Appunti di Calcolo Numerico A.A. 2009/2010

Numeri Reali e Numeri Finiti

L'impiego di un calcolatore permette di effettuare operazioni elementari fra numeri in tempi molto brevi offrendo la possibilità di risolvere problemi molto più complicati mediante l'esecuzione di algoritmi complessi, costituiti da operazioni elementari.

Tuttavia il calcolatore, essendo una macchina finita, è costretto ad operare con numeri rappresentati da un **numero finito di cifre**; ne consegue che in genere un numero reale introdotto nel calcolatore viene approssimato mediante un "numero finito" o numero di macchina.

Si pensi, ad esempio, ai numeri π o $\sqrt{2}$ che hanno un numero infinito di cifre, essi non potranno mai essere rappresentati correttamente su un calcolatore-

Se si eseguono, poi, operazioni elementari sui numeri finiti si possono ottenere risultati non più rappresentabili esattamente dal calcolatore. **Questo significa che eseguendo un algoritmo su un calcolatore si ha, in generale, una creazione e propagazione degli errori**.

Quindi, il risultato prodotto dall'algoritmo differisce dal risultato esatto, cioè da quel risultato che si otterrebbe lavorando con i numeri reali.

Il controllo degli errori introdotto dall'uso dei numeri finiti è molto importante al fine di determinare l'attendibilità dei risultati ottenuti.

1. Rappresentazione dei Numeri Naturali: basi di Rappresentazione

Si consideri il ***sistema decimale*** per la rappresentazione dei **numeri naturali**.

Un numero intero positivo *N* viene espresso nella forma

$$N = d_n \cdot 10^n + d_{n-1} \cdot 10^{n-1} + ... + d_1 \cdot 10^1 + d_0 \cdot 10^0$$

e rappresentato, mediante **notazione posizionale**, come

$$N = d_n d_{n-1} ..... d_1 d_0$$

dove le cifre d_i , $0 \le i \le n$, sono numeri interi compresi tra 0 e 9.

- Es. il numero naturale **425**, nel sistema decimale è espresso nella forma:

  425 = 400 + 20 + 5 =4*10^2 + 2*10^1 +5*10^0

  Nel sistema decimale si è scelta la **base** di rappresentazione uguale a 10.

Considerando un sistema avente base di rappresentazione β >1, il numero *N* viene espresso come

$$N = (d_n d_{n-1} ..... d_1 d_0)_\beta$$
$$= d_n \cdot \beta^n + d_{n-1} \cdot \beta^{n-1} + ... + d_1 \cdot \beta^1 + d_0 \cdot \beta^0$$

dove $0 \le d_i \le \beta - 1$, con $0 \le i \le n$.

Note basi di rappresentazione sono la base binaria (β =2), ottale (β =8) ed esadecimale(β=16).

- Es. Sia il numero binario N=$(100011111)_2$; una notazione equivalente è:

  N = 1*2^8 + 0*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0

$= 256 + 0 + 0 + 0 + 16 + 8 + 4 + 2 + 1 = (287)_{10}$

$\Rightarrow (287)_{10} = (100011111)_2$

Ora, fissata una base intera $\beta > 1$, vediamo come rappresentare in tale base un **numero reale α.**

Teorema: Rappresentazione dei Numeri Reali

Ogni numero reale $\alpha \neq 0$ può esprimersi univocamente, in base β, nella seguente forma:

$$\alpha = \pm(a_1\beta^{-1} + a_2\beta^{-2} + \ldots\ldots) \cdot \beta^p$$

$$= sign(\alpha) \cdot (\sum_{i=1}^{+\infty} a_i \beta^{-i}) \cdot \beta^p = sign(\alpha) \cdot m \cdot \beta^p \qquad (1)$$

dove

- $sign(x) = \begin{cases} 1 & se\ x > 0 \\ -1 & se\ x < 0 \end{cases}$ (funzione segno)
- $p \in Z^+$, numero intero positivo
- le cifre a_i sono numeri interi che soddisfano le seguenti condizioni:

  I. $0 \leq a_i \leq \beta - 1, \quad i = 1,2,.... \qquad a_1 \neq 0$

  II. può esistere, eventualmente, un indice $\mu > 1$ tale che $\forall$ i $\geq \mu$ sia $a_i = 0$, ma non esiste alcun indice $\nu > 0$ tale che $\forall$ i $\geq \nu$ sia $a_i = \beta - 1$.

Il numero $m = a_1\beta^{-1} + a_2\beta^{-2} + \ldots\ldots$ si chiama **mantissa** di α e soddisfa la condizione

$$\frac{1}{\beta} \leq m < 1 .$$

β^p si definisce **parte esponente** di α e p si chiama **esponente** di α.

