652 \longrightarrow 0.00046520$;
3. somma delle mantisse: $0.99830000 + 0.00046520 = 0.99876520$;
4. arrotondamento e normalizzazione del risultato: $c = 0.9988 \times 10^2$.

Si noti che durante il passo 4 sono state perse tre cifre significative del risultato e quindi c non è il risultato esatto dell'addizione di a e b, anche se a e b sono numeri macchina; tuttavia $c = fl(0.99876520 \times 10^2)$ e dunque, in questo caso, si ha la massima accuratezza[26].
Al contrario, se i registri hanno $t_{reg} = t$ si ottiene:

1. confronto degli esponenti: $2 > -1$;
2. shift della mantissa di b: $0.4652 \longrightarrow 0.0004$;
3. somma delle mantisse: $0.9983 + 0.0004 = 0.9987$;
4. arrotondamento e normalizzazione del risultato: $c = 0.9987 \times 10^2$;

quindi non si ha la massima accuratezza.

1.3 L'errore di roundoff

L'errore che si commette approssimando un numero reale x mediante $fl(x)$ si dice *errore di roundoff*; più precisamente, si ha:

Definizione 1.4. (Errore di roundoff)
Considerato un sistema aritmetico f.p. a precisione finita, con $F(\beta, t, emin, emax)$, sia x un numero reale appartenente all'insieme di rappresentabilità di F e sia $fl(x)$ la sua rappresentazione in F. Si dice errore assoluto di roundoff il numero

$$|fl(x) - x|\,,$$

e si dice errore relativo di roundoff il numero

$$\frac{|fl(x) - x|}{|x|}\,.$$

[26] E' possibile dimostrare che $t_{reg} = t + 3$, insieme con accorgimenti detti *sticky bit* e *rounding bit*, è tale da garantire la massima accuratezza in qualsiasi operazione f.p. [13].

$x = \hat{m} \times \beta^e$	$fl_T(x) = m_T \times \beta^e$	$fl_A(x) = m_A \times \beta^e$
-0.115237×10^{-1}	-0.11523×10^{-1}	-0.11524×10^{-1}
0.1111248×10^{4}	0.11112×10^{4}	0.11112×10^{4}
0.5723378×10^{2}	0.57233×10^{2}	0.57234×10^{2}
-0.461775×10^{1}	-0.46177×10^{1}	-0.46178×10^{1}

Tabella 1.1: Rappresentazione floating-point normalizzata: troncamento ed arrotondamento della mantissa

L'errore di roundoff si genera sia quando un numero reale, dato nella usuale rappresentazione decimale, viene rappresentato in F, cioè nel passaggio dalla rappresentazione esterna a quella interna (*errore di roundoff di rappresentazione*), sia nell'esecuzione di ogni operazione f.p. (*errore di roundoff delle operazioni f.p.*).

1.3.1 L'errore di roundoff di rappresentazione

Siano $\beta = 10$ e $t = 5$ e si indichino con $fl_T(x) = m_T \times \beta^e$ e $fl_A(x) = m_A \times \beta^e$ le rappresentazioni f.p.n. di un numero reale x ottenute, rispettivamente, per troncamento e per arrotondamento, e con $\hat{m}$ la mantissa normalizzata di x in aritmetica ordinaria (cioè ad infinite cifre).[27] Si considerino i numeri riportati in Tabella 1.1.

E' immediato osservare che $\hat{m}$ e m_T differiscono di una quantità inferiore a 10^{-5}, mentre $\hat{m}$ e m_A differiscono di una quantità non superiore a 0.5×10^{-5}; l'errore relativo di roundoff è minore di 10^{-4} nel primo caso e di 0.5×10^{-4} nel secondo (Tabella 1.2).

Si vuole ora stabilire in generale il massimo errore relativo che si commette approssimando un numero reale $x \neq 0$ con $fl(x)$ in un sistema aritmetico con $F(\beta, t, emin, emax)$.

Si consideri, ad esempio, il caso dell'arrotondamento; siano

$$\begin{aligned} x &= \hat{m} \times \beta^e, \\ fl(x) &= m_A \times \beta^e \end{aligned}$$

rispettivamente la rappresentazione f.p.n. di x (in aritmetica ordinaria) e la rappresentazione f.p.n. a precisione finita (in F) corrispondente. Dunque

$$|\hat{m}| \geq \beta^{-1}$$

[27] Da ora in poi il termine mantissa è utilizzato sia per indicare la mantissa di un numero in rappresentazione f.p.n., sia per indicare la parte frazionaria del numero in tale rappresentazione.

x	$\lvert\hat{m} - m_T\rvert$	$\lvert\hat{m} - m_A\rvert$	$\lvert fl_T(x) - x\rvert/\lvert x\rvert$	$\lvert fl_A(x) - x\rvert/\lvert x\rvert$
-0.115237×10^{-1}	0.70×10^{-5}	0.30×10^{-5}	0.61×10^{-4}	0.26×10^{-4}
0.1111248×10^{4}	0.48×10^{-5}	0.48×10^{-5}	0.43×10^{-4}	0.43×10^{-4}
0.5723378×10^{2}	0.78×10^{-5}	0.22×10^{-5}	0.14×10^{-4}	0.38×10^{-5}
-0.461775×10^{1}	0.50×10^{-5}	0.50×10^{-5}	0.11×10^{-4}	0.11×10^{-4}

Tabella 1.2: Rappresentazione floating-point normalizzata: errore di round-off assoluto e relativo

e $\hat{m}$ e m_A differiscono per una quantità non superiore a $0.5 \times \beta^{-t}$, cioè

$$|\hat{m} - m_A| \leq \frac{1}{2}\beta^{-t}.$$

Si ha:

$$\frac{|fl(x) - x|}{|x|} = \frac{|m_A - \hat{m}|}{|\hat{m}|} \leq \frac{\frac{1}{2}\beta^{-t}}{\beta^{-1}} = \frac{1}{2}\beta^{1-t},$$

da cui risulta che $\frac{1}{2}\beta^{1-t}$ è il *massimo errore relativo* che si commette approssimando x con $fl(x)$. Analogo risultato sussiste per il troncamento:

$$\frac{|fl(x) - x|}{|x|} \leq \beta^{1-t}.$$

Definizione 1.5. (Massima accuratezza relativa)
Si dice massima accuratezza relativa di un sistema aritmetico f.p. a precisione finita con $F(\beta, t, emin, emax)$ il massimo errore che si commette nel rappresentare un numero x nel sistema F

$$u = max\frac{|fl(x) - x|}{|x|} = \frac{1}{2}\beta^{1-t}, \quad nel\ caso\ dell'arrotondamento.$$
$$u' = max\frac{|fl(x) - x|}{|x|} = \beta^{1-t}, \quad nel\ caso\ del\ troncamento.$$

Come si vedrà in seguito, la massima accuratezza relativa è una delle costanti fondamentali di un sistema aritmetico f.p. a precisione finita.

♣ **Esempio 1.17.** Siano $\beta = 10$ e $t = 5$. La massima accuratezza relativa è

$$u = 0.5 \times 10^{-4} \quad \text{(arrotondamento)},$$
$$u' = 10^{-4} \quad \text{(troncamento)}.$$

♣

Si osservi che, posto $\delta = (fl(x) - x)/x$, sussiste la relazione:

$$fl(x) = x(1+\delta)\,, \qquad |\delta| \leq u \quad (|\delta| \leq u')\,. \tag{1.9}$$

Da ora in poi si prenderà in considerazione solo l'arrotondamento, perché è lo schema usato nella maggior parte degli calcolatori.

1.3.2 L'errore di roundoff delle operazioni floating-point

In un sistema aritmetico f.p. a precisione finita si introduce un errore nel risultato di ogni operazione aritmetica. Come sarà chiarito in seguito, anche la risoluzione di un problema apparentemente semplice, quale ad esempio un'equazione algebrica di secondo grado, può risultare tutt'altro che banale, in quanto nel progetto di un algoritmo è essenziale tenere conto della presenza dell'errore di roundoff, che altrimenti può condurre ad una soluzione completamente errata.

Si supponga che il sistema aritmetico garantisca la massima accuratezza dinamica; allora, indicata con $\#$ una qualsiasi delle quattro operazioni elementari e con $\circledast$ la corrispondente operazione f.p. , se a e b sono due numeri macchina si ha:

$$a \circledast b = fl(a\#b) = (a\#b)(1+\delta)\,, \tag{1.10}$$

con $|\delta| \leq u$.

♣ **Esempio 1.18.** Si considerino $\beta = 10$ e $t = 4$ e si sommino i numeri $a = 0.5496{\times}10^2$, $b = 0.8714{\times}10^1$ e $c = 0.1493 \times 10^{-1}$.
Eseguendo le addizioni secondo l'ordine $(a \oplus b) \oplus c$, con un registro a doppia precisione, si ha:

$$\begin{aligned} & (0.5496 \times 10^2 \oplus 0.8714 \times 10^1) \oplus 0.1493 \times 10^{-1} = \\ = \;& 0.6367 \times 10^2 \oplus 0.1493 \times 10^{-1} = 0.6368 \times 10^2\,, \end{aligned}$$

calcolando invece $a \oplus (b \oplus c)$ si ha:

$$\begin{aligned} & 0.5496 \times 10^2 \oplus (0.8714 \times 10^1 \oplus 0.1493 \times 10^{-1}) = \\ = \;& 0.5496 \times 10^2 \oplus 0.8729 \times 10^1 = 0.6369 \times 10^2\,. \end{aligned}$$

I risultati ottenuti nei due casi differiscono di 1 unità sull'ultima cifra significativa e quindi non vale la proprietà associativa dell'addizione.

♣

♣ **Esempio 1.19.** Siano $\beta = 10$ e $t = 4$ e si considerino i numeri $a = 0.2240 \times 10^2$, $b = 0.7953 \times 10^1$ e $c = 0.3329 \times 10^2$.
Utilizzando un registro a doppia precisione, si ha:

$$\begin{aligned} & (a \oplus b) \otimes c = \\ = \;& (0.2240 \times 10^2 \oplus 0.7953 \times 10^1) \otimes 0.3329 \times 10^2 = \\ = \;& 0.3035 \times 10^2 \otimes 0.3329 \times 10^2 = 0.1010 \times 10^4 \end{aligned}$$

e

$$\begin{aligned}&(a \otimes c) \oplus (b \otimes c) = \\ &= \quad (0.2240 \times 10^2 \otimes 0.3329 \times 10^2) \oplus (0.7953 \times 10^1 \otimes 0.3329 \times 10^2) = \\ &= \quad 0.7457 \times 10^3 \oplus 0.2648 \times 10^3 = 0.1011 \times 10^4 \,.\end{aligned}$$

Le due sequenze di operazioni non conducono allo stesso risultato e quindi non è valida la proprietà distributiva della moltiplicazione rispetto all'addizione.

♣

1.4 L'epsilon macchina

Supponiamo di eseguire, in un sistema aritmetico f.p. con $\beta = 10$ e $t = 3$, dotato di massima accuratezza dinamica, l'addizione:

$$0.994 \times 10^0 \oplus 0.177 \times 10^{-3} = 0.994 \times 10^0 \oplus 0.000177 \times 10^0 = 0.994 \times 10^0 \,.$$

In generale, quando si effettua un'addizione f.p. tra numeri aventi ordini di grandezza differenti può accadere che il risultato sia uguale all'addendo maggiore (in valore assoluto). Questo fenomeno è dovuto alla precisione finita del sistema aritmetico utilizzato ed esiste anche se il sistema garantisce la massima accuratezza dinamica.
In particolare, è interessante studiare il caso in cui l'addendo maggiore in valore assoluto è il numero 1.

Definizione 1.6. (Epsilon macchina)
In un sistema aritmetico f.p. con $F(\beta, t, emin, emax)$, si dice epsilon macchina il più piccolo numero $\epsilon \in F$ tale che

$$1 \oplus \epsilon = fl(1 + \epsilon) > 1 \,. \tag{1.11}$$

Si supponga, ad esempio, $\beta = 10$ e $t = 3$ e si calcoli il più grande numero $\eta \in F$ tale che[28]

$$1 \oplus \eta = 1 \,. \tag{1.12}$$

Chiaramente $0 < \eta = 0.\alpha_1\alpha_2\alpha_3 \times 10^p$, con $p < 0$. Si supponga inoltre che l'addizione f.p. sia eseguita con un registro a doppia precisione.

[28] E' facile convincersi che la limitatezza del sistema f.p. a precisione finita implica l'esistenza di elementi diversi dallo zero che si comportano come lo zero nella somma f.p. con 1. In realtà esiste un insieme di numeri macchina che godono di questa proprietà.

Affinché sia valida (1.12), l'operazione $1 \oplus \eta$ deve essere effettuata come segue:

$$\begin{array}{cccccccccc} 0. & 1 & 0 & 0 & 0 & 0 & 0 & \times & 10^1 & + \\ 0. & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \times & 10^1 & = \\ \hline 0. & 1 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \times & 10^1 & \end{array} \tag{1.13}$$

e la terza cifra significativa del risultato non deve essere modificata quando si effettua l'arrotondamento a 3 cifre. Quest'ultima condizione equivale a imporre che $\alpha_1 < 5$, mentre da (1.13), avendo effettuato uno shift della mantissa di η di tre cifre verso destra, si ricava

$$p + 3 = 1, \quad \text{da cui} \quad p = 1 - 3 = -2.$$

Pertanto $\eta = 0.499 \times 10^{-2}$ e quindi $\epsilon = 0.500 \times 10^{-2}$.

In generale, in un sistema aritmetico f.p. con $F(\beta, t, emin, emax)$, si ha:

$$\epsilon = \frac{1}{2}\beta^{1-t},$$

nel caso dell'arrotondamento, mentre,

$$\epsilon = \beta^{1-t}$$

nel caso del troncamento.

L'epsilon macchina coincide con la massima accuratezza relativa u. Pertanto, nel seguito, i termini epsilon macchina e massima accuratezza relativa saranno considerati sinonimi.

Nota la base di numerazione, conoscere l'epsilon macchina equivale a conoscere la precisione del sistema aritmetico; d'altra parte la definizione stessa di epsilon macchina suggerisce un modo per calcolare tale quantità nel sistema aritmetico di un calcolatore, quando non è nota la precisione. L'algoritmo per il calcolo dell'epsilon macchina è infatti basato sulla definizione 1.6; il valore finale di u è, appunto, l'epsilon macchina.

```
procedure precrel(out: u)
  /# SCOPO: calcolo dell'epsilon macchina
  /# SPECIFICHE PARAMETRI:
    var: u    : reale   {epsilon macchina}
  /# VARIABILI LOCALI:
    var: u1   : reale   {variabile di appoggio}
    var: u2   : reale   {variabile di appoggio}
  /# INIZIO ISTRUZIONI:
      u1 := 1.;
      repeat
        u := u1;
        u1 := u1/β;
        u2 := u1 + 1.;
      until(u2 = 1.)
  end precrel
```

Procedura 1.1: Algoritmo per il calcolo dell'epsilon macchina

Implementando l'algoritmo in Fortran su un calcolatore IBM RISC System/6000, in singola ed in doppia precisione, si ha rispettivamente: [29]

$$\begin{aligned} u &= 2^{-23} \simeq 0.1192093 \times 10^{-6}\,, \\ u &= 2^{-52} \simeq 0.2220446 \times 10^{-15}\,. \end{aligned} \tag{1.14}$$

Con un ragionamento analogo al precedente si verifica che, se $x, y \in F$ e $x = m \times \beta^p$, risulta:

$$x \oplus y \neq x \Longleftrightarrow |y| \geq \epsilon_x = \frac{1}{2}\beta^{p-t}\,,$$

[29] E' interessante notare che l'implementazione diretta dell'algoritmo descritto nella Procedura 1.1, può portare a risultati diversi sullo stesso calcolatore se si usano compilatori diversi. Alcuni compilatori (ad es. Microsoft Fortran 6.0) ottimizzano infatti il programma eseguibile mantenendo i valori di $u1$ ed $u2$ sempre nei registri. In tal caso l'algoritmo deve essere implementato "costringendo" il compilatore a conservare $u2$ in memoria centrale; si vuole calcolare l'epsilon macchina relativo alla precisione delle locazioni di memoria, non a quella dei registri!

cioè, se $|y| < \epsilon_x$, y non dà contributo alla somma. Quindi, dato un qualunque numero macchina x, esiste un insieme di numeri macchina che si comportano come lo zero nell'addizione f.p. con x. Questa osservazione è di importanza fondamentale nel progetto di algoritmi numerici.

Si noti che

$$\epsilon_x = \frac{1}{2}\beta^{p-t} = \beta^{p-1} \times \frac{1}{2}\beta^{1-t} = \beta^{p-1} \times u \leq |x| \times u\,;$$

in generale, quindi, si assume

$$\epsilon_x = |x| \times u\,,$$

in quanto tale numero si calcola immediatamente a partire da x e da u.

Si supponga di voler calcolare e^x, con $x > 0$. Questo problema matematico può essere approssimato dal problema numerico "calcolo di una somma" del tipo:

$$S_N = \sum_{n=0}^{N} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \ldots + \frac{x^N}{N!}\,. \tag{1.15}$$

S_N è un'approssimazione di e^x, la cui "bontà" dipende da N ed è tanto migliore quanto maggiore è N [30].

Da un punto di vista algoritmico il calcolo di S_N è basato sullo schema:

$$\begin{aligned} &S_0 = 1 \\ &S_n = S_{n-1} + x^n/n!\,, \quad n = 1, 2, \ldots, N\,. \end{aligned} \tag{1.16}$$

Una formula del tipo della (1.16) sarà detta *formula ricorrente* e la computazione basata su di essa sarà detta *processo iterativo*.

L'aspetto centrale dal punto di vista numerico consiste nella determinazione del numero N degli addendi da sommare. In Tabella 1.3 sono riportati i valori di $x^n/n!$ e di S_n per $x = 5$ e $n = 1, 2, ..., 23$, calcolati utilizzando 7 cifre decimali significative.[31]

Si noti che a partire da $n = 21$, S_n non varia, ovvero il termine $x^n/n!$ non dà contributo alla somma $S_{n-1} + x^n/n!$; i calcoli effettuati a partire da $n = 21$ sono dunque inutili.

Sulla base dell'osservazione precedente, si può affermare che piuttosto che fissare a priori N, è più conveniente che sia l'algoritmo stesso a determinare dinamicamente, cioè nel corso della sua esecuzione, il valore più opportuno di N. Nel caso non sia nota alcuna informazione sull'errore di troncamento analitico, la scelta più naturale è che l'algoritmo continui il processo iterativo (1.16) arrestandosi al più piccolo valore $\overline{n}$ per cui si abbia

[30] In questo caso si ha $\lim_{N\to\infty} S_N = e^x$

[31] Implementazione su IBM RISC System/6000, in Fortran, utilizzando la singola precisione.

n	$x^n/n!$	S_n
0	0.1000000×10^1	0.1000000×10^1
1	0.5000000×10^1	0.6000000×10^1
2	0.1250000×10^2	0.1850000×10^2
3	0.2083333×10^2	0.3933334×10^2
4	0.2604167×10^2	0.6537500×10^2
5	0.2604166×10^2	0.9141666×10^2
6	0.2170139×10^2	0.1131180×10^3
7	0.1550099×10^2	0.1286190×10^3
8	0.9688119×10^1	0.1383072×10^3
9	0.5382288×10^1	0.1436895×10^3
⋮	⋮	⋮
18	0.5958255×10^{-3}	0.1484129×10^3
19	0.1567962×10^{-3}	0.1484131×10^3
20	0.3919905×10^{-4}	0.1484132×10^3
21	0.9333107×10^{-5}	0.1484132×10^3
22	0.2121161×10^{-5}	0.1484132×10^3
23	0.4611219×10^{-6}	0.1484132×10^3

Tabella 1.3: Valori di $x^n/n!$ e di S_n ottenuti con un programma in Fortran, utilizzando una workstation IBM RISC System/6000

$$\frac{|x|^{\overline{n}}}{\overline{n}!} < \epsilon_{S_{\overline{n}-1}} = |S_{\overline{n}-1}| \times u\,, \tag{1.17}$$

cioè quando l'addendo $x^{\overline{n}}/\overline{n}!$ si comporta come lo zero nella somma f.p. con $S_{\overline{n}-1}$. Inoltre la (1.17) vale per $n > \overline{n}$ e quindi se $n > \overline{n}$ il processo iterativo non modifica alcuna cifra significativa della soluzione calcolata dall'algoritmo. L'algoritmo ha dunque determinato dinamicamente il miglior valore di N ($N = \overline{n}$). La (1.17) rappresenta un esempio di condizione che, se verificata, comporta la fine del processo iterativo (cfr. Procedura 1.2).

Si dirà *criterio di arresto* di un algoritmo basato su un processo iterativo l'insieme delle condizioni che, se verificate, determinano l'arresto del processo iterativo. L'individuazione di un criterio di arresto efficiente (cioè che non comporti iterazioni inutili) ed affidabile (cioè che non arresti prematuramente le iterazioni) è uno degli aspetti principali del progetto degli algoritmi basati su processi iterativi.
L' algoritmo descritto nella Procedura 1.2 usa il criterio di arresto precedente, che si dirà *criterio di arresto naturale*:

```
procedure espo(in: x, u ; out: sum)
 /# SCOPO: calcolo del valore della funzione esponenziale
 /# SPECIFICHE PARAMETRI:
 /# PARAMETRI DI INPUT:
   var: x     : reale    {argomento della funzione}
                         {esponenziale}
   var: u     : reale    {epsilon macchina}
 /# PARAMETRI DI OUTPUT:
   var: sum   : reale    {valore assunto dalla funzione}
                         {esponenziale in x}
 /# VARIABILI LOCALI:
   var: n     : intero   {numero di termini della serie}
   var: add   : reale    {termine generale della serie}
 /# INIZIO ISTRUZIONI:
      sum := 0.;
      n := 0;
      add := 1.;
      repeat
        sum := sum + add;
        n := n + 1;
        add := add × (x/n);              {calcolo di x^n/n!}
      until(|add| < |sum| × u)   {criterio di arresto naturale}
 end espo
```

Procedura 1.2: Algoritmo per il calcolo di $\mathbf{e^x}$

Le considerazioni precedenti si applicano in generale ad un processo iterativo basato su una formula ricorrente del tipo

$$S_{n+1} = S_n + a_n \,. \tag{1.18}$$

Il criterio di arresto naturale è

$$|a_n| < \epsilon_{S_n} \,,$$

dove

$$\epsilon_{S_n} = |S_n| \times u\,.$$

E' opportuno notare che anche il calcolo di ϵ_{S_n} è sottoposto alle limitazioni del sistema aritmetico f.p. a precisione finita.

♣ **Esempio 1.20.** Si consideri un sistema aritmetico f.p. con $F(10,7,-9,9)$ e si calcolino i valori di ϵ_{S_n} in relazione ai numeri $y = -0.1238452 \times 10^2$ e $z = 0.2356112 \times 10^{-8}$. Si ricordi che l'epsilon macchina ed il minimo numero reale positivo rappresentabile in F sono, rispettivamente, $u = 0.5 \times 10^{-6}$ e $rmin = 0.1 \times 10^{-9}$. Si ha:

$$\epsilon_y = (0.1238452 \times 10^2) \times (0.5 \times 10^{-6}) = 0.6192260 \times 10^{-5}\,,$$
$$\epsilon_z = (0.2356112 \times 10^{-8}) \times (0.5 \times 10^{-6}) = 0.1178056 \times 10^{-14} < rmin$$

e quindi il calcolo di $fl(\epsilon_z)$ provoca underflow. ♣

L'algoritmo descritto nella Procedura 1.3 calcola ϵ_x per un valore generico $x \in F$ in modo accurato.
Inoltre, l'algoritmo richiede che sia noto il valore di $rmin$, che può essere calcolato con l'algoritmo descritto nella Procedura 1.4. [32]

Si supponga di voler utilizzare la (1.16) per calcolare un'approssimazione di e^x con p cifre decimali corrette. Bisogna dunque arrestare il procedimento iterativo quando

$$E = |e^x - S_n| < 10^{-p} = tol\,.$$

Il valore di E non è noto e bisogna quindi calcolarne una stima.[33] A tal fine si ricordi che

$$e^x - S_n = R_n(x)\,,$$

e che al divergere di n il resto della formula di Taylor è infinitesimo

$$\lim_{x\to 0} \frac{R_n(x)}{x^n} = 0\,;$$

[32] Tale algoritmo si basa sull'ipotesi, generalmente verificata, che, in caso di underflow, ad r venga assegnato il valore 0.

[33] Una stima di E è fornita dalla (1.5) (cfr. esempio 1.6):

$$E = |e^x - S_n| = |R_n(x)| \le e^a \frac{|x|^{n+1}}{(n+1)!}\,,$$

con a opportunamente scelto. Come già osservato nell'esempio 1.6, il calcolo effettivo di tale stima richiede però la determinazione di a, o meglio di un valore di a per cui $e^a|x|^{n+1}/(n+1)!$ non sia "eccessivamente grande". Tale problema non è di facile risoluzione e quindi in generale non si ricorre alla stima suddetta.

```
procedure epsrel(in: x, rmin, u ; out: ε_x)
 /# SCOPO: calcolo dell'epsilon macchina relativo a x
 /# SPECIFICHE PARAMETRI:
 /# PARAMETRI DI INPUT:
   var: x    : reale  {numero macchina di cui calcolare}
                      {l'epsilon macchina relativo}
   var: rmin : reale  {minimo numero reale positivo}
                      {rappresentabile }
   var: u    : reale  {epsilon macchina}
 /# PARAMETRI DI OUTPUT:
   var: ε_x  : reale  {epsilon macchina relativo a x}
 /# INIZIO ISTRUZIONI:
      if |x| > rmin/u then
        ε_x := |x| × u;
      else
        ε_x := rmin;
      endif
 end epsrel
```

Procedura 1.3: Algoritmo per il calcolo di ϵ_x

concludendo, a meno del fattore costante e^a (o e^ξ con $\xi \in [-a, a]$), vale l'approssimazione:

$$E = |e^x - S_n| \simeq \frac{|x|^{n+1}}{(n+1)!} = |S_{n+1} - S_n| \,.$$

La quantità

$$|S_{n+1} - S_n|$$

si assume pertanto come stima dell'errore assoluto E al passo n e si utilizza il criterio d'arresto

$$|S_{n+1} - S_n| < tol \,. \tag{1.19}$$

```
procedure rmin(in: β ; out: rmim)
 /# SCOPO: calcolo del minimo numero reale positivo
           rappresentabile
 /# SPECIFICHE PARAMETRI:
 /# PARAMETRI DI INPUT:
   var: β      : intero   {base del sistema aritmetico}
 /# PARAMETRI DI OUTPUT:
   var: rmin   : reale    {minimo num. reale positivo rappr.}
 /# VARIABILI LOCALI:
   var: r      : reale    {variabile di appoggio}
 /# INIZIO ISTRUZIONI:
      r := 1.;
      repeat
         rmin := r;
         r := r/β;
      until (r = 0.)
end rmin
```

Procedura 1.4: Algoritmo per il calcolo di $rmin$

Per quanto detto sul criterio d'arresto naturale, non ha senso richiedere che

$$tol < \varepsilon_{S_n} = |S_n| \times u\,. \tag{1.20}$$

Analogamente, se si vuole calcolare un'approssimazione di e^x con p cifre significative corrette, si utilizza il criterio d'arresto

$$\frac{|S_{n+1} - S_n|}{|S_n|} < 10^{-p} = tol \tag{1.21}$$

e in questo caso non ha senso richiedere che

$$tol < u\,. \tag{1.22}$$

e^5	tol	E'
0.1483096×10^3	10^{-3}	0.70×10^{-3}
0.1484029×10^3	10^{-4}	0.69×10^{-4}
0.1484124×10^3	10^{-5}	0.51×10^{-5}
0.1484131×10^3	10^{-6}	0.40×10^{-6}
0.1484132×10^3	10^{-7}	0.26×10^{-6}
0.1484132×10^3	10^{-8}	0.26×10^{-6}
0.1484132×10^3	10^{-9}	0.26×10^{-6}
0.1484132×10^3	10^{-10}	0.26×10^{-6}
0.1484132×10^3	10^{-20}	0.26×10^{-6}
0.1484132×10^3	10^{-30}	0.26×10^{-6}

Tabella 1.4: Valori di e^5 ottenuti con un programma in Fortran, utilizzando una workstation IBM RISC System/6000 e diversi valori di tolleranza

In Tabella 1.4 sono riportati i valori di e^5, calcolati utilizzando la Procedura 1.2 con il criterio d'arresto (1.21) [34], per valori di tol differenti, e sono indicati gli errori relativi, E', corrispondenti. Si osservi che per $10^{-6} \leq tol \leq 10^{-3}$ l'errore relativo è dell'ordine di tol, mentre per $tol \leq 10^{-7}$ tale errore è sempre 0.26×10^{-6}, in accordo col valore di u riportato in (1.14).[35]

Pertanto, i criteri d'arresto (1.19) e (1.21) devono essere modificati, per evitare che sia richiesta un'accuratezza che non può essere ottenuta:

$$|S_{n+1} - S_n| < tol + |S_n| \times u\,,$$

$$\frac{|S_{n+1} - S_n|}{|S_n|} < tol + u\,.$$

Quanto detto sul criterio di arresto naturale per il calcolo di e^x si estende in generale al progetto dei criteri di arresto di algoritmi basati su processi iterativi, che approssimano un valore x come limite di una successione $\{x_n\}_{n \in \mathcal{N}}$. In tal caso, si assumono come stime dell'errore assoluto e dell'errore relativo rispettivamente le quantità

$$|x_{n+1} - x_n|$$

e

$$\frac{|x_{n+1} - x_n|}{|x_n|}$$

[34] Implementazione su un'IBM RISC System/6000, in Fortran, utilizzando la singola precisione.

[35] Gli errori relativi sono stati calcolati utilizzando un'approssimazione di e^5 con 9 cifre significative corrette ($e^5 \simeq 0.148413159 \times 10^3$).

e si utilizzano i criteri d'arresto[36]

$$|x_{n+1} - x_n| < tol + |x_n| \times u \tag{1.23}$$

e

$$\frac{|x_{+1} - x_n|}{|x_n|} < tol + u\,. \tag{1.24}$$

1.5 Il condizionamento di un problema matematico

Nella risoluzione computazionale di un problema, i dati sono affetti dall'errore di roundoff. In generale, essi sono affetti anche da errori sperimentali, cioè dovuti ai procedimenti di misurazione utilizzati per la loro rilevazione; tali errori, inevitabili, non saranno esplicitamente considerati, in quanto assimilabili agli errori di roundoff.
In base a quanto detto, ci si augura che, in un problema matematico, un errore relativo (assoluto) nei dati di un certo ordine di grandezza si traduca in un errore relativo (assoluto) dello stesso ordine nella soluzione.

Definizione 1.7. (Buon condizionamento di un problema)
Un problema si dice ben condizionato se l'errore relativo (assoluto) nella soluzione ha al più lo stesso ordine di grandezza dell'errore relativo (assoluto) nei dati.

In accordo con la definizione precedente, un problema per il quale l'errore relativo nella soluzione ha ordine di grandezza maggiore rispetto all'errore relativo nei dati si dice *mal condizionato.*

♣ **Esempio 1.21.** Il sistema di equazioni lineari:

$$\begin{cases} 2.1 \quad x \quad + \quad 3.5 \quad y \quad = \quad 8 \\ 4.19 \quad x \quad + \quad 7 \quad\;\; y \quad = \quad 15 \end{cases} \tag{1.25}$$

ammette come soluzione $(100, -57.714\ldots)$. La risoluzione di (1.25) è un problema i cui dati sono i coefficienti della matrice del sistema ed il vettore dei termini noti.

[36] Si osservi che i criteri d'arresto (1.23) e (1.24) garantiscono che il processo iterativo si arresti quando due approssimazioni successive sono uguali a meno di una tolleranza prefissata, ma non assicurano che l'errore sia minore di tale tolleranza. Per tale motivo essi generalmente si utilizzano in combinazione con altri criteri, che tengono conto, ad esempio, di caratteristiche particolari del processo iterativo in esame. Si osservi inoltre che se $x_n = 0$ i criteri suddetti risultano inadeguati (il criterio (1.24) è addirittura inutilizzabile!). Un modo per superare tale problema può essere quello di utilizzare un criterio d'arresto del tipo

$$\frac{|x_{n+1} - x_n|}{|x_n| + 1} < tol + u\,.$$

Si consideri ora il sistema:

$$\begin{cases} 2.1 & x & + & 3.5 & y & = & 8 \\ 4.192 & x & + & 7 & y & = & 15 \end{cases} \tag{1.26}$$

che ammette come soluzione $(125, -72.714\ldots)$. Il sistema (1.26) può essere visto come una perturbazione del sistema (1.25) (nel coefficiente di x nella seconda equazione è stato introdotto un errore relativo dell'ordine di 10^{-3}) e la risoluzione di (1.26) come la risoluzione di un problema perturbato rispetto al problema di riferimento (la risoluzione di (1.25)). La perturbazione introdotta, corrispondente, da un punto di vista geometrico, ad una modifica nell'inclinazione della seconda delle due rette rappresentate dalle equazioni, comporta un'amplificazione dell'errore notevole, dovuta al fatto che le due rette sono "quasi parallele" e quindi una lieve pertubazione del coefficiente angolare di una delle due provoca uno spostamento del punto di intersezione. Tale situazione è illustrata in Figura 1.5.

♣

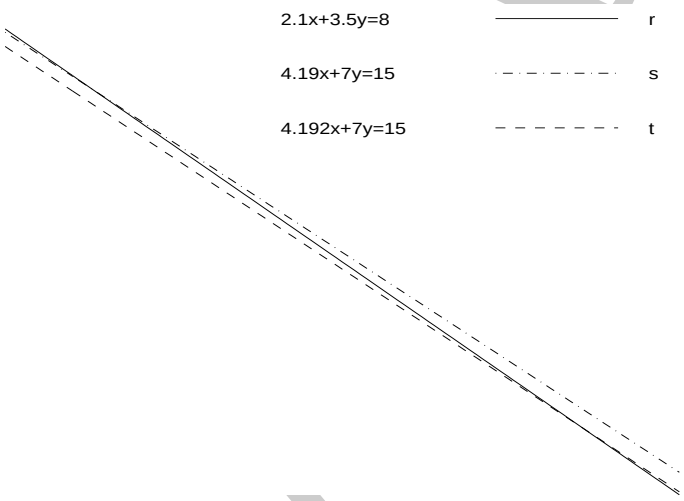


Figura 1.5: Interpretazione geometrica del condizionamento: mal condizionamento di un sistema lineare.

