CORSO DI
ANALISI NUMERICA

A.A. 2011-2012

(Prof. A. Bellen)

CAPITOLO 1

TEORIA DEGLI ERRORI

In questo capitolo ci occuperemo della rappresentazione dei numeri reali in una base assegnata, della loro rappresentazione approssimata nelle unità aritmetiche dei computers e delle conseguenze che tale approssimazione produce nella esecuzione di una sequenza di calcoli progettata per risolvere un dato problema .

1. RAPPRESENTAZIONE POSIZIONALE DEI NUMERI REALI

Sia dato un intero B, che chiameremo **base**, ed un insieme di B simboli distinti, $C=\{0,1,2,.....B-1\}$, che chiameremo **cifre** della base B, rappresentanti i primi B numeri interi, da 0 a B-1. E' noto che ogni numero reale x si può rappresentare, nella base B, con il seguente allineamento di cifre di B:

$$x=a_n a_{n-1}...a_0,a_{-1}a_{-2}.....$$

intendendo con ciò:

$$x=a_n B^n+a_{n-1}B^{n-1}+.....+a_0B^0+a_{-1}B^{-1}+a_{-2}B^{-2}+..... \qquad (0.1)$$

La rappresentazione è unica quando si escluda l'allineamento periodico di periodo coincidente con la cifra massima della base. Infatti si dimostra facilmente (utilizzando la somma della serie geometrica) che:

$$0,(B-1)(B-1)…..(B-1)…=1$$

Quindi, in accordo con la (0.1), una rappresentazione periodica di periodo B-1 coincide con la rappresentazione ottenuta incrementando di una unità la cifra che precede il periodo, seguito dallo zero periodico (che naturalmente non si usa scrivere). Per esempio, nella base 8, si ha

$$x=453,6257777.... = 453,626000....$$

Quando le cifre che seguono la virgola sono tutte nulle il numero è **intero**, in caso contrario si dice **decimale**. In ogni caso il numero $y=a_n a_{n-1}...a_0$ rappresenta la *parte intera* di x mentre il numero $z=0,a_{-1}a_{-2}....$, che risulta sempre <1, ne rappresenta la *parte decimale*. Il numero x è sempre dato dalla somma della sua parte intera con la parte decimale. E' evidente che il carattere intero o decimale di x, ed anche la separazione della sua parte intera da quella decimale, non dipende dalla base B. Di conseguenza nei cambi di base è sufficiente trasformare separatamente la parte intera e la parte decimale di x.

A tale scopo osserviamo che per un numero intero $y=a_n a_{n-1}...a_0$ si ha:

$$y=a_nB^n+a_{n-1}B^{n-1}+.....+a_1B+a_0=(a_nB^{n-1}+a_{n-1}B^{n-2}+.....+a_1)B+a_0$$

da cui, essendo $a_0<B$, si deduce che a_0 è il resto della divisione di y per B il cui quoziente q_0 è dato da:

$$q_0=a_nB^{n-1}+a_{n-1}B^{n-2}+.....+a_1$$

A sua volta

$$q_0=(a_nB^{n-2}+a_{n-1}B^{n-3}+.....+a_2)B+a_1.$$

da cui si ricava che a_1 è il resto di q_0/B. Così procedendo, fino ad incontrare un quoziente minore di B, si ottengono tutte le cifre della rappresentazione cercata.

Per un numero decimale $z=0,a_{-1}a_{-2}.... =a_{-1}B^{-1}+a_{-2}B^{-2}+.....$ l'algoritmo per la determinazione delle cifre è diverso. Moltiplicando z per la base si ha:

$$Bz=B(a_{-1}B^{-1}+a_{-2}B^{-2}+.....)= a_{-1}+(a_{-2}B^{-1}+a_{-3}B^{-2}+ .....).$$

Essendo evidentemente

$$(a_{-2}B^{-1}+a_{-3}B^{-2}+ .....)<1,$$

la cifra a_{-1} rappresenta la parte intera del prodotto Bz, mentre

$$r_1=a_{-2}B^{-1}+a_{-3}B^{-2}+ .....$$

ne rappresenta la parte decimale. Analogamente a_{-2} è ottenuto come parte intera di Br_1 e così di seguito.