Usando la **notazione posizionale** il numero $\alpha \neq 0$ si rappresenta

$$\alpha = \pm 0.a_1 a_2 a_3 \ldots \beta^p$$

Rappresentazione Normalizzata

- Esempio: Il numero reale π può essere espresso in base β=10 nella forma

$$\pi = +(3 \cdot 10^{-1} + 1 \cdot 10^{-2} + 4 \cdot 10^{-3} + 1 \cdot 10^{-4} + 5 \cdot 10^{-5} + 9 \cdot 10^{-6} .....) 10^1$$

e in rappresentazione normalizzata diviene

$$\pi = +(0.314159.....) 10^1 = 3.14159$$

2. I Numeri Finiti: Il Sistema Floating Point

Un **numero reale è caratterizzato da segno, esponente e mantissa**, in cui la mantissa può essere costituita da un numero illimitato di cifre.
Lavorando con il calcolatore, comunque, ognuna di queste parti deve essere limitata, ovvero rappresentata da un numero finito di bits. Da ciò nasce l'impossibilità di rappresentare nel calcolatore l'insieme dei numeri reali. Occorre, quindi, definire una rappresentazione approssimata dei numeri reali costituita da numeri la cui mantissa è rappresentata da un numero finito di cifre, che indichiamo con *t*.

Un numero finito si memorizza in un calcolatore in una parola costituita da n_b bits; in ogni bit si memorizza una cifra binaria 0 oppure 1.
Consideriamo un calcolatore con β=2 e parola di lunghezza n_b.
Tre sono le quantità da memorizzare nella parola: il segno, l'esponente e la mantissa.

Se la parola del calcolatore è di $n_b = 32$ bits la suddivisione standard è del tipo:

n_b varia nel calcolatore a seconda della precisione con cui si lavora:

$n_b = 32$ in singola precisione (1 bit per il segno; 8 bits per l' esponente, 23 bits per la mantissa)

$n_b = 64$ in doppia precisione (1 bit per il segno; 11 bits per l'esponente, 52 bits per la mantissa).

Quando inseriamo un numero reale α in un calcolatore, poiché per la rappresentazione dell' esponente vengono riservati solo un numero limitato di bits, si possono verificare i seguenti casi:

1. Il numero α è tale che il suo esponente p sia compreso tra un lower bound L ed un upper bound U, che indicano il minimo ed il massimo esponente rappresentabili con i bits a disposizione, cioè tale che $L \leq p \leq U$.

   $\Rightarrow$ allora α è un numero di macchina ed è rappresentabile nel calcolare considerato.

2. Se $p \notin [L,U]$. Il numero non può essere rappresentato nel calcolatore.

   - Se $p<L$ si verifca un **underflow** e solitamente si assume lo zero come valore approssimato, e ciò viene solitamente segnalato mediante warning.

   - Se $p>U$ si verifica un **overflow;** non si approssima, ma si segnala l'evento con un arresto del calcolo.

3. Nel caso in cui $p \in [L,U]$, cioè il numero α sia rappresentabile nel calcolatore, si possono verificare due casi:

a. le cifre a_i, per i>t sono tutte nulle, cioè α è rappresentabile esattamente con un numero finito t di cifre.

b. le cifre a_i, per $i>t$ non sono tutte nulle. In questo caso la rappresentazione di α sul calcolatore è data da $fl(\alpha)$, che si esprime nella forma,

$$fl(\alpha) = \begin{cases} 0 & se\ \alpha = 0 \\ \pm\, m_t \beta^p & se\ \alpha \neq 0 \end{cases}$$

$$m_t = a_1\beta^{-1} + a_2\beta^{-2} + \ldots\ldots + \tilde{a}_t\beta^{-t} \qquad con\ a_1 \neq 0.$$

dove

- nel caso di troncamento $\tilde{a}_t = a_t$;
- nel caso di arrotondamento: $\tilde{a}_t = a_t$ del numero $\tilde{\alpha} = \alpha + \frac{1}{2}\beta^{-t}$

In notazione posizionale si può esprimere come

$$fl(\alpha) = \pm 0.a_1a_2\ldots\tilde{a}_t\beta^p \ .$$

ES:

$\alpha = 0.54361$ t = 3

Troncamento $fl(\alpha)$=0.543

Arrotondamento $fl(\alpha)$=Tronc(0.54361+0.0005)=Tronc(0.5441)=0.544

Def: Insieme Floating Point

Ad ogni calcolatore che utilizza la base di rappresentazione β, usa t cifre per rappresentare la mantissa, ed in cui L ed U rappresentano il minimo ed il massimo esponente rappresentabile, si associa l' insieme dei Numeri di Macchina (**Insieme Floating Point**), cioè l'insieme dei numeri reali esattamente rappresentabili con quel calcolatore.

Esso è così definito:

$$F(\beta,t,L,U) = \{0\} \cup \left\{ \alpha \in \Re : \alpha = sign(\alpha)\left(\sum_{i=1}^{t} a_i \beta^{-i} \right)\beta^{p} \right\}$$

$$0 \le a_i \le \beta - 1, \quad i = 1,2,.... \qquad a_1 \neq 0 \text{ , } L \le p \le U.$$

L'insieme $F(\beta,t,L,U)$ è un sottoinsieme di R (insieme di tutti i numeri reali) ed ha cardinalità finita, come ci mostra il seguente teorema.

Teorema: Cardinalità dei numeri floating point

Sia $F(\beta,t,L,U)$ l'insieme dei numeri floating point.

Allora esso possiede $fl(0)$ e $(\beta\text{-}1)\cdot\beta^{t-1}$ (U-L+1) numeri positivi non uniformemente distribuiti in $[\beta^{L-1},\beta^{U})$ e altrettanti numeri negativi non uniformemente distribuiti in $(-\beta^{U},-\beta^{L-1}]$, ovvero

$$\#F = 2\cdot(\beta-1)\beta^{t-1}(U-L+1)+1$$

Dim:

Ragioniamo solo sui numeri positivi.