♣ **Esempio 1.22.** Consideriamo ora il sistema di equazioni lineari:

$$\begin{cases} -3.5 & x & + & 2.1 & y & = & 15 \\ 2.1 & x & + & 3.5 & y & = & 8 \end{cases} \tag{1.27}$$

che ammette come soluzione $(-0.6723, 2.6891)$, ed il sistema:

$$\begin{cases} -3.502 & x & + & 2.1 & y & = & 15 \\ 2.1 & x & + & 3.5 & y & = & 8 \end{cases} \tag{1.28}$$

che ammette come soluzione $(-0.6720, 2.6889)$. Come nell'esempio precedente, il sistema (1.28) può essere visto come una perturbazione del sistema (1.27) (anche in questo esempio, nel coefficiente di x nella seconda equazione è stato introdotto un errore relativo dell'ordine di 10^{-3}) e la risoluzione di (1.28) come la risoluzione di un problema perturbato rispetto al problema di riferimento (la risoluzione di (1.27)). La perturbazione introdotta non comporta questa volta un'amplificazione dell'errore,

dovuta al fatto che in questo caso le due rette sono "quasi ortogonali" [37] e quindi una lieve pertubazione del coefficiente angolare di una delle due non provoca uno spostamento significativo del punto di intersezione. Tale situazione è illustrata in Figura 1.6.

♣

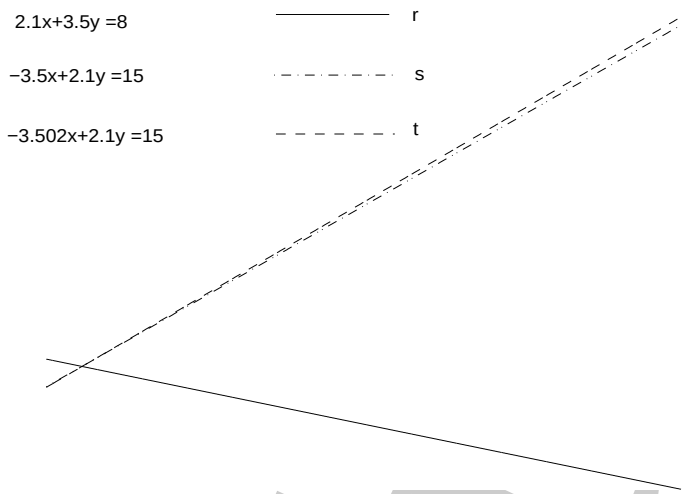


Figura 1.6: Interpretazione geometrica del condizionamento: ben condizionamento di un sistema lineare.

♣ **Esempio 1.23.** Si consideri l'equazione:

$$(x-2)^4 = 0\,, \tag{1.29}$$

che ha quattro radici coincidenti uguali a 2. Si consideri poi l'equazione:

$$(x-2)^4 = 10^{-8}\,, \tag{1.30}$$

che può essere vista come una perturbazione della precedente, in quanto è ottenuta sommando 10^{-8} al secondo membro di (1.29); è immediato osservare che (1.30) ha due soluzioni reali (1.99 e 2.01) e due soluzioni complesse ($2 - 0.01i$ e $2 + 0.01i$). La perturbazione introdotta ha quindi modificato il campo di appartenenza delle soluzioni ed ha introdotto nelle radici reali una perturbazione di un ordine di grandezza 4 volte superiore.

♣

♣ **Esempio 1.24.** Si consideri il problema del calcolo della differenza tra due numeri.
Siano $x = 12345678.0$ e $y = 12345677.0$ e sia z la differenza

$$z = x - y = 1.00000000\,;$$

[37] Per due rette ortogonali di coefficiente angolare, rispettivamente, m e m' sussiste la relazione: $m \cdot m' = -1$.

si consideri ora il problema, che si ottiene perturbando il precedente,

$$z' = x' - y' \,, \quad x' = 12345678.1, \;\; y' = 12345676.9 \,,$$

la cui soluzione è

$$z' = 1.20000000 \,.$$

Si osservi che una perturbazione nella nona cifra significativa dei dati ha provocato una perturbazione nella seconda cifra significativa della soluzione, dunque l'errore relativo nei dati

$$\frac{|x - x'|}{|x|} = 0.81 \times 10^{-8}, \qquad \frac{|y - y'|}{|y|} = 0.81 \times 10^{-8}$$

si è amplificato in

$$\frac{|z - z'|}{|z|} = 0.2.$$

Se si considera il numero delle cifre significative corrette degli addendi e del risultato, si ha che si passa da otto cifre, in x' e y', ad una cifra, in z', cioè sono "cancellate"[38] sette cifre significative. Tale fenomeno è detto, appunto, *cancellazione* ed una giustificazione di esso è data nell'esempio 1.27.

♣

Dagli esempi precedenti si deduce che nei problemi mal condizionati "piccoli errori" nei dati possono risultare amplificati nella soluzione, rendendola talvolta inaccettabile.

Detto δ l'errore nei dati e σ l'errore corrispondente nella soluzione, e posto

$$\sigma = \mu \cdot \delta,$$

μ è detto *indice di condizionamento* del problema; risulta, quindi, che:

se $\mu \leq 1$ il problema è *ben condizionato*;
se $\mu > 1$ il problema è *mal condizionato* [39].

Si ricordi che, essendo interessati alla risoluzione di problemi mediante calcolatore, *l'analisi del condizionamento deve essere fatta sempre*, perché i dati sono affetti almeno dall'errore di roundoff. Ad esempio, il problema (1.30) può essere visto come una versione di (1.29) perturbata dal solo errore di roundoff nel secondo membro.

È quindi importante riuscire a dare una stima quantitativa dell'indice di condizionamento.

[38] La cancellazione delle cifre significative è dovuta al fatto che nell'effettuare la sottrazione in corrispondenza delle cifre uguali (rispettivamente in x e in y) si ottiene una cifra nulla. A seguito della normalizzazione le cifre nulle vengono traslate verso sinistra e pertanto corrispondono a posizioni che vengono perse.

[39] La maggior parte dei problemi dell'Analisi Numerica risulta, più o meno, mal condizionata ed è sempre $\delta > 0$.

Si supponga di valutare una funzione f, reale di variabile reale, in un punto x. In generale, nell'approccio computazionale, f non sarà valutata in x, ma in $x+\Delta x$ a causa dell'errore di roundoff. L'errore Δx induce quindi un errore nel risultato. Si noti che la funzione f rappresenta il problema matematico, x i suoi dati (Δx è la loro perturbazione assoluta) e $f(x)$ la soluzione. Posto

$$\Delta f = f(x+\Delta x) - f(x)\,, \tag{1.31}$$

se f è derivabile in x, si ha:

$$\Delta f = f'(x)\Delta x + R(\Delta x)\,, \qquad \lim_{\Delta x \to 0} \frac{R(\Delta x)}{\Delta x} = 0$$

e quindi

$$|\Delta f| \simeq |f'(x)\Delta x|\,;$$

$|f'(x)|$ è, allora, il fattore di amplificazione nella soluzione dell'errore assoluto presente nei dati ed è, quindi, l'*indice di condizionamento assoluto*. In molti casi è più interessante considerare l'errore relativo, per il quale vale la relazione

$$\left|\frac{\Delta f}{f}\right| \simeq \left|\frac{f'(x)x}{f(x)}\right| \cdot \left|\frac{\Delta x}{x}\right|\,. \tag{1.32}$$

La quantità

$$C(f,x) = |f'(x)x|/|f(x)| \tag{1.33}$$

è il fattore di amplificazione nella soluzione dell'errore relativo nei dati ed è, quindi, l'*indice di condizionamento relativo*. $C(f,x)$ quantifica appunto la sensibilità del problema agli errori nei dati.

E' utile ricordare la seguente regola operativa facilmente deducibile dalla (1.32): se i dati del problema hanno p cifre significative corrette ($p \leq t$, con t precisione del sistema aritimetico f.p.), allora un indice di condizionamento relativo $C = 10^q$ indica che la soluzione non può essere ottenuta con più di $p-q$ cifre significative corrette. Si può concludere allora che un problema mal condizionato è completamente intrattabile se $q \simeq p$; altrimenti esso è trattabile a patto che si sia disposti a sopportare una perdita di q cifre significative nella soluzione.

♣ **Esempio 1.25.** Sia $f(x) = x^2$. Risulta

$$C(f,x) = 2 \;\; \forall x \in \Re\,.$$

Più in generale, $\forall\, n \in \mathcal{N}$, $C(x^n, x) = n$ e quindi il condizionamento del problema della valutazione della funzione x^n non dipende da x; inoltre il mal condizionamento della funzione cresce al crescere di n.

♣

♣ **Esempio 1.26.** Sia $f(x) = \sqrt{1-x}$, $x \leq 1$. In questo caso

$$C(f,x) = \frac{|x|}{2(1-x)};$$

per $x = 0.9999$, $C(f,x) = 4999.5$ e quindi il problema è mal condizionato per $x \simeq 1$. ♣

Per problemi più complessi della semplice valutazione di una funzione scalare di una sola variabile[40] si può cercare di valutare il fattore di amplificazione dell'errore generalizzando l'approccio qui presentato. Un caso semplice da analizzare è quello della sottrazione tra due numeri.

[40] Assegnata una funzione vettoriale ad m componenti, di n variabili:

$$\begin{aligned} \underline{f} &= (f_1, \ldots, f_m) \\ f_i &= f_i(\underline{x}) \qquad i = 1, \ldots, m \\ \underline{x} &= (x_1, \ldots, x_n) \end{aligned}$$

dallo sviluppo in serie di Taylor ed adottando, inoltre, per due vettori assegnati

$$\begin{aligned} w &= (w_1, \ldots, w_k) \\ v &= (v_1, \ldots, v_k) \end{aligned}$$

la notazione

$$\frac{w}{v} = \left(\frac{w_1}{v_1}, \ldots, \frac{w_k}{v_k}\right)$$

si ha

$$\frac{\Delta \underline{f}}{\underline{f}} \simeq \left(\frac{\delta f_i}{\delta x_j} \cdot \frac{x_j}{f_i}\right) \frac{\Delta \underline{x}}{\underline{x}} \tag{1.34}$$

dove con $\Delta\underline{x}$ e $\Delta\underline{f}$ si indicano, rispettivamente:

$$\begin{aligned} \Delta \underline{x} &= (\Delta x_1, \ldots, \Delta x_n) \\ \Delta \underline{f} &= (\Delta f_1, \ldots, \Delta f_m) \end{aligned}$$

da cui segue

$$\Delta f_i = f_i(\underline{x} + \Delta \underline{x}) - f_i(\underline{x}) \qquad i = 1, \ldots, m.$$

Passando alle norme infinito nella (1.34), si ottiene

$$\left\| \frac{\Delta \underline{f}}{\underline{f}} \right\|_\infty \leq \left\| \left(\frac{\delta f_i}{\delta x_j} \cdot \frac{x_j}{f_i}\right) \right\|_\infty \cdot \left\| \frac{\Delta \underline{x}}{\underline{x}} \right\|_\infty$$

La quantità

$$C(\underline{f}, \underline{x}) = \left\| \left(\frac{\delta f_i}{\delta x_j} \cdot \frac{x_j}{f_i}\right) \right\|_\infty$$

è detta *indice di condizionamento relativo* della funzione $\underline{f}$ nel punto $\underline{x}$.

♣ **Esempio 1.27.** Sia $f(x,y) = x - y$. Posto $\Delta f = f(x + \Delta x, y + \Delta y) - f(x,y)$, si ha:

$$\Delta f = x + \Delta x - y - \Delta y - x + y = \Delta x - \Delta y,$$

e quindi

$$\begin{aligned} \frac{|\Delta f|}{|f|} &\leq \frac{|\Delta x| + |\Delta y|}{|x - y|} = \frac{|x| + |y|}{|x - y|} \cdot \frac{|\Delta x| + |\Delta y|}{|x| + |y|} = \\ &= \frac{|x f_x| + |y f_y|}{|f(x,y)|} \cdot \frac{||(\Delta x, \Delta y)||}{||(x,y)||}, \end{aligned}$$

dove $||\cdot|| = ||\cdot||_1$ in $\Re^2$, $f_x = \frac{\partial f}{\partial x}$ e $f_y = \frac{\partial f}{\partial y}$. In questo caso l'indice di condizionamento relativo è

$$C(f,x,y) = (|x| + |y|)/|x - y|.$$

E' evidente che $C(f,x,y)$ cresce al diminuire della distanza tra x e y; dunque il calcolo della differenza tra due numeri è un problema mal condizionato quando i due numeri sono "vicini". Ciò spiega il fenomeno della cancellazione illustrato nell'esempio 1.24.

♣

♣ **Esempio 1.28.** Risulta, spesso, utile nelle applicazioni, determinare una stima dell'indice di condizionamento, per il problema

$$y = Mx,$$

in cui sono noti la matrice M ed il vettore x e si desidera determinare il vettore y, attraverso il loro prodotto. Se, dunque, si legge il problema come:

$$y = Mx = f(x),$$

in base a quanto descritto l'indice di condizionamento assoluto è dato da

$$|f'(x)| = \|M\|,$$

mentre l'indice di condizionamento relativo risulta

$$\left| \frac{f'(x)x}{f(x)} \right| = \frac{\|Mx\|}{\|Mx\|} = 1.$$

♣

1.6 La stabilità di un algoritmo

Anche se il problema da risolvere è ben condizionato, la bontà del risultato computazionale dipende dal "comportamento" dell'algoritmo utilizzato.

♣ **Esempio 1.29.**

$$\begin{cases} 0.0001 & x & + & y & = & 1 \\ & x & + & y & = & 2 \end{cases}$$

Si può dimostrare che questo problema è ben condizionato; la sua soluzione è il vettore (1.0001000..., 0.99989999...).

Sostituendo alla seconda equazione la prima equazione moltiplicata per 10^4 e sottratta dalla seconda, si ha il sistema equivalente

$$\begin{cases} 0.0001 & x & + & & y & = & 1 \\ & & & -9999 & y & = & -9998 \end{cases}$$

da cui

$$\begin{cases} y & = & \frac{-9998}{-9999} & & = & 1.0001000\ldots \\ x & = & \frac{1}{0.0001}(1 - \frac{9998}{9999}) & & = & 0.99989998\ldots \end{cases}$$

Eseguendo questo algoritmo in un sistema aritmetico f.p. con precisione $t = 3$ si ottiene:

$$\begin{cases} y & = & \frac{-0.100\times 10^5}{-0.100\times 10^5} & = & 0.100 \times 10^1 & = & 1 \\ x & = & \frac{0.100\times 10^1}{0.100\times 10^{-3}}(0.100 \times 10^1 - 0.100 \times 10^1) & & & = & 0 \end{cases}$$

cioè il vettore soluzione calcolato dall'algoritmo ha la seconda componente senza alcuna cifra significativa corretta rispetto alla soluzione esatta del problema.

Invertendo l'ordine delle equazioni e moltiplicando per 10^{-4} la prima equazione, si ottiene, nello stesso sistema f.p., la soluzione $x = 1$, $y = 1$, che è un'approssimazione accettabile della soluzione.

Questo fenomeno è dovuto al modo in cui si è propagato l'errore di roundoff durante l'esecuzione dell'algoritmo. Infatti, nel primo caso la divisione per il coefficiente 0.0001 ha prodotto come fattore moltiplicativo 10^4 amplificando l'errore di roundoff. Invertendo l'ordine delle equazioni, il fattore moltiplicativo diventa 10^{-4} che, al contrario del primo caso, riduce notevolmente la propagazione dell'errore.

♣

In generale, due algoritmi per la risoluzione di uno stesso problema possono produrre risultati differenti, in quanto gli errori complessivi di roundoff sono diversi. Il concetto di *stabilità* (e quindi di instabilità) si riferisce alla sensibilità di un algoritmo alla precisione finita, cioè agli errori di roundoff dei dati e delle operazioni f.p..

Definizione 1.8. (Instabilità di un algoritmo)
Un algoritmo si dice instabile, se gli errori di roundoff introdotti nei dati si propagano amplificandosi in maniera tale che i risultati siano inaccettabili.

In altri termini, un algoritmo è instabile se, a causa della propagazione dell'errore di roundoff, il risultato dell'algoritmo differisce sostanzialmente dalla soluzione del problema numerico.

Un altro esempio di algoritmo instabile è illustrato qui di seguito.

n	I_n
0	$+0.6321206 \times 10^{0}$
1	$+0.3678795 \times 10^{0}$
2	$+0.2642411 \times 10^{0}$
3	$+0.2072767 \times 10^{0}$
4	$+0.1708932 \times 10^{0}$
5	$+0.1455340 \times 10^{0}$
6	$+0.1267958 \times 10^{0}$
7	$+0.1124296 \times 10^{0}$
8	$+0.1005630 \times 10^{0}$
9	$+0.9493256 \times 10^{-1}$
10	$+0.5067444 \times 10^{-1}$
11	$+0.4425812 \times 10^{0}$
12	-0.4310974×10^{1}
13	$+0.5704266 \times 10^{2}$
14	-0.7975973×10^{3}

Tabella 1.5: Calcolo di un integrale per ricorrenza: risultati ottenuti implementando la formula (1.36) con $n = 14$ su una IBM RISC System/6000

♣ **Esempio 1.30.** Si consideri l'integrale

$$I_n = \int_0^1 x^n e^{x-1} dx\,, \quad n \geq 0\,.$$

Integrando per parti si ha:

$$\int_0^1 x^n e^{x-1} dx = \left[x^n e^{x-1}\right]_0^1 - n \int_0^1 x^{n-1} e^{x-1} dx = 1 - n \int_0^1 x^{n-1} e^{x-1} dx\,,$$

cioè:

$$I_n = 1 - n I_{n-1}\,, \quad n \geq 1\,. \tag{1.35}$$

Dato che

$$I_0 = \int_0^1 e^{x-1} dx \; = 1 - \frac{1}{e}\,,$$

si può calcolare I_n mediante un algoritmo basato sulla formula ricorrente

$$\begin{aligned} &I_0 = 1 - 1/e\,, \\ &I_n = 1 - n I_{n-1}, \quad n \geq 1\,. \end{aligned} \tag{1.36}$$

Si consideri, ad esempio, il calcolo di I_{14}. Implementando il procedimento in Fortran, su un'IBM RISC System/6000, utilizzando la singola precisione, si ottengono i risultati riportati in Tabella 1.5.

Il valore calcolato di I_{14} è completamente errato; infatti I_{14} deve essere positivo. Più in generale, i valori ottenuti per $n > 9$ sono completamente errati.[41] Perché si verifica questa situazione? Per dare una risposta al quesito occorre analizzare la propagazione dell'errore di roundoff dell'unico dato di input dell'algoritmo, I_0, supponendo, per semplicità, di effettuare le operazioni in aritmetica esatta. L'errore ϵ_0 introdotto in $1/e$, e quindi in I_0, è tale che:

$$\overline{I}_0 = I_0 + \epsilon_0\,, \quad |\epsilon_0| \simeq 10^{-6}\,,$$

dove $\overline{I}_0$ denota il valore calcolato di I_0. Quindi, per i termini successivi si ha:

$$\begin{aligned}
&\overline{I}_1 = 1 - 1 \times \overline{I}_0 = (1 - I_0) - \epsilon_0 = I_1 - \epsilon_0\,,\\
&\overline{I}_2 = 1 - 2 \times \overline{I}_1 = (1 - 2 \times I_1) + 2 \times \epsilon_0 = I_2 + 2 \times \epsilon_0\\
&\vdots\\
&\overline{I}_n = 1 - n \times \overline{I}_{n-1} = I_n + (-1)^n n! \times \epsilon_0\,.\\
&\vdots
\end{aligned}$$

Indicato con ϵ_n l'errore assoluto al passo n, per $n = 10$ risulta $\epsilon_n \simeq 1$ e, a partire da $n = 10$, ϵ_n cresce "enormemente", vanificando i calcoli successivi. In particolare, per $n = 14$ $\epsilon_n \simeq 14! \times 10^{-6} \simeq 87000$. L'algoritmo basato sulla formula (1.36) è dunque instabile e non può essere utilizzato.[42] Si deve quindi progettare un algoritmo diverso basato su una formula che non amplifichi l'errore di roundoff.

Si osservi che la relazione di ricorrenza (1.35) può essere riscritta nel modo seguente:

$$I_{n-1} = (1 - I_n)/n \qquad n \geq 1\,. \tag{1.37}$$

Dato che la successione $\{I_n\}$ è decrescente e tende a 0, si può porre $I_{20} = 0$ e calcolare poi $I_{19}, I_{18}, \ldots, I_{14}$ utilizzando la (1.37). L'implementazione di questo algoritmo sull'IBM RISC System/6000 calcola una buona approssimazione di I_{14} e, più in generale, fornisce risultati accettabili per $n \leq 17$ (Tabella 1.6).[43]

[41] Ciò è evidente per $n > 10$, in quanto, per ogni n, I_n deve essere positivo e tale che $I_{n+1} < I_n$. Nella seguente tabella sono riportati i valori corretti di I_n, per $n = 0, 1, 2, \ldots, 14$, arrotondati a 7 cifre sigificative:

n	I_n
0	$.6321206 \times 10^{0}$
1	$.3678794 \times 10^{0}$
2	$.2642411 \times 10^{0}$
3	$.2072766 \times 10^{0}$
4	$.1708934 \times 10^{0}$
5	$.1455329 \times 10^{0}$
6	$.1268024 \times 10^{0}$
7	$.1123835 \times 10^{0}$
8	$.1009320 \times 10^{0}$
9	$.9161229 \times 10^{-1}$
10	$.8387707 \times 10^{-1}$
11	$.7735223 \times 10^{-1}$
12	$.7177325 \times 10^{-1}$
13	$.6694770 \times 10^{-1}$
14	$.6273216 \times 10^{-1}$

[42] Il valore di n a partire dal quale i risultati non sono più attendibili dipende dal sistema f.p. utilizzato.

n	I_n
20	0.0000000
19	0.5000000×10^{-1}
18	0.5000000×10^{-1}
17	0.5277778×10^{-1}
16	0.5571895×10^{-1}
15	0.5901757×10^{-1}
14	0.6273216×10^{-1}

Tabella 1.6: Calcolo di un integrale per ricorrenza: risultati ottenuti implementando la formula (1.37) con $n = 14$ su una IBM RISC System/6000

L'algoritmo è infatti stabile; l'errore ϵ_{20} introdotto in I_{20} si propaga nel modo seguente:

$$
\begin{aligned}
&\overline{I}_{20} = I_{20} + \epsilon_{20}\,,\\
&\overline{I}_{19} = (1 - \overline{I}_{20})/20 = I_{19} - \epsilon_{20}/20\,,\\
&\overline{I}_{18} = (1 - \overline{I}_{19})/19 = I_{18} + \epsilon_{20}/(20 \times 19)\,,\\
&\vdots\\
&\overline{I}_{n-1} = (1 - \overline{I}_{n})/n = I_{n-1} + (-1)^{n-1}\epsilon_{20}/(20 \times 19 \times \cdots n)\,,\\
&\vdots
\end{aligned}
$$

L'errore iniziale ϵ_{20} al generico passo n è diviso per $20 \times 19 \times \cdots \times (n+1)$ e quindi non è amplificato, anzi è ridotto.

♣

Infatti, in un sistema aritmetico a precisione più elevata il valore di n aumenta; si osservi però che qualunque sia la precisione finita esiste sempre un valore di n per cui il risultato è inaccettabile.

[43] I valori corretti di I_n, per $n = 20, 19, \ldots, 14$, arrotondati a 7 cifre significative, sono riportati nella tabella seguente:

n	I_n
20	$.4554488 \times 10^{-1}$
19	$.4772276 \times 10^{-1}$
18	$.5011985 \times 10^{-1}$
17	$.5277112 \times 10^{-1}$
16	$.5571935 \times 10^{-1}$
15	$.5901754 \times 10^{-1}$
14	$.6273216 \times 10^{-1}$

Si noti che i valori calcolati con la (1.37) per $n = 20, 19$ sono inattendibili solo perché il valore iniziale I_{20} è arbitrario; a differenza dell'algoritmo precedente, si può calcolare con la massima accuratezza possibile il valore di qualunque I_n, a patto di considerare come dato iniziale $I_{\overline{n}} = 0$ con $\overline{n}$ sufficientemente maggiore di n.

n	$x^n/n!$	S_n
0	$+0.1000000 \times 10^{1}$	$+0.1000000 \times 10^{1}$
1	-0.1000000×10^{2}	-0.9000000×10^{1}
2	$+0.5000000 \times 10^{2}$	$+0.4100000 \times 10^{2}$
3	-0.1666667×10^{3}	-0.1256667×10^{3}
4	$+0.4166666 \times 10^{3}$	$+0.2910000 \times 10^{3}$
$\vdots$	$\vdots$	$\vdots$
32	$+0.3800392 \times 10^{-3}$	$+0.3651343 \times 10^{-4}$
33	$-0.1151634 \times 10^{-3}$	$-0.7864997 \times 10^{-4}$
34	$+0.3387159 \times 10^{-4}$	$-0.4477838 \times 10^{-4}$
35	$-0.9677598 \times 10^{-5}$	$-0.5445597 \times 10^{-4}$
$\vdots$	$\vdots$	$\vdots$
41	$-0.2989313 \times 10^{-8}$	$-0.5234286 \times 10^{-4}$
42	$+0.7117412 \times 10^{-9}$	$-0.5234215 \times 10^{-4}$
43	$-0.1655212 \times 10^{-9}$	$-0.5234231 \times 10^{-4}$
44	$+0.3761846 \times 10^{-10}$	$-0.5234228 \times 10^{-4}$

Tabella 1.7: Valori di $x^n/n!$ e di S_n ottenuti con un programma in Fortran, utilizzando una workstation IBM RISC System/6000

Un altro esempio di algoritmo instabile è l'algoritmo per il calcolo di un'approssimazione di e^x, con $x < 0$, basato sullo sviluppo in serie di Mac Laurin.

♣ **Esempio 1.31.** Si supponga $t = 7$ e $x = -10$ e si approssimi e^x con la somma

$$S_N(x) = \sum_{n=0}^{N} \frac{x^n}{n!} = \sum_{n=0}^{N} \frac{(-1)^n |x|^n}{n!} \,. \tag{1.38}$$

Un algoritmo basato sul calcolo diretto della (1.38) con il criterio d'arresto naturale (cfr. Procedura 1.2) fornisce il risultato $-0.5234228 \times 10^{-4}$, mentre il valore corretto di e^{-10}, arrotondato a 7 cifre significative, è 0.4539993×10^{-4}. L'errore relativo è dunque circa 0.2152915×10 ed il risultato dell'algoritmo non è sicuramente una buona approssimazione di e^{-10}. Il problema risiede nel fatto che i termini dello sviluppo in serie di e^{-10} hanno segni alterni e la sequenza di somme effettuate non è altro che una sequenza di sottrazioni tra numeri "abbastanza vicini" (Tabella 1.7).

La causa dell'instabilità di questo algoritmo sta nella progressiva perdita di cifre significative e quindi nel conseguente aumento dell'errore relativo; si verificano cioè successivamente vari fenomeni di cancellazione nelle sottrazioni. Si ricordi che la cancellazione è la manifestazione di una caratteristica intrinseca dell'operazione aritmetica di sottrazione, che è mal condizionata quando gli operandi sono "vicini" (cfr. esempi 1.24 e 1.27). Il fatto che si usi un sistema aritmetico a precisione finita implica solo che vi è certamente un errore (quello di roundoff) negli addendi e quindi il mal condizionamento della sottrazione si manifesta effettivamente. Sono proprio gli errori negli addendi le cause della cancellazione, mentre l'operazione sottrazione, nell'unità aritmetica, è completamente incolpevole.
Un algoritmo stabile per il calcolo di un'approssimazione di e^x per $x < 0$, utilizzando sempre lo sviluppo in serie di Mac Laurin, è basato sull'osservazione che $e^x = 1/e^{-x}$ ed il problema è trasformato in quello

del calcolo di e^x per $x > 0$. L'algoritmo descritto dalla Procedura 1.2 del §1.4, che è stabile se $x > 0$ (gli addendi sono tutti positivi e quindi non c'è fenomeno di cancellazione), è usato per il calcolo dell'esponenziale positivo ed il reciproco del risultato è il risultato del nuovo algoritmo stabile per e^x con $x < 0$.

♣

E' importante osservare che, a differenza del condizionamento, l'instabilità è una caratteristica dell'algoritmo e non del problema e quindi l'instabilità può essere evitata con opportune modifiche dell'algoritmo, come, appunto, nell'esempio 1.31.

1.7 L'Aritmetica Standard IEEE

Nel 1982 il *Floating - point Working Group dell'IEEE* [44] *Computer Society's Microprocessor Standard Committee* propose uno standard di **sistema aritmetico floating point binario**, per il progetto delle unità aritmetiche dei calcolatori[45].

L'aritmetica Standard IEEE descrive un modello di sistema aritmetico floating point (cioè definisce la rappresentazione dei dati numerici e le operazioni aritmetiche eseguite su tali dati) le cui caratterististiche fondamentali si possono sintetizzare nel modo seguente:

- massima accuratezza statica, per la rappresentazione dei numeri floating point;
- massima accuratezza dinamica, per le operazioni aritmetiche;
- gestione delle eccezioni;
- gestione del *gradual underflow*;
- quattro schemi per l'arrotondamento.

Nel seguito si descrive, brevemente, lo standard IEEE, riassumendo le proprietà della rappresentazione dei numeri floating point, la gestione delle situazioni eccezionali, il gradual underflow e gli schemi dell'arrotondamento.

1. **Rappresentazione dei numeri floating point**: l'aritmetica IEEE definisce tre formati:
   - singola precisione,
   - doppia precisione,
   - precisione estesa.

[44] **I**nstitute of **E**lectrical and **E**lectronic **E**ngineering

[45] L'aritmetica standard IEEE è anche nota come aritmetica *KCS* dalle iniziali di W. Kahan, J. Coonen, studente di W. Kahan presso l'Università della California di Berkeley, e del prof. H. Stone. W. Kahan ha ricevuto per questo motivo il premio **TURING** dall'ACM, nel 1989.

Mentre i primi due sono strettamente obbligatori, il terzo formato è facoltativo in quanto viene utilizzato al più nei registri all'interno dell'unità aritmetico-logica per la rappresentazione dei risultati intermedi in una successione di operazioni floating point.

Nella tabella seguente riportiamo le caratteristiche del formato esteso della singola e della doppia precisione:

Tipo	Lunghezza parola	precisione	esponente min	esponente max
singola	≥ 43	≥ 32	$\leq$ -1021	≥ 1024
doppia	≥ 79	≥ 64	≤ -16381	≥ 16384

La locazione di memoria per i formati in singola e doppia precisione è suddivisa nel modo seguente:

Tipo	Segno	Esponente	Mantissa
singola	$\pm$	8	23
doppia	$\pm$	11	52

Organizzazione della locazione di memoria per la rappresentazione dei numeri floating point nello standard IEEE

Tenendo conto dell'organizzazione della locazione di memoria, il numero di bit riservati per l'esponente è 8 per la singola precisione e 11 per la doppia; pertanto l'intervallo di rappresentabilità è delimitato dagli esponenti -127 e $+128$, in base 2, per la singola precisione e da -1023 a $+1024$ per la doppia precisione. Questo corrisponde, in base 10, ad una fascia esponenziale che varia da -38 a $+38$ per la singola precisione e da -308 a $+308$ per la doppia precisione. Inoltre, un solo bit viene riservato per il segno della mantissa. Per l'esponente si usa la notazione ad *eccesso 127* nella singola precisione e ad *eccesso 1023* nella doppia precisione[46]. Ciò significa che invece di rappresentare l'esponente e con $-127 \leq e \leq 128$, il valore che viene memorizzato è un intero senza segno $e^{'}$ con $0 \leq e^{'} \leq 255$, per la singola precisione. Per quanto riguarda la doppia precisione, invece di rappresentare l'esponente e con $-1023 \leq e \leq 1024$ viene memorizzato un numero positivo, $e^{'}$, con $0 \leq e^{'} \leq 2047$. Questo significa considerare anche il bit dedicato al segno come un bit per il campo dell'esponente e calcolare il numero decimale ottenuto dall'intera sequenza di bit, considerato come intero senza segno.

[46] Questo concetto è già stato introdotto nella nota (23) del paragrafo 1.2.2.

♣ **Esempio 1.32.** Considerati 3 bit, di cui il primo per il segno ed i restanti due per il numero stesso, si ottengono i seguenti numeri naturali:

segno + rapp.bin	rapp. dec.
0 00	-0
0 01	-1
0 10	-2
0 11	-3
1 00	+0
1 01	+1
1 10	+2
1 11	+3

Utilizzando invece la notazione senza segno, la stessa sequenza di bit rappresenta un diverso intervallo di numeri naturali (positivi):

segno + rapp.bin	rapp. dec. senza segno
0 00	0
0 01	1
0 10	2
0 11	3
1 00	4
1 01	5
1 10	6
1 11	7

In questo caso l'intervallo di rappresentabilità viene slittato da $[-3, +3]$ a $[0, 7]$.

♣

L'aritmetica IEEE si avvale della rappresentazione floating point normalizzata, in cui, per la rappresentazione della mantissa, si fa uso del **bit implicito**:[47] il bit iniziale ha sempre valore 1 e non è esplicitamente rappresentato. Questo comporta la possibilità di rappresentare effettivamente 24 bit per la singola precisione e 53 per la doppia precisione.

Relativamente all'esecuzione delle **operazioni aritmetiche floating point**, l'aritmetica IEEE utilizza, per la rappresentazione delle mantisse degli operandi, registri aritmetici con 2 *guard digits* più uno *sticky-bit* per garantire la massima accuratezza dinamica. I due bit aggiuntivi (i *guard digits*) sono utilizzati per rappresentare i primi due bit significativi della parte di mantissa (degli operandi e del

[47] Questo concetto è già stato introdotto nella nota (21) del paragrafo 1.2.2 nell'ambito della descrizione di un qualsiasi sistema aritmetico floating point.

risultato) che viene rimossa a causa dell'arrotondamento. Il terzo bit, detto *stichy bit*, è ottenuto come OR logico di tutti i bit rimanenti. Come anche già detto si può dimostrare che tre bit sono sufficienti a garantire la massima accuratezza dinamica.

2. **Gestione delle situazioni eccezionali**:
   Tali situazioni sono classificate in:

   - operazione non valida (INVALID),
   - divisione per zero,
   - overflow,
   - underflow.

   Ad ogni eccezione sono associati un *flag* ed un *trap*. Il *flag* indica quale eccezione si è verificata. Il *trap* attiva la scelta tra due possibilità: chiamata ad una procedura di gestione dell'eccezione con interruzione dell'esecuzione oppure nessuna interruzione dell'esecuzione.