I numeri decimali si distinguono in **periodici** e non periodici. A loro volta i numeri periodici possono essere divisi tra quelli di periodo 0, e quindi finiti, e gli altri, ma quest'ultima non e' una suddivisione significativa dei numeri. Infatti nei cambiamenti di base valgono le seguenti proposizioni:

Se x *è razionale, allora è periodico in qualunque base*.
Se x *è periodico (in qualche base) allora è razionale*.

Di conseguenza un numero periodico rimane tale in qualunque base. La periodicita' non è dunque una caratteristica della rappresentazione ma dipende dalle proprieta' algebriche del numero.

$$x \text{ razionale} \Leftrightarrow x \text{ periodico.}$$

In generale non viene però conservato il carattere finito dei numeri periodici. Si pensi per esempio al numero x=1/3 che in base 10 si esprime con 0,3333...ed in base 3 con 0,1 (ma ricordiamoci che 0,1 non e' che l'abbreviazione convenzionale di 0,1000…..).

2. RAPPRESENTAZIONE "FLOATING POINT" E NUMERI DI MACCHINA

E' evidente che la moltiplicazione o la divisione di un numero x per la sua base di rappresentazione ha l'effetto di spostare la virgola di una posizione a destra o, rispettivamente, a sinistra nella sua rappresentazione posizionale.
Per esempio:

$$x=a_n a_{n-1}...a_0,a_{-1}a_{-2}..... \quad \Rightarrow \quad Bx=a_n a_{n-1}...a_0 a_{-1},a_{-2}.....$$

Di conseguenza ogni numero reale x, eccetto lo zero, si potra rappresentare come:

$$x=\pm B^p \cdot b_0,b_1...b_n..... \qquad \text{con } b_0 \neq 0$$

per una opportuna scelta dell'esponente p. Tale rappresentazione dei numeri reali è detta **rappresentazione normale**. Il numero p è detto **esponente** ed il numero b_0, $b_1...b_n.....$ è detto **mantissa** di x.
Nei calcolatori elettronici la base più comunemente usata è B=2 e le cifre sono indicate con 0 ed 1. A volte si usano altre basi, ma le cifre relative a queste basi sono sempre espresse in forma binaria (**rappresentazione mista**).

Per esempio in base B=10 il numero x=945 sarà rappresentato attraverso la sequenza delle sue tre cifre espresse in forma binaria:

1001 (=9), 0100 (=4), 0101 (=5):
x= 1001 0100 0101.

E' evidente lo spreco di memoria ti tale rappresentazione mista nella quale ogni cifra, per essere espressa in base 2, ha bisogno di 4 posizioni. Queste 4 posizioni non saranno mai sfruttate in tutte le loro possibili configurazioni, poichè la cifra massima è 1001 (=9).
Se la base è una potenza di 2, $B=2^p$ allora la rappresentazione mista è ottimale e coincide con quella in base 2.
Ad esempio in base B=8 (p=3) consideriamo il numero

$$x=537= 5 \cdot 8^2 +3 \cdot 8^1 +7$$

Le sue cifre, espresse in forma binaria, necessitano di 3 posizioni e sono:

101 (=5) 011 (=3) 111 (=7)

La rappresentazione mista sarà dunque:

x=101 011 111.

La situazione è ottimale in quanto le cifre della base 8 sono espresse da tutte e sole le terne di cifre binarie

E' facile verificare che tale allineamento di cifre binarie è lo stesso della rappresentazione di x in base 2. A tale scopo si osservi che $8=2^3$ e $8^2=2^6$ e, sviluppando le cifre di x in potenze di 2, si ha:

$$x= (101)\, 2^6 +(011)\, 2^3 +(111)=(1\cdot 2^2 +0\cdot 2+1)2^6 +(0\cdot 2^2 +1\cdot 2+1)2^3+(1\cdot 2^2 +1\cdot 2+1)$$
$$= 1\cdot 2^8 +0\cdot 2^7+1\cdot 2^6+0\cdot 2^5 +1\cdot 2^4+1\cdot 2^3+1\cdot 2^2 +1\cdot 2+1.$$

Quindi il passaggio di x da base 2 a base 2^p si ottiene direttamente raccogliendo le cifre di x "a gruppi di p" a partire dalla fine, ed esprimendo ogni gruppo con la corrispondente cifra della base 2^p