La mantissa più piccola che si può scrivere in base β e con t cifre è

$$1 \cdot \beta^{-1} + 0 \cdot \beta^{-2} + ...0 \cdot \beta^{-t} = \beta^{-1}$$

La mantissa più grande è

$$(\beta - 1) \cdot \beta^{-1} + (\beta - 1) \cdot \beta^{-2} + ...(\beta - 1) \cdot \beta^{-t} = \beta^{0} - \beta^{-1} + \beta^{-1} - \beta^{-2} + ... + \beta^{-t+1} - \beta^{-t} = 1 - \beta^{-t}$$

Il numero totale di mantisse è quindi

$$(1 - \beta^{-t})\beta^{t} - \beta^{-1}\beta^{t} + 1 = \beta^{t} - \beta^{0} - \beta^{t-1} + 1 = \beta^{t-1}(\beta - 1)$$

- Es: base β = 10, t =3 cifre per la matissa,

  la mantissa più piccola è $0.10000 = 10^{-1}$

  la mantissa più grande è $0.999 = 1-10^{-3}$

  il numero totale di mantisse è

  $(1 - 10^{-3})10^{3} - (10^{-1})10^{3} + 1 = 10^{3} - 1 - 10^{3} + 1 = 10^{3} - 10^{2} = 10^{2}(10-1)) = 900$

Poiché per ogni mantissa si hanno (U-L+1) esponenti possibili

$\Rightarrow$

$$\#\text{ elementi positivi in F} = (\beta-1)\,\beta^{t-1}\,(U-L+1)$$

$$\#F = 2 \cdot (\beta-1)\,\beta^{t-1}\,(U-L+1) + 1.$$

■

Si osserva che i numeri dell'insieme F sono ugualmente distribuiti per ciascun valore dell'esponente della base, ma non su tutto il range.

- Es:

  β=2, t=3, L=-1 e U=2

  Le possibili mantisse sono:

  m= .100, m= .101, m=.110, m=.111

Ogni mantissa è moltiplicata per $\beta^p \in \{2^{-1}, 2^0, 2^1, 2^2\}$

$\Rightarrow$ 4 mantisse possibili, 4 possibili esponenti

Da cui 16 numeri positivi, 16 negativi e lo zero

Da cui segue che #F=33

Tali numeri sono:

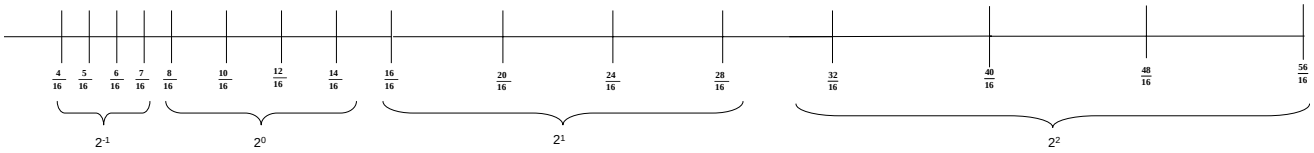
$$\left.\begin{aligned} .100\ 2^{-1} &= (1*2^{-1})\,2^{-1} = 1/2 \cdot 1/2 = 4/16 \\ .101\ 2^{-1} &= (1*2^{-1} + 1*2^{-3})\,2^{-1} = 5/16 \\ .110\ 2^{-1} &= (1*2^{-1} + 1*2^{-2})\,2^{-1} = 6/16 \\ .111\ 2^{-1} &= (1*2^{-1} + 1*2^{-2} + 1*2^{-3})\,2^{-1} = 7/16 \end{aligned}\right\}$$ 4 numeri relativi a 2^{-1}

$$\left.\begin{aligned} .100\ 2^{0} &= (1*2^{-1})\,1 = 1/2 = 8/16 \\ .101\ 2^{0} &= (1*2^{-1} + 1*2^{-3})\,1 = 10/16 \\ .110\ 2^{0} &= (1*2^{-1} + 1*2^{-2})\,1 = 12/16 \\ .111\ 2^{0} &= (1*2^{-1} + 1*2^{-2} + 1*2^{-3})\,1 = 14/16 \end{aligned}\right\}$$ 4 numeri relativi a 2^{0}

$$\left.\begin{aligned} .100\ 2^{1} &= (1*2^{-1})\,2 = 1/2 \cdot 2 = 16/16 \\ .101\ 2^{1} &= (1*2^{-1} + 1*2^{-3})\,2 = 20/16 \\ .110\ 2^{1} &= (1*2^{-1} + 1*2^{-2})\,2 = 24/16 \\ .111\ 2^{1} &= (1*2^{-1} + 1*2^{-2} + 1*2^{-3})\,2 = 28/16 \end{aligned}\right\}$$ 4 numeri relativi a 2^{1}

$$
\left.
\begin{aligned}
&.100\ \ 2^2 = (1*2^{-1})\ 4 = 1/2 \cdot 4 = 32/16 \\
&.101\ \ 2^2 = (1*2^{-1} + 1*2^{-3})\ 4 = 40/16 \\
&.110\ \ 2^2 = (1*2^{-1} + 1*2^{-2})\ 4 = 48/16 \\
&.111\ \ 2^2 = (1*2^{-1} + 1*2^{-2} + 1*2^{-3})\ 4 = 56/16
\end{aligned}
\right\} \text{4 numeri relativi a } 2^2
$$

Tali numeri sono equispaziati relativamente ad ogni singola base ma non globalmente.