   Un'operazione INVALID produce un **NaN** (Not a Number), la cui rappresentazione in memoria è la seguente:

Segno	Esponente	Mantissa
$\pm$	$1 \ldots 1$	qualsiasi combinazione di bit

Rappresentazione di NaN nell'aritmetica IEEE

   Un risultato NaN è ottento effetuando operazioni non definite come ad esempio $0 \times \infty$, $0/0$, ∞/∞, $+\infty - \infty$.

   La divisione per zero produce come risultato **Inf**, la cui rappresentazione in memoria è la seguente:

Segno	Esponente	Mantissa
$\pm$	$1 \ldots 1$	$0 \cdots\cdots 0$

Rappresentazione di Inf nell'aritmetica IEEE

   Il segno di **Inf** è l'usuale segno di un quoziente pertanto la divisione per zero è l'unica operazione algebrica che rivela il segno di zero.

   Lo standard IEEE introduce alcune funzioni di ambiente per la gestione delle eccezioni; tra queste particolarmente utili sono `FINITE(x)` che restituisce il valore *vero*

se l'argomento è rappresentabile, *falso* in caso contrario, e la funzione `ISNAN(x)` che ritorna il valore *vero* se l'argomento è un NaN, *falso* altrimenti.

Sia **NaN** che **Inf** possono essere utilizzati nell'esecuzione delle usuali operazioni aritmetiche. Tipicamente il risultato di un'operazione che coinvolge un **NaN** continua ad essere **NaN**, lo stesso vale per **Inf** con le dovute eccezioni quali, ad esempio, nella divisione. In tal caso se il numeratore è diverso da zero ed il denominatore è **Inf**, il quoziente è zero.

3. **Gradual underflow**:
   Lo standard IEEE introduce i cosiddetti *numeri denormalizzati*, aventi, come esponente, l'esponente minimo consentito ed una mantissa non normalizzata. Questo significa che dopo $rmin = 0.1 \times \beta^{emin}$ sono ancora rappresentabili i numeri del tipo:

$$\begin{array}{rrl} 0.01 & \times\beta^{emin} = & 0.1 \times \beta^{emin-1} \\ 0.001 & \times\beta^{emin} = & 0.1 \times \beta^{emin-2} \\ 0.0001 & \times\beta^{emin} = & 0.1 \times \beta^{emin-3} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots\ldots\ldots \\ 0.00\ldots1 & \times\beta^{emin} = & 0.1 \times \beta^{emin-t} \end{array}$$

   Come si può notare in questo caso la soglia di underflow viene ridotta: in particolare, per la singola precisione, essendo $t = 7$, essa risulta pari a $10^{-38-7} = 10^{-45}$, mentre per la doppia precisione ($t = 16$), si ha il valore $10^{-308-16} = 10^{-324}$.
   I numeri denormalizzati sono particolarmente utili nell'esecuzione di sommatorie in cui uno degli addendi può andare in underflow. In tal caso l'underflow graduale consente di non perdere completamente il contributo di un tale addendo, fornendo quindi un risultato più accurato.

4. **Schemi di arrotondamento**
   Nella rappresentazione dei numeri floating point, usualmente, si utilizza lo schema dell'arrotondamento. Esso è utilizzato per default nello standard IEEE, ed è indicato anche come *Rounded to the Nearest (RN)*[48]. Lo standard IEEE prevede la possibilità di utilizzare tre ulteriori **schemi di arrotondamento**, indicati, rispettivamente, come *Rounded towards Zero (RZ)*, *Rounded towards Plus* ∞ *(RP)* e *Rounded towards Minus* ∞ *(RM)*. Il primo dei tre, RZ, è l'usuale **troncamento**[49],

[48] *Rounded to Nearest (RN)* significa *Approssimazione al più vicino*, principio sul quale si basa la tecnica di arrotondamento per la rappresentazione di un qualsiasi numero reale (appartenente all'insieme di rappresentabilità) mediante il numero macchina ad esso più vicino.

[49] *Rounded towards Zero (RZ)* significa *Approssimazione verso lo zero*, principio sul quale si basa la tecnica del troncamento, in base alla quale un numero reale rappresentabile viene approssimato con il numero macchina più vicino che, in valore assoluto, risulta più piccolo.

mentre gli altri due scelgono come approssimazione floating point rispettivamente il numero più grande[50] ed il più piccolo[51].

♣ **Esempio 1.33.** Sia F=(10,6,-9,9) e $x = .123456789$, si ha

$$fl(x) = \begin{cases} .123457 & \text{per RN e RP} \\ .123456 & \text{per RZ e RM} \end{cases}$$

Se consideriamo $x = -.123456789$, si ha :

$$fl(x) = \begin{cases} -.123457 & \text{per RN e RM} \\ -.123456 & \text{per RZ e RP} \end{cases}$$

♣

Nella tabella seguente riportiamo le caratteristiche dei numeri floating point nello standard IEEE e la loro tipica rappresentazione:

Argomento	Singola precisione	Doppia precisione
Bit di segno	1	1
Bit per l'esponente	8	11
Bit per la mantissa	23	52
Bit totali	32	64
Rappr. esponente	Eccesso 127	Eccesso 1023
Fascia esponenziale	da -127 a +128	da -1023 a + 1024
Minimo normalizzato	-2^{-126}	-2^{-1022}
Max normalizzato	$(2-2^{-23})2^{127}$	$(2-2^{-52})2^{1023}$
Fascia decimale	da 10^{-38} a 10^{+38}	da 10^{-308} a 10^{+308}
Minimo denormalizzato	10^{-45}	10^{-324}

Lo standard IEEE

[50] *Rounded towards Plus* ∞ *(RP)* significa *Approssimazione verso più infinito*, e consiste nell'approssimare un numero reale rappresentabile con il più vicino numero macchina situato nella direzione positiva dell'asse reale.

[51] *Rounded towards Minus* ∞ *(RM)* significa *Approssimazione verso meno infinito*, e consiste nell'approssimare un numero reale rappresentabile con il più vicino numero macchina situato nella direzione negativa dell'asse reale.

1.8 Analisi della propagazione dell'errore di roundoff

L'analisi della propagazione dell'errore di roundoff, cioè lo studio dell'errore globale di roundoff inteso come somma (algebrica) di tutti gli errori di roundoff generati dall'algoritmo, riveste un ruolo fondamentale nella progettazione e sviluppo di un algoritmo. E' dunque importante avere a disposizione strumenti che consentano di valutare gli effetti di tale errore sulla soluzione calcolata dall'algoritmo (eseguito in aritmetica a precisione finita).
Nel seguito sono presentati due metodi di analisi della propagazione dell'errore di roundoff:

- la *forward error analysis*;
- la *backward error analysis*.

♣ **Esempio 1.34.** Sia assegnato un sistema aritmetico f. p.

$$F : (\beta = 10, t = 3, \underbrace{t_{reg} = 2 \cdot t}_{max.acc.din.} = 6),$$

in cui $u = 0.5\beta^{1-t} = 0.5 \times 10^{-2}$. Se $x = 3.453$ risulta

$$x^* = fl(x) = 0.345 \times 10^1 = 3.45,$$

Si desidera calcolare il quadrato di x. In aritmetica *esatta* $s = x^2 = 11.923209$. In F, invece,

$$s^* = x^* \otimes x^* = 0.345 \times 10 \otimes 0.345 \times 10 = 0.119 \times 10^2 = 11.9$$

L'errore relativo di roundoff su s^* è, dunque,

$$E'_{s^*} = \frac{|s - s^*|}{s} = 0.195 \times 10^{-2} \leq 3 \times \underbrace{(0.5 \times 10^{-2})}_{u} = 3u = 1.5 \times 10^{-2}\,.$$

Segue, allora, che

$$E'_{s^*} \simeq ku \quad \text{con} \quad k = 3$$

cioè la stima dell'errore relativo è dello stesso ordine di grandezza dell'errore di roundoff sul dato, per cui la soluzione si ritiene *accettabile*. ♣

Effettuiamo l'**analisi della propagazione dell'errore di roundoff**, ovvero studiamo in che modo gli errori introdotti dal sistema aritmetico a precisione finita si propagano sulla soluzione. Nel calcolo di

$$s^* = x^* \oplus y^*$$

siano:

1. δ_1,δ_2: errori di *rappresentazione* dei dati:

$$\begin{aligned} x^* &= fl(x) = x(1+\delta_1) \quad \text{con} \quad |\delta_1| \leq u \\ y^* &= fl(y) = y(1+\delta_2) \quad \text{con} \quad |\delta_2| \leq u \end{aligned}$$

2. ρ_1: errore sulle *operazioni floating-point* in F:

$$\begin{aligned} x^* &= fl(x) \in F \\ y^* &= fl(y) \in F \end{aligned}$$

per cui si ricava:

$$\begin{aligned} fl(x) \oplus fl(y) &= fl(fl(x) \# fl(y)) = (fl(x) \# fl(y))(1+\rho_1) \\ \text{con} \quad & |\rho_1| \leq u \end{aligned}$$

supponendo di essere in presenza di un sistema che realizza la massima accuratezza dinamica. In particolare, volendo eseguire il calcolo del quadrato di un dato x:

$$x^* = fl(x) = x(1+\delta_1)$$

eseguendo la moltiplicazione

$$s^* = x^* \otimes x^* = fl(x^* \times x^*) = fl(fl(x) \times fl(x))$$

si ottiene

$$\begin{aligned} fl(fl(x) \times fl(x)) &= (\underbrace{x(1+\delta_1)}_{err.\,di\,rappr.} \cdot \underbrace{x(1+\delta_1)}_{err.\,di\,rappr.}) \cdot \underbrace{(1+\rho_1)}_{err.\,sulla\,moltipl.} = \\ &= x^2(1+\delta_1)^2(1+\rho_1) \end{aligned}$$

con $|\delta_1|, |\rho_1| \leq u$, da cui:

$$s^* = x^* \otimes x^* = x^2(1+\delta_1)^2(1+\rho_1).$$

L'errore relativo risulta:

$$\begin{aligned} E'_{s^*} &= \frac{|s^* - s|}{|s|} = \frac{|x^2(1+\delta_1)^2(1+\rho_1) - x^2|}{x^2} = \\ &= |1 + \rho_1 + \delta_1^2 + \rho_1\delta_1^2 + 2\delta_1 + 2\delta_1\rho_1 - 1| \leq \\ &\leq 3u + 3u^2 + u^3 = 3u + O(u^2) \simeq \mathbf{3u} \end{aligned}$$

ovvero E'_{s^*} è dello stesso ordine di grandezza della massima accuratezza relativa; la *soluzione* può, dunque, ritenersi *accettabile*.

♣ **Esempio 1.35.** Assegnato un sistema aritmetico f.p.

$$F : (\beta = 10, t = 4, \underbrace{t_{reg} = 2 \cdot t}_{} = 8),$$

in cui $u = 0.5\beta^{1-t} = 0.5 \times 10^{-3}$, ed i dati:

$$x = 7.41330 \Rightarrow x^* = fl(x) = 0.7413 \times 10^1$$
$$y = 3.453749 \Rightarrow y^* = fl(y) = 0.3454 \times 10^1$$

calcolare la differenza

$$z^* = x^* - y^*$$

In aritmetica a precisione infinita $z = x - y = 3.959551$; nel sistema f.p. considerato:

$$\begin{aligned} z^* &= x^* - y^* = \\ &= 0.7413 \times 10^1 - 0.3454 \times 10^1 \\ &= 0.3959 \times 10^1 \end{aligned}$$

L'errore relativo di roundoff su z^* risulta:

$$\begin{aligned} E'_{z^*} &= \frac{|z^* - z|}{|z|} = \frac{|0.3959 \times 10^1 - 3.959551|}{3.959551} = \\ &= 0.1392 \times 10^{-3} \simeq 0.3 \times \underbrace{0.5 \times 10^{-3}}_{u} \end{aligned}$$

ovvero

$$E'_{z^*} \simeq ku \quad \text{con} \quad k = 0.3$$

La stima ottenuta è, dunque, dello stesso ordine di grandezza dell'errore di roundoff sul dato, per cui la *soluzione* può ritenersi *accettabile*. ♣

♣ **Esempio 1.36.** Assegnato un sistema aritmetico f.p.

$$F : (\beta = 10, t = 4, \underbrace{t_{reg} = 2 \cdot t}_{} = 8),$$

in cui $u = 0.5\beta^{1-t} = 0.5 \times 10^{-3}$, ed i dati:

$$\begin{aligned} x = 3.453749 &\Rightarrow x^* = fl(x) = 0.3454 \times 10^1 \\ y = 3.453432 &\Rightarrow y^* = fl(y) = 0.3453 \times 10^1 \end{aligned}$$

calcolare la differenza

$$z^* = x^* - y^*$$

In aritmetica a precisione infinita $z = x - y = 0.000317$; nel sistema f.p. considerato:

$$\begin{aligned} z^* &= x^* - y^* = \\ &= 0.3454 \times 10^1 - 0.3453 \times 10^1 \\ &= 0.1 \times 10^{-2} \end{aligned}$$

L'errore relativo di roundoff su z^* risulta:

$$\begin{aligned} E'_{z^*} &= \frac{|z^* - z|}{|z|} = \frac{|0.1 \times 10^{-2} - 0.000317|}{0.000317} = \\ &= 0.2155 \times 10^1 = 4.31 \times 10^3 \times \underbrace{0.5 \times 10^{-3}}_{u} \end{aligned}$$

ovvero

$$E'_{z^*} \simeq ku \quad \text{con} \quad k = 4310$$

La stima ottenuta è, dunque, molto più grande dell'errore di roundoff sul dato (è circa di tre ordini di grandezza più grande), per cui la *soluzione* può ritenersi *non accettabile*. ♣

Riguardando i due esempi precedenti si può dedurre che, nello stesso sistema aritmetico a precisione finita, l'algoritmo per il calcolo di $x - y$ produce un *risultato accettabile* in corrispondenza di

$$\begin{aligned} x &= 7.41330 \\ y &= 3.453749 \end{aligned}$$

ed un *risultato NON accettabile* per

$$\begin{aligned} x &= 3.453749 \\ y &= 3.453432 \end{aligned}$$

Ha senso, allora, chiedersi se l'algoritmo per il calcolo della differenza sia stabile o non lo sia. Effettuiamo, quindi, l'analisi della propagazione dell'errore di roundoff, nel calcolo della differenza tra due numeri, x e y.
Siano:

$$\begin{aligned} x^* = fl(x) = x(1 + \delta_1) \\ y^* = fl(y) = y(1 + \delta_2) \end{aligned}$$

si ha:

$$\begin{aligned} z^* &= x^* \ominus y^* = \\ &= fl(x^* - y^*) = \\ &= fl(fl(x) - fl(y)) = \\ &= (x(1 + \delta_1) - y(1 + \delta_2))(1 + \rho_1) \end{aligned}$$

con $|\delta_1|, |\delta_2|, |\rho_1| \leq u$. Dall'espressione dell'errore relativo si deduce in che modo, tali errori, si propagano sulla soluzione z^*:

$$\begin{aligned} E'_{z^*} &= \frac{|z^* - z|}{|z|} = \frac{|(x(1 + \delta_1) - y(1 + \delta_2))(1 + \rho_1) - (x - y)|}{|x - y|} = \\ &= \frac{|x(\delta_1 + \rho_1 + \delta_1\rho_1) - y(\delta_2 + \rho_1 + \delta_2\rho_1)|}{|x - y|} \leq \\ &\leq \frac{|x|(2u + u^2) + |y|(2u + u^2)}{|x - y|} = (2u + O(u^2))\frac{|x| + |y|}{|x - y|} \end{aligned}$$

ovvero

$$E'_{z^*} = (2u + O(u^2))\frac{|x| + |y|}{|x - y|}$$

Si osservi, dunque, che l'errore relativo nel calcolo della differenza tra due numeri, dipende da x e y. In altre parole, non si può concludere se la soluzione sia accettabile o no, indipendentemente dai valori di x e y. Posto

$$k = \frac{|x| + |y|}{|x - y|},$$

in corrispondenza di

$$\begin{aligned} x &= 7.41330 \\ y &= 3.453749 \end{aligned}$$

si ottiene $k = 2.7445$ e, quindi, il risultato della differenza $x - y$ può ritenersi accettabile. Al contrario, per

$$\begin{aligned} x &= 3.453749 \\ y &= 3.453432 \end{aligned}$$

si ha $k = 2.17 \times 10^4$, costante grande per cui il risultato della differenza può ritenersi NON accettabile.

L'analisi della propagazione dell'errore introdotto dalla precisione finita, *durante l'esecuzione delle operazioni richieste dall'algoritmo*, viene detta **forward error analysis (f.e.a.)** (analisi in avanti). Analizziamo, ora, quando un algoritmo si può dire stabile nel senso della forward error analysis.

Si indichino con $S(d)$ e $S_c(d)$, rispettivamente, la soluzione esatta e quella calcolata. La f.e.a. si propone di fornire una stima dell'errore relativo (assoluto):

$$E'_s = \frac{||S(d) - S_c(d)||}{||S(d)||} \quad (||S(d) - S_c(d)||), \tag{1.39}$$

dove $||\cdot||$ è una norma fissata, seguendo, passo dopo passo, l'introduzione di errori nel processo risolutivo, a partire dall'errore nei dati e la loro propagazione. In particolare, se

$$E'_s = \frac{||S(d) - S_c(d)||}{||S(d)||} \leq k \cdot u$$

qualora k sia una costante "sufficientemente piccola", l'algoritmo si dice stabile nel senso della f.e.a.; se, invece, k è una costante "grande", l'algoritmo si dice instabile nel senso della f.e.a.. Ad esempio, si osserva che *l'algoritmo per il calcolo del quadrato di un numero è stabile*, essendo

$$E'_{s^*} = 3u.$$

Nel calcolo della sottrazione, invece, applicando la f.e.a. si è ottenuto:

$$E'_{z^*} = (2u + O(u^2)) \underbrace{\frac{|x| + |y|}{|x - y|}}_{k}.$$

Lo stesso algoritmo ha prodotto un risultato accettabile nel primo caso, essendo

$$E'_{z^*} \simeq ku \quad \text{con} \quad k = 0.3$$

ovvero l'algoritmo è stabile nel senso della f.e.a., mentre, nel secondo caso, il risultato NON è accettabile essendo

$$E'_{z^*} \simeq ku \quad \text{con} \quad k = 4310$$

ovvero l'algoritmo è instabile nel senso della f.e.a. . La f.e.a. fornisce, quindi, in relazione al calcolo della differenza, un fattore di amplificazione dell'errore che dipende sia dall'algoritmo che dai dati del problema.
Nasce, quindi, traendo spunto da questo caso, la necessità di isolare l'amplificazione dell'errore introdotto dalle operazioni dell'algoritmo, indipendentemente dai dati del problema. A tale scopo un'idea è quella di ricondurre tutti gli errori introdotti dalle operazioni floating point a precisione finita, ad errori sui dati, assumendo, quindi, che le operazioni dell'algoritmo siano, di fatto, eseguite in aritmetica a precisione *infinita*. In tal modo, infatti, la perturbazione, oramai ricondotta sul dato iniziale, sarà quella indotta dall'algoritmo ed una sua (eventuale) amplificazione sulla soluzione sarà quella indotta dal problema, ovvero dal (mal) condizionamento del problema. L'analisi dell'errore basata sull'idea di considerare la soluzione, calcolata dall'algoritmo in un sistema aritmetico a precisione finita, come soluzione esatta (ovvero ottenuta in aritmetica a precisione infinita) di un problema dello stesso tipo, con dati perturbati, viene detta **backward error analysis (b.e.a.)** (analisi all'indietro).

♣ **Esempio 1.37.** Assegnato un sistema aritmetico f.p.

$$F : (\beta = 10, t = 4, \underbrace{t_{reg} = 2 \cdot t}_{max.acc.din.} = 8),$$

in cui $u = 0.5\beta^{1-t} = 0.5 \times 10^{-3}$, ed i dati:

$$\begin{aligned} x = 3.453749 \quad &\Rightarrow \quad x^* = fl(x) = 0.3454 \times 10^1 \\ y = 3.453432 \quad &\Rightarrow \quad y^* = fl(y) = 0.3453 \times 10^1 \end{aligned}$$

calcolare la differenza

$$x^* \ominus y^*$$

In aritmetica esatta $z = x - y = 0.000317$; nel sistema f.p. considerato:

$$\begin{aligned} z^* &= x^* \ominus y^* = \\ &= 0.1 \times 10^{-2} \end{aligned}$$

Volendo analizzare gli errori introdotti nel calcolo di $z^* = x^* \ominus y^*$, bisogna ricordare che l'errore sul dato x risulta:

$$\begin{aligned}\delta_1 &= \left|\frac{x - fl(x)}{x}\right| = \\ &= \left|\frac{0.3453749 \times 10^1 - 0.3454 \times 10^1}{0.3453749 \times 10^1}\right| = \\ &= 0.7 \times 10^{-4}\end{aligned}$$

da cui, in particolare,

$$fl(x) = x(1 + \delta_1) \Leftrightarrow 0.3454 \times 10^1 = 3.453749(1 + 0.7 \times 10^{-4})$$

Analogamente l'errore sul dato y risulta:

$$\begin{aligned}\delta_2 &= \left|\frac{y - fl(y)}{y}\right| = \\ &= \left|\frac{0.3453432 \times 10^1 - 0.3453 \times 10^1}{0.3453432 \times 10^1}\right| = \\ &= 0.1 \times 10^{-3}\end{aligned}$$

da cui, in particolare,

$$fl(y) = y(1 + \delta_2) \Leftrightarrow 0.3453 \times 10^1 = 3.453432(1 + 0.1 \times 10^{-3})$$

Poiché

$$\begin{aligned}z^* &= fl(x) \ominus fl(y) = \\ &= 0.3454 \times 10^1 \ominus 0.3453 \times 10^1 \\ &= 0.1 \times 10^{-2}\end{aligned}$$

si ha

$$\begin{aligned}\rho_1 &= \frac{|z^* - z|}{|z|} = \left|\frac{0.1 \times 10^{-2} - 0.000317}{0.000317}\right| = \\ &= 0.2 \times 10^1\end{aligned}$$

e, quindi,

$$z^* = z(1 + \rho_1) \Leftrightarrow 0.1 \times 10^{-2} = 0.000317(1 + 0.2 \times 10^1)$$

Quindi:

$$\begin{aligned} fl(x) = x(1 + \delta_1) &\Rightarrow 0.3454 \times 10 = (0.3453749 \times 10)(1 + 0.7 \times 10^{-4}) \\ fl(y) = y(1 + \delta_2) &\Rightarrow 0.3453 \times 10 = (0.3453432 \times 10)(1 + 0.1 \times 10^{-3}) \\ z^* = fl(fl(x) - fl(y)) = z(1 + \rho_1) &\Rightarrow 0.1 \times 10^{-2} = 0.000317(1 + 0.2 \times 10) \end{aligned} \quad (1.40)$$

Effettuando le operazioni in aritmetica a precisione infinita, dalle (1.40) segue:

$$\begin{aligned}z^* &= (0.3453749 \times 10)(1 + 0.7 \times 10^{-4}) \ominus (0.3453432 \times 10)(1 + 0.1 \times 10^{-3}) = \\ &= [(0.3453749 \times 10)(1 + 0.7 \times 10^{-4}) - (0.3453432 \times 10)(1 + 0.1 \times 10^{-3})] \times (1 + 0.2 \times 10) \\ &\simeq \underbrace{0.000317(1 + 0.1 \times 10^{-3})(1 + 0.7 \times 10^{-4})}_{(x-y)(1+\delta_1)(1+\delta_2)} \underbrace{(1 + 0.2 \times 10)}_{(1+\rho_1)}\end{aligned}$$

da cui,

$$\begin{array}{rcl} z^* & = & x^* \ominus y^* = (x-y)(1+\delta_1)(1+\delta_2)(1+\rho_1), \\ & \text{con} & |\delta_1| \leq u, |\delta_2| \leq u, |\rho_1| \leq u \end{array}$$

Ovvero:

$$z^* = \underbrace{[x(1+\delta_1)(1+\delta_2)(1+\rho_1)]}_{\underline{x}} - \underbrace{[y(1+\delta_1)(1+\delta_2)(1+\rho_1)]}_{\underline{y}}$$

In questo modo gli errori introdotti dalla sottrazione eseguita a precisione finita sono stati ricondotti ad errori su x e su y. Per stimare quanto è grande la perturbazione ricondotta sui dati iniziali x e y calcoliamo la distanza relativa tra (x,y) e $(\underline{x},\underline{y})$.

$$\begin{array}{rcl} E'_{(\underline{x},\underline{y})} & = & \dfrac{\|(\underline{x},\underline{y})-(x,y)\|}{\|(x,y)\|} = \\ & = & \dfrac{|x(\delta_1+\rho_1+\delta_1\rho_1)-y(\delta_2+\rho_1+\delta_2\rho_1)|}{|x|+|y|} \leq \\ & \leq & \dfrac{|x(2u+u^2)|+|y(2u+u^2)|}{|x|+|y|} = 2u+u^2 = 2u+O(u^2) \simeq 2u \end{array} \qquad (1.41)$$

Essendo, dunque,

$$E'_{(\underline{x},\underline{y})} \simeq k'u \quad \text{con} \quad k'=2$$

la stima dell'errore relativo ottenuta è dello stesso ordine di grandezza della massima accuratezza relativa e, quindi, $(\underline{x},\underline{y})$ è un'approssimazione di (x,y) che dista da (x,y) di una quantità dello stesso ordine di grandezza della massima accuratezza relativa. Sostanzialmente $(\underline{x},\underline{y})$ è un'approssimazione di (x,y) *accettabile*. In questo caso, la perturbazione introdotta dalle operazioni eseguite dall'algoritmo e poi ricondotta sul dato iniziale, è dello stesso ordine di grandezza dell'errore di roundoff sul dato iniziale per cui **l'algoritmo della sottrazione è stabile nel senso della b.e.a.**. ♣

In generale, considerato un algoritmo, che risolve un problema di dati iniziali d, se si indicano con d^* e d, rispettivamente il dato perturbato ed il dato iniziale, la b.e.a. si propone di fornire una stima dell'errore relativo (assoluto):

$$\frac{\|d-d^*\|}{\|d\|} \quad (\|d-d^*\|),$$

dove $\|\cdot\|$ è una norma fissata. In dettaglio, se

$$E'_s = \frac{\|d-d^*\|}{\|d\|} \leq k \cdot u$$

qualora k sia una costante "sufficientemente piccola", l'algoritmo si dice stabile nel senso della b.e.a.; se, invece, k è una costante "grande", l'algoritmo si dice instabile nel senso della b.e.a..
Nell'esempio relativo al calcolo della sottrazione:

$$E'_{z^*} \simeq 2u \underbrace{\frac{|x|+|y|}{|x-y|}}_{k}$$

Ricordiamo che, dall'analisi del condizionamento del problema della sottrazione, l'indice di condizionamento della sottrazione è

$$k = \frac{|x| + |y|}{|x - y|},$$

per cui, in corrispondenza dei dati

$$\begin{aligned} x &= 7.41330 \\ y &= 3.453749 \end{aligned}$$

si deduce $k = 2.7445$ ed il problema risulta *ben condizionato*; al contrario, per

$$\begin{aligned} x &= 3.453749 \\ y &= 3.453432 \end{aligned}$$

si ha $k = 2.17 \times 10^4$, costante "grande" per cui il problema è *mal condizionato*. Quindi, per l'algoritmo della sottrazione, dalla **f.e.a.** si deduce:

$$E'_s \simeq 2\frac{|x| + |y|}{|x - y|}u$$

ovvero *l'errore si approssima con una costante che dipende dall'algoritmo e dal condizionamento del problema*; dalla **b.e.a.**:

$$E'_d \simeq 2u$$

ovvero *l'errore si approssima con una costante che dipende solo dall'algoritmo.* In conclusione l'algoritmo della sottrazione è stabile nel senso della b.e.a. ma può fornire un risultato inaccettabile se i numeri sono vicini, perché è mal condizionato.
In generale, la f.e.a. fornisce:

$$E'_s \simeq ku$$

ovvero *l'errore si approssima con una costante k che dipende dall'algoritmo e dal condizionamento del problema*; per la **b.e.a.**, invece,

$$E'_d \simeq k'u$$

ovvero *l'errore si approssima con una costante k' che dipende solo dall'algoritmo.*
Siano

$$\begin{aligned} d &= \text{dati iniziali} \\ S(d) &= \text{soluzione esatta} \\ \tilde{d} &= \text{dati perturbati} \\ \tilde{S}(d) &= \text{soluzione calcolata} \end{aligned}$$

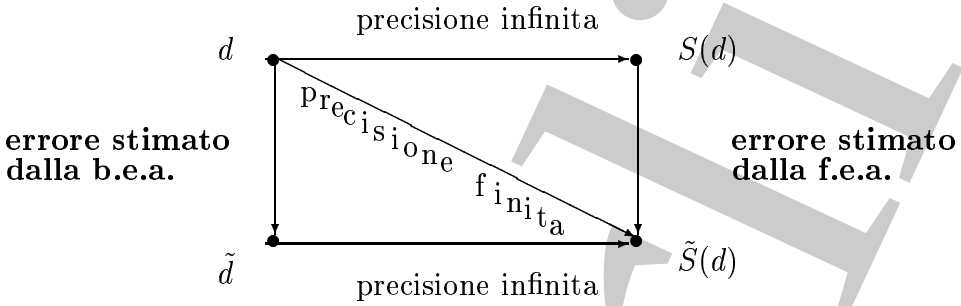


Figura 1.7: Errori stimati con la forward error analysis e con la backward error analysis.

Vale, dunque, lo schema seguente:

$$\begin{array}{rcl} E'_{\tilde{S}(d)} = E'_{S(\tilde{d})} & & \Rightarrow \text{ Backward error analysis:} \\ \Downarrow & & \text{analisi dell' algoritmo} \\ E'_{S(\tilde{d})} = \mu E'_{\tilde{d}} & & \Rightarrow \text{ Stima del condizionamento:} \\ \Downarrow & & \text{analisi del problema} \\ E'_{\tilde{d}} = ku & & \Rightarrow \text{ risultato della b.e.a.} \\ \hline \Downarrow & & \\ E'_{\tilde{S}(d)} = \mu k u & & \end{array}$$

con

μ dipendente dal problema (indice di cond. del probl.)
k dipendente dall' algoritmo (fattore di amplif. dell'errore)
u dipendente dalla precisione del sistema (max acc. rel.)

Se il problema è ben condizionato ($\mu \leq 1$) l'errore sui dati non viene amplificato ed *un algoritmo stabile nel senso della b.e.a. ($k < 1$) fornisce un risultato accettabile*; se il problema è mal condizionato ($\mu > 1$) l'errore sui dati viene amplificato ed *anche un algoritmo stabile nel senso della b.e.a. ($k < 1$) fornisce un risultato inaccettabile.*

Una rappresentazione grafica degli errori stimati dalla f.e.a e dalla b.e.a e, più in generale, delle relazioni che intercorrono tra i due metodi di analisi dell'errore, è fornita in Figura 1.7.