ESEMPIO: da base 2 a base 8

x= 11111101001
x= 3 7 5 1

ESEMPIO: da base 2 a base 16($=2^4$). Supponiamo che le cifre della base 16 siano (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

$$x=\underbrace{1000}_{8}\underbrace{1111}_{F}\underbrace{0010}_{2}\underbrace{1011}_{B}$$

$$x= 8\ F\ 2\ B$$

Poiche i registri delle unità aritmetiche e i dispositivi di memoria di un calcolatore consentono l'allocazione di un *numero finito* di cifre binarie non si possono, in generale, memorizzare tutte le cifre presenti nella rappresentazione normale del numero. Vengono quindi assegnate delle posizioni di memoria per il segno, per l'esponente e per la mantissa del numero. Schematizziamo, per il momento, la rappresentazione del numero nella forma:

segno	esponente	mantissa

L'insieme M dei numeri rappresentabili in tal modo su un determinato calcolatore sono detti **numeri di macchina** e la loro rappresentazione, detta rappresentazione in **virgola mobile** o **floating point**, e' del tipo:

$$z=\pm B^p \cdot b_0,b_1...b_t \qquad \text{con } b_0\neq 0$$

Ovviamente M e' un sottinsieme finito dei numeri reali e si osservi che la distanza tra due numeri di macchina successivi non e' costante ma dipende dal numero stesso. Infatti, detto z' il numero di macchina successivo a z in modulo

$$z'= \pm B^p \cdot(b_0,b_1...b_t+0,0\ldots1).$$

si ha:

$$|z-z'| = B^p \cdot 0,0\ldots1 = B^{p-t}$$

Ogni calcolatore adotta una particolare rappresentazione per i propri numeri di macchina e la funzione che associa al numero reale x il numero di macchina z che lo rappresenta o che lo approssima viene indicata genericamente con z=fl(x).

Standard IEEE

L'ampiezza dello spazio dedicato all'esponente ed alla mantissa, nonche' le modalita' della loro memorizzazione, varia da calcolatore a calcolatore e ne rappresenta una caratteristica fondamentale. Una configurazione molto diffusa è quella stabilita dallo standard dell'IEEE (Institute of Electrical and Electronic Engineers) che assegna un'area di 32 posizioni per la *semplice precisione* e 64 per la *doppia precisione* così distribuite

1 posizione per il segno
8 o 11 posizioni per l'esponente
23 o 52 posizioni per la mantissa.

Nella semplice precisione, le 8 posizioni binarie dedicate all'esponente consentono di rappresentare numeri da 0 a 2^8-1=255. In realtà gli estremi 0 e 255 sono riservati ad altri scopi e quindi i numeri utili sono compresi tra 1 e 254. Se p è l'esponente della rappresentazione normale di x, nell'area in oggetto verrà allocato il numero q=p+127. Di conseguenza p può variare tra -126 e 127.
Per quanto riguarda la mantissa della rappresentazione floating point di x, poiche' la parte intera e' sempre 1, nello spazio ad essa dedicato verra' memorizzata solo la sua parte decimale.

Poichè in base 2 la mantissa più piccola è 1.00000...0 e la più grande è 1.11111...1 (<2), il più piccolo numero positivo rappresentabile è 2^{-126} mentre il più grande è quasi 2^{128}. Analogamente nel caso della doppia precisione.

Se durante le operazioni si va al di sotto o al di sopra di tali valori, l'unità aritmetica segnala rispettivamente errore di **underflow** o **overflow**. Lo zero e' rappresentato dalla sequenza di 32 o 64 zeri.

Se le cifre decimali della mantissa di x sono in numero superiore a t dobbiamo operare qualche approssimazione per la sua memorizzazione. Tra i vari modi di operare ci sono sostanzialmente 2 strategie, quella del **troncamento** e quella dell' **arrotondamento**. Sia

$$x=\pm 2^{p} \cdot 1,b_1...b_t\, b_{t+1}....$$

il numero reale da memorizzare.