Poiché in un calcolatore il numero reale $\alpha \neq 0$ è sostituito con $fl(\alpha)$, è fondamentale **stabilire una stima della differenza tra α e $fl(\alpha)$ relativa al valore di α stesso.**

Definizione: Si definisce **Errore Relativo** di arrotondamento di $\alpha \neq 0$ la quantità

$$\left| \frac{\alpha - fl(\alpha)}{\alpha} \right|$$

Teorema: Sia $\alpha \neq 0$ un numero reale rappresentato in base β come

$$\alpha = \pm\left(\sum_{i=1}^{+\infty} a_i \beta^{-i}\right) \cdot \beta^p, \quad a_1 \neq 0, \quad p \in [L, U]$$

Allora, se non si verifica un Overflow, si ha:

$$\left|\frac{\alpha - fl(\alpha)}{\alpha}\right| \leq K \cdot \beta^{1-t} = eps \qquad (3)$$

Con K=1 nel caso di troncamento

K=1/2 nel caso di arrotondamento

Dimostrazione:

a) Consideriamo il caso del troncamento

$$\alpha = \pm .a_1 a_2 .... \beta^p \qquad fl(\alpha) = \pm .a_1 a_2 ... a_t \beta^p$$

$$\alpha - fl(\alpha) = .a_{t+1} a_{t+2} ... \beta^{-t} \beta^p$$

$$\left|\frac{\alpha - fl(\alpha)}{\alpha}\right| = \frac{.a_{t+1} a_{t+2} ... \beta^{-t} \beta^p}{.a_1 a_2 ..... \beta^p} \quad con \quad .a_{t+1} a_{t+2} ... \leq 1 \quad e \quad .a_1 a_2 ..... \geq \beta^{-1}$$

$$\Rightarrow \quad \left|\frac{\alpha - fl(\alpha)}{\alpha}\right| \leq \frac{\beta^{-t}}{\beta^{-1}} = \beta^{1-t}$$

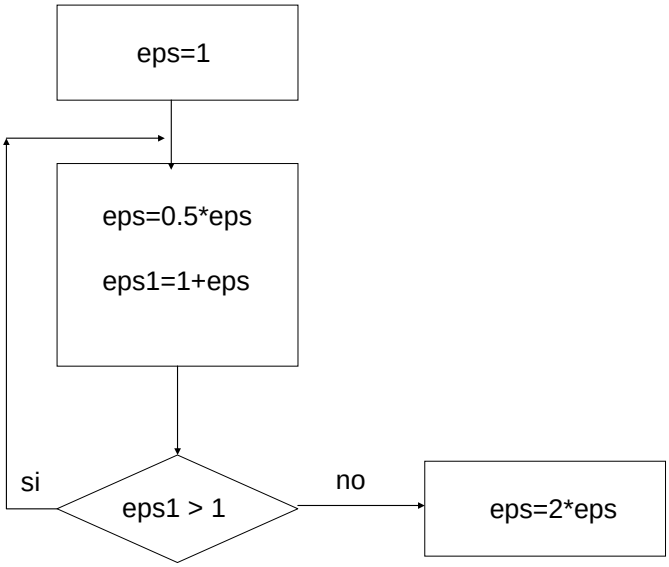
In modo analogo si dimostra per il caso dell'arrotondamento.

■

La quantità $eps = K \cdot \beta^{1-t}$ è detta **<u>precisione di macchina</u>** nel sistema floating point: **eps è il più piccolo numero positivo di macchina tale che sommato all'unità rende una quantità più grande di 1**:

***fl*(1+eps)>1**

Di seguito diamo l'algoritmo per il calcolo della **precisione di macchina**



Se poniamo $\varepsilon = \dfrac{\alpha - fl(\alpha)}{\alpha}$, dalla (3) si ha

$$|\varepsilon| \le eps \qquad \text{e} \quad fl(\alpha)=\alpha(1+\varepsilon)$$

Cioè il numero finito *fl(α)* è una perturbazione del numero reale α corrispondente.

3. ARITMETICA IN VIRGOLA MOBILE

Dato l'insieme dei numeri di macchina $F(\beta,t,L,U)$, è necessario definire su di esso delle nuove operazioni aritmetiche, cioè le operazioni in aritmetica di macchina, in quanto il risultato di un'operazione aritmetica classica eseguita fra numeri di macchina può non essere un numero di macchina.

Siano quindi $fl(x)\ e\ fl(y)\ \in F(\beta,t,L,U)$ e il risultato di un'operazione aritmetica tra *fl*(x) e *fl*(y), sia tale che $p \in [L,U]$

Si definiscono le operazioni di macchina (Operazioni Floating Point) e si indicano con i simboli $+^*$, $-^*$, $\times^*$, $/^*$, come segue:

$$fl(x) +^* fl(y) = fl(fl(x) + fl(y))$$
$$fl(x) \times^* fl(y) = fl(fl(x) \times fl(y))$$
$$fl(x) -^* fl(y) = fl(fl(x) - fl(y))$$
$$fl(x) /^* fl(y) = fl(fl(x) / fl(y))$$

Ricordando il risultato precedente, possiamo scrivere che

$$fl(x) \times^* fl(y) = fl(fl(x) \times fl(y)) = (fl(x) \times fl(y))(1+\varepsilon)$$

cioè il risultato di un'operazione Floating Point è una perturbazione di quello della corrispondente operazione reale.