♣ **Esempio 1.38.** Sia assegnato un sistema aritmetico f. p.

$$F : (\beta = 10, t = 3, \underbrace{t_{reg} = 2 \cdot t}_{max.acc.din.} = 6),$$

in cui $u = 0.5\beta^{1-t} = 0.5 \times 10^{-2}$. Se $x = 3.453$ risulta

$$x^* = fl(x) = 0.345 \times 10^1 = 3.45.$$

Calcolare il quadrato di x. In aritmetica *esatta* $x^2 = 3.453 \times 3.453 = 11.923209$. In F, invece,

$$s^* = x^* \otimes x^* = 0.345 \times 10 \otimes 0.345 \times 10 = 0.119 \times 10^2 = 11.9$$

Per studiare gli errori introdotti nel calcolo di x^2 in F, si ricorda che, l'errore sul dato è il seguente:

$$\begin{aligned} \delta_1 &= \left|\frac{x - fl(x)}{x}\right| = \\ &= \left|\frac{0.3453 \times 10^1 - 0.345 \times 10^1}{0.3453 \times 10^1}\right| = \\ &= 0.8 \times 10^{-4} \end{aligned}$$

da cui,

$$fl(x) = x(1+\delta_1) \Leftrightarrow 0.345 \times 10 = 3.453(1 + 0.8 \times 10^{-4}) \tag{1.42}$$

mentre, dall'essere

$$s^* = fl(fl(x) \times fl(x)) = 0.345 \times 10 \otimes 0.345 \times 10 = 0.119 \times 10^2 \tag{1.43}$$

l'errore sulla operazione risulta:

$$\begin{aligned} \delta_2 &= \left|\frac{s - s^*}{s}\right| = \\ &= \frac{0.11923209 \times 10^2 - 0.119 \times 10^2}{0.11923209 \times 10^2} = 0.2 \times 10^{-2} \end{aligned}$$

da cui:

$$s^* = s(1+\delta_2) \Leftrightarrow 0.119 \times 10^2 = (0.11923209 \times 10^2)(1 + 0.2 \times 10^{-2}) \tag{1.44}$$

Dalla (1.43), tenendo presente le (1.42) e (1.44), eseguendo i calcoli in precisione infinita, si ottiene:

$$\begin{aligned} &= 3.453(1 + 0.8 \times 10^{-4}) \otimes 3.453(1 + 0.8 \times 10^{-4}) = \\ &= [3.453(1 + 0.8 \times 10^{-4}) \times 3.453(1 + 0.8 \times 10^{-4})] \times (1 + 0.2 \times 10^{-2}) = \\ &= \underbrace{11.923209(1 + 0.8 \times 10^{-4})^2}_{x^2(1+\delta_1)^2} \underbrace{(1 + 0.2 \times 10^{-2})}_{(1+\delta_2)} \end{aligned}$$

Sostanzialmente risulta:

$$s^* = x^* \otimes x^* = x^2(1+\delta_1)^2(1+\delta_2) \quad \text{con} \quad |\delta_1| \leq u, |\delta_2| \leq u$$

Più in dettaglio:

$$\begin{aligned} s^* &= 11.923209(1 + 0.8 \times 10^{-4})^2(1 + 0.2 \times 10^{-2}) = \\ &= (11.923209)^{\frac{1}{2}}(1 + 0.8 \times 10^{-4})(1 + 0.2 \times 10^{-2})^{\frac{1}{2}} \times \\ &\times (11.923209)^{\frac{1}{2}}(1 + 0.8 \times 10^{-4})(1 + 0.2 \times 10^{-2})^{\frac{1}{2}} \end{aligned}$$

e quest'ultimo è il risultato della moltiplicazione eseguita in aritmetica a precisione infinita:

$$s^* = \underbrace{[x(1+\delta_1)\sqrt{1+\delta_2}]}_{\underline{x}} \times \underbrace{[x(1+\delta_1)\sqrt{1+\delta_2}]}_{\underline{x}}$$

Con il procedimento descritto, gli errori introdotti dalla moltiplicazione eseguita a precisione finita sono ricondotti ad errori su x. Per eseguire una stima della perturbazione ricondotta sul dato iniziale x si può calcolare la distanza relativa tra x e $\underline{x}$, ovvero l'errore relativo su x:

$$\begin{aligned} E'_{\underline{x}} &= \frac{|\underline{x} - x|}{|x|} = \frac{|x(1+\delta_1)\sqrt{1+\delta_2} - x|}{|x|} \\ &\leq |\sqrt{1+\delta_2} - 1| + |\delta_1\sqrt{1+\delta_2}| \leq \\ &\leq 2u + u^2 = 2u + O(u^2) \simeq 2u \end{aligned}$$

da cui si deduce che l'errore relativo in $\underline{x}$ è maggiorato, a meno di $O(u^2)$, da $2u$; sostanzialmente $\underline{x}$ è un'approssimazione di x che dista da x di una quantità dello stesso ordine di grandezza della massima accuratezza relativa, cioè:

$$E'_{\underline{x}} \simeq k'u \quad \text{con} \quad k' = 2$$

ovvero $\underline{x}$ *è un'approssimazione di* x *accettabile.* In questo caso la perturbazione introdotta dalle operazioni eseguite dall'algoritmo, e poi ricondotta sul dato iniziale, è dello stesso ordine di grandezza dell'errore di roundoff sul dato iniziale, per cui **l'algoritmo per il calcolo del quadrato è stabile nel senso della b.e.a.**. ♣

♣ **Esempio 1.39.** Siano assegnati x, y, $z \in F$, dove F è un opportuno sistema aritmetico f.p., ovvero siano

$$\begin{aligned} \overline{x} &= fl(x) = x(1+\delta_1) \\ \overline{y} &= fl(y) = y(1+\delta_2) \\ \overline{z} &= fl(z) = z(1+\delta_3) \end{aligned}$$

con $\delta_i = 0, \quad i = 1, 2, 3$. Calcolare, in F,

$$\tilde{a} = x \oplus y \otimes z$$

Nel sistema aritmetico F, si devono eseguire, dunque, una *moltiplicazione f.p.* ed un'*addizione f.p.*. Posto

$$\tilde{w} = yz(1+\rho_1),$$

il calcolo di $\tilde{a}$ si può ricondurre all'operazione f.p.

$$\tilde{a} = x \oplus \tilde{w} = (x + \tilde{w})(1+\rho_2)$$

e, quindi,

$$\tilde{a} = (x + (yz(1+\rho_1)))(1+\rho_2)$$

Applicando la b.e.a. si ha

$$\begin{aligned} \tilde{a} &= (x + (yz(1+\rho_1)))(1+\rho_2) = (x + yz + yz\rho_1)(1+\rho_2) = \\ &= x + yz + yz\rho_1 + x\rho_2 + yz\rho_2 + yz\rho_1\rho_2 = \\ &= x + x\rho_2 + yz + yz\rho_1 + yz\rho_2 + yz\rho_1\rho_2 = \\ &= x(1+\rho_2) + y(1+\rho_2)z(1+\rho_1) \end{aligned} \tag{1.45}$$

Allora $\tilde{a}$ si può considerare come il risultato *esatto* delle operazioni eseguite tra

$$\overline{x} = x(1+\rho_2), \qquad \overline{y} = y(1+\rho_2) \qquad \text{e} \qquad \overline{z} = z(1+\rho_1).$$

♣

1.9 Esempio di studio: risoluzione di un'equazione di secondo grado

Si consideri la generica equazione di secondo grado:

$$ax^2 + bx + c = 0\,, \tag{1.46}$$

dove, per semplicità, si suppongono a, b e c reali. E' noto che, se $a \neq 0$, le radici di (1.46) hanno la seguente espressione:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}\,, \tag{1.47}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}\,. \tag{1.48}$$

E' naturale scrivere un algoritmo risolutivo che applichi direttamente (1.47) e (1.48):

$$\begin{array}{l}
\vdots \\
q := -b \\
y := b^2 \\
w := 4 \times a \\
v := w \times c \\
p := y - v \\
d := \sqrt{p} \\
s_1 := q + d \\
s_2 := q - d \\
r := 2 \times a \\
\overline{x}_1 := s_1/r \\
\overline{x}_2 := s_2/r \\
\vdots
\end{array}$$

Procedura 1.5: Algoritmo per la risoluzione di un'equazione di secondo grado

Fissato il sistema aritmetico f.p. con $F(10, 8, 50, -50)$, i quattro esempi seguenti mostrano l'applicazione dell'algoritmo 5 alla risoluzione di alcune equazioni di secondo grado.

1. $6x^2 + 5x - 4 = 0$.
   Le radici sono:

$$x_1 = 0.5\,, \qquad x_2 = -1.3333333\ldots\,,$$

quelle ottenute come risultato dell'algoritmo 5 sono:

$$\overline{x}_1 = 0.50000000\,, \qquad \overline{x}_2 = -1.3333333\,;$$

in questo caso l'algoritmo 5 fornisce la massima accuratezza del risultato.

2. $x^2 - 10^5 x + 1 = 0$.
L'equazione ha le seguenti radici, rappresentate con undici cifre significative:

$$x_1 = 99999.999990\,, \qquad x_2 = 0.000010000000001\,.$$

L'esecuzione dell'algoritmo 5 fornisce:

$$\begin{aligned}
&q = -b = 0.10000000 \times 10^6 \\
&y = b^2 = 0.10000000 \times 10^{11} \\
&w = 4 \times a = 0.40000000 \times 10^1 \\
&v = w \times c = 0.40000000 \times 10^1 \\
&\boxed{p = y - v = 0.10000000 \times 10^{11} - 0.40000000 \times 10^1 = \\
&\qquad = 0.10000000 \times 10^{11}} \\
&d = \sqrt{p} = 0.10000000 \times 10^6 \\
&s_1 = q + d = 0.20000000 \times 10^6; \qquad s_2 = q - d = 0 \\
&r = 2 \times a = 0.20000000 \times 10^1 \\
&\boxed{\overline{x}_1 = s_1/r = 0.10000000 \times 10^6}\,; \qquad \boxed{\overline{x}_2 = s_2/r = 0}
\end{aligned}$$

Gli errori relativi delle soluzioni calcolate dall'algoritmo sono:

$$E'_1 = \frac{|x_1 - \overline{x}_1|}{|x_1|} \simeq 10^{-7}, \qquad E'_2 = \frac{|x_2 - \overline{x}_2|}{|x_2|} = 1\,.$$

Dunque l'algoritmo fornisce un'ottima approssimazione di x_1 (massima accuratezza), ma fornisce un'approssimazione di x_2 affetta da un errore relativo del 100%. Ciò è dovuto al fatto che nel calcolo di p, in aritmetica a precisione finita, si perde completamente il contributo di v e, quindi, il calcolo di s_2 diventa una sottrazione tra due numeri uguali (fenomeno di cancellazione).

3. $x^2 - 4x + 3.9999999 = 0$.
Le radici, rappresentate con otto cifre significative, sono:

$$x_1 = 2.0003162\,, \qquad x_2 = -1.9996838\,.$$

Applicando l'algoritmo 5 si ha:

$$\begin{aligned}
&q = -b = 0.40000000 \times 10^1 \\
&y = b^2 = 0.16000000 \times 10^2 \\
&w = 4 \times a = 0.40000000 \times 10^1
\end{aligned}$$

$$\boxed{v = w \times c = (0.40000000 \times 10^1) \times (0.39999999 \times 10^1) = \\ = 0.16000000 \times 10^2}$$

$$p = y - v = 0$$
$$d = \sqrt{p} = 0$$
$$s_1 = q + d = s_2 = q - d = 0.40000000 \times 10^1$$
$$r = 2 \times a = 0.20000000 \times 10^1$$
$$\boxed{\overline{x}_1 = s_1/r = \overline{x}_2 = s_2/r = 0.20000000 \times 10^1}$$

L'algoritmo calcola una radice doppia, invece di due radici semplici, e tale radice ha 4 cifre decimali in comune con x_1, cioè solo metà della precisione di F. In questo caso la difficoltà risiede nel fatto che per rappresentare la mantissa di v si hanno a disposizione solo 8 cifre.

4. $10^{40}x^2 - 5 \times 10^{40}x + 6 \times 10^{40} = 0$.
Le radici esatte dell'equazione sono:

$$x_1 = 3\,, \qquad x_2 = 2\,;$$

applicando l'algoritmo 5 si ha:

$$q = -b = 0.50000000 \times 10^{41}$$
$$\boxed{y = b^2 = 0.25000000 \times 10^{82} > \beta^{emax} = 10^{50}}$$
$$\vdots$$

cioè si verifica un overflow. Anche se i cofficienti e le radici sono rappresentabili in F, l'algoritmo produce la situazione eccezionale di overflow, non fornendo alcuna informazione sulla soluzione. In generale, dato che le formule (1.47) e (1.48) richiedono il calcolo di b^2 e $4ac$, si può verificare overflow o underflow anche quando i coefficienti e le radici sono, in modulo, "molto distanti" da β^{emax} e β^{emin} rispettivamente.

Tranne che nel caso 1, l'applicazione diretta delle formule (1.47) e (1.48) non fornisce risultati soddisfacenti.

I principali requisiti di un "buon" algoritmo, per il calcolo delle radici di un polinomio di secondo grado $ax^2 + bx + c$, sono i seguenti:

I. se $a = b = c = 0$,
l'algoritmo deve segnalare che l'equazione è soddisfatta da ogni numero complesso x;

II. se $a = b = 0$ e $c \neq 0$,
deve segnalare che l'equazione non ammette soluzioni;

III. se $a = 0$ e $b \neq 0$,
deve segnalare che l'equazione è di primo grado e calcolarne la soluzione con la formula $x = -c/b$;

IV. se $a \neq 0$,
deve calcolare le radici con un'accuratezza adeguata (evitando, quindi, cancellazioni e situazioni di overflow ed underflow nelle operazioni), se queste sono rappresentabili nel sistema aritmetico f.p. utilizzato e deve segnalare il verificarsi di situazioni eccezionali in caso contrario[52].

Lo sviluppo di un algoritmo che abbia tutti i requisiti I-IV non è affatto semplice. Tralasciando il requisito dell'accuratezza in IV, è possibile tuttavia utilizzare "semplici" accorgimenti per superare alcune difficoltà come quelle incontrate nei precedenti esempi 2-4.

Le formule (1.47) e (1.48) possono essere riscritte, razionalizzando i numeratori, nel modo seguente:

$$x_1 = \frac{2c}{-b - \sqrt{b^2 - 4ac}}, \tag{1.49}$$

$$x_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}}. \tag{1.50}$$

Se nell'esempio 2 si utilizza (1.50) per calcolare x_2 si ottiene:

$$x_2^* = \frac{0.20000000 \times 10^1}{0.10000000 \times 10^6 + 0.10000000 \times 10^6} = 0.10000000 \times 10^{-4},$$

cioè si calcola un valore di x_2 con tutte le 8 cifre della mantissa corrette. In generale, supposto $b \neq 0$ e $b^2 - 4ac > 0$, se $b < 0$ le formule (1.50) e (1.47) evitano la sottrazione $-b - \sqrt{b^2 - 4ac}$, che può essere causa di un fenomeno di cancellazione; analogo discorso vale per le formule (1.48) e (1.49) nel caso $b > 0$.

Un altro modo per evitare un eventuale fenomeno di cancellazione consiste nel calcolare una delle radici con (1.47) o (1.48), a seconda che sia $b < 0$ o $b > 0$, e nel calcolare l'altra mediante la relazione

$$x_1 x_2 = c/a\ . \tag{1.51}$$

Utilizzando (1.51) per calcolare x_2, nell'esempio 2, si ha lo stesso valore ottenuto con la formula (1.50):

$$x_2^* = \frac{0.10000000 \times 10^1}{(0.10000000 \times 10^6) \times (0.10000000 \times 10^1)} = 0.10000000 \times 10^{-4}\ .$$

[52] Una precisa specificazione di tale requisito è discussa in [5], dove si richiede ad esempio che, utilizzando un sistema aritmetico f.p. con $F(\beta, t, emin, emax)$, l'errore sulle ultime 2 cifre di ciascuna radice calcolata x^* non sia più grande di $\beta + 1$ unità, cioè

$$\frac{|x - x^*|}{|x|} \leq 2u\frac{\beta + 1}{\beta}.$$

La difficoltà che si presenta nell'esempio 3 può essere superata eseguendo in doppia precisione le operazioni relative al calcolo di v:

$$q = -b = 0.40000000 \times 10^1$$

$$\boxed{\begin{array}{l} y = b^2 = 0.1600000000000000 \times 10^2 \\ w = 4 \times a = 0.4000000000000000 \times 10^1 \\ v = w \times c = 0.1599999960000000 \times 10^2 \\ p = y - v = 0.4000000000000000 \times 10^{-6} \end{array}}$$

$$d = \sqrt{p} = 0.6325 \times 10^{-3}$$

$$s_1 = q + d = 0.40006325 \times 10^1; \qquad s_2 = q - d = 0.39993675 \times 10^1$$

$$r = 2 \times a = 0.20000000 \times 10^1$$

$$\boxed{\overline{x}_1 = s_1/r = 0.20003163 \times 10^1}; \qquad \boxed{\overline{x}_2 = s_2/r = 0.19996838 \times 10^1}$$

Tale tecnica permette di calcolare due soluzioni distinte, commettendo al più un errore sull'ultima cifra significativa.

E' da osservare che le difficoltà relative all'esempio 3 sono legate, più che all'algoritmo 5, all'equazione, ovvero al suo condizionamento. Secondo un approccio di tipo b.e.a., le radici coincidenti $\overline{x}_1 = \overline{x}_2 = 2$ si possono vedere come soluzioni esatte dell'equazione

$$0.999999992x^2 - 3.999999968x + 3.999999968 = 0\,,$$

i cui coefficienti differiscono dai coefficienti omologhi dell'altra equazione per meno di 1 unità sull'ottava cifra significativa; quindi, a piccole variazioni nei dati corrispondono sensibili variazioni nei risultati.

Per finire, si consideri nuovamente l'esempio 4. Moltiplicando i coefficienti dell'equazione per 10^{-40} si ha:

$$x^2 - 5x + 6 = 0\,,$$

le cui soluzioni si possono calcolare esattamente con l'algoritmo 5. In questo caso si dice che è stato effettuato uno *scaling dei coefficienti*. Per evitare gli errori di roundoff connessi con la moltiplicazione f.p., si scelgono fattori di scaling che sono potenze della base del sistema aritmetico utilizzato (10^{-40} nell'esempio). Si noti, tuttavia, che non esiste una tecnica di scaling applicabile in generale.[53]

[53] Lo scaling dei coefficienti non risolve sempre i problemi di overflow od underflow. Si consideri ad esempio l'equazione

$$10^{-30}x^2 - 10^{30}x + 10^{30} = 0\,,$$

le cui radici x_1 e x_2 sono "prossime" a 10^{60} e ad 1. Un buon algoritmo risolutivo deve essere in grado di calcolare un'approssimazione di x_2 conformemente alle richieste del punto IV; d'altra parte ogni tentativo di scalare i coefficienti dell'equazione moltiplicandoli per uno stesso fattore non migliora la situazione, anzi può provocare un overflow od un underflow. Questa equazione è infatti un difficile test per un algoritmo risolutivo.

Altre tecniche sono utilizzabili per evitare l'overflow e l'underflow. Ad esempio, se il calcolo di b^2 provoca overflow, mentre a, b, c e $4ac$ sono rappresentabili, se $b \neq 0$, si può calcolare $\sqrt{b^2 - 4ac}$ secondo la formula:

$$\sqrt{b^2 - 4ac} = |b|\sqrt{1 - a\frac{2}{b}\frac{2}{b}c}\,;$$

se, invece, $b \neq 0$, a, b, c sono rappresentabili e $4ac$ produce underflow, si può calcolare una delle radici con la formula

$$x_i = -b/a \tag{1.52}$$

e l'altra sfruttando (1.51).[54]

A conclusione di questa discussione, si può, quindi, evidenziare come, anche la risoluzione di un problema matematico semplice, quale è la risoluzione di un'equazione di secondo grado, possa risultare estremamente complesso utilizzando un sistema aritmetico f.p. a precisione finita.

In generale, in casi come questo, si può cercare di effettuare uno *scaling dei coefficienti e dell'incognita*, cioè di trasformare (1.46) in un'equazione

$$a'y^2 + b'y + c' = 0\,,$$

con

$$a' = \beta^h a\,, \quad b' = \beta^h b\,, \quad c' = \beta^h c\,, \quad x = \beta^k y\,,$$

che sia di più facile risoluzione [13].

[54] In tal modo si trascura $4ac$ in (1.47) se $b < 0$ o in (1.48) se $b > 0$. D'altra parte, nel sistema aritmetico di un elaboratore, un valore che provoca underflow viene posto, generalmente, uguale a 0 e, quindi, la (1.52) si può vedere come conseguenza immediata dell'applicazione della corrispondente formula risolutiva.

1.10 Esercizi sull'aritmetica Floating-Point

Nel corso dell'intero paragrafo dedicato agli esercizi, si indicherà con $\Im$ un sistema aritmetico *floating-point* a precisione finita

$$\Im = (\beta, t, emin, emax)$$

con β=base, t= precisione, $emin$ = esponente minimo, $emax$ = esponente massimo; i valori numerici dei parametri che lo caratterizzano saranno specificati, laddove occorre.

1.10.1 Alcuni esercizi numerici

Esercizio 1 Utilizzando il troncamento rappresentare:

- In $\Im$: $\beta = 10$, t=5, $emin = -50$, $emax = +50$

  a) 2.718218285
  b) -1073741824
  c) 0.577216
  d) -123×10^{-45}

  Risposta

  a) 0.27182×10^{1}
  b) -0.10737×10^{10}
  c) 0.57722×10^{0}
  d) -0.123×10^{-42}

- In $\Im$: $\beta = 10$, t=4, $emin = -9$, $emax = +9$

  a) 989273
  b) 0.0000000000001
  c) -0.34×10^{5}
  d) -23×10^{-2}

  Risposta

  a) 0.9893×10^{6}
  b) **Underflow**
  c) -0.3400×10^{5}
  d) -0.2300×10^{0}

- In $\Im$: $\beta = 10$, t=3, $emin = -2$, $emax = 2$, si rappresentino i seguenti numeri e si calcoli *il minimo numero rappresentabile* e *la massima accuratezza relativa*:

$$x_1 = 222.13$$
$$x_2 = 0.2$$
$$x_3 = 0.056$$
$$x_4 = 3.467864$$

Risposta

$\tilde{x}_1 \rightarrow$**Overflow**		Non Rappresentabile in $\Im$
$\tilde{x}_2 = 0.200$	$\times 10^0$	Esattamente rappresentabile in $\Im$
$\tilde{x}_3 = 0.560$	$\times 10^{-1}$	Esattamente rappresentabile in $\Im$
$\tilde{x}_4 = 0.346$	$\times 10^1$	Non esattamente rappresentabile in $\Im$

Il minimo numero rappresentabile in $\Im$ (troncamento) è:

$$\beta^{emin-1} = 10^{-2-1} = 10^{-3}$$

La massima accuratezza relativa è:

$$\beta^{1-t} = 10^{1-3} = 10^{-2}$$

Esercizio 2 Sia $\Im$:

$$\Im : \beta = 10, t = 3, emin = -9, emax = 9$$

si calcolino:

(a) il massimo numero reale positivo rappresentabile;

(b) il minimo numero reale positivo rappresentabile;

(c) l'ϵ_{mac} in $\Im$;

(d) si esegua l'addizione seguente e si calcoli l'errore relativo introdotto nel risultato:

$$57.46 + 1.8888$$

Risposta

(a) il massimo numero rappresentabile è:

$$rmax = \beta^{emax}(1 - \beta^{-t}) = 0.999 \times 10^9$$

(b) il minimo numero rappresentabile è:

$$rmin = \beta^{emin-1} = 0.1 \times 10^{-10}$$

(c) l'ϵ_{mac} è:

$$\beta^{1-t} = 10^{1-3} = 10^{-2}$$

(d) Eseguiamo l'addizione una volta rappresentati in $\Im$ i numeri:

$$\begin{array}{ll} x_1 = 57.46 & \tilde{x}_1 = 0.574 \times 10^2 \\ x_2 = 1.8888 & \tilde{x}_2 = 0.188 \times 10^1 \end{array}$$

$$\tilde{x}_1 + \tilde{x}_2 = 0.574 \times 10^2 + 0.018 \times 10^2 = 0.592 \times 10^2$$

L'errore relativo è:

$$E_{rel} = \frac{|0.593488 \times 10^2 - 0.592 \times 10^2|}{0.593488 \times 10^2} \simeq 0.0025072 \simeq 0.250 \times 10^{-2}$$

Esercizio 3 Sia

$$\Im : \quad \beta = 10, t = 3, emin = -3, emax = 3.$$

Si rappresentino in $\Im$ i numeri seguenti e si indichi l'errore relativo che si commette nella loro rappresentazione.

a) $x_1 = 12.13$
b) $x_2 = 1.2$
c) $x_3 = 0.005678$
d) $x_4 = 3467.864$

Risposta

a)	$\tilde{x}_1 = 0.121$	$\times 10^3$	Non rappresentabile esattamente in $\Im$
b)	$\tilde{x}_2 = 0.120$	$\times 10^2$	Rappresentabile esattamente in $\Im$
c)	$\tilde{x}_3 = 0.567$	$\times 10^{-2}$	Non rappresentabile esattamente in $\Im$
d)			*Overflow*

Per quanto riguarda gli errori relativi si ha:

$$E_{1rel} = \left|\frac{0.1213\times10^2 - 0.121\times10^2}{0.1213\times10^2}\right| = 0.00003639 = 0.363 \times 10^{-4}$$

$$E_{2rel} = \left|\frac{0.12\times10^2 - 0.12\times10^2}{0.12\times10^2}\right| = 0$$

$$E_{3rel} = \left|\frac{0.5678\times10^{-2} - 0.567\times10^{-2}}{0.5678\times10^{-2}}\right| = 0.00045424 = 0.454 \times 10^{-3}$$

Esercizio 4 Siano I ed $\Im$ rispettivamente un sistema aritmetico intero ed uno floating-point a precisione finita, di parametri rispettivamente:

$$I: \quad \beta = 10 \quad t = 3 \quad imin = -999 \quad imax = 999$$

$$\Im: \quad \beta = 10 \quad t = 3 \quad emin = -9 \quad emax = 9$$

Si eseguano le seguenti operazioni segnalando eventuali "situazioni eccezionali".

1.) **In I**:

(a) $\frac{2}{4}$;

(b) -$\frac{96}{55}$;

(c) 50 × (-600).

2.) **In $\Im$**:

(a) $\frac{2}{4}$;

(b) 0.0001 × 0.000000986;

(c) 2.568+355.66 .

Risposta

1.) In I si ha:

(a) $\frac{2}{4}$=0;

(b) -$\frac{96}{55}$=-1;

(c) 50 × (-600)= -30000 (Overflow).

2.) In $\Im$ si ha:

(a) $\frac{2}{4}$=0.5 $\times 10^0$;

(b) $(0.1 \times 10^{-3}) \times (0.986 \times 10^{-6})$ =(Underflow) = 0

(c) $x_1 = 2.568 \quad \tilde{x}_1 = 0.256 \times 10^1$

$x_2 = 355.66 \quad \tilde{x}_2 = 0.355 \times 10^3$

$$\tilde{x}_1 + \tilde{x}_2 = 0.355 \times 10^3 + 0.00256 \times 10^2 = 0.357 \times 10^2$$

Esercizio 5 Sia:

$$\Im : \quad \beta = 10, \quad t = 3, \quad emin = -9, \quad emax = 9.$$

Rappresentare x, y, z e w dove:

$$x = \tfrac{1}{2}, \; y = -78666666/0.0078949, \; z = -0.0001 \times 0.000000789$$

$$w = 4.567 + 255.89$$

Risposta

$$\begin{aligned}
&\tilde{x} = 0.5 \times 10^0 \\
&\tilde{y} = 0.997 \times 10^{10} \;\; (Overflow) \\
&z = -0.789 \times 10^{-10} \;\; (Underflow) \\
&\tilde{w} = 0.457 \times 10^1 + 0.256 \times 10^3 = \\
&= 0.00457 \times 10^3 + 0.256 \times 10^3 = \\
&= 0.260 \times 10^3
\end{aligned}$$

Esercizio 6 Siano $\Im_1$ e $\Im_2$ due sistemi aritmetici floating-point a precisione finita di parametri:

$$\begin{aligned}
&\Im_1 : \quad \beta_1 = 2 \quad t_1 = 23 \\
&\Im_2 : \quad \beta_2 = 10 \quad t_2 = ?
\end{aligned}$$

Determinare t_2 in maniera che i due sistemi abbiano la stessa massima accuratezza relativa.

Risposta

Sia u_1 la massima accuratezza relativa di $\Im_1$ ed u_2 la massima accuratezza relativa di $\Im_2$. Si ha:

$$u_1 = \frac{1}{2} \times \beta_1^{1-t_1} \Rightarrow \quad u_1 = \frac{1}{2} \times 2^{1-23} = 2^{-23}$$

Poiché:

$$u_1 = u_2 = \frac{1}{2} \times 10^{1-t_2} \Rightarrow 10^{1-t_2} = 2u_1 = 2^{-22}$$

da cui,

$$t_2 = 1 - \log_{10} 2u_1 \Rightarrow t_2 = 1 - \log_{10} 2^{-22} = 1 + 22 \log_{10} 2$$

quindi

$$t_2 = 1 + 22 \cdot \frac{\log_{10} 10}{\log_{10} 2} \simeq 1 + 6 = 7$$

Esercizio 7 Sia:

$$\Im : \quad \beta = 10, \quad t = 2, \quad emax = 3, \quad emin = -3.$$

quanti numeri macchina contiene $\Im$?

Risposta

I numeri macchina presenti in $\Im$ sono:

$$2(\beta - 1)\beta^{t-1}(emax - emin + 1) + 1$$

e dunque:

$$2 \times 9 \times 10 \times 7 + 1 = 1261$$

Esercizio 8 Sia:

$$\Im : \beta = 10, t = 4, emax = 9, emin = -9.$$

dotato della massima accuratezza dinamica. Rappresentare i numeri:

$$x = 764.65 \qquad y = 0.0000000098701$$

e determinare gli epsilon macchina eps_x e eps_y relativi a ciascuno di essi (supponendo $t_{reg} > t$).

Risposta

$$\widetilde{x} = 0.7646 \times 10^3; \quad \widetilde{y} = 0.9870 \times 10^{-8}$$
$$\varepsilon_{mac} = \frac{1}{2}\beta^{1-t} = 0.5 \times 10^{1-4} = 0.5 \times 10^{-3}$$
$$eps_x = 0.7646 \times 10^3 \times 0.5 \times 10^{-3} = 0.3823 \times 10^0$$
$$eps_y = 0.987 \times 10^{-8} \times 0.5 \times 10^{-3} = UFL\ (Underflow)$$

Esercizio 9 Sia:

$$\Im : \beta = 10, t = 2, emax = 9, emin = -9.$$

determinare quanti sono i numeri macchina in $\Im$ e rappresentare il numero $x = 2.12$ in tale sistema aritmetico.

Risposta

I numeri macchina in $\Im$ sono:

$$2(\beta - 1)\beta^{t-1}(emax - emin + 1) + 1$$

ovvero

$$2 \times 9 \times 10 \times 18 + 1 = 3241$$

inoltre dato $x = 2.12$, la sua rappresentazione in $\Im$ è:

$$\tilde{x} = 0.21 \times 10^1$$

dunque x non è rappresentabile esattamente in $\Im$ e

$$E_{rel} = \frac{|0.212 \times 10^1 - 0.21 \times 10^1|}{0.21 \times 10^1} = 0.009433 \approx 0.943 \times 10^{-2}$$

Esercizio 10 Sia:

$$\Im : \beta = 10, t = 6, emax = 20, emin = -20.$$

eseguire in $\Im$ le seguenti operazioni:

$$\begin{array}{ll} a) & 1 + 10^7 \\ b) & 1 + 10^3 \\ c) & 1 + 10^{-7} \end{array}$$

Risposta

Eseguendo le operazioni, si ottiene:

a)$0.1 \times 10^1 + 0.1 \times 10^8 = 0.00000001 \times 10^8 + 0.1 \times 10^8 = 0.1 \times 10^8$
b)$0.1 \times 10^4 + 0.1 \times 10^1 = 0.1000 \times 10^4 + 0.0001 \times 10^4 = 0.1001 \times 10^4$
c)$0.1 \times 10^1 + 0.1 \times 10^{-6} = 0.1 \times 10^1 + 0.00000001 \times 10^1 = 0.1 \times 10^1$

Esercizio 11 Sia:

$$\Im : \quad \beta = 10, \quad t = 4, \quad emax = 9, \quad emin = -9.$$

si supponga che in tale sistema aritmetico sia presente un *registro* per eseguire i calcoli in doppia precisione con precisione $t_{reg} = 8$. Verificare che il sistema aritmetico considerato non gode della proprietà associativa calcolando $a + b + c$, con a, b, e c:

$$a = 0.6472 \times 10^0$$
$$b = 0.4685 \times 10^{-4}$$
$$c = 0.3297 \times 10^{-4}$$

Risposta

Sommiamo nell'ordine (a+b)+c:

Dati	*Memoria*	*Registro*
a	0.6472×10^0	
b	0.4685×10^{-4}	0.00004685×10^0
$a + b$		0.64724685×10^0
$fl(a + b)$	0.6472×10^0	
c	0.3297×10^{-4}	0.00003297×10^0
$fl(a + b) + c$		0.64723297×10^0
$fl(fl(a + b) + c)$		0.6472×10^0

Il risultato è uguale ad a perché b e c sono entrambi molto più piccoli del primo. Sommiamo adesso nell'ordine (b+c)+a:

Dati	*Memoria*	*Registro*
c	0.3297×10^{-4}	0.00003297×10^{0}
$b+c$		0.00007982×10^{0}
$fl(b+c)$	0.7982×10^{-4}	0.00007982×10^{0}
$a+fl(b+c)$		0.64727982×10^{0}
$fl(a+fl(b+c))$		0.6473×10^{0}

Il risultato di $(b+c)+a$ differisce da $(a+b)+c$. Resta dunque provato che il sistema aritmetico considerato *non è* ***associativo***.

Esercizio 12 Sia:

$$\Im : \beta = 2, t = 7, emax = 7, emin = -7.$$

dotato di massima accuratezza dinamica. Rappresentare $x = 9.6$ e $z = 4.2$ in $\Im$. Trovare due numeri $y \in \Im$ e $t \in \Im$ tali che $x + y = x$ e $z + t = z$.

Risposta

Si ha:

$$9 = 1001_2 \quad 0.6 = 0.\overline{1001}_2$$

$$4 = 100_2 \quad 0.2 = 0.\overline{0011}_2$$

per cui:

$$x = 0.1001100 \times 2^{100}$$

$$y = 0.125 = 0.1 \times 2^{-011} \quad x + y = 0.1001100 \times 2^{100} + 0.00000001 \times 2^{100} = x$$

$$x = 0.1001100 \times 2^{100}$$

$$z = 0.0325 = 0.1 \times 2_2^{-101} \quad z + t = 0.1001100 \times 2^{011} + 0.000000001 \times 2^{011} = z$$

Esercizio 13 Dati $x = 123.4578$ e $\overline{x} = 123.4599$ calcolare l'errore assoluto E_a e l'errore relativo E_r. Contare il numero di cifre significative e decimali.

Risposta

Si ha $E_a = |123.4599 - 123.4578| = 0.0021 = 0.21 \times 10^{-2}$ da cui si ha conferma che l'approssimazione è corretta a **2 cifre decimali**, mentre $E_r = E_a/123.4578 = 0.000017.. = 0.17... \times 10^{-4}$, da cui si ricava che l'approssimazione è corretta a **5 cifre significative**.

Esercizio 14 Dati $x = 0.000418$ e $\overline{x} = 0.000412$ calcolare l'errore assoluto E_a e l'errore relativo E_r. Contare il numero di cifre significative e decimali.

Risposta

Si ha $E_a = |0.000418 - 0.000412| = 0.000006 = 0.6 \times 10^{-5}$ da cui si ha conferma che l'approssimazione è corretta a **5 cifre decimali**, mentre $E_r = E_a/0.000418 = 0.01435 = 0.1435... \times 10^{-1}$, da cui si ricava che l'approssimazione è corretta a **2 cifre significative**.

Esercizio 15 Sia:

$$\Im : \quad \beta = 10, \quad t = 5, \quad e_{min} = -9 \quad e_{max} = 9.$$

Si determini la massima accuratezza relativa μ. Rappresentare, mediante arrotondamento, i numeri $x_1 = 123.4578$, $x_2 = 0.0215984$, calcolare inoltre l'errore relativo di round-off e verificare che esso è minore della massima accuratezza relativa.

Risposta

Si ha $\varepsilon = 0.5 \times 10^{-4}$.

Inoltre è $x_1 = 0.1234578 \times 10^3$ e $\overline{x_1} = 0.12346 \times 10^3$.

L'errore relativo allora è

$$\frac{|0.1234578 \times 10^3 - 0.12346 \times 10^3|}{0.1234578 \times 10^3} = 0.000017... = 0.17... \times 10^{-4} < \mu$$

Si ha poi $x_2 = 0.215984 \times 10^{-1}$ e quindi $\overline{x_2} = 0.21598 \times 10^{-1}$. L'errore relativo allora è

$$\frac{|0.215984 \times 10^{-1} - 0.21598 \times 10^{-1}|}{0.215984 \times 10^{-1}} = 0.000018.. = 0.18.. \times 10^{-4} < \mu$$

Esercizio 16 Sommare le seguenti coppie di numeri in un sistema aritmetico a massima accuratezza dinamica: $0.54321 \times 10^{-1} + 0.76543 \times 10^{2}$ e $0.49854 \times 10^{3} + 0.21485 \times 10^{-3}$.

Risposta

La prima somma è 0.76597×10^{2}, mentre la seconda è 0.49854×10^{3}.