Il troncamento consiste semplicemente nel trascurare tutte le cifre decimali della mantissa successive alla t-esima. Avremo quindi:

$$tr(x)=\pm 2^{p} \cdot 1,b_1...b_t$$

e la sua memorizzazione nel registro sarà:

±	q	b_1.....b_t

con un errore

$$|x-tr(x)| \leq 2^{p-t}$$

L'arrotondamento invece è definito nel seguente modo:

$arr(x)=\pm 2^p \cdot 1,b_1...b_t$ se $b_{t+1}=0$

$arr(x)=\pm 2^p (1,b_1...b_t + 0,0\ldots1)$ se $b_{t+1}=1$

In altre parole arr(x) e' definito come il troncamento z=tr(x) se la t+1–esima cifra è 0 oppure il numero di macchina successivo di tr(x) se la t+1–esima cifra è 1, e piu' precisamente, quello dei due che e' più vicino ad x. Quindi l'errore sara' al più pari alla metà della distanza tra il numero z e il suo successivo (z').
Si osservi che, rispetto ai moduli, nel primo caso si ha una approssimazione per difetto (in senso debole), nel secondo per eccesso (in senso forte). In quest'ultimo caso si ha:

$$arr(x)=\pm 2^p \cdot 1,b_1...b_t \ \pm \ 2^p \cdot 0,0...1 = \pm\ 2^r \cdot 1,a_1...a_t$$

e la sua schematizzazione sarà:

segno	esponente	a_1.....a_t

I numeri tr(x) e arr(x) sono detti entrambi rappresentazioni in **virgola mobile** (**floating point**) del numero x.

L'insieme dei numeri rappresentabili su un determinato calcolatore sono detti **numeri di macchina**. Ogni calcolatore adotta una rappresentazione per i propri numeri di macchina e la funzione che associa al numero reale x il numero di macchina tr(x) oppure arr(x) viene indicata genericamente con fl(x) (approssimazione: **floating point** del numero reale x). Vogliamo ora analizzare l'errore che si compie nel sostituire x con fl(x).

Come abbiamo gia' osservato, rispetto ai moduli il troncamento è un'approssimazione sempre per difetto, mentre nell'arrotondamento il tipo di approssimazione dipende dalla prima cifra trascurata.

Per quanto riguarda il modulo dell'errore si ha:

$$|x - tr(x)| = 2^p \cdot 1,b_1...b_tb_{t+1}.... \ - \ 2^p \cdot 1,b_1...b_t = 2^p \cdot 0,0...0b_{t+1}.... \leq 2^{p-t}$$

$$|x-arr(x)| \leq \frac{1}{2}|z - tr(x)| \leq 2^{p-t-1}.$$

L'ultima relazione e' giustificata dal fatto che arr(x) e' dato da tr(x) o dal suo successivo
In generale:

$$|z-fl(z)| \leq \rho 2^{p-t} \qquad \text{dove } \rho = \begin{cases} 1 & \text{nel troncamento} \\ \frac{1}{2} & \text{nell'arrotondamento} \end{cases}$$

L'errore dipende dunque non solo dalla lunghezza t della mantissa ma anche da p, cioè dall'ordine di grandezza del numero z. Per quanto riguarda invece l'errore relativo, poichè $z \geq 2^p$, si ha:

$$\frac{|z-fl(z)|}{|z|} \leq \rho 2^{-t}$$

La quantità $\varepsilon := \rho 2^{-t}$ è detta **precisione di macchina** o **epsilon di macchina** o **unità di arrotondamento** ed è un parametro caratteristico del calcolatore.
In conclusione la relazione che lega un numero reale z alla sua rappresentazione fl(z) è:

$$fl(z)=z(1+u) \qquad \text{con } |u| \leq \varepsilon \qquad (0.2)$$

Si vede facilmente che ε rappresenta il più piccolo numero reale che sommato ad 1 da come risultato un numero floating maggiore di 1.
La conoscenza di questo parametro è essenziale per un controllo della propagazione dell'errore nell'esecuzione dei vari algoritmi che vedremo nel corso. L'epsilon di macchina si può calcolare con il seguente algoritmo:

```
e=1
i=0
while 1+e > 1
    e=e/2
    i=i+1
end
print (i-1,2*e)
end
```

I valori in output i-1 e 2*e sono rispettivamente il numero delle cifre decimali della mantissa e l'epsilon di macchina. La doppia precisione dello standard IEEE fornisce il valore $\varepsilon = 2^{-52} = 2.22044.... \cdot 10^{-16}$.