Ovviamente questo vale per tutte le quattro operazioni floating point.

Osservazione:

Per le operazioni floating point non valgono le proprietà dell'aritmetica reale: in particolare non vale la proprietà associativa come dimostra il seguente esempio:

- **Es:** Sia β=2, e t=8. Dati i numeri di macchina

  a=$0.23371258*10^{-4}$

  b=$0.33678429*10^{2}$

  c=$-0.33677811*10^{2}$

Si valuti

1) $(a \oplus b) \oplus c =$

$$\begin{array}{r} 0.00000023371258 \cdot 10^2 \\ 0.33678429 \quad \cdot 10^2 \\ \hline 0.33678452371258 \cdot 10^2 \end{array}$$

$fl(a+b)=0.33678452 \cdot 10^2$

$$\begin{array}{rr} fl(a+b) & 0.33678452 \cdot 10^2 \\ c & -0.33677811 \cdot 10^2 \\ \hline & 0 \cdot 0000064110^2 \end{array}$$

$fl(\ fl(a+b)+c)= 0.64100000 \cdot 10^{-3}$

2) $a \oplus (b \oplus c)$

$$\begin{array}{r} 0.33678429 \cdot 10^2 \\ -0.33677811 \cdot 10^2 \\ \hline 0.00000618 \cdot 10^2 \end{array}$$

$fl(b+ c)=0.61800000 \cdot 10^{-3}$

$$\begin{array}{ll} a & 0.023371258\cdot 10^{-3} \\ fl(b+c) & 0.61800000\ \cdot 10^{-3} \\ \hline & 0.641371258\cdot 10^{-3} \end{array}$$

$fl($ a+fl(b+ c))= $0.64137126\cdot 10^{-3}$

Il risultato esatto è a+b+c = $0.641371258\cdot 10^{-3}$

Perciò si osserva che il risultato ottimale si ottiene con la 2), mentre con la 1) si ottiene un risultato con solo 3 cifre esatte.

Poiché tutte le volte che si esegue una operazione in aritmetica floating point si ottiene un risultato approssimato, uno dei principali compiti del calcolo numerico è quello di studiare come questi singoli errori si propagano e quale effetto danno sul risultato finale. Consideriamo il seguente esempio di radici di equazioni di II° grado.

Si vogliono calcolare le radici di equazione del tipo: $ax^2 + bx + c = 0$.

La formula risolvente è:

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

La formula risolutiva ha un grandissimo vantaggio: suggerisce una precisa successione di passi , i quali, partendo dai coefficienti *a,b,c* conducono alle radici x_1 e x_2.

In matematica pura è sottointeso che nell'algoritmo descritto si faccia uso dei numeri reali e si eseguano operazioni aritmetiche esatte, inclusa l'estrazione della radice. Come abbiamo visto però un calcolatore non è in grado di fare queste operazioni e nemmeno di memorizzare i numeri reali *a,b,c* in modo esatto ma deve sostituire i numeri reali con numeri in virgola mobile e le operazioni aritmetiche esatte con le operazioni floating point.

Si consideri l'equazione

$$x^2 - 6.433x + 0.009474 = 0$$

e si consideri il sistema decimale con t=4 cifre.

Applicando la formula risolutiva si ha

$$\alpha = (6.433)^2 = 0.6433 \cdot 10^1 \times 0.6433 \cdot 10^1 = 0.41383489 \cdot 10^1$$
$$\beta = 4 \times 0.009474 = 0.4 \cdot 10^1 \times 0.9474 \cdot 10^{-2} = 0.37896 \cdot 10^{-1}$$
$$\gamma = fl(\alpha) = 0.4138 \cdot 10^2$$
$$\delta = fl(\beta) = 0.3789 \cdot 10^{-1}$$
$$\gamma - \delta = 0.4138 \cdot 10^2 - 0.0003789 \cdot 10^2 = 0.4134211 \cdot 10^2$$
$$\eta = fl(\gamma - \delta) = 0.4134 \cdot 10^2$$
$$fl(\sqrt{\eta}) = fl(0.64296.. \cdot 10^1) = 0.6429 \cdot 10^1$$

Da cui si ottiene la radice $x_1 = \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$ pari a

$$x_1 = fl\left(\frac{0.6433 \cdot 10^1 - 0.6429 \cdot 10^1}{2}\right) = fl\left(\frac{0.0004 \cdot 10^1}{2}\right) = 0.2000 \cdot 10^{-2}$$

La sottrazione di due numeri quasi uguali ha portato alla cancellazione di cifre significative.

In aritmetica reale si sarebbe ottenuto:

$$x_1 = \frac{6.433 - \sqrt{41.383489 - 0.037896}}{2} = \frac{6.433 - \sqrt{41.345593}}{2}$$
$$= \frac{6.433 - 6.4300538}{2} = 0.0014731...$$

Il risultato ottenuto in aritmetica floating point è molto diverso da quello vero. Ciò è dovuto in particolare all'ultimo passaggio, dove facciamo la differenza fra numeri molto simili in modulo. Questo produce il fenomeno della "**cancellazione di cifre significative**", che è un'operazione molto pericolosa, che se è possibile va evitata.