Esercizio 17 Assumiamo un sistema f.p. con base $\beta = 10$, $t = 4$, precisione relativa della macchina $\epsilon_{mac} = 10^{-5}$ ed esponente massimo e minimo $emax = +20$ ed $emin = -20$, rispettivamente. Rappresentare il risultato delle operazioni f.p. seguenti

(a) $1 + 10^{-7}$ (b) $1 + 10^{3}$
(c) $1 + 10^{7}$ (d) $10^{10} + 10^{3}$
(e) $10^{10}/10^{15}$ (f) $10^{-10} \times 10^{-15}$

Risposta

Si ottiene

(a) 0.1000×10^{1} (b)0.1001×10^{4}
(c) 0.1000×10^{8} (d) 0.1000×10^{11}
(e) $0.1 \times 10^{-4} = \epsilon_{mac}$ (f) $0.1 \times 10^{-24} = 0$

Esercizio 18 In un sistema aritmetico avente una soglia di $UFL = 10^{-38}$ si eseguano le operazioni:

(I) $a = \sqrt{y^2 + z^2}$ (II) $a = \sqrt{w^2 + z^2}$ (III) $u = (v \times y)/(w \times z)$

con $y = 1$,$v = 10^{-15}$, $w = 10^{-20}$ *e* $\quad z = 10^{-25}$

Risposta

Si ottiene:

(I) $a = \sqrt{y^2 + z^2} = \sqrt{1 + (10^{-25})^2} = \sqrt{1 + 10^{-50}} = \sqrt{1 + 0} = 1$

(II) $a = \sqrt{w^2 + z^2} = \sqrt{10^{-40} + 10^{-50}} = \sqrt{0 + 0} = 0$

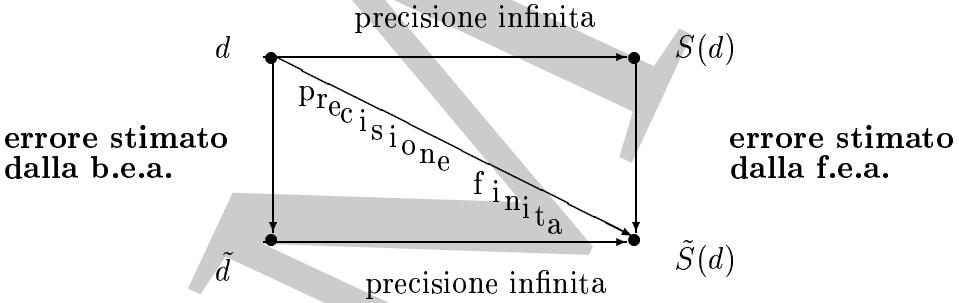
(III) $u = \frac{10^{-15}}{0} = INF$

1.10.2 Alcuni quesiti

1. *Quando la soluzione calcolata di un dato problema può essere considerata accurata nel senso della* Backward Error Analysis?

   - Risposta

     Nella Backward Error Analysis si assume che la soluzione calcolata di un dato problema sia accurata, se essa è la soluzione esatta (ottenuta in un sistema aritmetico a precisione infinita) del medesimo problema perturbato, ovvero dello stesso problema in cui sia stata introdotta una variazione nei dati comparabile con l'errore di *round-off* di rappresentazione dei dati.
     Indicata con f la funzione che descrive il problema da risolvere, x i dati esatti, $\hat{x}$ i dati affetti da errore, $\hat{f}$ la funzione che descrive il problema perturbato risulta: $\hat{f}(x) = f(\hat{x}) = f(x+\delta)$, con $|\delta| \leq u$, dove u è la precisione del sistema.
     In figura è descritto il diagramma operativo della Backward Error Analysis, confrontato con l'altra tecnica di analisi della propagazione dell'errore, cioé la Forward Analysis.



Errori stimati con la forward error analysis e con la backward error analysis.

2. *In un sistema aritmetico a precisione finita cosa è più pericoloso durante l'esecuzione di un'operazione: un* underflow *o un* overflow? *E perché*?

   - Risposta

     Un overflow è un evento più difficile da gestire durante il calcolo. Esso infatti, può comportare l'interruzione dell'esecuzione del calcolo; alcuni sistemi utilizzano degli opportuni indicatori di tali situazioni eccezionali, come ad esempio nell'IEEE il numero macchina INF. Quando si realizza un underflow invece, la maggior parte dei sistemi aritmetici assegna al risultato del calcolo il **valore 0**. Questo valore può essere plausibile e quindi non inficiare l'esecuzione delle successive operazioni. Tuttavia anche in tal caso va verificato se il numero che è andato in underflow non sia poi utilizzato come divisore di una successiva operazione e creare, dunque, un' ulteriore situazione d'errore.

3. *In un sistema aritmetico f.p. a precisione finita, il prodotto di due numeri macchina non è, di solito, rappresentabile esattamente. Perché?*

   - Risposta

     Consideriamo il caso in cui non si verifichino situazioni eccezionali, cioé il risultato, r, dell'operazione sia ancora rappresentabile in $\Im$. Supponiamo inoltre che il sistema aritmetico a precisone finita $\Im$ garantisca la massima accuratezza dinamica, ovvero sia tale che il risultato $\tilde{r}$ di un'operazione f.p. differisca dal risultato calcolato in aritmetica a precisione infinita, di una quantità che è il solo errore di rappresentazione di r in $\Im$, avviene cioé:

     $$\tilde{r} = x \otimes y = fl(r) = fl(x \times y)$$

     con

     $$\begin{aligned} x &= fl(x) \\ y &= fl(y) \end{aligned}$$

     Il risultato di un prodotto tra due numeri esattamente rappresentabili, generalmente non *è esattamente rappresentabile* in quanto se, ad esempio, si moltiplicano due numeri di t cifre il risultato può essere (al più) composto da 2t cifre ed è dunque necessario un *troncamento* (o un *arrotondamento*) se t è la precisione del sistema aritmetico.

     Fissato un sistema aritmetico a precisione finita di base $\beta = 10$, precisione $t = 2$ ed $emin = -2$, $emax = 2$, che utilizzi il troncamento, sia:

     $$\begin{aligned} x &= 0.25 \times 10^2 \\ y &= 0.25 \times 10^2 \end{aligned}$$

allora

$$\tilde{r} = fl((0.25 \times 10^2) \times (0.25 \times 10^2)) = 0.62 \times 10^3 \quad \neq \quad r = 0.625 \times 10^3$$

4. *Si dia un esempio di operazione f.p. che possa produrre i valori INF e NaN.*

   - Risposta

   I valori indicati sono introdotti in un sistema aritmetico standard IEEE quando si eseguono delle operazioni del tipo:

$$NaN = \begin{cases} \frac{0}{0} \\ 0 * \infty \\ \infty * \infty \end{cases}$$

$$Inf = \frac{k}{0}, \qquad k \in \Im - \{0\}$$

   Se si considera un sistema aritmetico f.p. con soglia di UFL=10^{-12} ed una costante reale $\alpha = 0.2 \times 10^{-14}$, l'operazione seguente produce NaN:

$$\frac{\alpha}{\alpha} = \frac{0}{0} = NaN$$

   Mentre il valore *INF* si ottiene quando si effettua una divisione per zero nel sistema aritmetico f.p. utilizzato:

$$INF = \frac{k}{0}, \qquad k \in \Im - \{0\}.$$

5. *Si spieghi la differenza tra l'epsilon macchina ϵ_{mac} ed il livello di underflow UFL in un sistema aritmetico f.p. Di queste due quantità:*

   (a) Quale delle due dipende solo dal numero di cifre della mantissa?

   (b) Quale delle due dipende solo dall'esponente?

   (c) Quale delle due non dipende dal tipo di arrotondamento utilizzato?

   (d) Quale delle due non cambia quando si utilizzano dei numeri denormalizzati?

   - Risposta

   Il livello di underflow (UFL) è il più piccolo numero macchina positivo, ovvero:

$$UFL = 0.1 \times \beta^{emin} = \beta^{emin-1}$$

   L'epsilon macchina (ϵ_{mac}) è il più piccolo numero macchina positivo che dà contributo alla somma f.p. con 1; ovvero:

$$1 \oplus \epsilon_{mac} = fl(1 + \epsilon_{mac}) > 1$$

Risulta:

$$\epsilon_{mac} = \beta^{1-t} \ \ con \ troncamento$$

(a) L'ϵ_{mac}.
(b) L'UFL.
(c) L'UFL.
(d) La precisione relativa della macchina non varia se si utilizzano numeri denormalizzati. Al contrario l'introduzione dei numeri denormalizzati permette al sistema di avere un *Underflow graduale*, che è implementato riservando valori speciali al campo dell'esponente. Dunque la soglia di UFL varia.
Se si considera un sistema aritmetico a precisione finita

$$\Im = (\beta, t, emin, emax)$$

che adoperi numeri denormalizzati allora è possibile rappresentare:

$$\begin{aligned}
0.\underbrace{01.....0}_{\text{t}} \times \beta^{emin} &= \quad 0.1 \times \beta^{emin-1} \\
0.\underbrace{001...0}_{\text{t}} \times \beta^{emin} &= \quad 0.1 \times \beta^{emin-2} \\
\ldots & \qquad\qquad \ldots \\
0.\underbrace{00.....1}_{\text{t}} \times \beta^{emin} &= \quad 0.1 \times \beta^{emin-t}
\end{aligned}$$

dunque la soglia di UFL diventa $0.1 \times \beta^{emin-t}$.

6. *Si spieghi il fenomeno della cancellazione.*

- Risposta

La cancellazione è un fenomeno tipico dei sistemi a *precisione finita*. Esso si verifica a causa del numero di cifre finito della mantissa per rappresentare un numero f.p. e può determinare la perdita di cifre significative quando si esegue una sottrazione tra numeri "vicini".
Ad esempio, considerato un sistema aritmetico a precisione finita con $\beta = 10$, $t = 4$, $emin = -12$, $emax = 12$, eseguendo la sottrazione f.p. dei due numeri seguenti:

$$0.1234 \times 10^3 - 0.1233 \times 10^3 = 0.0001 \times 10^3 = 0.1000 \times 10^0$$

si ottiene un numero con una sola cifra significativa, in altre parole, si perdono 3 cifre significative nella rappresentazione del risultato.
In generale se

$$f \ : \ \Re \times \Re \ \rightarrow \ \Re$$

è una funzione che alla coppia di numeri reali (x, y) associa il numero reale $z = x - y$, è noto (cfr. cap.1) che l'indice di condizionamento del problema, è dato da

$$C(f, x, y) = \frac{(|x| + |y|)}{|x - y|}$$

Tale quantità cresce al diminuire della distanza tra x e y. Cioé la cancellazione è l'effetto del **mal-condizionamento** della sottrazione tra due numeri f.p. "vicini".

7. *Per calcolare il punto medio m di un intervallo [x,y], quale delle seguenti due formule è preferibile utilizzare in un sistema aritmetico f.p. a precisione finita? Perché?*

$$(a)\ m = f_1(x, y) = (x + y)/2\ , \quad (b)\ m = f_2(x, y) = x + (y - x)/2$$

- Risposta

Calcoliamo l'indice di condizionamento relativo $C(f, x, y)$ delle due formule. [55] Per la formula (a) si ha:

$$C(f_1, x, y) = \left| \frac{x}{\frac{x+y}{2}} \cdot \frac{1}{2} \right| + \left| \frac{y}{\frac{x+y}{2}} \cdot \frac{1}{2} \right| = \frac{|x| + |y|}{|x + y|} \tag{1.53}$$

Applicando la proprietà triangolare dei valori assoluti al denominatore di $C(f_1, x, y)$ abbiamo che la (1.53) risulta limitata inferiormente e superiormente dalle quantità seguenti:

[55] Relativamente al problema di valutare una funzione f, reale di variabile reale, in un punto x, si ricava, come *indice di condizionamento relativo*, la quantità:

$$C(f, x) = |f'(x)x|/|f(x)$$

$$\underbrace{\frac{|x|+|y|}{|x|+|y|}}_{1} \leq \frac{|x|+|y|}{|x+y|} \leq \frac{|x|+|y|}{||x|-|y||}$$

Se x ed y sono "vicini" tra loro in modulo, $|x| \approx |y|$, $C(f_1, x, y)$ non è limitato superiormente:

$$\lim_{|y|\to|x|} \frac{|x|+|y|}{||x|-|y||} = \frac{2|x|}{0} = +\infty$$

quindi in questo caso la formula (a) non è ben condizionata ed il calcolo di m con tale formula, in un sistema aritmetico $\Im$ a precisione finita, puó condurre ad un risultato non accettabile.
La formula (b) ha invece indice di condizionamento uguale a:

$$\begin{aligned} C(f_2, x, y) &= \left|\frac{x}{x+\frac{y-x}{2}} \cdot \left(1-\frac{1}{2}\right)\right| + \left|\frac{y}{x+\frac{y-x}{2}} \cdot \frac{1}{2}\right| = \\ &= \frac{1}{2} \cdot \left|\frac{x}{x+\frac{y-x}{2}}\right| + \frac{1}{2} \cdot \left|\frac{y}{x+\frac{y-x}{2}}\right| = \\ &= \frac{1}{2} \cdot \frac{|x|+|y|}{\left|x+\frac{y-x}{2}\right|} \end{aligned} \tag{1.54}$$

Applicando la proprietá triangolare dei valori assoluti al denominatore di $C(f_2, x, y)$, la (1.54) risulta limitata inferiormente e superiormente dalle quantità seguenti:

$$\frac{|x|+|y|}{2|x|+|y-x|} \leq \frac{1}{2} \cdot \frac{|x|+|y|}{\left|x+\frac{y-x}{2}\right|} \leq \frac{|x|+|y|}{|2|x|-|y-x||}$$

In questo caso, se x ed y sono "vicini" tra loro, $x \approx y$, allora:

* $\lim_{y\to x} \frac{|x|+|y|}{2|x|+|y-x|} = \frac{2|x|}{2|x|} = 1$
* $\lim_{y\to x} \frac{|x|+|y|}{|2|x|-|y-x||} = \frac{2|x|}{2|x|} = 1$

e per il teorema del confronto si ha:

$$\lim_{y\to x} \frac{1}{2} \cdot \frac{|x|+|y|}{\left|x+\frac{y-x}{2}\right|} = 1 \quad .$$

Pertanto, nel caso in cui x ed y siano "vicini" tra loro, la formula (b) risulta ben condizionata e dunque il calcolo di m con tale formula, in un sistema aritmetico $\Im$ a precisione finita, conduce ad un risultato accettabile.

Consideriamo un sistema aritmetico $\Im$ a precisione finita caratterizzato dai seguenti parametri:

$$\Im : \beta = 10,\ t = 4,\ emin = -9,\ emax = +9,$$

e supponiamo che in $\Im$ venga utilizzata, per la rappresentazione dei numeri, la funzione di *troncamento*. Calcoliamo in $\Im$ il punto medio m per le seguenti coppie di numeri (in parentesi è riportato il valore di m che si ottiene in aritmetica a precisione infinita), distinguiamo due casi:

1.) x ed y "vicini" tra loro in modulo:
   1.a) $x = 3.7678,\ y = 3.7668,\ (m = 0.37673 \times 10^1)$;
   1.b) $x = 3.7678,\ y = -3.7668,\ (m = 0.5 \times 10^{-3})$;

2.) x ed y "distanti":
   2.a) $x = 3.7678,\ y = 2.6211,\ (m = 0.319445 \times 10^1)$;
   2.b) $x = 2.9768,\ y = -6.7678,\ (m = -0.18955 \times 10^1)$;
   2.c) $x = 3.7678,\ y = 0.2611 \times 10^{-2},\ (m = 0.18852055 \times 10^1)$;
   2.d) $x = 2.9775,\ y = -61.3267,\ (m = -0.291746 \times 10^2)$;
   2.e) $x = -2.7502,\ y = -6.36852,\ (m = -0.455936 \times 10^1)$.

Indichiamo con $\tilde{x}$ e $\tilde{y}$, la rappresentazione di x e y in $\Im$ ed analizziamo le soluzioni $\tilde{m}_a$, $\tilde{m}_b$ ottenute in tale sistema rispettivamente con le formule (a) e (b):

	$\tilde{x}$	$\tilde{y}$	$\tilde{m}_a$	$\tilde{m}_b$
1.a)	0.3767×10^1	0.3766×10^1	0.3766×10^1	0.3767×10^1
1.b)	0.3767×10^1	-0.3766×10^1	0.0×10^0	0.1×10^{-3}
2.a)	0.3767×10^1	0.2621×10^1	0.3194×10^1	0.3194×10^1
2.b)	0.2976×10^1	-0.6767×10^1	-0.1895×10^1	-0.1895×10^1
2.c)	0.3767×10^1	0.2611×10^{-2}	0.1884×10^1	0.1885×10^1
2.d)	0.2977×10^1	-0.6132×10^2	-0.2922×10^2	-0.2916×10^2
2.e)	-0.2750×10^1	-0.6368×10^1	-0.4559×10^1	-0.4559×10^1

Di seguito sono riportati gli errori relativi E_a ed E_b, che si commettono nell'approssimare m, rispettivamente con $\tilde{m}_a$ ed $\tilde{m}_b$:

	E_a	E_b
1.a)	0.345×10^{-3}	0.769×10^{-4}
1.b)	0.1×10^1	0.8×10^0
2.a)	0.1408×10^{-3}	0.1408×10^{-3}
2.b)	0.2637×10^{-3}	0.2637×10^{-3}
2.c)	0.6394×10^{-3}	0.109×10^{-3}
2.d)	0.1576×10^{-3}	0.146×10^{-3}
2.e)	0.789×10^{-4}	0.789×10^{-4}

Osservando la tabella, entrambe le formule producono risultati accettabili, in particolare per i casi che abbiamo distinto:

1.) x ed y "vicini" tra loro in modulo ($|x| \approx |y|$):
la formula (b) produce risultati piú accurati (1.a), 1.b)), infatti gli errori ottenuti sono dell'ordine della precisione del sistema, o al piú il risultato differisce dalla soluzione esatta, sull'ultima cifra significativa;

2.) se x ed y sono distanti tra loro (concordi o discordi):
entrambe le formule producono lo stesso risultato nei casi 2.a), 2.b) e 2.e), mentre nel caso 2.c) e 2.d) è piú accurata la formula (b);

8. *Si elenchino due situazioni in cui il calcolo della formula:*

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

può dare luogo ad errori, in aritmetica floating-point.

- RISPOSTA

(I) Se b è sufficientemente grande il calcolo di b^2 potrebbe comportare un overflow.

(II) Ci potrebbero essere errori di cancellazione distruttiva quando si effettua il calcolo del discriminante.

Consideriamo, ad esempio, un sistema aritmetico a precisione finita con $\beta = 10$, $t = 4$, $emin = -10$, $emax = 10$; allora se:

$$(1)\ a = 0.1 \times 10^1 \quad b = 0.2 \times 10^6 \quad c = 0.3 \times 10^{-1}$$
$$(2)\ a = 0.1 \times 10^1 \quad b = 0.1 \times 10^4 \quad c = 0.3 \times 10^1$$

si ha:

$$(1)\ b^2 = 0.4 \times 10^{11} \qquad \text{(Overflow)}$$
$$(2)\ \Delta = 0.1 \times 10^7 - 0.12 \times 10^2 = 0.1 \times 10^7 \qquad \text{(Cancellazione)}$$

9. *Si supponga di risolvere un'equazione di secondo grado completa del tipo:*

$$ax^2 + bx + c = 0$$

utilizzando un sistema aritmetico f.p. (normalizzato) con $\beta = 10$ $t = 3$ *e* $a = 1.22$, $b = 3.34$, $c = 2.28$.

(a) Qual è il valore calcolato del discriminante $b^2 - 4ac$*?*

(b) Qual è il valore corretto del discriminante in aritmetica a precisione infinita?

(c) Qual è l'errore relativo nel valore calcolato del discriminante?

- RISPOSTA

  Indicando con $\tilde{\Delta}$ il discriminante calcolato e con Δ quello in aritmetica a precisione infinita si ottiene:

  $$(a)\ \tilde{\Delta} = 0.111556 \times 10^2 - 0.111164 \times 10^2 = 0$$
  $$(b)\ \Delta = 0.292 \times 10^{-1}$$
  $$(c)\ E_{rel} = \frac{|\Delta - \tilde{\Delta}|}{|\Delta|} = 1 = 100\,\%$$

10. *Si consideri l'espressione:*

$$\frac{1}{1-x} - \frac{1}{1+x},$$

(a) Per quale intervallo di valori di x è difficile un calcolo accurato in aritmetica f.p.?

(b) Si dia un'espressione equivalente della formula in modo che, per i valori di x del punto (a), il calcolo sia più accurato in aritmetica f.p.

- RISPOSTA

  (a) Si possono avere problemi nel calcolo dell'espressione quando $fl(x)$ non dà contributo nella somma/differenza con 1. Infatti se $|fl(x)| < \epsilon_{macc}$:

  $$\frac{1}{1-fl(x)} - \frac{1}{1+fl(x)} \equiv 0$$

  Ad esempio, se consideriamo un sistema aritmetico a precisione finita di base $\beta = 10$, $t = 8$ $emin = -20$, $emax = 20$, se $x = 10^{-7}$ allora:

  $$\frac{1}{1-10^{-7}} - \frac{1}{1+10^{-7}} \equiv 1 - 1 \equiv 0$$

  (b) Se, invece, si considera l'espressione identica:

  $$\left(\frac{1}{1-x} - \frac{1}{1+x}\right) = \frac{2x}{1-x^2}$$

  in questo caso, nello stesso sistema aritmetico del punto (a) scegliendo ancora $x = 10^{-7}$:

  $$\frac{2 \cdot 10^{-7}}{1 - 10^{-14}} = \frac{2 \cdot 10^{-7}}{1-0} = 2 \cdot 10^{-7} \neq 0.$$

11. *Se $x \approx y$ ci si aspetta un errore di cancellazione nel calcolo di:*

$$z = log(x) - log(y)$$

Tale problema si può risolvere osservando che:

$$z = log(x) - log(y) = log\left(\frac{x}{y}\right).$$

Con questo accorgimento si può affermare che il calcolo di log(x/y) è più accurato?

- Risposta
  Si, perché se

$$x = y \cdot o(\epsilon)$$

  allora

$$log(x) = log(y) + log(o(\epsilon)),$$

  e il calcolo di $z = log(x) - log(y)$ può creare problemi di cancellazione distruttiva. Ciò non accade se z viene valutato nel modo seguente:

  (a) Calcolo di $\frac{x}{y}$

  (b) Calcolo di $z = log\left(\frac{x}{y}\right)$.

  Consideriamo il seguente sistema aritmetico a precisione finita, $\Im : \beta = 10$, $t = 7$, $emin = -10$, $emax = 10$, siano:

$$x = 0.2509999 \times 10^1, \; y = 0.2510001 \times 10^1$$

  Calcoliamo i logaritmi di x e y:

$$log(x) = 0.3996735 \times 10^0$$

$$log(y) = 0.3996739 \times 10^0$$

  Nel calcolo della differenza fra questi valori, si perdono tutte le cifre significative, a causa della cancellazione distruttiva; infatti si ha:

$$\begin{aligned} log(x) - log(y) &= 0.3996735 \times 10^0 - 0.3996739 \times 10^0 = \\ &= -0.00000004 \times 10^0 = -0.4 \times 10^{-6} \end{aligned}$$

Calcoliamo ora il rapporto tra x e y:

$$\frac{x}{y} = 0.9999992 \times 10^0$$

da cui:

$$log\left(\frac{x}{y}\right) = -0.3474357 \times 10^{-8}$$

il risultato ottenuto ha 7 cifre significative ed è più accurato del precedente, in quanto il risultato, in aritmetica a precisione infinita, è:

$$log\left(\frac{x}{y}\right) = log(x) - log(y) = -0.3474357245069005... \times 10^{-8}$$

12. *Se x è un vettore di $\Re^n$, la norma Euclidea è definita da:*

$$\|y\|_2 = \left(\sum_{i=1}^{n} y_i^2\right)^{1/2}$$

Come è possibile evitare un overflow nel calcolo di tale norma?

- Risposta

Calcolando la norma Euclidea con l'accorgimento seguente, posto $\tilde{y} = \max_{i=1,n} y_i$:

$$\|y\|_2 = \tilde{y} \cdot \sqrt{\sum_{i=1}^{n} \left(\frac{y_i}{\tilde{y}}\right)^2}$$

Dividendo per il più grande degli y_i riduciamo l'ordine di grandezza dei numeri e quindi evitiamo l'overflow.
Consideriamo il sistema aritmetico a precisione finita $\Im : \beta = 10, t = 5$, $emin = -4$, $emax = +4$, ed il vettore di $\Re^3$:

$$y = (0.1 \times 10^2,\ 0.2 \times 10^3,\ 0.6 \times 10^1)$$

Consideriamo il quadrato delle sue componenti:

$$y_1^2 = 0.1 \times 10^3\,, y_2^2 = 0.4 \times 10^5,\ y_3^2 = 0.36 \times 10^2$$

la componente y_2 al quadrato genera un overflow.
Scaliamo adesso il vettore per la componente di valor massimo:

$$\tilde{y}_1 = \frac{0.1 \times 10^1}{0.2 \times 10^3} = 0.5 \times 10^{-1},\ \tilde{y}_2 = 0.1 \times 10^1,\ \tilde{y}_3 = \frac{0.6 \times 10^1}{0.2 \times 10^3} = 0.3 \times 10^{-1}$$

Calcoliamo il quadrato delle componenti, $\tilde{y}_i$:

$$\tilde{y}_1^2 = 0.25 \times 10^{-3},\ \tilde{y}_2^2 = 0.1 \times 10^1,\ \tilde{y}_3^2 = 0.9 \times 10^{-3}$$

in questo modo viene evitato l'overflow. Calcoliamo la norma, in aritmetica a precisione infinita:

$$\|y\|_2 = 0.20033971 \times 10^3$$

mentre con l'accorgimento proposto, si ottiene:

$$\|\tilde{y}\|_2 = 0.20034 \times 10^3$$

come si può notare, l'errore che si commette è sulla quinta cifra significativa, ed è trascurabile perché compatibile con la precisione del sistema.

13. *Assumiamo un sistema aritmetico f.p. normalizzato con $\beta = 10$, $t = 3$, e emin $= -98$.*

    (a) Qual è la soglia di UFL per questo sistema?

    (b) Se $x = 6.87 \times 10^{-97}$ e $y = 6.81 \times 10^{-97}$, qual è il risultato di $x - y$?

    (c) Quale sarebbe il risultato di x - y se il sistema avesse un underflow graduale?

    - Risposta

      (a) $UFL = 0.1 \times 10^{-97}$
      (b) $x - y = 0.6 \times 10^{-99} = INF = 0$
      (c) In un sistema che permette l'underflow graduale risulta:
      $x - y = 0.006 \times 10^{-97} = 0.6 \times 10^{-99}$

14. *Sia $\{x_k\}$ una successione monotona decrescente finita di numeri positivi ($x_k > x_{k+1}$ per ogni k). In quale ordine gli elementi della successione devono essere sommati per minimizzare l'errore di arrotondamento?*

    - Risposta

      Sommando i numeri in ordine crescente, ovvero dal più piccolo al più grande, si sommano numeri approssimativamente dello stesso ordine di grandezza. Si riducono quindi eventuali errori di cancellazione dovuti all'arrotondamento. Infatti sia $\Im$ un sistema aritmetico a precisione finita di parametri $\beta = 10$,

$t = 7$, $emin = -10$, $emax = +10$. Sommiamo i primi 1000 termini nell'ordine **naturale**:

$$T_{1000} = \sum_{k=0}^{1000} \frac{5}{(2k+1)^3} = 5 + \frac{5}{3^3} + \cdots + \frac{5}{2001^3}$$

$$T_{1000} = 0.5258986 \times 10^1$$

Se invece si sommano i termini in maniera **decrescente** si ha:

$$T_{1000} = 0.525899 \times 10^1$$

Si assume che il valore "esatto" sia:

$$T^*_{1000} = 0.525899\ldots \times 10^1$$

dove tutte le cifre indicate sono esatte.
Quindi l'errore assoluto che si commette utilizzando tale strategia ha un ordine di grandezza di 10^{-5}, ovvero:

$$E = |T_{1000} - T^*_{1000}| = 0.12 \times 10^{-5}$$

Mentre l'errore, nel sistema aritmetico $\Im$, utilizzando la seconda strategia, risulta nullo.

15. *Si spieghi perché la serie divergente:*

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

possa avere una somma finita in aritmetica f.p. A partire da quale indice gli addendi non danno più contributo alla sommatoria con i termini precedenti?

- Risposta
  Volendo eseguire la somma in un sistema aritmetico a precisione finita, esiste un indice n_0 per cui il termine $1/n_0$ puó non dare contributo alla somma con:

$$\sum_{k=1}^{n-1} \frac{1}{k}$$

  per due motivi:
  - perché $\frac{1}{n_0} <$ underflow, e quindi ad esso viene assegnato il valore zero;

– perché:

$$\left|\frac{1}{n_0}\right| < \epsilon_{mac} \left|\sum_{k=1}^{n-1} \frac{1}{k}\right|$$

e dunque il temine n_0-esimo della serie non dà contributo alla somma con i termini precedenti.

16. *Se f è una funzione reale derivabile, di variabile reale, si ha:*

$$f'(x) = \frac{f(x+h) - f(x)}{h} + E(x), \qquad E(x) = O(h^2)$$

dove l'errore $E(x)$ tende a zero quando $h \to 0$. Una sua approssimazione è, quindi,

$$\frac{f(x+h) - f(x)}{h}$$

Sotto quali condizioni, nel sistema aritmetico a precisione finita $\Im$, tale espressione ha significato?

- Risposta

In un sistema aritmetico $\Im$ a precisione finita i due numeri macchina x e $x+h$ sono distinti se

$$h \geq \epsilon_{mac} \cdot |x|$$

cioé, risultano distinti se h è un numero macchina che dà contributo alla somma con x. Consideriamo, ad esempio, un sistema aritmetico $\Im$ a precisione finita caratterizzato dai parametri

$$\Im : \beta = 10, t = 4, emin = -10, emax = +10,$$

e sia

$$f(x) = x^2.$$

Calcoliamo il più piccolo numero h che dà contributo alla somma con

$$x = 0.3 \times 10^1.$$

In $\Im$ risulta

$$\epsilon_{mac} = 0.5 \times 10^{-3},$$

eseguendo le operazioni in doppia precisione e cioé $t_{reg} = 8$; si ottiene, dunque:

$$h \geq \epsilon_{mac} \cdot |x| = 0.5 \times 10^{-4} \times 0.3 \times 10^{1} = 0.15 \times 10^{-3}$$

per cui è sufficiente che sia

$$h = 0.15 \times 10^{-3}$$

e, di conseguenza,

$$x + h = 0.300015 \times 10^{1} \neq x,$$

perché abbia significato la valutazione della funzione f in due valori distinti. D'altra parte, risulta

$$|f(x+h)| \geq \epsilon_{mac} \cdot |f(x)| \Rightarrow f(x+h) \neq f(x)$$

pertanto deve risultare che:

$$f(x+h) \geq \epsilon_{mac} \cdot f(x)$$

perché abbia significato il numeratore del rapporto incrementale.

17. *Illustrare un metodo per evitare il fenomeno della cancellazione distruttiva nel calcolo di una somma finita del tipo:*

$$\sum_{k=0}^{N} \frac{(-1)^k}{2k+1} \qquad (1)$$

- Risposta

Per evitare il fenomeno della **cancellazione distruttiva** occorre raggruppare i termini dello stesso segno e sommarli in maniera decrescente.
Ciò significa:

(a) **raggruppare** i termini dello stesso segno;
(b) **eseguire** la somma parziale dei numeri in maniera decrescente;
(c) **eseguire** la differenza tra le somme parziali.

Supponiamo di voler valutare la somma (1) per $n = 20$

$$\sum_{k=0}^{20} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots - \frac{1}{41}$$

Nei passi (a) e (b) si calcolano le somme parziali seguenti:

$$\sum_{k=pari}^{k=20\ldots0} \frac{(-1)^k}{2k+1} = \frac{1}{39} + \frac{1}{35} + \cdots + \frac{1}{9} + \frac{1}{5} + 1$$

$$-\sum_{k=dispari}^{k=19\ldots1} \frac{(-1)^k}{2k+1} = \frac{1}{41} + \frac{1}{37} + \cdots + \frac{1}{11} + \frac{1}{7} + \frac{1}{3}$$

nel passo (c) si calcola la differenza delle somme parziali:

$$\sum_{k=0}^{20} \frac{(-1)^k}{2k+1} = \sum_{k=pari}^{k=19\ldots0} \frac{(-1)^k}{2k+1} + \sum_{k=dispari}^{k=20\ldots1} \frac{(-1)^k}{2k+1}$$

18. *Il condizionamento è un effetto della precisione aritmetica con cui si risolve il problema.*

    - RISPOSTA [FALSO] . Il condizionamento è una caratteristica intrinseca del problema e dipende solo dalla natura del problema. La precisione aritmetica fa sí che si evidenzino gli effetti del mal condizionamento.

19. *Il condizionamento di un problema dipende dall'algoritmo utilizzato per risolverlo.*

    - RISPOSTA [FALSO]. Il condizionamento dipende solo dal problema.

20. *Utilizzando la massima precisione di un sistema aritmetico floating-point a precisione finita, si puó migliorare il condizionamento di un problema* ***mal-condizionato***.

    - RISPOSTA [FALSO]. Il fattore di amplificazione degli errori sui dati dipende dal problema e non dal sistema aritmetico utilizzato. Pertanto, anche utilizzando una maggiore precisione aritmetica, l'amplificazione dell'errore sulla soluzione non puó essere evitata.

21. *Un "buon" algoritmo produce una soluzione accurata indipendentemente dal problema che si sta risolvendo.*

    - RISPOSTA [FALSO]. L'accuratezza della soluzione di un problema dipende oltre che dall'algoritmo anche dal condizionamento del problema. Pertanto, un "buon" algoritmo (cioè un algoritmo stabile), se viene utilizzato per risolvere un problema mal-condizionato, può fornire una soluzione inaccurata.

22. *La scelta di un algoritmo per risolvere un problema ha effetto sulla propagazione degli errori nei dati.*

- RISPOSTA [VERO]. In generale due algoritmi per la risoluzione di uno stesso problema possono produrre risultati diversi in quanto gli errori di **round-off**, dovuti al modo differente con cui vengono effettuate le operazioni, possono propagarsi in maniera diversa.