Un algoritmo più efficiente per il calcolo delle radici di un'equazione di II° grado è il seguente:

calcolo la radice che, in base al segno di b, non porta problemi con la formula risolvente,

$$x_1 = \frac{-b + sign(b)\sqrt{b^2 - 4ac}}{2a}$$

mentre l'altra radice viene calcolata tenendo presente che: $x_1 x_2 = \frac{c}{a}$ o $x_1 + x_2 = -\frac{b}{a}$.

In generale, lo studio della propagazione degli errori è un argomento matematicamente difficile, per cui ci limitiamo a considerare solo i due casi che seguono.

Errore nel sommare n numeri finiti

Studieremo ora qual è l'errore da cui è affetta la somma "finita" (cioè ottenuta in aritmetica finita) di *n* numeri finiti.

Siano $x_1, x_2, ...x_n$ n numeri finiti da sommare.

L'algoritmo somma è il seguente:

S:=x_1

for i=2,…,n

S:=S+x_i

endfor

I passi eseguiti dall'algoritmo sono:

$S_2 = fl(S_1 + x_2)$

$S_3 = fl(S_2 + x_3)$

.

.

$S_n = fl(S_{n-1} + x_n)$

Quindi il risultato finale S_n differisce dal risultato teorico S.

L'errore relativo è $\left|\frac{S - S_n}{S}\right|$ è dato da:

$$\left|\frac{S - S_n}{S}\right| \le \frac{\left(|x_1|n + |x_2|(n-1) + \ldots + |x_n|\right)}{|S|} \cdot 1.01 \cdot eps \qquad (*)$$

Dimostrazione:

Per la dimostrazione ci si avvale del risultato del seguente lemma.

Lemma:

$\forall\ \varepsilon_i, i = 1,\ldots,k \quad |\varepsilon_i| \le eps$, se $k \cdot eps \le 0.01$, allora si ha che

$$\prod_{i=1}^{k}(1+\varepsilon_i) = 1 + \sigma_k \ ,$$

$$|\sigma_k| \le 1.01 \cdot k \cdot eps$$

Dall'algoritmo della somma si ha:

$S_2 = fl(S_1 + x_2) = (S_1 + x_2)(1+\varepsilon_2) = x_1(1+\varepsilon_1)(1+\varepsilon_2) + x_2(1+\varepsilon_2)$

$S_3 = fl(S_2 + x_3) = (S_2 + x_3)(1+\varepsilon_3) = x_1(1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3) + x_2(1+\varepsilon_2)(1+\varepsilon_3) + x_3(1+\varepsilon_3)$

.

$S_n = fl(S_{n-1} + x_n) = (S_{n-1} + x_n)(1+\varepsilon_n) = x_1(1+\varepsilon_1)\ldots(1+\varepsilon_n) + x_2(1+\varepsilon_2)\ldots(1+\varepsilon_n) + \ldots + x_n(1+\varepsilon_n)$

Dal lemma si ottiene:

$S_n - S = x_1(1+\sigma_n) + x_2(1+\sigma_{n-1}) + \ldots + x_n(1+\sigma_1) - x_1 - x_2 - \ldots - x_n =$

$S_n - S = x_1\sigma_n + x_2\sigma_{n-1} + \ldots + x_n\sigma_1$

$$\left|\frac{S - S_n}{S}\right| \le \frac{\left(|x_1|n + |x_2|(n-1) + \ldots + |x_n|\right)}{|S|} \cdot 1.01 \cdot eps$$

Da qui si vede che ogni addendo è moltiplicato per un peso fisso. Tale formula mostra che, assegnati i numeri finiti $x_1, x_2, ...x_n$, la maggiorazione **dell'errore relativo della loro somma "finita" è minima se si sommano questi numeri in modo che i loro valori assoluti siano in ordine crescente**, cioè: $|x_1| \le |x_2| \le ... \le |x_n|$. Infatti, così operando,ai pesi (1.01 eps)n, (1.01 eps)(n-1) più grandi si associano i numeri $|x_1|, |x_2|, ..., |x_n|$ più piccoli. Ne segue che l'errore pesa meno e si avranno perciò risultati più attendibili.

- **Es:** Assegnati i seguenti numeri finiti

  $x_1 = 0.9763 \cdot 10^{-5}$ $x_2 = 0.9895 \cdot 10^{-3}$ $x_3 = 0.1289 \cdot 10^{-1}$ $x_4 = 0.3429 \cdot 10^{0}$ $x_5 = 0.1234 \cdot 10^{1}$

  appartenenti all'insieme F(10,4,-6,2), la loro somma si determina, in aritmetica in base 10 con 4 cifre, costruendo la seguente sequenza di somme parziali.

$$x_1 = 0.9763 \cdot 10^{-5} \quad = 0.009763 \cdot 10^{-3}$$
$$x_2 = 0.9895 \cdot 10^{-3} \quad = 0.9895 \cdot 10^{-3}$$
$$x_1 + x_2 = 0.999263 \cdot 10^{-3}$$

$$s_2 = fl(x_1 + x_2) = 0.9992 \cdot 10^{-3} = 0.009992 \cdot 10^{-1}$$
$$x_3 = 0.1289 \cdot 10^{-1} \quad = 0.1289 \cdot 10^{-1}$$
$$s_2 + x_3 = 0.138892 \cdot 10^{-1}$$

$$s_3 = fl(s_2 + x_3) = 0.1388 \cdot 10^{-1} = 0.01388 \cdot 10^{0}$$
$$x_4 = 0.3429 \cdot 10^{0} \quad = 0.3429 \cdot 10^{0}$$
$$s_3 + x_4 = 0.35678 \cdot 10^{0}$$

$$s_4 = fl(s_3 + x_4) = 0.3567 \cdot 10^{0} = 0.03567 \cdot 10^{1}$$
$$x_5 = 0.1234 \cdot 10^{1} \quad = 0.1234 \cdot 10^{1}$$
$$s_4 + x_5 = 0.15907 \cdot 10^{1}$$

$$s_5 = fl(s_4 + x_5) = 0.1590 \cdot 10^{1}$$

I numeri sono stati sommati in ordine crescente. La soluzione esatte è S=0.1590789263. Perciò possiamo dire che la somma approssimata s_5 ne è una buona approssimazione a 4 cifre.

Se sommiamo i numeri in ordine decrescente si ha

$$s_2 = fl(x_4 + x_5) = fl(0.3429 \cdot 10^0 + 0.1234 \cdot 10^1) = fl(0.03429 \cdot 10^1 + 0.1234 \cdot 10^1) = 0.1576 \cdot 10^1$$

$$s_3 = fl(s_2 + x_3) = fl(0.1576 \cdot 10^1 + 0.1289 \cdot 10^{-1}) = fl(0.1576 \cdot 10^1 + 0.001289 \cdot 10^1) = 0.1588 \cdot 10^1$$

$$s_4 = fl(s_3 + x_2) = fl(0.1588 \cdot 10^1 + 0.9895 \cdot 10^{-3}) = fl(0.1588 \cdot 10^1 + 0.00009895 \cdot 10^1) = 0.1588 \cdot 10^1$$

$$s_5 = fl(s_4 + x_1) = fl(0.1588 \cdot 10^1 + 0.9763 \cdot 10^{-5}) = fl(0.1588 \cdot 10^1 + 0.0000009763 \cdot 10^1) = 0.1588 \cdot 10^1$$

La nuova somma s_5 è una cattiva approssimazione della somma esatta.

La formula (*) può essere ulteriormente modificata se sostituiamo x_i con $x_{\max}$

$$\left|\frac{S - S_n}{S}\right| \leq \frac{|x_{\max}|}{|S|} \cdot 1.01 \cdot eps \cdot \frac{n(n+1)}{2}$$

Se $|x_{\max}|$ è elevato ma S è piccola, come può avvenire se si sommano addendi di segno opposto e modulo simile

$$\Rightarrow \frac{|x_{\max}|}{|S|} \text{ è grande}$$

cioè l'errore **relativo può crescere molto.**

La somma di numeri di modulo simile ma a segni alterni può avere un errore molto grande perché la differenza porta a cancellazione di cifre significative.

In generale l'operazione di cancellazione è pericolosa se viene eseguita su numeri già arrotondati poiché porta all'amplificazione degli errori. Essa quindi è la causa

principale dell'instabilità degli algoritmi poiché produce un'amplificazione eccessiva degli errori che rende i risultati inattendibili.

Se dobbiamo effettuare differenze, occorre prima sommare tutti i termini positivi, poi sommare i moduli di tutti i termini negativi ed effettuare un'unica differenza.

- **Esempio 1:**

  Sia β=10, t=5.

  Calcolare e^x con lo sviluppo in serie $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... = \sum_{k=0}^{\infty} \frac{x^k}{k!}$

  in x = -5.5.

  $e^{-5.5} = 1.0000 - 5.5000 + 15.1250 - 27.730 + 38.1290 - 41.9420 + 38.4460 - 30.208 + 20.768 - 12.692 + 6.9803 - 3.4902.....$

  Se si effettua le somma dei primi 25 termini si ottiene $e^{-5.5} \approx 0.26363 \cdot 10^{-2}$

  Il valore corretto è $e^{-5.5} = 0.408677 \cdot 10^{-2}$

  Quando x è negativo lo sviluppo in serie è un procedimento che numericamente porta problemi.

  Poiché

  $$e^{-5.5} = \frac{1}{e^{5.5}} = \frac{1}{1 + 5.5 + 15.125 + 27.730 + 38.129 + ...} = 0.40865 \cdot 10^{-2}$$

  Possiamo applicare la formula usando tutti termini positivi. Il risultato che otteniamo è molto più preciso.

  Un'altra possibilità è quella di sommare tutti i termini positivi e tutti i termini negativi e fare un'unica differenza finale.

Errore nel moltiplicare n numeri finiti.

Siano $x_1, x_2, ...x_n$ n numeri di macchina

L'algoritmo prodotto è il seguente:

P:=x_1

for i=2,…,n

P := P×x_i

endfor

I passi eseguiti dall'algoritmo sono:

$P_2 = fl(x_1 \times x_2)$

$P_3 = fl(P_2 \times x_3)$

.

$P_n = fl(P_{n-1} \times x_n)$

L'errore relativo è

$$\left|\frac{P - P_n}{P}\right| \leq 1.01 \cdot (n-1) \cdot eps$$

Dimostrazione:

$P_2 = fl(x_1 \times x_2) = x_1 \times x_2 \times (1+\varepsilon_2)$

$P_3 = fl(P_2 \times x_3) = fl(x_1 \times x_2 (1+\varepsilon_2) \times x_3) = x_1 \times x_2 \times x_3 \times (1+\varepsilon_2) \times (1+\varepsilon_3)$

.