23. *Se due numeri reali sono esattamente rappresentabili come numeri f.p. in un sistema aritmetico a precisione finita, allora il risultato di un'operazione aritmetica su di loro può non essere esattamente rappresentabile.*

- RISPOSTA [VERO]. Adoperando un sistema aritmetico a *precisione finita* quanto affermato è sicuramente vero.
  Ad esempio, fissata la base $\beta = 10$, la precisione $t = 7$ e gli esponenti minimo e massimo rispettivamente $emin = -2$, $emax = +2$, considerati i due numeri macchina

$$x_1 = 0.3354110 \times 10 \quad \text{e} \quad x_2 = 0.7666607 \times 10$$

  il prodotto risulta

$$x_1 \times x_2 = 25.71464320477000 = 0.2571464320477 \times 10^2$$

  e, quest'ultimo è, effettivamente un numero rappresentabile, essendo il suo esponente $emin \leq 2 \leq emax$, ma non esattamente, avendo un numero di cifre significative maggiore di t; applicando uno schema di arrotondamento si trova il risultato

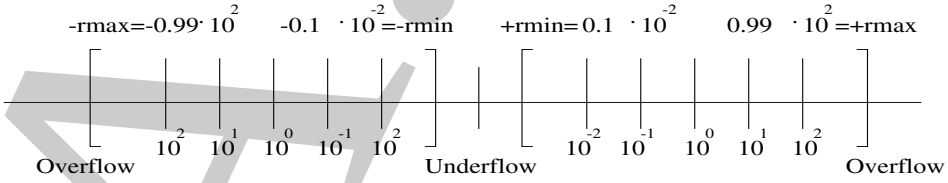
$$fl(x_1 \times x_2) = 0.2571464 \times 10^2.$$

24. *I numeri f.p. sono uniformemente distribuiti sulla retta reale* $\Re$.

- RISPOSTA [FALSO]. Essi non sono uniformemente distribuiti, ma lo sono solo tra successive potenze della base di numerazione, β, del sistema. Se, ad esempio, fissiamo un sistema aritmetico a precisione finita di base $\beta = 10$, precisione $t = 2$ ed $emin = -2$, $emax = 2$, allora tutti e soli i numeri rappresentabili in questo sistema sono quelli appartenenti all'insieme $\Im$:

$$\Im := [-0.99 \cdot 10^2, -0.1 \cdot 10^{-2}] \cup \{0\} \cup [0.1 \cdot 10^{-2}, 0.99 \cdot 10^2]$$

  Tali numeri, come si può osservare, non sono equidistanziati sull'asse reale $\Re$.



Rappresentazione grafica dell'insieme $\Im$

1.10.3 Alcuni problemi da risolvere con il calcolatore

Problema 1 Si scriva un programma in Fortran per il calcolo di un valore approssimato del numero di Nepero e, utilizzando la definizione:

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n.$$

(a) Calcolare la quantità $\left(1 + \frac{1}{n}\right)^n$ al crescere di n per $n = 10^k$ con $k = 1, 2, \ldots, 20$

(b) porre particolare attenzione (effettuando un opportuno controllo) al contributo dato alla somma di $\frac{1}{n}$ con 1;

(c) stampare, in corrispondenza di ogni approssimazione,

- l'indice del passo corrente,
- il valore stimato,
- l'errore relativo prodotto nel risultato.

confrontando, ad ogni passo, il valore dedotto con il valore e^1 ottenuto utilizzando la funzione *built-in* del Fortran exp();

(d) Dopo quanti passi conviene terminare l'esecuzione del programma? Perché?

(e) In corrispondenza di quale passo si raggiunge il minimo errore relativo?

(f) Cosa accade nei passi successivi e cosa si può dedurre sull'accuratezza del risultato?

Problema 2 Si consideri la funzione:

$$f(x) = \frac{e^x - 1}{x}$$

Utilizzando la regola de L'Hôspital si prova che:

$$\lim_{x\to 0} f(x) = 1.$$

(a) Si provi questo risultato implementando un programma in Fortran per il calcolo di f(x) per $x = 10^{-k}$, k=1, ... ,16.
I risultati trovati concordano con quelli teorici? Perché?

(b) Valutare f(x) utilizzando la formula: $f(x) = \frac{e^x - 1}{log(e^x)}$.
Questa formula dà risultati piú accurati? Perché?

Problema 3 Si supponga di generare n+1 punti equidistanziati di $h = \frac{b-a}{n}$, nell'intervallo [a,b].

(a) In un sistema aritmetico f.p. Si quale dei due algoritmi seguenti è piú accurato,

$$x_0 = a, \qquad x_k = x_{k-1} + h, \qquad k+1, \ldots, n$$

oppure

$$x_k = a + hk, \qquad k = 0, \ldots, n\,?$$

(b) Si scriva un programma che implementi entrambi gli algoritmi e si trovi un esempio, con $a = 0$ e $b = 1$, che ne illustri le differenze.

Problema 4 Si scriva un programma per il calcolo della derivata di una funzione f(x) utilizzando l'approssimazione che segue:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

Si provi il programma usando la funzione $ln(x)$, per $x = 2$.

(a) Si ripeta l'esercizio utilizzando l'approssimazione:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Problema 5 Si consideri la serie armonica:

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

(a) Provare che la serie è divergente.

(b) Spiegare perché la serie ha somma finita in aritmetica f.p.

(c) Si dica quando i termini della sommatoria non danno più contributo, nel sistema aritmetico standard IEEE, in singola ed in doppia precisione.

(d) Si scrivano due programmi per il calcolo della serie armonica uno in singola e l'altro in doppia precisione. Si controlli il processo di calcolo riportando l'indice e la somma parziale ad ogni iterazione. Quale criterio di arresto può essere utilizzato? Si confrontino i risultati con i valori trovati.

Problema 6 Si scriva un programma per il calcolo della media $\bar{x}$ e della varianza σ^2 di una successione finita x_i. $\bar{x}$ è un vettore di dimesione n, per il calcolo della varianza si utilizzino entrambe le formule:

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$\sigma^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)$$

Si confrontino i risultati per le due scelte.

1.10.4 Risposte

Di seguito verranno proposti i programmi in Fortran e le risposte ai relativi problemi illustrati nel paragrafo precedente. I risultati numerici sono stati ottenuti implementando gli algoritmi in Fortran su un calcolatore IBM RISC System/6000.

Problema 1: calcolo del numero di Nepero

```
      program calcolo numero di Nepero

C Fase di dichiarazione delle variabili
      integer k,n
      real N1,nep,e
      real err1,epsilon

  write(*,*)    '                                                    '
  write(*,*)    '  Calcolo del numero  di Nepero mediante il limite  '
  write(*,*)    '  (1) n -> inf  di  (1+1/n)**n                      '
  write(*,*)    '                                                    '
  write(*,*)    '                                                    '

C Calcolo del numero di Nepero tramite funzione interna
      n = 1
      e=exp(1.0)
C Calcolo dell'epsilon macchina
      epsilon = epsilon(1.0)
      print*, 'Epsilon macchina = ',epsilon
      k=0
      print*, 'Numero di Nepero mediante funzione interna',e
C Fase di calcolo
   5  continue
      k=k+1
      n=10**k
      N1=1.0/n
      if (N1 .ge. epsilon ) then
         N1=1.0+N1
         nep=N1**n
C Calcolo dell'errore relativo
         err1 = abs(e-nep)/abs(e)
C Stampa del valore calcolato del numero di Nepero
         print*,'######################'
         print*,'iterazione numero',k
         print*,'def(1)=',nep
         print*,'Errore relativo ',err1
         print*,'######################'
         print*
      else
         print*, '1/n e" minore dell"epsilon macchina'
         print*, 'def(1)=',nep
      endif
C Terminazione del ciclo di calcolo
      if( k.lt.21 )goto 5
      stop
      end
```

Prova di esecuzione

Calcolo del numero di Nepero mediante il limite
(1) $n \to inf$ di $(1+1/n)^n$

Epsilon macchina = 1.1920929E-07
Numero di Nepero mediante funzione interna 2.71828175

```
######################
iterazione numero 1
def(1)=  2.59374309
Errore relativo   0.0458152145
######################

######################
iterazione numero 2
def(1)=  2.704813
Errore relativo   0.00495487358
######################

######################
iterazione numero 3
def(1)=  2.71704197
Errore relativo   0.000456088339
######################

######################
iterazione numero 4
def(1)=  2.7180233
Errore relativo   9.50768808E-05
######################

######################
iterazione numero 5
def(1)=  2.72152352
Errore relativo   0.00119258335
######################

######################
iterazione numero 6
def(1)=  2.59460497
Errore relativo   0.0454981439
######################

 1/n è minore dell'epsilon macchina
def(1)=  2.59460497
```

Osservazione 1

Approssimando il numero di Nepero con la formula (1), l'algoritmo produce l'errore minimo all'iterazione numero 4 dove:

l'approssimazione vale	2.7180233
l'errore relativo	9.50768808E-05

A partire dalla quinta iterazione l'errore aumenta e quindi l' approssimazione del numero di Nepero diventa sempre meno accurata. Ciò è dovuto alla propagazione dell'errore di round-off durante l'esecuzione dell'algoritmo. Infatti, posto

$$x^{(k)} = 1 + \frac{1}{n},$$

consideriamo l'errore relativo di rappresentazione, nel dato, alla k-esima iterazione:

$$\widetilde{x}^{(k)} = x^{(k)}(1 + \delta_k) \quad \text{con} \quad |\delta_k| \leq u\,.$$

È noto (cfr. cap.1) che l'indice di condizionamento relativo della funzione $(x^{(k)})^n$ è dato da:

$$\left| \frac{z \cdot nz^{n-1}}{z^n} \right| = n$$

ponendo $z = x^{(k)}$, per cui il mal condizionamento del problema cresce con n. Posto, dunque, $n = 10^k$ calcolando la potenza n-esima l'errore relativo di rappresentazione, nella soluzione, sarà:

$$|E_{rapp}| = n \cdot |\delta_k| = 10^k \cdot |\delta_k| \quad \text{con} \quad |\delta_k| \leq u$$

Pertanto, alla quinta iterazione cioè per $k = 5$, si ottiene:

$$|E_{rapp}| = 10^5 \cdot |\delta_5| \leq 10^5 \cdot u = 10^5 \cdot 10^{-7} = 10^{-2}$$

minore della massima accuratezza relativa, in un sistema aritmetico a precisione finita con base $\beta = 10$ e precisione $t = 8$, per cui $u = \beta^{1-t} = 10^{-7}$. Quanto detto, è confermato dal risultato della sperimentazione numerica, che per $k = 5$ produce:

valore calcolato	$2.72152352 = 0.27215235 \cdot 10$
errore relativo	$0.00119258335 = 0.11925834 \cdot 10^{-2}$

ovvero quest'ultimo è dello stesso ordine di grandezza del massimo errore di rappresentazione.

L'algoritmo descritto è pertanto **instabile**!

Osservazione 2

Dall'iterazione numero 7 in poi, si ottengono i risultati seguenti:

valore calcolato	2.59460497
errore relativo	0.0454981439

cioè il valore dell'approssimazione non varia in quanto $1/n$, non dà più contributo alla somma con 1, perché $1/n$ risulta minore dell'epsilon macchina.

Problema 2: approssimazione di un limite

```
      program  approssimazione di un limite

C Fase  di dichiarazione delle variabili
      integer k,n
      real N1,nep,nep1,aus,e,err1,err2
C Stampa a video del problema da risolvere

  write(*,*)    '                                                  '
  write(*,*)    '  ------------------------------------------------'
  write(*,*)    '   Calcolo del limite                             '
  write(*,*)    '   (1) x -> 0  di   (EXP(x)-1)/x                  '
  write(*,*)    '   (2) x -> 0  di   (EXP(x)-1)/log(EXP(x))        '
  write(*,*)    '                                                  '
  write(*,*)    '  ------------------------------------------------'

C Fase di  calcolo
      k = 0
   5  continue
      k=k+1
      n=10**(k)
      N1=1.0/n
      aus=exp(N1)
      nep1=aus-1
      if(nep1 .ne. 0.0) then
        e=nep1/(log(aus))
        aus=nep1/N1
        err1 = abs(1-aus)
        err2 = abs(1-e)
C Stampa a video della soluzione calcolata
       print*,'k  =',k
       print*,'limite  (1)=',aus,'  errore assoluto = ',err1
       print*,'limite  (2)=',e,'  errore assoluto = ',err2
       print*,'####################'
      else
C Risultato dovuto alla precisione finita
       print*,'k  =',k
       print*,'limite  (1) = limite (2) = 0'
       print*,'####################'
      endif
C Terminazione della fase di calcolo
      if(k .lt. 16)goto 5
      stop
      end
```

Prova di esecuzione

```
Calcolo del limite
(1) x → 0  di   (EXP(x)-1)/x
(2) x → 0  di   (EXP(x)-1)/log(EXP(x))

k  = 1
limite  (1)=  1.05170965  errore assoluto =   0.0517096519
limite  (2)=  1.05170918  errore assoluto =   0.0517091751
######################
k  = 2
limite  (1)=  1.00501776  errore assoluto =   0.00501775742
limite  (2)=  1.0050168  errore assoluto =   0.00501680374
######################
k  = 3
limite  (1)=  1.00052357  errore assoluto =   0.0005235672
limite  (2)=  1.0005002  errore assoluto =   0.000500202179
######################
k  = 4
limite  (1)=  1.00016594  errore assoluto =   0.000165939331
limite  (2)=  1.00004995  errore assoluto =   4.99486923E-05
######################
k  = 5
limite  (1)=  1.00135803  errore assoluto =   0.00135803223
limite  (2)=  1.00000501  errore assoluto =   5.00679016E-06
######################
k  = 6
limite  (1)=  0.953674316  errore assoluto =   0.0463256836
limite  (2)=  1.00000048  errore assoluto =   4.76837158E-07
######################
k  = 7
limite  (1)=  1.1920929  errore assoluto =   0.192092896
limite  (2)=  1.00000012  errore assoluto =   0.0
######################
k  = 8
limite  (1) = limite (2) = 0
######################
```

Osservazione 1

L'errore minimo che si commette approssimando il limite assegnato mediante la formula (1), si ottiene all'iterazione numero 4:

valore calcolato del limite	1.00016594
errore assoluto	0.000165939331

ottenendo un'accuratezza fino alla quarta cifra decimale.
Al crescere del numero di iterazioni il risultato diventa sempre meno accurato, a causa della propagazione dell'errore di round-off. Inoltre, dall'ottava iterazione in poi il risultato è del tutto **errato**, infatti l'algoritmo produce:

$$\frac{e^x - 1}{x} \overset{x \to 0}{\to} 0$$

Analizziamo nel dettaglio cosa avviene.
In un sistema aritmetico a precisione finita, due numeri y e t coincidono quando:

$$y \equiv t \Longleftrightarrow y = t(1+\delta) \ , \ |\delta| < \epsilon_y$$

dove $\epsilon_y = \epsilon \cdot |y|$, e ϵ è l'epsilon macchina.
Nel sistema aritmetico IEEE, utilizzando la singola precisione, il valore di e^x con $x = 10^{-8}$, calcolato mediante la routine *built-in* del FORTRAN restituisce il valore 1:

$$e^{10^{-8}} \equiv 1$$

quindi:

$$e^{10^{-8}} - 1 = 1 - 1 = 0$$

pertanto, a partire dall'iterazione $k = 8$, il valore calcolato del numeratore è 0.

Osservazione 2
La formula (2) fornisce un risultato più accurato della formula (1) fino all'iterazione numero 7. Inoltre, come sottolineato nell'Osservazione 1, all'aumentare del numero di iterazioni, il numeratore risulta nullo, per cui, per $k \geq 8$, la formula (1) restituisce valore zero, essendo non nullo il denominatore, x, mentre la (2) restituisce un NaN, che denota la presenza di una situazione eccezionale, dovuta al tentativo di eseguire la divisione 0/0. In effetti, a partire dall'iterazione 8, anche il denominatore $\log(\exp(x))$ risulta nullo. Fino all'iterazione 7, inoltre, pur essendo uguale, per entrambe le formule, il valore assunto dal numeratore e, a meno dell'errore di round off, anche quello assunto dal denominatore, le due formule forniscono, al crescere di k (o, equivalentemente al tendere di x a zero), il valore del limite con un'accuratezza diversa e questo

è dovuto esclusivamente all'operazione di divisione tra due numeri reali floating point. [56]

Supponiamo che e^x e $logx$ vengano calcolati con un errore relativo minore della precisione della macchina u. Vogliamo calcolare la seguente quantità:

$$f(x) = \frac{e^x - 1}{loge^x}$$

indicando con $\tilde{y} = e^x(1+\delta)$ con $|\delta| < u$, il valore calcolato di e^x, ed essendo $fl(log\tilde{y}) = x(1+\eta)$ con $|\eta| < u$, si ha:

$$\begin{aligned}
\tilde{f}(x) &= fl\left(\frac{fl(\tilde{y}-1)}{fl(log\tilde{y})}\right) = fl\left(\frac{(\tilde{y}-1)(1+\epsilon_1)}{x(1+\eta)}\right) = \\
&= \frac{(\tilde{y}-1)(1+\epsilon_1)}{x(1+\eta)}(1+\epsilon_2) = \\
&= \frac{(e^x(1+\delta)-1)}{x}(1+\epsilon_1)(1+\eta)^{-1}(1+\epsilon_2) \approx \\
&\approx \frac{e^x-1}{x}(1+\epsilon_1)(1+\eta)^{-1}(1+\epsilon_2) \simeq f(x)(1+\theta_3)
\end{aligned}$$

con $|\epsilon_i| < u$ e $|\theta_3| \leq \frac{3u}{1-3u} = \mathcal{O}(u)$. Segue che l'errore relativo che si commette nell'approssimare $f(x)$ con $\tilde{f}(x)$ è:

$$\frac{|f(x) - \tilde{f}(x)|}{|f(x)|} \approx \mathcal{O}(u).$$

[56] In effetti, implementando le funzioni *built-in* del FORTRAN $\log(\cdot)$ e $\exp(\cdot)$, in singola precisione, è possibile osservare che la funzione $\log(\exp(x))$, che rappresenta il denominatore della formula (2), e la x, denominatore della formula (1), forniscono, praticamente, lo stesso numero macchina, a meno dell'errore di round-off.

Problema 3: generazione di punti equidistanziati

```
      program  punti equidistanziati

C Fase  di dichiarazione delle variabili
      integer k,n,i,j
      real  a,b,h,x(0:100000),y(0:100000)
C Stampa a video del problema da risolvere

  write(*,*)    '                                                    '
  write(*,*)    '  Formule che generano i punti                      '
  write(*,*)    '  (1)    x0 = a   x(k) = x(k - 1) + h   k = 1, ... , n '
  write(*,*)    '  (2)    x(k) = a + h*k   k = 0, ... , n            '
  write(*,*)    '                                                    '
  write(*,*)    '                                                    '

C Fase di  calcolo
      print*,'Inserire gli estremi a,b'
      read*,a,b
      print*,a,b
      print*,'Inserire il numero dei punti desiderati'
      read*,n
      print*,n

      x(0)=a
      y(0)=a
      h=(b-a)/n

C Generazione dei numeri mediante le due formule

      do 10 k=1,n
      x(k)=x(k-1)+h
      y(k)=y(0)+h*k
10     continue

C Stampa a video della soluzione calcolata

       do 20 k=0,n
       print*,'Punto ',k
       print*,'Punti con la formula  (1)='
       write(6,*)x(k)
       print*,'Punti con la formula  (2)='
       write(6,*)y(k)
       print*,'############################'
20     continue
      stop
      end
```

Prova di esecuzione

Formule che generano i punti
(1) $x0 = a \quad x(k) = x(k-1) + h \quad k = 1, \ldots, n$
(2) $x(k) = a + h{*}k \quad k = 0, \ldots, n$

```
Inserire gli estremi a,b
 0.  1.
Inserire il numero dei punti desiderati
1001
Punto  0
Punti con la formula  (1)=  0.
Punti con la formula  (2)=  0.
############################
Punto  1
Punti con la formula  (1)=  0.000999000971
Punti con la formula  (2)=  0.000999000971
############################
Punto  2
Punti con la formula  (1)=  0.00199800194
Punti con la formula  (2)=  0.00199800194
############################
Punto  3
Punti con la formula  (1)=  0.00299700303
Punti con la formula  (2)=  0.00299700303
############################
Punto  4
Punti con la formula  (1)=  0.00399600388
Punti con la formula  (2)=  0.00399600388
############################
Punto  5
Punti con la formula  (1)=  0.00499500474
Punti con la formula  (2)=  0.00499500474
############################
```

```
⋮
############################
Punto  996
Punti con la formula  (1)=  0.994992673
Punti con la formula  (2)=  0.995004952
############################
Punto  997
Punti con la formula  (1)=  0.995991647
Punti con la formula  (2)=  0.996003985
############################
Punto  998
Punti con la formula  (1)=  0.996990621
Punti con la formula  (2)=  0.997002959
############################
Punto  999
Punti con la formula  (1)=  0.997989595
Punti con la formula  (2)=  0.998001993
############################
Punto  1000
Punti con la formula  (1)=  0.998988569
Punti con la formula  (2)=  0.999000967
############################
Punto  1001
Punti con la formula  (1)=  0.999987543
Punti con la formula  (2)=  1.
############################
```

Osservazione 1

Supponiamo di generare 1001 punti nell'intervallo [0,1]. Per semplicità e brevità di scrittura riportiamo solo i primi cinque e gli ultimi cinque elmenti da generare:

(0 0.001 0.002 0.003 0.004 ... 0.996 0.997 0.998 0.999 1).

Sempre per brevità di scrittura, abbiamo riportato i risultati relativi alle due formule solo delle prime cinque iterazioni dell'algoritmo.

Come si può osservare dalla prova di esecuzione, la formula (2) è più accurata della (1). Effettuiamo k passi dell'algoritmo utilizzando la formula (1) e facciamo un'analisi della propagazione dell'errore di round-off, indichiamo con ϵ_a e ϵ_h, l'errore che si commette nel rappresentare rispettivamente a ed h in un sistema aritmetico a precisione finita, mentre con δ_{S_k} l'errore che si commette eseguendo la somma al passo $k-esimo$. Si ha:

$$\tilde{x}_0 = fl(a) = a(1+\epsilon_a)$$
$$\tilde{x}_1 = fl(\tilde{x}_0 + \tilde{h}) = (a(1+\epsilon_a) + h(1+\epsilon_h))(1+\delta_{S_1})$$
$$\vdots$$
$$\tilde{x}_k = fl(\tilde{x}_{k-1} + \tilde{h}) = (\tilde{x}_{k-1} + h(1+\epsilon_h))(1+\delta_{S_k}) =$$
$$= a(1+\epsilon_a+\delta_{S_1}+\cdots+\delta_{S_k}) + h(1+\epsilon_h+\delta_{S_1}+\cdots+\delta_{S_k})$$

$$\text{con } |\epsilon_a| \leq u \, |\epsilon_h| \leq u \, |\delta_{S_k}| \leq u \; k = 1, .., n$$

Si noti che al passo $k-esimo$ su a e su h si accumulano tutti gli errori di round-off δ_{S_k} dovuti alle somme effettuate nei $k-1$ passi precedenti. Eseguiamo gli stessi k passi, utilizzando invece la formula (2) e effettuiamo l'analisi della propagazione dell'errore di round-off:

$$\tilde{x}_0 = fl(a) = a + \epsilon_0$$
$$\tilde{x}_k = fl(\tilde{a} + \tilde{h} \cdot \tilde{k}) = (a(1+\epsilon_0) + (h(1+\epsilon_h) \cdot k(1+\epsilon_k))(1+\delta_P))(1+\delta_{S_k})$$

abbiamo indicato con ϵ_a, ϵ_h, ϵ_k gli errori che si commettono nel rappresentare rispettivamente a, h e k in un sistema aritmetico a precisione finita, mentre con δ_P l'errore che si commette eseguendo il prodotto $h \cdot k$ e con δ_{S_k} l'errore che si commette eseguendo la somma al passo $k-esimo$. Si osserva facilmente che utilizzando la formula (2), al generico passo k sono presenti sia δ_P che δ_{S_k}, ma non si ha l'accumulo degli errori δ_{S_i} $i = 1, \cdots, k-1$ delle somme effettuate nei $k-1$ passi precedenti, che si verifica invece utilizzando la formula (1).

Problema 4: calcolo della derivata

```
      program  calcolo della derivata

C Fase  di dichiarazione delle variabili
      integer n,i,j
      real  x,h,d1,d2,k
C Stampa a video del problema da risolvere

  write(*,*)    '                                                    '
  write(*,*)    '  Formule  per il calcolo della derivata            '
  write(*,*)    '  (1) (f(x+h) - f(x))/h                             '
  write(*,*)    '  (2) (f(x+h) - f(x-h))/2h                          '
  write(*,*)    '                                                    '
  write(*,*)    '                                                    '

C Fase di  calcolo
      print*,'Inserire il numero di passi ed il punto di valutazione'
      read*,n,x
      print*,n,x
      k=2.
10     continue
            h=1./k
            call der(x,h,d1,d2)
            print*,'incremento h =',h
            print*,'Derivata con la formula  (1)=',d1
            print*,'Derivata con la formula  (2)=',d2
            print*,'############################'
            k=2*k
      if(k.le.n) goto 10
      stop
      end

C Routine per il calcolo della derivata della funzione logaritmo
      subroutine der(x,h,d1,d2)
      real x,h,d1,d2
      d1=(log(x+h)-log(x))/h
      d2=(log(x+h)-log(x-h))/(2*h)
      return
      end
```

Prova di esecuzione

Formule per il calcolo della derivata
(1) $(f(x+h)-f(x))/h$
(2) $(f(x+h)-f(x-h))/2h$

```
Inserire il numero di passi ed il punto di valutazione
10000  2.
incremento h =  0.5
Derivata con la formula  (1)=  0.446287155
Derivata con la formula  (2)=  0.510825634
##########################################
incremento h =  0.25
Derivata con la formula  (1)=  0.47113204
Derivata con la formula  (2)=  0.502628803
##########################################
incremento h =  0.125
Derivata con la formula  (1)=  0.484996796
Derivata con la formula  (2)=  0.500652552
##########################################
incremento h =  0.0625
Derivata con la formula  (1)=  0.492346764
Derivata con la formula  (2)=  0.500163078
##########################################
incremento h =  0.03125
Derivata con la formula  (1)=  0.496133804
Derivata con la formula  (2)=  0.500041008
##########################################
incremento h =  0.015625
Derivata con la formula  (1)=  0.498058319
Derivata con la formula  (2)=  0.500011444
##########################################
incremento h =  0.0078125
Derivata con la formula  (1)=  0.499511719
Derivata con la formula  (2)=  0.5
##########################################
```

```
##################################
incremento h =  0.00390625
Derivata con la formula  (1)=  0.499023438
Derivata con la formula  (2)=  0.5
##################################
incremento h =  0.001953125
Derivata con la formula  (1)=  0.499755859
Derivata con la formula  (2)=  0.5
##################################
incremento h =  0.0009765625
Derivata con la formula  (1)=  0.49987793
Derivata con la formula  (2)=  0.5
##################################
incremento h =  0.00048828125
Derivata con la formula  (1)=  0.49987793
Derivata con la formula  (2)=  0.5
##################################
incremento h =  0.000244140625
Derivata con la formula  (1)=  0.5
Derivata con la formula  (2)=  0.5
##################################
incremento h =  0.000122070312
Derivata con la formula  (1)=  0.5
Derivata con la formula  (2)=  0.5
##################################
```

Osservazioni

Sia la formula (1) che la formula (2), costituiscono un'approssimazione della derivata. Per entrambe è possibile stimare il massimo errore di discretizzazione che si commette. La formula di Taylor al secondo ordine, di punto iniziale $\xi \in (x, x+h)$, è:

$$f(x) = f(\xi) + f'(\xi)(x-\xi) + f''(\xi)\frac{(x-\xi)^2}{2!} + O((x-\xi)^3)$$

sottraendo ad ambo i membri $f(\xi)$ e $f'(\xi)(x-\xi)$ otteniamo:

$$f(x) - f(\xi) - f'(\xi)(x-\xi) = f''(\xi)\frac{(x-\xi)^2}{2!} + O((x-\xi)^3)$$

Trascurando gli infinitesimi di ordine superiore al secondo e dividendo ambo i membri per $(x-\xi)$:

$$\frac{f(x) - f(\xi)}{(x-\xi)} - f'(\xi) = f''(\xi)\frac{(x-\xi)}{2}$$

passando ai valori assoluti:

$$\left|\frac{f(x)-f(\xi)}{(x-\xi)}-f'(\xi)\right| = \left|f''(\xi)\frac{(x-\xi)}{2}\right|$$

Maggioriamo il secondo membro, sostituendo a $(x-\xi)$ l'ampiezza h dell'intervallo $(x, x+h)$, otteniamo:

$$\left|\frac{f(x)-f(\xi)}{(x-\xi)}-f'(\xi)\right| \quad \leq \quad \left|f''(\xi)\frac{h}{2}\right|$$

Al primo membro abbiamo l'errore assoluto che si commette nell'approssimare la derivata prima $f'(x)$ con la formula (1), nota come formula della *differenza in avanti su due punti*, pertanto:

$$E \leq \left|\frac{h\cdot f''(\xi)}{2}\right|, \qquad x<\xi<x+h$$

La formula (2) viene dal calcolo della formula della *differenza centrale*. La formula di Taylor al terzo ordine, di punto iniziale x è:

$$f(x\pm h) = f(x)\pm f'(x)h + f''(x)\frac{h^2}{2!}\pm f'''(x)\frac{h^3}{3!}+O(h^4) \tag{1.55}$$

da cui si ha la formula cosiddetta della *differenza in avanti*:

$$\frac{f(x+h)-f(x)}{h} \simeq f'(x)+f''(x)\frac{h}{2!}+f'''(x)\frac{h^2}{3!}+O(h^3)$$

e quella della cosiddetta della *differenza all'indietro*:

$$\frac{f(x)-f(x-h)}{h} \simeq f'(x)-f''(x)\frac{h}{2!}+f'''(x)\frac{h^2}{3!}-O(h^3)$$

Sommando membro a membro e dividendo per 2 le espressioni in avanti ed all'indietro, trascurando gli infinitesimi di ordine superiore al terzo, si ottiene:

$$\frac{f(x+h)-f(x-h)}{2h} \cong f'(x)+f'''(x)\frac{h^2}{6}$$

Sottraendo $f(x)$ ad ambo i membri e passando ai valori assoluti si ha:

$$\left|\frac{f(x+h)-f(x-h)}{2h}-f'(x)\right| \cong \left|f'''(x)\frac{h^2}{6}\right|.$$

Al primo membro abbiamo l'errore assoluto che si commette nell'approssimare la derivata prima $f'(x)$ con la formula (2), nota come *formula della differenza centrale*, al secondo una sua stima.

In generale le formule per approssimare le derivate che coinvolgono più punti sono più accurate.

La funzione test utilizzata per approssimare il calcolo della derivata è $f(x) = ln(x)$, la cui derivata è $f'(x) = \frac{1}{x}$; nel punto $x = 2$ risulta $f'(2) = \frac{1}{2} = 0.5$. Riportiamo per ogni incremento h l'errore relativo che si commette utilizzando la formula (1) o la (2):

h	$\frac{f(x+h)-f(x)}{h}$	Err. relativo	$\frac{f(x+h)-f(x-h)}{2h}$	Err. relativo
0.5	0.446287155	0.1074×10^{0}	0.510825634	0.2170×10^{-1}
0.25	0.47113204	0.577×10^{-1}	0.502628803	0.5257×10^{-2}
0.125	0.484996796	0.3006×10^{-1}	0.500652552	0.1305×10^{-2}
0.0625	0.492346764	0.1531×10^{-1}	0.500163078	0.3260×10^{-3}
0.03125	0.496133804	0.7732×10^{-2}	0.500041008	0.8200×10^{-4}
0.015625	0.498058319	0.3883×10^{-2}	0.500011444	0.228×10^{-4}
0.0078125	0.499023438	0.1953×10^{-2}	0.5	0.
0.00390625	0.499511719	0.9766×10^{-3}	0.5	0.
0.001953125	0.499755859	0.4884×10^{-3}	0.5	0.
0.0009765625	0.49987793	0.2442×10^{-3}	0.5	0.
0.00048828125	0.49987793	0.2442×10^{-3}	0.5	0.
0.000244140625	0.5	0.	0.5	0.
0.0001220703125	0.5	0.	0.5	0.

Come si può notare la formula (2) è più accurata della (1) e converge più velocemente.

Problema 5: calcolo della Serie Armonica

```
      program calcolo Serie Armonica

C Fase di dichiarazione delle variabili
      integer i,flag
      real e,epsilon,Nmax
      real add,Sum,Sumold,n
C Stampa a video del problema

  write(*,*)    '                                        '
  write(*,*)    '   Calcolo della Serie Armonica         '
  write(*,*)    '   n -> inf  di  S(1/n)                 '
  write(*,*)    '                                        '
  write(*,*)    '                                        '

      write(*,*)'Inserire massimo numero di iterazioni'
      read(*,*)Nmax
      write(*,*)Nmax
C  Inizializzazione dell'addendo n-esimo
      n = 1.0
      add = n
      i = 0
      flag = 0
C  Inizializzazione della Somma
      Sum = 1.0
C Calcolo dell'epsilon macchina
      epsilon = eps()
      print*, 'Epsilon macchina = ',epsilon
C Fase di Calcolo
   5  continue
      i = i + 1
      n= n + 1.0
      add=1.0/n
      if (add .ge. Sum*epsilon ) then
         Sumold = Sum
         Sum = Sum + add
```

```
C Stampa del valore calcolato della Serie
          print*,'######################'
          print*,'Approssimazione della serie =',Sum
          print*,'Iterazioni eseguite = ',i
          print*,'######################'
       else
          print*, ' 1/n è minore di (S_n-1)*epsilon macchina'
          print*,'Approssimazione della serie =',Sum
          flag = 1
       endif
C Terminazione del ciclo di calcolo
       if( i .le. Nmax .and. flag .eq. 0 ) goto 5
       print*,'Iterazioni eseguite = ',i
       stop
       end
```

Prima di descrivere una prova di esecuzione dell'algoritmo descritto, si riportano le risposte ai quesiti proposti nel paragrafo 1.10.3:

(a) La serie armonica:

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

è una serie a termini positivi, pertanto la successione delle somme parziali $\{s_n\}$, con $s_n = \sum_{k=1}^{n} \frac{1}{k}$ è strettamente crescente, e quindi la serie diverge positivamente:

$$\lim_{n \to \infty} s_n = +\infty$$

(b) In un sistema aritmetico a precisione finita la serie ha somma finita in quanto:

$$s_{k+1} = s_k + \frac{1}{k+1} \neq s_k \iff \frac{1}{k+1} \geq s_k \cdot \epsilon_{mac}$$

dove ϵ_{mac} è l'epsilon macchina.

(c) Per il punto (b), in un sistema aritmetico a precisione finita, la somma non cambia all'iterazione $k - esima$, per cui si verifica la disuguaglianza:

$$\frac{1}{k+1} < s_k \cdot \epsilon_{mac}$$

e per ogni $n \geq k$ risulta:

$$s_n = s_k$$

(d) Poiché la serie è divergente, il calcolo della serie può essere arrestato o quando k raggiunge un massimo numero di iterazioni assegnato o quando k raggiunge il valore che verifica il punto (c).