$P_n = fl(P_{n-1} \times x_n) = x_1 \times x_2 \ldots \times x_n \times (1+\varepsilon_2) \times (1+\varepsilon_3) \ldots (1+\varepsilon_n) = P \times (1+\sigma_{n-1})$

$\Rightarrow$

$$\left|\frac{P - P_n}{P}\right| = \left|\frac{P - P(1+\sigma_{n-1})}{P}\right| \leq 1.01 \cdot (n-1) \cdot eps$$

L'errore relativo nel prodotto quindi cresce linearmente con *n*.

Perciò posso concludere che il prodotto è un'operazione più sicura.

4. Propagazione dell'errore e Stabilità

Abbiamo visto che, nella somma di n numeri, per certi valori dei dati, l'errore relativo sul risultato può essere molto più grande dell'errore relativo sui dati.
E' necessario quindi studiare una nuova proprietà degli algoritmi: la **stabilità** che esprime il comportamento dell'algoritmo considerato rispetto alla propagazione degli errori.
Se un **algoritmo è stabile** la successione con cui esegue le operazioni non amplifica più dell'inevitabile gli errori iniziali. Se un **algoritmo è instabile,** il risultato che produce non è accettabile in quanto la successione delle operazioni che esegue amplifica in modo esagerato gli inevitabili errori dovuti all'utilizzo dell'aritmetica floating point.
Pertanto un algoritmo si dice stabile se l'influsso totale degli errori di arrotondamento sul risultato finale è maggiorabile dall'errore iniziale sui dati moltiplicato per una certa costante dipendente solo dalla struttura dell'algoritmo.

Uno dei principali compiti dell'analisi numerica è di trovare **algoritmi stabili (ben fatti)** cioè algoritmi in cui gli errori dovuti alla necessità di operare con l'aritmetica dei numeri finiti siano dell'ordine degli errori inevitabili.

Esempio:

Si vogliono calcolare gli integrali

$$E_n = \int_0^1 x^n e^{x-1} dx \quad n = 1,2,.....$$

Integrando per parti si ottiene

$$\int_0^1 x^n e^{x-1} dx = x^n e^{x-1} \Big|_0^1 - \int_0^1 n\, x^{n-1} e^{x-1} dx$$

$$E_n = 1 - nE_{n-1} \quad n = 2,3,.... \quad con \;\; E_1 = \frac{1}{e}.$$

Usando $\beta = 10$ **e** $t = 6$ si può usare questa formula ricorsiva per calcolare $E_1 \ldots E_9$

$$E_1 \approx 0.367879$$
$$E_2 \approx 0.264242$$
$$E_3 \approx 0.207274$$
$$E_4 \approx 0.170904$$
$$E_5 \approx 0.145480$$
$$E_6 \approx 0.127120$$
$$E_7 \approx 0.110160$$
$$E_8 \approx 0.118720$$
$$E_9 \approx -0.0684800$$

Inaspettatamente troviamo che, sebbene l'integrando $x^9 e^{x-1}$ sia positivo nell'intervallo (0,1), il nostro valore calcolato E_9 risulta negativo.

Il valore esatto è $E_9 = 0.0916123$

Gli unici errori che intervengono sono quello iniziale nel calcolare 1/e in quanto la formula è esatta nell'aritmetica finita. L'amplificazione dell'errore è dovuto al fatto che l'errore iniziale su E1 è stato amplificato della moltiplicazione per 9!, cioè

$9!\text{x } E1 = 9! \text{ x } 4.412\ 10^{-7} = 0.1601$

Bisogna cercare un altro algoritmo.

Si può pensare di partire da E_{20} e procedere all'indietro $E_{n-1} = \frac{1 - E_n}{n} \quad n = 20,...,3,2$

In questo modo l'errore su E_n diminuisce di un fattore 1/n. Si deve perciò cominciare con E_n con n >> 1 e l'errore diminuirà ad ogni passo. Per ottenere un valore iniziale si può osservare che

$$E_n = \int_0^1 x^n e^{x-1} dx \quad \leq \int_0^1 x^n dx = \frac{x^{n+1}}{n+1}\Big|_0^1 = \frac{1}{n+1}$$

cioè E_n va a 0 con $n \to +\infty$.

Partendo con un valore iniziale di $E_{20}=0$ in soli 5 passi l'errore è pari a $4*10^{-8}$ che è minore dell'errore di arrotondamento. In questo modo si ottiene $E_9 \approx 0.0916123$, che è accurato fino alla sesta cifra.

L'analisi della stabilità di un metodo numerico, iniziata da Von Neumann e Goldstine nel 1947, si può effettuare seguendo strategie alternative:

Analisi in avanti:

si stimano le variazioni sulla soluzione dovute sia a perturbazione nei dati, sia ad errori intrinseci al metodo numerico.

Analisi all'indietro:

si considera la soluzione calcolata con i numeri finiti come soluzione esatta di un problema perturbato e si studia a quale perturbazione sui dati corrisponde la soluzione trovata. Vedremo un'applicazione di questa tecnica nello studiare la stabilità della fattorizzazione di una matrice.