Riportiamo di seguito alcune prove di esecuzione dell'algoritmo che calcola il valore della serie armonica in precisione finita. Per semplicità di scrittura, dato l'elevato numero di iterazioni eseguite dall'algoritmo, vengono riportate le somme parziali relative alle ultime 4 iterazioni eseguite.

Prova di esecuzione

```
Calcolo della Serie Armonica
n → inf  di  S(1/n)

Inserire massimo numero di iterazioni
1000000
Epsilon macchina =1.192029E-07


######################
Approssimazione della serie =  13.8890009
Iterazioni eseguite =  603970
######################


######################
Approssimazione della serie =  13.8890028
Iterazioni eseguite =  603971
######################

######################
Approssimazione della serie =  13.8890047
Iterazioni eseguite =  603972
######################

######################
Approssimazione della serie =  13.8890066
Iterazioni eseguite =  603973
######################

 1/n e' minore di (S_n-1) * epsilon macchina
Approssimazione della serie =  13.8890066
Iterazioni eseguite =  603974
```

Osservazione
L'algoritmo si arresta all'iterazione numero $k+1 = 603974+1 = 603975$, cioè quando risulta:

$$\frac{1}{k+1} < s_k \cdot \epsilon_{mac}$$

Quindi il punto (c) si realizza all'iterazione numero 603975:

Approssimazione della serie	13.8890066
Errore relativo	1.90734863E-06

Riportiamo di seguito lo stesso algoritmo per il calcolo della serie armonica, sviluppato in doppia precisione.

```
      program calcolo Serie Armonica

C Fase di dichiarazione delle variabili
      integer i,flag
      double precision e,epsilon,Nmax
      double precision add,Sum,Sumold,n


  write(*,*)     '                                        '
  write(*,*)     '   Calcolo della Serie Armonica         '
  write(*,*)     '   n -> inf   di  S(1/n)                '
  write(*,*)     '                                        '
  write(*,*)     '                                        '

      write(*,*)'Inserire massimo numero di iterazioni'
      read(*,*)Nmax
      write(*,*)Namx

C  Inizializzazione dell'addendo n-esimo
      n = 1.0
      add = n
      i = 0
      flag = 0

C  Inizializzazione della Somma
      Sum = 1.0

C Calcolo dell'epsilon macchina precisone doppia
      epsilon = epsdouble()
      print*, 'Epsilon macchina = ',epsilon

C Fase di calcolo
   5  continue
      i = i + 1
      n= n + 1.0
      add=1.0/n
      if (add .ge. Sum*epsilon ) then
         Sumold = Sum
         Sum = Sum + add

C Calcolo dell'errore assoluto
         err1 = abs(Sum - Sumold)
```

```
C Stampa del valore calcolato della Serie
          print*,'#####################'
          print*,'Approssimazione della serie =',Sum
          print*,'Iterazioni eseguite ',i
          print*,'#####################'
          print*
       else
          print*, ' 1/n e" minore di S_(n-1)*epsilon macchina'
          print*,'Approssimazione della serie =',Sum
          flag = 1
       endif

C Terminazione del ciclo di calcolo
       if( i .le. Nmax .and. flag .eq. 0 ) goto 5
       print*,'Iterazioni eseguite = ',i
       stop
       end
```

Prova di esecuzione

Si descrive, di seguito, un esempio di esecuzione di cui si riportano, per semplicità, dato l'elevato numero di iterazioni eseguite, solo i risultati relativi alle ultime cinque iterazioni.

Calcolo della Serie Armonica
$n \to inf$ di S(1/n)

Inserire massimo numero di iterazioni
10000000
Epsilon macchina = 2.22044605E-16

####################
Approssimazione della serie = 16.6953111
Iterazioni eseguite 9999996
####################

```
######################
Approssimazione della serie =  16.6953112
Iterazioni eseguite   9999996
######################

######################
Approssimazione della serie =  16.6953113
Iterazioni eseguite   9999997
######################

######################
Approssimazione della serie =  16.6953114
Iterazioni eseguite   9999998
######################

######################
Approssimazione della serie =  16.6953115
Iterazioni eseguite   9999999
######################

Iterazioni eseguite =  10000000
```

Osservazione

Eseguendo i calcoli in doppia precisione l'algoritmo si arresta all'iterazione numero 10000000, fornendo come risultato: 16.6953115. Anche in precisione doppia valgono le stesse considerazioni fatte per i risultati ottenuti dall'algoritmo in singola precisione. In questo caso l'algoritmo si arresta perché è stato raggiunto il massimo numero di iterazioni.

Problema 6: calcolo della Varianza

```
      program  calcolo della Varianza

C Fase  di dichiarazione delle variabili
      integer n1,k
      real  x(100),sigma1,sigma,med,n

C Stampa a video del problema da risolvere

  write(*,*)    '  ------------------------------------------------  '
  write(*,*)    '  Calcolo della Varianza                            '
  write(*,*)    '  (1) 1/(n-1) [sum_{i=1,n} (x(i)-med)^2]            '
  write(*,*)    '  (2) 1/(n-1) [sum_{i=1,n} x(i)^2 +                 '
  write(*,*)    '                    -1/n (sum_{i=1,n} x(i))^2]     '
  write(*,*)    '  ------------------------------------------------  '

C Fase di  calcolo
      print*,'Inserire la lunghezza del vettore'
      read*,n1
      print*,n1
      print*,'Inserire le componenti del vettore'
      do 10 i=1,n1
        read*,x(i)
        print*,x(i)
10     continue
      med=0
      do 20 k=1,n1
         med=med+x(k)
20     continue
      n=n1
      med=med/n
      sigma=0
      sigma1=0
      do 30 k=1,n
        sigma= sigma+((x(k)-med)**2)
        sigma1=sigma1+(x(k)**2)
30     continue
      sigma= (1./(n-1))*sigma
      sigma1=(1./(n-1))*(sigma1-n*(med**2))
C Stampa a video dei risultati
      print*,'Media =',med
      print*,'Varianza con la formula  (1)=',sigma
      print*,'Varianza con la formula  (2)=',sigma1
      print*,'============================='
      stop
      end
```

Prova di esecuzione

```
Calcolo della Varianza
(1) 1/(n-1) [sum_{i=1,n} (x(i)-med)^2]
(2) 1/(n-1) [sum_{i=1,n} x(i)^2 +
                -1/n (sum_{i=1,n} x(i))^2]

Inserire la lunghezza del vettore
5
Inserire le componenti del vettore
 2.
 4.
 7.
 8.
 10.
Media =  6.19999981
Varianza con la formula  (1)=  10.1999998
Varianza con la formula  (2)=  10.2000008
==================================

Calcolo della Varianza
(1) 1/(n-1) [sum_{i=1,n} (x(i)-med)^2]
(2) 1/(n-1) [sum_{i=1,n} x(i)^2 +
                -1/n (sum_{i=1,n} x(i))^2]

Inserire la lunghezza del vettore
3
Inserire le componenti del vettore
 10000.
 10001.
 10002.
Media =  10001.
Varianza con la formula  (1)=  1.
Varianza con la formula  (2)=  0.
==================================
```

Osservazioni

Le formule utilizzate dall'algoritmo sono:

$$(1)\ \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

$$(2)\ \sigma^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)$$

analizziamo le due prove di esecuzione separatamente.
Nella prima prova il vettore è $\overline{x} = (2, 4, 7, 8, 10)$. In aritmetica a precisione infinita $\overline{x}$ ha media $m = 6.2$ e varianza $\sigma^2 = 10.2$. L'errore relativo che si commette utilizzando la formula (1) è:

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|10.2 - 10.1999998|}{10.2} = 0.2 \times 10^{-8}$$

utilizzando invece la seconda formula

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|10.2 - 10.2000008|}{10.2} = 0.7 \times 10^{-8}$$

in entrambi i casi il risultato è accettabile, si ottiene un' accuratezza fino alla settima cifra significativa.
Analizziamo ora la seconda prova. Il vettore è $\overline{x} = (10000,\ 10001,\ 10002)$. In aritmetica a precisione infinita $\overline{x}$ ha media $m = 10001$ e varianza $\sigma^2 = 1$. L'errore relativo che si commette utilizzando la formula (1) è:

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|1.0 - 1.0|}{1.0} = 0$$

utilizzando invece la seconda formula

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|1.0 - 0.0|}{1.0} = 0.1 \times 10^{1}$$

Questo secondo esempio, mette in luce il fatto che in aritmetica a precisione finita la formula (2) può generare effetti di cancellazione distruttiva e condurre ad un risultato del tutto scorretto.

1.11 Due disastri causati dall'aritmetica a precisione finita del calcolatore

1.11.1 Quando un *piccolo* errore può essere fatale

Di seguito vengono riportati due episodi recentemente accaduti. Lo scopo è quello di dimostrare che lavorando in un sistema aritmetico a precisione finita, la facilità di commettere errori, sia pur *piccoli*, può compromettere un risultato, le cui conseguenze non si limitano purtroppo, al solo errore numerico.

1.11.2 *Un missile Patriot manca il bersaglio*

Il 25 Febbraio 1991, durante la guerra del golfo, una batteria americana di missili *Patriot* a Dharan, Arabia Saudita, non riuscì ad intercettare un missile Scud iracheno. Lo Scud colpì un acquartieramento dell'esercito americano uccidendo 28 soldati. Un rapporto del GAO-IMTEC-92-26, l'ufficio generale di contabilità, dal titolo "*Missile Difensivo Patriot: un problema software ha condotto ad un malfunzionamento del sistema a Dharan, Arabia Saudita*", segnalò la causa del fallimento. La causa era da attribuirsi ad un calcolo inesatto del tempo di *boot* di caricamento del sistema, inesattezza dovuta alla precisione finita del sistema aritmetico del calcolatore. Tale tempo, misurato dall'orologio interno in decimi di secondo, era moltiplicato per 1/10 per produrre il tempo in secondi. La rappresentazione binaria di 1/10 ha infinite cifre, indichiamola con x:

$$x = \frac{1}{10} = (0.1)_{10} = (0.0001100110011...)_2 = (0.0\overline{0011})_2$$

I registri utilizzati avevano una lunghezza di 24 bit, utilizzando un bit per il segno la rappresentazione di 1/10 fu troncata alla ventitreesima cifra dopo la virgola. Indichiamo la rappresentazione di 1/10 nei registri con $\tilde{x}$:

$$\tilde{x} = 0.00011001100110011001100$$

L'errore introdotto dal troncamento è:

$$\begin{aligned} E &= |x - \tilde{x}| = (0.0000000000000000000000110011...)_2 \\ &\approx (0.000000095)_{10} = (0.95 \times 10^{-7})_{10} \end{aligned}$$

Questo *piccolo* errore di troncamento, moltiplicato per il tempo in decimi di secondo in cui era stata attiva la batteria dei *Patriot*, e cioé circa 100 ore uguali a $100 \times 60 \times 60 \times 10 = 360000$ decimi di secondo, produce un errore sul tempo risultante di circa 0.34 secondi:

$$0.000000095 \times 100 \times 60 \times 60 \times 10 = 0.342$$

Moltiplicando questo tempo, per la velocità con cui viaggia uno *Scud*, e cioé circa 1676 metri al secondo, otteniamo:

$$\text{Spazio} = \text{velocità} \times \text{tempo} = 1676\, m/s \times 0.34\, s = 569.84\, m$$

cioè uno *Scud* che viaggia a 1676 metri al secondo, percorre in 0.34 secondi uno spazio di circa mezzo chilometro in avanti nella sua direzione, e questo spazio fu sufficiente per uscire fuori dalla zona individuata dal *Patriot*.

1.11.3 *Esplosione dell'Ariane 5*

Il 4 Giugno 1996 l'Ente Spaziale Europeo lanciò nello spazio un razzo senza equipaggio, l'*Ariane 5*.

Figura 1.8: L'Ariane 5

Lo sviluppo di Ariane 5 venne iniziato nel 1984 con l'idea, non realizzata, che dovesse servire a immettere in orbita la navetta europea Hermes e per questo motivo aveva i due booster laterali sovradimensionati rispetto al normale carico di satelliti commerciali. Il suo primo lancio (Ariane V88/501), il 4 giugno del 1996, terminò dopo circa 39 secondi dal decollo, quando il razzo raggiunse un'altezza di 2.5 *miglia;* un meccanismo di autodistruzione *pose fine all'* Ariane 5*, insieme con la perdita di* 4 *costosi e non assicurati satelliti scientifici Cluster dell'ESA che si trovavano a bordo. Il progetto era durato circa 10 anni e costato 7 miliardi di dollari. A tale somma vanno aggiunti il costo di costruzione del razzo e quello del carico, che furono stimati con un valore complessivo di 500 milioni di dollari. Dopo un controllo completo è tornato in sevizio il 30 ottobre 1997 e da quel momento ha effettuato lanci senza intoppi. Visto la sua enorme potenza di solito viene usato per mettere in orbita due satelliti per volta e dal 2004 lancerà verso la ISS-Alpha la capsula di rifiorimento automatica ATV* [57].

[57] [9]

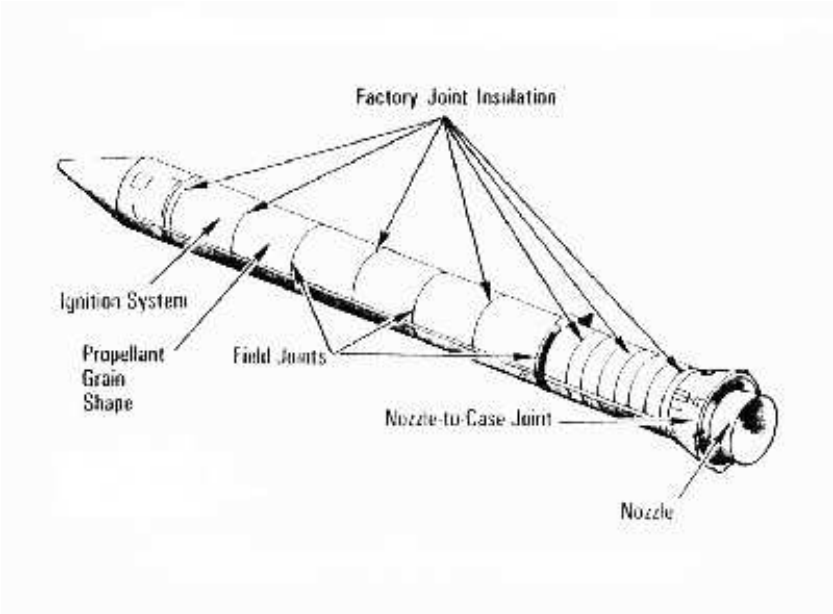


Figura 1.9: I *boosters*

Una commissione d'inchiesta studiò le cause dell'esplosione e dopo due settimane pubblicò un rapporto[58]. *L'eplosione avvenne a causa di un errore del software del sistema inerziale di riferimento.* Il comportamento del *Flight Control System* dell'Ariane 5 ed il suo moto nello spazio, erano misurati da un *Inertial Reference System* (SRI). Quest'ultimo disponeva di un computer interno in cui venivano misurati sia gli angoli di inclinazione assunti dal razzo durante il volo che le velocità raggiunte, sulla base di informazioni provenienti dalla piattaforma di lancio. I dati del SRI venivano trasmessi al computer di bordo (OBC, *On Board Computer*), che controllava il programma di volo ed i dispositivi ad esso correlati. Per migliorare l'affidabilità era prevista una quantità sovrabbondante di apparecchiature. C'erano, infatti, due SRI che lavoravano in parallelo, basati su software ed hardware identici. Uno di essi era attivo, il secondo in stand-by, in modo tale che, qualora il computer OBC individuasse un guasto od un errore nel sistema SRI, immediatamente si collegasse al secondo, a condizione che quest'ultimo funzionasse correttamente. Inoltre c'erano due OBC così come erano state duplicate anche un certo numero di unità del *Flight Control System.* Il razzo cominciò a disintegrarsi circa 39 secondi dopo il lancio, a causa di dati trasmessi dal sistema attivo SRI 2, che, in quel momento, non contenevano informazioni coerenti con il volo ma furono interpretate come tali; il motivo per cui il sistema SRI 2 attivo non inviò informazioni corrette consisteva nel fatto che l'unità aveva dichiarato la presenza di una situazione eccezionale, durante l'esecuzione del software, per cui SRI 2 si era interrotto. **L'anomalia nel software SRI interno si verificò durante la conversione di un dato da *floating-point a 64 bit* ad un valore *intero a 16 bit.* Il numero**

[58] [1]

floating-point convertito presentava un valore più grande di quello che sarebbe potuto essere rappresentato da un intero a 16 bit (praticamente si verificò un errore di overflow). L'errore si verificò a causa di un risultato inaspettato di una funzione interna di allineamento, chiamata Horizontal Bias, BH, correlata alla velocità orizzontale percepita dalla piattaforma di lancio; il valore di BH si presentò molto più alto di quanto ci si aspettasse. Questo provocò un errore, *operand error*, che le istruzioni di conversione dei dati non erano predisposte ad evitare. Sebbene ne sia stata individuata l'origine, in realtà non fu esattamente questo a determinare il fallimento della missione. Il meccanismo destinato al trattamento di situazioni eccezionali prevedeva di individuare l'errore nel flusso di dati, immagazzinare il complesso dei dati affetti da errore in un'area di memoria esterna al sistema ed, infine, determinare lo spegnimento del sistema SRI. E fu proprio la decisione di interrompere le operazioni di quest'ultimo che, essenzialmente, si dimostrò fatale, dal momento che sarebbe stato impossibile ricalcolare la posizione, dopo lo spegnimento. Da quel momento in poi, dunque, il sistema SRI risultò inutile. Il motivo di quella tragica disposizione va ricercato nel fatto che il programma della missione Ariane prevedeva di affrontare solo fallimenti hardware.

Questo è uno dei motivi per cui si devono sempre prevedere situazioni eccezionali nella realizzazione di un software.

1.12 Il calcolo numerico di π

Riportiamo alcuni classici metodi per il calcolo numerico di π, poiché riteniamo utile, dal punto di vista didattico, evidenziare come risultino ancora attuali alcune questioni di calcolo numerico dell'antichità. Il problema del calcolo delle cifre di π è oggetto di studio sin dall'antichità. Il valore di π ha attirato l'attenzione di molti matematici, dai tempi di Archimede ai nostri giorni e sono state utilizzate diverse metodologie che hanno condotto ad approssimazioni sempre piú accurate del numero di cifre di π[59].

Uno dei primi risultati significativi è dovuto ad Archimede che riuscì a dimostrare che:

$$\frac{223}{71} = 3 + \frac{10}{71} < \pi < 3 + \frac{1}{7} = \frac{22}{7}$$

approssimando la lunghezza della circonferenza con raggio unitario con il perimetro dei poligoni inscritti e circoscritti, arrivando ad utilizzare poligoni fino a 96 lati.

Riportiamo una breve tabella che indica **lo sviluppo storico** del calcolo delle cifre di π:

[59] L'interesse che accomuna i matematici antichi e quelli moderni, nei confronti del calcolo dei decimali di π, è legato sia all'uso che ne hanno fatto le civiltà nel corso dei secoli (basti pensare che già i costruttori Egizi stabilirono l'inclinazione delle pareti della piramide di Cheope in modo tale che il rapporto tra l'altezza e la larghezza della base corrispondesse a quello esistente tra il raggio e la circonferenza di un cerchio), che a quello che se ne fa attualmente in numerosi campi, tra i quali, ad esempio, la crittografia; d'altra parte, lo stimolo a determinare sempre più cifre di quello sviluppo infinito che rappresenta lo stesso π, è legato anche alle problematiche ancora aperte, che riguardano l'espressione e la sequenza dei decimali del numero. Ci si chiede, ad esempio, se ciascuna delle cifre $0, 1, 2, \ldots, 9$, si ripete infinitamente spesso, o se π è *semplicemente normale in base 10*, nel senso che ogni sua cifra appare ugualmente spesso, in senso asintotico, nel suo sviluppo decimale, oppure se è *normale in base 10*, cioè se ogni blocco di cifre di una fissata lunghezza appare ugualmente spesso, in senso asintotico, nel suo sviluppo decimale, o anche se π è *normale*, intendendo con ciò, che ogni blocco di cifre di una fissata lunghezza appare ugualmente spesso, in senso asintotico, nella sua rappresentazione in una qualsiasi base.

Autore	**Anno**	**Risultato**	**Cifre corrette**
Re Salomone	975 a.c	3.0	1
Archimede	250 a.c.	3.14	3
Aryabhata	499 d.c.	3.1416	4
Fibonacci	1220	3.141818	4
Viete	1593	3.141592653	9
Newton	1665	3.1415926535897932	16
Sharp	1699	3.(72 cifre)	71
Machin	1706	3.(100 cifre)	100
Eulero	1755	3.(20 cifre)	20
Vega	1789	3.(140 cifre)	126
Vega	1794	3.(137 cifre)	136
Ferguson	07-1946	3.(700 cifre)	620

Tra gli autori indicati in tabella è curioso ricordare che Eulero, che adottò la lettera greca π, ottenne il suo risultato in solo un'ora!
Il valore ottenuto da Ferguson nel Luglio del 1946 rappresenta l'ultimo calcolo delle cifre di π eseguito a mano.

Nella tabella seguente si riportano, invece, alcuni dei risultati ottenuti mediante calcolatore.

Autore	**Anno**	**Cifre corrette**	**Macchina**
Ferguson	01-1947	710	Desk calculator
Ferguson e Wrench	09-1947	808	Desk calculator
Shants e Wrench Jr	07-1961	100 265	IBM 7090
Guilloud e Filliatre	02-1966	250 000	IBM 7030
Guilloud e Dichampt	02-1967	500 000	CDC 6600
Kanada e Ushiro	10-1983	10 013 395	Hitachi S-810/20
Kanada et al.	01-1987	134 214 700	NEC SX-2
Kanada e Takahashi	09-1999	206 158 430 000	Hitachi SR8000
Kanada et al.	09-2002	1 241 100 000 000	Hitachi SR8000/MP

In particolare il risultato raggiunto da Y. Kanada e Y. Ushiro nel 1983 è il primo ottenuto mediante *supercalcolatore*. Dal 1980 Yasumasa Kanada è uno dei maggiori esponenti, tra gli interessati al calcolo di π con un elevato numero di cifre. La maggior parte dei suoi calcoli è stata realizzata su supercalcolatori e basata su moderni algoritmi iterativi.

Le prime 151 cifre del numero π calcolate su un elaboratore IBM 704 sono riportate qui

di seguito[60]:

$$\pi = 3.14159265358979323856264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848\cdots$$

[60]Per una breve storia sul calcolo delle cifre di π mediante calcolatore si suggerisce il sito

`http://pw1.netcom.com/~hjsmith/`

ed, in particolare, la pagina

`http://pw1.netcom.com/~hjsmith/Pi.html`

L'ultimo risultato risale al Settembre 2002, quando il Prof. Yasumasa Kanada dell'Università di Tokyo, *Information Technology Center*, insieme ad altri nove collaboratori, realizzò il calcolo di π con 1.241.100.000.000 cifre decimali, superando di piu' di sei volte il numero delle cifre del loro stesso *Guinness World Record*, costituito da 206.158.430.000 cifre decimali e raggiunto nel 1999. Il calcolo richiese circa 602 ore e fu realizzato con un supercomputer Hitachi (Hitachi SR8000), con accesso alla memoria di circa un terabyte. Tali informazioni sono reperibili alla pagina web

`http://www.sciencenews.org/articles/20021214/mathtrek.asp`

1.12.1 *Metodo I - Archimede 240 a.C.*

Si consideri una circonferenza di raggio r. Essendo

$$A_{cerchio} = \pi \cdot r^2, \ se \ r = 1 \ \Rightarrow A_{cerchio} = \pi$$

Il metodo proposto da Archimede è quello di approssimare il valore di π attraverso l'area, A_n, del poligono regolare di 2^n lati, inscritto nella circonferenza di raggio unitario. Più precisamente:

$$\pi = \lim_{n \to \infty} A_n$$

Sia $n = 2$, A_2 è l'area del quadrato inscritto nella circonferenza.

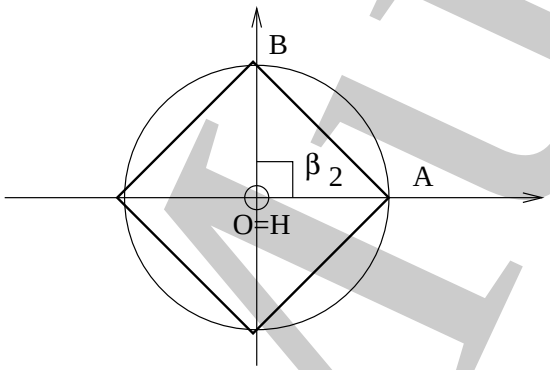


Figura 1: Quadrato inscritto nella circonferenza

Osservando la Figura 1, l'area del triangolo AOB può essere ottenuta come:

$$\frac{AO \cdot BH}{2}$$

$BH = AO = 1$ perché raggi di una circonferenza unitaria. Inoltre: $\beta_2 = \frac{2\pi}{4} = \frac{\pi}{2}$, pertanto $sen(\beta_2) = 1$.
Risulta quindi:

$$\frac{AO \cdot BH}{2} = \frac{1 \cdot sen(\beta_2)}{2} \ \Rightarrow \ A_2 = 2^2 \cdot \frac{1 \cdot sen(\beta_2)}{2}.$$

dove abbiamo posto $BH = sen(\beta_2)$.
Se dividiamo ciascun lato del quadrato in due, si ottiene il poligono con 2^3 lati per cui la situazione è ora quella illustrata in Fig. 2.

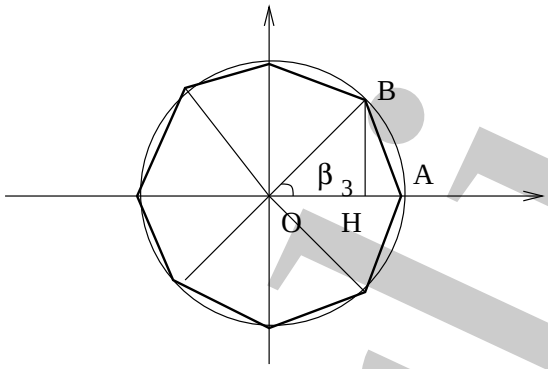


Figura 2: Metodo di Archimede per il calcolo di π

Ragionando come prima, l'area del triangolo AOB è data da:

$$\frac{AO \cdot BH}{2} = \frac{1 \cdot sen(\beta_3)}{2} \Rightarrow A_3 = 2^3 \cdot \frac{1 \cdot sen(\beta_3)}{2}$$

essendo $\beta_3 = \frac{2\pi}{8}$.

In generale, per un fissato n, l'area del poligono con 2^n lati, indicata con A_n, è 2^n volte l'area dei triangoli che lo compongono, ovvero, come indicato dalla Fig. 2, si ha:

$$(1) \quad A_n = \frac{2^n AO \cdot BH}{2} = 2^{n-1} sen(\beta_n)$$

dove $\beta_n = \frac{2\pi}{2^n}$.
Posto:

$$(2) \quad b_n = 2^{n-1}$$

la (1) diventa:

$$(3) \quad A_n = b_n \cdot sen(\beta_n)$$

Il problema del calcolo di A_n è ricondotto al calcolo di un valore approssimato di $sen(\beta_n)$.

Nel metodo proposto da Archimede, il valore di $sen(\beta_n)$ viene calcolato a partire da quello di $sen(2\beta_n)$ attraverso la formula di bisezione degli angoli:

$$(4) \quad sen(\beta_n) = \sqrt{\frac{1 - cos(2\beta_n)}{2}} = \sqrt{\frac{1 - \sqrt{1 - sen^2(2\beta_n)}}{2}}$$

dove $0 \leq \beta_n < \pi/2$.

Posto $s_n = sen(\beta_n)$ si ha :

$$s_2 = sen(\beta_2) = 1$$

e, in generale:

$$sen(2\beta_n) = sen\left(2\frac{2\pi}{2^n}\right) = sen(\beta_{n-1}) = s_{n-1}$$

Dalla (4) si ottiene la formula ricorrente:

$$s_2 = 1, \qquad s_n = \sqrt{\frac{1-\sqrt{1-s_{n-1}^2}}{2}} \quad n > 2$$

Si ha anche:

$$b_2 = 2, \qquad b_i = 2 \times b_{i-1,} \quad i > 2$$

pertanto la (3), diventa:

$$A_n = b_n \cdot s_n = 2 \cdot b_{n-1} \cdot \sqrt{\frac{1-\sqrt{1-s_{n-1}^2}}{2}}$$

In conclusione una stima di π si ottiene attraverso approssimazioni successive ottenute dallo schema iterativo seguente:

$$(5) \quad \begin{cases} b_2 = & 2\,, \quad s_2 = 1 \\ b_i = & 2 \cdot b_{i-1} \\ s_i = & \sqrt{\frac{1-\sqrt{1-s_{i-1}^2}}{2}}, \quad i > 2 \\ A_i = & b_i \cdot s_i, \quad i > 2 \end{cases}$$

Analisi della stabilità dell'algoritmo

Volendo implementare lo schema iterativo descritto dalla (5) in un sistema aritmetico a precisione finita $\Im$, bisogna tenere conto della propagazione dell'errore di *round-off*. Consideriamo quindi il passo generico dell'algoritmo di Archimede:

$$
\begin{aligned}
&A_2 = b_2 \cdot s_2 = 2 \cdot 1 = 2 \\
&A_3 = b_3 \cdot s_3 = 2b_2 \cdot \sqrt{\tfrac{1-\sqrt{1-s_2^2}}{2}} = 2^2 \cdot \sqrt{\tfrac{1}{2}} = 2^2 \cdot \tfrac{\sqrt{2}}{2} = 2 \cdot \sqrt{2} \\
&A_4 = b_4 s_4 = 2b_3 s_4 = 2^3 \cdot \sqrt{\tfrac{1-\sqrt{1-s_3^2}}{2}} = 2^3 \cdot \sqrt{\tfrac{1-\sqrt{1-\frac{1}{2}}}{2}} = 2^2 \cdot \sqrt{2-\sqrt{2}} \\
&A_5 = b_5 s_5 = 2b_4 s_5 = 2^4 \cdot \sqrt{\tfrac{1-\sqrt{1-s_4^2}}{2}} = 2^3 \cdot \sqrt{\tfrac{1-\sqrt{1-\frac{2-\sqrt{2}}{4}}}{2}} = 2^2 \cdot \sqrt{2-\sqrt{2-\sqrt{2}}} \\
&\vdots \\
&A_i = b_i \cdot s_i = 2b_{i-1} \cdot s_i = 2^{i-1} \cdot \sqrt{\tfrac{1-\sqrt{1-s_{i-1}^2}}{2}} = 2^{i-2} \cdot \underbrace{\sqrt{2-\sqrt{2-\ldots-\sqrt{2}}}}_{i-2 \text{ radicali innestati}} \quad , i > 2
\end{aligned}
$$

Si osserva che al passo i-esimo si moltiplica il termine precedente per:

$$\mu = 2^{i-2}$$

che è, quindi, anche il fattore di amplificazione dell'errore di round-off. Lo stesso fattore, in base 10, è:

$$\mu \simeq 10^{0.301 \cdot (i-2)}$$

segue allora che l'algoritmo descritto è **instabile**.
Il valore di π fornito dalla procedura descritta potrebbe essere, quindi,
del tutto errato.

Calcoliamo quando il fattore di amplificazione dell'errore di round-off, comporta un risultato errato del 100%.
Poiché:

$$\pi \approx 3.1415.....$$

la perturbazione introdotta deve essere dell'ordine di 10^1:

$$10^{0.301 \cdot (i-2)} = 10^1 \quad \Leftrightarrow \quad 0.301 \cdot (i-2) = 1 \quad \Leftrightarrow$$

$$\Leftrightarrow \quad i \approx \frac{10}{3} + 2 \quad \Leftrightarrow \quad i \approx 6$$

Dunque l'algoritmo produce un risultato sbagliato dopo circa 6 passi. Tale risultato è confermato dalle esperienze numeriche riportate nel paragrafo 1.12.8.

Analisi dell'errore di troncamento analitico

Per dare una stima dell' errore di troncamento che si commette arrestando al passo n il processo iterativo descritto, si utilizza lo sviluppo in serie di Mc-Laurin della funzione $sen(x)$.
Lo sviluppo in serie di Mc-Laurin della funzione $sen(x)$ è dato da :

$$(6) \qquad sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Ricordando la (1) si ha:

$$A_n = 2^{n-1} sen \frac{\pi}{2^{n-1}}$$

ed, utilizzando la (6), calcolata in $\frac{\pi}{2^{n-1}}$, segue che:

$$\begin{aligned} A_n &= 2^{n-1}\left[\frac{\pi}{2^{n-1}} - \frac{1}{6}\frac{\pi^3}{8^{n-1}} + \frac{1}{120}\frac{\pi^5}{32^{n-1}} - \dots\right] \\ &= \left[\pi - \frac{1}{6}\frac{\pi^3}{4^{n-1}} + \frac{1}{120}\frac{\pi^5}{16^{n-1}} - \dots\right] \end{aligned}$$

quindi l'errore di troncamento analitico:

$$\begin{aligned} e_n &= \pi - A_n = \\ &= \frac{1}{6}\frac{\pi^3}{4^{n-1}} - \dots \end{aligned}$$

e risulta:

$$|e_n| \simeq \left|\frac{\pi^3}{6} \cdot \frac{1}{4^{n-1}}\right| < 21 \times \frac{1}{4^n}$$

ovvero

$$e_n = O\left(\frac{1}{4^n}\right);$$

al passo successivo:

$$|e_{n+1}| < 21 \times \frac{1}{4^{n+1}} = 21 \times \frac{1}{4}\frac{1}{4^n} = O\left(\frac{1}{4^{n+1}}\right)$$

cioè:

$$e_{n+1} < \frac{1}{4} O\left(\frac{1}{4^n}\right) = \frac{1}{4} e_n$$

Quindi, ad ogni passo n, l'errore di troncamento analitico si riduce di un quarto (ordine di convergenza 1).

1.12.2 *Metodo II - Viete 1593*

Si consideri una circonferenza C di centro l'origine e di raggio r = 1. Il metodo proposto da Viete è quello di approssimare il valore di π con il semiperimetro p_n del poligono regolare con 2^n lati inscritto nella circonferenza. Indicata con $l(C)$ la lunghezza della circonferenza C

$$l(C) = 2 \cdot \pi \cdot r$$

se $r = 1$ allora :

$$\pi = \frac{l(C)}{2}$$

e $l(C) \approx p_n$.
Per ogni fissato valore di n, si indichi con a_n la lunghezza del lato del poligono di 2^n lati e con p_n il suo semiperimetro, si ottiene :

$$(1) \quad p_n = \frac{2^n a_n}{2} = 2^{n-1} a_n,$$

La relazione che lega il lato a_n del poligono di 2^n lati al lato a_{n+1} del poligono con 2^{n+1} lati è la seguente:

$$(2) \quad a_{n+1} cos\left(\frac{\pi}{2^{n+1}}\right) = \frac{a_n}{2},$$

da cui si ottiene :

$$(2') \quad a_n = 2 \cdot a_{n+1} cos\left(\frac{\pi}{2^{n+1}}\right),$$

Infatti come si osserva dalla Figura 3, se $n = 2$, a_2 è il lato del quadrato inscritto e a_3 è il lato dell'ottagono inscritto.

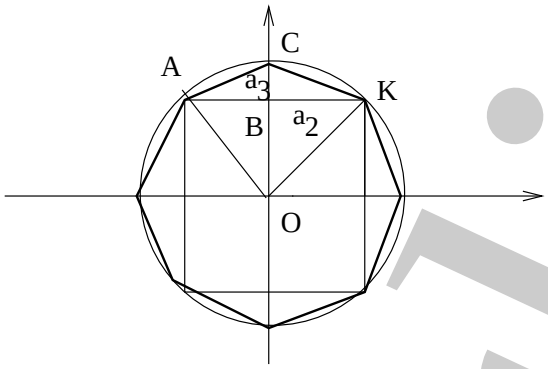


Figura 3: Metodo di Viete per il calcolo di π

Se si considerano i triangoli AKC e AOB, gli angoli $A\hat{K}C$ ed $A\hat{O}B$ sono due angoli rispettivamente alla circonferenza e al centro che insistono sullo stesso arco $(\widehat{AC})$[61] per cui:

$$A\hat{K}C = \frac{A\hat{O}B}{2}, \quad A\hat{O}B = \frac{\pi}{4} \Rightarrow A\hat{K}C = \frac{\pi}{8}$$

L'angolo $C\hat{A}B = \frac{\pi}{8}$ è uguale all' angolo $A\hat{K}C$, essendo angoli alla base di un triangolo isoscele (ACK); per cui se $\frac{a_2}{2}$ è un cateto del triangolo ABC, a_3 ne è l'ipotenusa, si ottiene che:

$$a_3 \cdot cos\left(\frac{\pi}{8}\right) = a_3 \cdot cos\left(\frac{\pi}{2^3}\right) = \frac{a_2}{2}.$$

Ponendo $\beta_i = \frac{\pi}{2^i}$ e $c_i = cos\,(\beta_i)$ ed utilizzando la formula di bisezione per il coseno:

$$cos(\beta_i) = \sqrt{\frac{1 + cos(2\beta_i)}{2}} = \sqrt{\frac{1 + cos(\beta_{i-1})}{2}}$$

poiché:

$$c_1 = cos\left(\frac{\pi}{2}\right) = 0,$$

si ha:

$$c_i = \sqrt{\frac{1 + c_{i-1}}{2}}, \quad i \geq 2$$

[61] Ricordiamo il seguente risultato di geometria elementare:

Teorema 1.1. *In una circonferenza, l'angolo alla circonferenza è la metà del corrispondente angolo al centro che insiste sullo stesso arco.*

Per $i = 2$, il triangolo OAB è isoscele e retto in B e l'angolo $O\widehat{A}B = \frac{\pi}{4}$, per cui essendo $AO = r$, raggio della circonferenza, si ha che:

$$\frac{a_2}{2} = r \cdot cos\left(\frac{\pi}{4}\right) \quad \Rightarrow \quad a_2 = 2 \cdot 1 \cdot \frac{\sqrt{2}}{2} = \sqrt{2}$$

per cui:

$$p_2 = 2 \cdot a_2 = 2 \cdot \sqrt{2}$$

Ricaviamo una relazione per $i > 2$, utilizzando la (1) e la (2) :

$$p_i = 2^{i-1} a_i = 2^{i-1} \cdot 2 \cdot cos(\beta_{i+1}) \cdot a_{i+1} = 2^i cos(\beta_{i+1}) \cdot a_{i+1} =$$

$$= p_{i+1} \cdot cos(\beta_{i+1}) = p_{i+1} \cdot c_{i+1}$$

da cui:

$$p_{i+1} = \frac{p_i}{c_{i+1}}$$

Per quanto detto resta individuata la seguente formula di ricorrenza:

$$p_{i+1} = \frac{p_i}{\sqrt{\frac{1+c_i}{2}}}, \qquad i \geq 2$$

con

$$p_2 = 2 \cdot \sqrt{2} \quad c_2 = \sqrt{\frac{1}{2}} = \frac{\sqrt{2}}{2}$$

Analisi della stabilità dell'algoritmo

Per quanto riguarda la propagazione dell'errore di *round-off*, essendo:

$$0 < c_i = cos\left(\frac{\pi}{2^i}\right) < 1, \qquad i \geq 3$$

al crescere di i le quantità calcolate c_i tendono ad 1:

$$c_i \to 1 \qquad \Rightarrow \qquad \frac{1}{c_i} \to 1$$

Pertanto non c'è amplificazione dell'errore di round-off nel calcolo di

$$p_{i+1} = p_i \cdot \frac{1}{c_i}$$

e il metodo di Viete per il calcolo di π è **stabile**.

Analisi dell'errore di troncamento analitico

Per determinare una stima dell'errore di troncamento analitico commesso ad ogni passo n si osserva che:

$$a_n = 2 \cdot sen\frac{2\pi}{2^{n+1}} \quad \Rightarrow \quad p_n = 2^{n-1}a_n \; = \; 2^n sen\frac{\pi}{2^n}$$

Utilizzando lo sviluppo in serie di Taylor della funzione seno, calcolato in $\frac{\pi}{2^n}$, si ha:

$$\begin{aligned} p_n &= 2^n \left[\frac{\pi}{2^n} - \frac{1}{6}\frac{\pi^3}{8^n} + \frac{1}{120}\frac{\pi^5}{16^n} - \dots\right] = \\ &= \pi - \frac{1}{6}\frac{\pi^3}{2^3} + \frac{1}{120}\frac{\pi^5}{2^5} - \dots \end{aligned}$$

quindi:

$$e_n = \pi - p_n$$

da cui:

$$|e_n| < \left|\frac{1}{6}\frac{\pi^3}{4^n} + \frac{1}{120}\frac{\pi^5}{16^n} - \dots\right| < 5.2 \times \frac{1}{4^n}$$

1.12.3 *Metodo III - Wallis 1655*

Nel suo libro *Arithmetica Infinitorium (1655),* John Wallis presentò l'identità seguente:

$$\frac{\pi}{2} = \frac{2}{1}\frac{2}{3}\frac{4}{3}\frac{4}{5}\cdots\frac{2n}{2n-1}\frac{2n}{2n+1}\cdots$$

il valore di π é calcolato mediante il prodotto di infinitesimi.
L'approssimazione del valore di π viene ottenuta considerando il prodotto parziale i-esimo p_i dei primi i fattori. E' possibile allora considerare la formula ricorrente seguente:

$$p_0 = 2, \quad p_i = p_{i-1}\frac{4i^2}{4i^2-1}$$

dove i=1,2, ... ,n.

Analisi della stabilità dell'algoritmo

Diamo una stima dell'errore di round-off. Si osservi che:

$$\frac{4i^2}{4i^2-1} = 1 + \frac{1}{4i^2-1} = 1 + \delta_i$$

dove $\frac{1}{4i^2-1} = \delta_i$.
Al crescere dell'indice i, per $i \to +\infty$, i valori δ_i tendono a zero ($\delta_i \to 0$) . L'algoritmo risulta dunque stabile.

Analisi dell'errore di troncamento analitico

Si può dimostrare che l'errore di troncamento analitico é :

$$|e_n| < \left|\frac{\pi}{2}n^{-1}\right| < 1.6n^{-1}$$

Tale errore è stato ricavato esprimendo la formula di Wallis, mediante l'identità di *Lord Brouncher*:

$$\frac{\pi}{2} = 2 \times \cfrac{1}{1+\cfrac{1^2}{2+\cfrac{3^2}{2+\cfrac{5^2}{2+\cfrac{7^2}{2+\cdots}}}}}$$

1.12.4 *Metodo IV - Leibniz 1688*

Il valore di π è ottenuto dallo sviluppo in serie di Mc Laurin[62] della funzione $arctg(x)$:

$$arctg(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$$

Per x=1, si ha infatti:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

indicata con q_i la somma parziale dei primi i addendi è possibile ricavare la formula ricorrente seguente:

$$(1) \quad q_1 = 1, \quad q_{i+1} = q_i + \frac{(-1)^{i-1}}{2i-1}$$

dove $i = 1, 2, ...n-1$.

Analisi della stabilità dell'algoritmo

In un sistema aritmetico a precisione finita $\Im$, l'errore di round-off E_n, nel calcolo di una somma di n numeri x_i, si può stimare attraverso la formula seguente:

$$|E_n| \leq (n-1)u \sum_{i=1}^{n} |x_i| + O(u^2), \quad \text{con} \quad u \textit{ massima accuratezza}$$

e, quindi, cresce linearmente con il numero degli addendi, n. Si osserva che

$$\frac{(-1)^{i-1}}{2i-1} \to 0, \quad \text{quando} \quad i \to \infty$$

per cui l'algoritmo per il calcolo della (1) risulta stabile.

Analisi dell'errore di troncamento analitico

L'errore di troncamento analitico è dato da:

$$e_n = \pi - 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \frac{(-1)^{n+1}}{2n+1}\right)$$

per cui:

$$|e_n| < \frac{4}{2n+1} < 2n^{-1}$$

[62] Al momento non siamo interessati al modo in cui è approssimata la funzione $arctg(\cdot)$ nei compilatori dei calcolatori. Possiamo solo accennare che la sua valutazione non viene eseguita mediante sviluppo in serie di Mc Laurin.

1.12.5 *Metodo V - Machin 1706*

Il metodo seguente è una variante di quello di Leibniz: il valore di π è calcolato effettuando la differenza tra due polinomi di Taylor della funzione $arctg(x)$.

Sia β l'arcotangente di $\frac{1}{5}$:

$$tg(\beta) = \frac{1}{5}$$

dalla formula di duplicazione degli angoli si ottiene:

$$tg(2\beta) = \frac{2tg(\beta)}{1 - tg^2(\beta)} = \frac{5}{12}$$

riapplicandola nuovamente :

$$tg(4\beta) = \frac{2tg(2\beta)}{1 - tg^2(2\beta)} = \frac{120}{119}.$$

L'idea di Machin fu quella di approssimare 1 (valore dell'arcotangente a $\frac{\pi}{4}$), con la frazione $\frac{120}{119}$, e calcolare la differenza $(\frac{120}{119} - 1)$ in termini di angoli, mediante la formula di sottrazione della tangente:

$$tg\left(4\beta - \frac{\pi}{4}\right) = \frac{tg(4\beta) - tg(\frac{\pi}{4})}{1 + tg(4\beta) \cdot tg(\frac{\pi}{4})} = \frac{tg(4\beta) - 1}{1 + tg(4\beta)}$$

si ottiene:

$$arctg\left(\frac{1}{239}\right) = 4\beta - \frac{\pi}{4} = 4arctg\left(\frac{1}{5}\right) - \frac{\pi}{4}$$

risolvendo rispetto a $\frac{\pi}{4}$:

$$\frac{\pi}{4} = 4arctg\left(\frac{1}{5}\right) - arctg\left(\frac{1}{239}\right).$$

Indicando con $T_n(x)$ il polinomio di Taylor di grado n dell'$arctg(x)$, possiamo riscrivere la relazione ottenuta, nel modo seguente:

$$\pi \approx 16 \cdot T_n\left(\frac{1}{5}\right) - 4 \cdot T_n\left(\frac{1}{239}\right)$$

Analisi dell'errore di troncamento analitico

Si puó calcolare la stima dell' errore di troncamento:

$$|e_n| < \frac{16}{(2n+1)5^{2n+1}} < 1.6n^{-1}25^{-n}$$

considerando l'errore di troncamento analitico che si commette approssimando π con il polinomio di Taylor di grado n che approssima:

$$16 \cdot arctg\left(\frac{1}{5}\right).$$

1.12.6 *Metodo VI*

Il valore di π è calcolato utilizzando lo sviluppo in serie di Taylor per la funzione arcoseno:

$$arcsenx = x + \frac{1}{2}\frac{x^3}{3} + \frac{1}{2}\frac{3}{4}\frac{x^5}{5} + \dots$$

Per x=1/2, si ha infatti:

$$\pi = 6\left(\frac{1}{2} + \frac{1}{2}\frac{1}{3}\frac{1}{2^3} + \frac{1}{2}\frac{3}{4}\frac{1}{5}\frac{1}{2^5} + \dots\right)$$

Indicata con q_i e t_i, rispettivamente la somma dei primi i termini e l'i-*esimo* fattore dello sviluppo in serie si ottiene la seguente formula iterativa:

$$(1) \quad t_1 = 1/2, \quad t_{i+1} = \frac{t_i(2i-1)}{8i}$$

$$(2) \quad q_0 = 0, \quad q_{i+1} = q_i + \frac{t_i}{2i-1},$$

ed inoltre:

$$p_n = 6q_n$$

dove i=1,..n.

Analisi della stabilità dell'algoritmo

In un sistema aritmetico a precisione finita $\Im$, l'errore di round-off E_n, nel calcolo di una somma di n numeri x_i, si può stimare attraverso la formula seguente:

$$|E_n| \leq (n-1)u\sum_{i=1}^{n}|x_i| + O(u^2), \quad \text{con} \quad u \textit{ massima accuratezza}$$

e, quindi, cresce linearmente con il numero degli addendi, n. Si osserva che

$$\frac{(2i-1)}{8i} \to \frac{1}{4}, \quad \text{quando} \quad i \to \infty$$

per cui l'algoritmo per il calcolo della (1) risulta stabile;
analogamente

$$\frac{t_i}{2i-1} \to 0, \quad \text{quando} \quad i \to \infty$$

per cui l'algoritmo per il calcolo della (2) risulta stabile.

Analisi dell'errore di troncamento analitico

Valutiamo l'errore di troncamento analitico che si commette approssimando π con il polinomio di grado n di Taylor della funzione $arcsen(x)$ in $x = \frac{1}{2}$:

$$|e_n| = \left| \pi - 6 \left(\frac{1}{2} + \frac{1}{2}\frac{1}{3}\frac{1}{2^3} + \frac{1}{2}\frac{3}{4}\frac{1}{5}\frac{1}{2^5} \right) \right|$$

per cui :

$$|e_n| < 0.7 \cdot n^{-1/2} \cdot 4^{-n}$$

1.12.7 *Metodo VII - Integrazione Numerica*

Il valore di π è calcolato attraverso una formula di quadratura trapezoidale. È possibile appossimare $\pi/4$ con la somma s_n delle aree dei trapezi di altezza $\frac{1}{n}$ inscritti in un quarto di circonferenza di raggio unitario.
Infatti la funzione che definisce un arco di circonferenza unitaria nel I quadrante è:

$$f(x) = \sqrt{1 - x^2}$$

Per cui integrando numericamente $f(x)$ in $[0, 1]$ si ottiene che:

$$\int_0^1 \sqrt{1 - x^2} dx \approx \frac{\pi}{4} \tag{1.56}$$

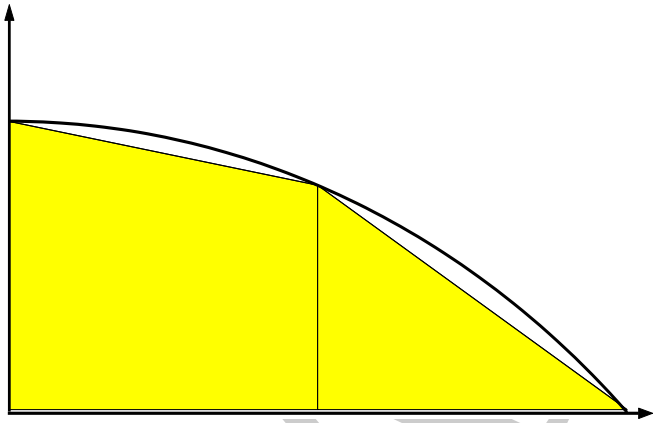


Figura 4: Metodo di integrazione

Utilizzando la formula di quadratura trapezoidale per il calcolo numerico dell'integrale (1.56) si ottiene, per s_n, l'espressione seguente:

$$s_n = \frac{1}{2^n}[1 + 2b_1 + 2b_2 + \cdots + 2b_{n-1}]$$

dove:

$$b_i = \sqrt{1 - \frac{i^2}{n^2}}$$

Analisi della stabilità dell'algoritmo

In un sistema aritmetico a precisione finita $\Im$, l'errore di round-off E_n, nel calcolo della somma s_n, si può stimare nel modo seguente:

$$|E_n| \leq \frac{1}{2^{n-1}}(n-1)u \sum_{i=1}^{n-1} |b_i| + O(u^2)$$

e, quindi, cresce linearmente con n, numero degli addendi. Pertanto l'algoritmo risulta stabile.

Analisi dell'errore di troncamento analitico

L'errore di troncamento analitico che si commette è:

$$|e_n| < k \cdot n^{-3/2}$$

essendo k una costante positiva ed e_n maggiorato dalla stima asintotica dei b_n.

1.12.8 Algoritmi implementati in FORTRAN

Di seguito vengono proposti alcuni programmi in Fortran relativi ad i metodi descritti nel paragrafo precedente.

Programma Fortran per il calcolo di π con il metodo di Archimede

```
=================================================================
      program  Calcolo Metodo I

C Fase  di dichiarazione delle variabili.

      integer n,step
      real  p,bi,si,error,pi,error1,n1

C Stampa a video del problema da risolvere

  write(*,*)    '                                                '
  write(*,*)    '    Calcolo di PI                               '
  write(*,*)    '    (1)   pi=b(i-1)s(i-1)                       '
  write(*,*)    '    (2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)       '
  write(*,*)    '                                                '
  write(*,*)    '                                                '

      write(*,*) 'Inserire il numero di lati del poligono'
      read*,n
      print*,n

C Fase di  Calcolo

      bi=2.
      si=1.
      step=0
      p=0

5     continue

      p=bi*si
      bi=2*bi
      si=sqrt((1-sqrt(1-si**2))/2)
      step=step+1

C Terminazione della fase di Calcolo.

      if(step.le.n) goto 5
```

```
      n1=n
      error=4**n1
      error=21*1/error
      pi=4*atan(1.)
      error1=(pi-p)/pi
      print*, 'PI calcolato=',p
      print*, 'Errore analitico stimato < di',error

C Confronto con la funzione di libreria

      print*, 'PI mediante funzioni built-in =',pi
      print*, 'Errore relativo',error1
      stop
      end
```

Prova di esecuzione

Consideriamo alcune prove di esecuzione dell'algoritmo descritto nel Metodo di Archimede, l'obiettivo è quello di mettere in evidenza l'**instabilità** del procedimento. Come vedremo per n = 7, il risultato è accurato a 5 cifre significative, mentre per n = 15, il risultato ottenuto è completamente errato.

```
Calcolo di PI
(1)   pi=b(i-1)s(i-1)
(2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)

Inserire il numero di lati del poligono
7
  PI calcolato=  3.141600132
  Errore analitico stimato < di  0.00128173828
  PI mediante funzione built-in=  3.14159274
  Errore relativo  0.235262069E-05
```

```
 Calcolo di PI
 (1)   pi=b(i-1)s(i-1)
 (2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)

 Inserire il numero di lati del poligono
 13
   PI calcolato=  2.82842708(*)
   Errore analitico stimato < di  3.12924385E-07
   PI mediante funzioni built-in=  3.14159274
   Errore relativo  0.0996837243
==============================================
```

(*) Il risultato della seconda esecuzione è del tutto errato. Infatti il sistema aritmetico del calcolatore esegue le operazioni nei registri a doppia precisione e per $n = 13$, l'errore di *round off* è del 100%, come descritto nel paragrafo 1.12.1 (algoritmo instabile).

Programma Fortran per il calcolo di π con il metodo di Viete

```
==========================================
      program  Calcolo Metodo II

C Fase  di dichiarazione delle variabili.

      integer n,step
      double precision  p,bi,si,error,pi,error1,n1

C Stampa a video del problema da risolvere

  write(*,*)    '                                    '
  write(*,*)    '   Calcolo di PI                    '
  write(*,*)    '   (1)   pi=p(i-1)/ci               '
  write(*,*)    '   (2)   c(i)=sqrt(1-c(i-1))/2 )    '
  write(*,*)    '                                    '
  write(*,*)    '                                    '

      write(*,*) 'Inserire il numero di lati del poligono'
      read*,n

C Fase di  Calcolo

      ci=0.
      per=2.
      step=1
      p=2.

5     continue

      step=step+1
      ci=sqrt((1+ci)/2)
      p=per/ci
      per=p

C Terminazione della fase di Calcolo.

       if(step.le.n) goto 5
       n1=n
       error=4**n1
       print*,'errore=',error
       error=5.2*1/error
       pi=4*atan(1.)
       error1=(pi-p)/pi
```

```
        print*, 'PI calcolato=',p
        print*, 'Errore analitico stimato < di',error

   C Confronto con la funzione di libreria

          print*, 'PI mediante funzioni built-in =',pi
          print*, 'Errore relativo',error1
          stop
          end
==================================================
```

Prova di esecuzione

Consideriamo alcune prove di esecuzione dell'algoritmo descritto nel Metodo di Viete, l'obiettivo è quello di mettere in evidenza la **stabilità** del procedimeto. Rispetto ai risultati del paragrafo precedente per valori più alti di n l'errore di round-off non compromette il risultato.

```
==================================================

   ------------------------------
    Calcolo di PI
    (1)   pi=p(i-1)/ci
    (2)   c(i)=sqrt(1-c(i-1))/2 )

   ------------------------------
    Inserire il numero di lati del poligono
              7
      PI calcolato=   3.141513586044312
      Errore analitico stimato < di  3.1738281250000001E-04
      PI mediante funzioni built-in =   3.141592741012573
      Errore relativo  0.25195808237132019E-04
==================================================


==================================================

   ------------------------------
    Calcolo di PI
    (1)   pi=p(i-1)/ci
    (2)   c(i)=sqrt(1-c(i-1))/2 )

   ------------------------------
    Inserire il numero di lati del poligono
             16
      PI calcolato=   3.141592502593994
      Errore analitico stimato < di  1.2107193470001221E-09
```

```
      PI mediante funzioni built-in =   3.141592741012573
      Errore relativo  0.75890988666060292E-07
==================================================


==================================================
   ---------------------------------
   Calcolo di PI
   (1)   pi=p(i-1)/ci
   (2)   c(i)=sqrt(1-c(i-1))/2 )

   ---------------------------------
   Inserire il numero di lati del poligono
             30
     PI calcolato=   3.141592502593994
     Errore analitico stimato < di  4.5102810375396986E-18
     PI mediante funzioni built-in =   3.141592741012573
     Errore relativo  7.5890988666060292E-08
==================================================
```

Osservazione

L'errore relativo che viene calcolato nel corso dell'algoritmo comprende anche l'errore di round-off, per cui risulta:

$$\begin{aligned} Errore\,relativo &= Errore\,di\,troncam.\,analitico + Errore\,di\,, round - off \\ &< 10^{-8} \end{aligned}$$

Programma Fortran per il calcolo di π con il metodo di Leibniz

A)Prima strategia

```
==============================================
       program  Calcolo Metodo IV strategia 1

   C Fase  di dichiarazione delle variabili.

      integer z
      real  p,i,error,pi,error1,n1,step,n

   C Stampa a video del problema da risolvere

     write(*,*)    '                                          '
     write(*,*)    '   Calcolo di PI                          '
     write(*,*)    '    Sviluppo in serie di Taylor di arctg(x)'
     write(*,*)    '                                          '
     write(*,*)    '                                          '
      write(*,*) 'Inserire ordine dello sviluppo'
      read*,n

   C Fase di  Calcolo

       qi=1.
       step=1

5      continue

       step=step+1
       i=(-1)**(step-1)
       i=i/(2*step-1)
       qi=qi+i

   C Terminazione della fase di Calcolo.

        if(step.le.n-1) goto 5
        p=4*qi
        n1=n
        error=1/n1
        error=2*error
        pi=4*atan(1.)
        error1=(pi-p)/pi
        print*, 'PI calcolato=',p
```

```
      print*, 'Errore analitico stimato < di',error

C Confronto con la funzione di libreria

      print*, 'PI mediante funzioni built-in =',pi
      print*, 'Errore relativo',error1
      stop
      end
```

==========

B) Seconda strategia

```
program  Calcolo Metodo IV strategia 2

C Fase  di dichiarazione delle variabili.

      integer z,step,n
      real p,i,error,pi,error1,n1
      real qi,qi1

C Stampa a video del problema da risolvere

  write(*,*)    '                                                    '
  write(*,*)    '    Calcolo di PI                                   '
  write(*,*)    '     Sviluppo in serie di Taylor di arctg(x)        '
  write(*,*)    '                                                    '
  write(*,*)    '                                                    '

    write(*,*) 'Inserire ordine dello sviluppo'
    read*,n
    print*,n

C Fase di  Calcolo

    qi=1.
    qi1=1.

    do 10 step=n,1,-1

    if(2*(step/2).eq.step) then
     i=1/(2*DFLOAT(step)-1)
     qi=qi+i
    else
```

```
      i=1/(2*DFLOAT(step)-1)
      qi1=qi1+i
      endif

10     continue

      qi=abs(qi-qi1)

C Terminazione della fase di Calcolo .

      p=4*qi
      n1=n
      error=1/n1

      error=2*error
      pi=4*atan(1.)
      error1=(pi-p)/pi
      print*, 'PI calcolato=',p
      print*, 'Errore analitico stimato < di',error

C Confronto con la funzione di libreria

      print*, 'PI mediante funzioni built-in =',pi
      print*, 'Errore relativo',error1
      stop
      end
```

Prova di esecuzione

A) Prima strategia

```
==============================================
 ------------------------------------------
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)
 ------------------------------------------
 Inserire ordine dello sviluppo
 70
   PI calcolato= 3.127307177
   Errore analitico stimato < di 0.2857142873E-01
   PI mediante funzioni built-in = 3.141592741
   Errore relativo 0.4547236487E-02
==============================================
```

```
================================================

 ------------------------------------
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)
 ------------------------------------

 Inserire ordine dello sviluppo
 200
   PI calcolato= 3.136592627
   Errore analitico stimato < di 0.9999999776E-02
   PI mediante funzioni built-in = 3.141592741
   Errore relativo 0.1591585809E-02
================================================


================================================

 ------------------------------------
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)
 ------------------------------------

 Inserire ordine dello sviluppo
 700
   PI calcolato= 3.140165091
   Errore analitico stimato < di 0.2857142827E-02
   PI mediante funzioni built-in= 3.141592741
   Errore relativo 0.4544352414E-03
================================================


================================================

 ------------------------------------
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)
 ------------------------------------

 Inserire ordine dello sviluppo
 500000
   PI calcolato= 3.141593933
   Errore analitico stimato < di 0.3999999990E-05
   PI mediante funzioni built-in = 3.141592741
   Errore relativo 0.3794549457E-06
================================================
```

B) Seconda strategia

```
================================================

 ------------------------------------------
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)
 ------------------------------------------
 Inserire ordine dello sviluppo
 70
   PI calcolato= 3.127307892
   Errore analitico stimato < di 0.2857142873E-01
   PI mediante funzioni built-in = 3.141592741
   Errore relativo 0.4547008779E-02
================================================


================================================

 ------------------------------------------
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)
 ------------------------------------------
 Inserire ordine dello sviluppo
 200
   PI calcolato= 3.136591911
   Errore analitico stimato < di 0.9999999776E-02
   PI mediante funzioni built-in = 3.141592741
   Errore relativo 0.1591813518E-02
================================================


================================================

 ------------------------------------------
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)
 ------------------------------------------
 Inserire ordine dello sviluppo
 700
   PI calcolato= 3.141160561
   Errore analitico stimato < di 0.2857142827E-2
   PI mediante funzioni built in = 3.141592741
   Errore relativo 0.4558771616E-04
================================================
```

```
==============================================

 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)

 Inserire ordine dello sviluppo
 500000
   PI calcolato= 3.141592026
   Errore analitico stimato < di 0.3999999990E-05
   PI mediante funzioni built-in = 3.141592741
   Errore relativo 0.2276729703E-07

==============================================
```

1.12.9 Conclusioni

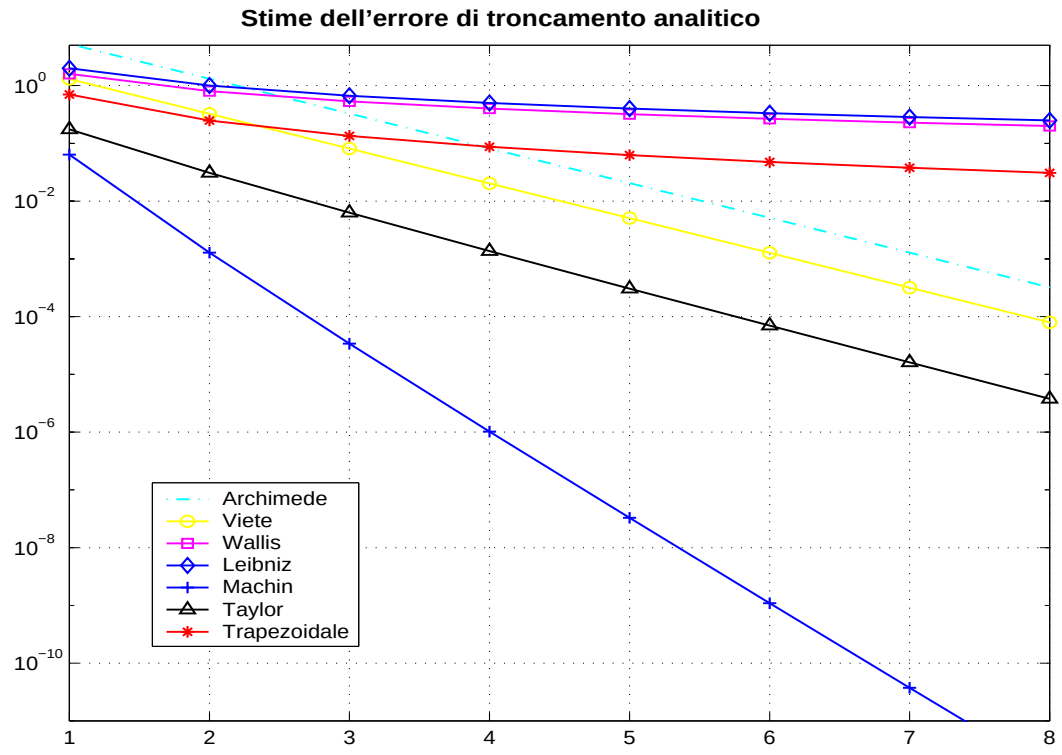
A partire da Leibniz, il calcolo delle cifre di π si basa sull'approssimazione della funzione arcotangente. Attualmente, negli algoritmi numerici, il modo più naturale per calcolare un'approssimazione di π secondo la precisione del sistema è

$$\pi = 4 \cdot arcotg(1)$$

che, ovviamente, fa uso dell'approssimazione della funzione arcotangente, considerata come funzione *built-in* del particolare compilatore utilizzato.

Si riporta una tabella contenente la stima dell'errore di troncamento analitico e la stabilità per i vari metodi descritti; segue, inoltre, il grafico delle curve corrispondenti, ottenute al variare di n. Da entrambi si deduce che il metodo di Machin fornisce una stima dell'errore di troncamento analitico più piccola rispetto agli altri metodi (invero questi ultimi sono stati elencati, in tabella, in ordine decrescente di velocità di convergenza) ovvero converge asintoticamente al valore π più velocemente rispetto ai metodi proposti.

Metodo	**Stima err. tronc.**	**Stabilità**
Machin (1706)	$1.6 \times n^{-1} \times 25^{-n}$	SI
Taylor	$0.7 \times n^{-1/2} \times 4^{-n}$	SI
Form. Trapezoidale	$K \times n^{-3/2}$	SI
Viete (1593)	5.2×4^{-n}	SI
Wallis (1655)	$1.6 \times n^{-1}$	SI
Leibniz (1688)	$2 \times n^{-1}$	SI
Archimede (240 a.c.)	21×4^{-n}	NO


Stime dell'errore di troncamento analitico
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
1
2
3
4
5
6
7
8
Archimede
Viete
Wallis
Leibniz
Machin
Taylor
Trapezoidale

Bibliografia

[1] *ARIANE 5 Flight 501 Failure, Report by the Inquiry Board, The Chairman of the Board: Prof. J. L. Lions*
http://sspg1.bnsc.rl.ac.uk/Share/ISTP/ariane5r.htm,
http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf, Paris, 1996 July 19.

[2] Cody W.J. et al. - *A Proposed Radix-and Wordlength-Independent Standard for Floating-Point Arithmetic* - IEEE Micro, 84, 1984.

[3] Dew P.M., James K.R. - *Introduction to Numerical Computation in Pascal* - Springer-Verlag, 1985.

[4] Forsythe A.I., Keenan T.A., Organick E.I., Stenberg W. - *Computer Science: a First Course.* - John Wiley & Sons, 1969.

[5] Forsythe G.E. - *What is a Satisfactory Quadratic Equation Solver?* - in `Constructive Aspects of the Fundamental Theorem of Algebra` - Proceedings of a Symposium conducted at the IBM Research Laboratory, Zürich-Rüschlikon, Switzerland, June 5-7, 1967. Edited by Bruno Dejon and Peter Henrici, pp. 53–61.

[6] Forsythe G.E., Malcom M.A., Moler C.B. - *Computer methods for mathematical computations* - Prentice-Hall, 1977.

[7] Gear C.W. - *Computer Organization and Programming* - McGraw-Hill, IV edition, 1985.

[8] Higham N.J. - *Accuracy and stability of numerical algorithms* - II ed., SIAM 2002.

[9] http://www.astronautica.us
http://web.tiscali.it/no-redirect-tiscali/astronautica1/astronautica_vettori.htm

[10] Kahan W., Coonen J.T., Stone H. - *A Proposed Standard Floating Point Arithmetic* - ACM SIGNUM Newsletter, special issue, Oct. 1979.

[11] Landau L., Lifchitz E. - *Phisique Théorique. Théorie de l'Élasticité* - tome VII, Mir, 1967.

[12] Lax P., Burstein S., Lax A. - *Analisi matematica con applicazioni e calcolo numerico* - Zanichelli, 1986.

[13] Sterbentz P. - *Floating point computation* - Prentice-Hall (1978)

[14] Wallis P.J.L. - *Improving Floating-Point Programming* - John Wiley & Sons, 1990.

[15] Wilkinson J.H. - *Rounding Errors in Algebraic Processes* - Prentice-Hall, 1963.