3. OPERAZIONI DI MACCHINA E PROPAGAZIONE DEGLI ERRORI.

E' chiaro che i numeri di macchina non sono un insieme chiuso rispetto alle operazioni elementari. Cioè il risultato di una operazione aritmetica tra due operandi appartenenti all'insieme dei numeri di macchina non è necessariamente un numero di macchina. Se per esempio si moltiplica per 2 il più grande numero di macchina ammissibile il risultato non sarà certamente un numero di macchina. Se si esegue la divisione di 1 per 3, pur essendo entrambi gli operandi numeri di macchina il risultato non lo è. Il prodotto di due numeri con t cifre decimali è, in generale, un numero con 2t cifre decimali, e così via. Detta $x*y$ una generica operazione tra due operandi presi nell'insieme dei numeri di macchina ($*:M\to R$, dove M è l'insieme dei numeri di macchina), indichiamo con $x\bar{*}y$ il risultato fornito dall'unità aritmetica. Nell'eseguire un'operazione $\bar{*}$, che chiameremo **operazione di macchina** ($\bar{*}:M\to M$), si commetterà in generale un'errore rispetto all'operazione $*$. Tale errore dipende dalla lunghezza dei registri dell'unità aritmetica che possono essere lunghi fino al doppio dei registri di memoria.

Per le nostre analisi faremo la seguente *ipotesi di lavoro*:

$$x\bar{*}y = fl(x*y)$$

In altre parole, una operazione di macchina coincide con l'approssimazione floating del risultato dell' operazione esatta. L'errore relativo sarà, in base alla (0.2),

$$\frac{x\bar{*}y - x*y}{x*y} = u \qquad \text{con } |u|\le\varepsilon.$$

Quindi una singola operazione tra numeri di macchina da luogo ad un errore relativo dell'ordine di ε ed è quindi accettabile dal punto di vista dell'approssimazione. Se però il risultato di questa operazione è a sua volta il dato di una operazione successiva, l'errore sul dato si *propagherà* nel risultato finale. La misura di tale propagazione dipenderà da quanto la seconda operazione è *sensibile* alle perturbazioni sui dati.
Si considerino per esempio le seguenti formule, che possono essere interpretate come tanti algoritmi per il calcolo dello stesso numero:

$$\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3 = \left(\sqrt{2}-1\right)^6 = \left(3-2\sqrt{2}\right)^3 = \left(5\sqrt{2}-7\right)^2 =$$

$$= 99-70\sqrt{2} = \frac{1}{\left(\sqrt{2}+1\right)^6} = \frac{1}{99+70\sqrt{2}} = 0.0050506339...$$

Tutte richiedono il calcolo preventivo di $\sqrt{2}$ (=1,4142135...) che sarà necessariamente dato in forma approssimata. Supponiamo inoltre che i calcoli necessari per ogni formula

siano eseguiti senza errore. Per esaltare l'effetto voluto consideriamo le seguenti approssimazioni di $\sqrt{2}$:

$\sqrt{2} \approx 1.4$ con errore relativo u=0,01
$\sqrt{2} \approx 1.414$ con errore relativo u=0.00015

Si ottengono i seguenti valori:

		$\sqrt{2} \approx 1.4$	errore relativo	$\sqrt{2} \approx 1.414$	errore relativo
1	$\left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3$	0,00463	0,1	0,0050441	0,002
2	$\left(\sqrt{2}-1\right)^6$	0,004096	0,2	0,0050349	0,002
3	$\left(3-2\sqrt{2}\right)^3$	0,008	0,6	0,0050884	0,006
4	$\left(5\sqrt{2}-7\right)^2$	0	1	0,0049	0,002
5	$99-70\sqrt{2}$	1	200	0,02	3
6	$\frac{1}{\left(\sqrt{2}+1\right)^6}$	0,00523	0,05	0,0050533	0,0006
7	$\frac{1}{99+70\sqrt{2}}$	0,0050761	0,0052	0,0050510	0,000074

Dalla precedente tabella si osserva che lo stesso errore sul dato $\sqrt{2}$ produce effetti molto diversi sulle varie formule proposte. Alcune di esse riducono l'errore sul dato, altre lo amplificano enormemente. Le formule 5 e 7 sono rispettivamente la più *instabile* e la più *stabile* rispetto alle perturbazioni sul dato $\sqrt{2}$. In particolare la formula 5 amplifica l'errore relativo su $\sqrt{2}$ di un fattore 20.000 mentre la formula 7 lo riduce di un fattore 0,5.
Per capire l'origine di tale diversità di comportamento, basta interpretare le 7 formule proposte come i valori delle seguenti funzioni nel punto x=$\sqrt{2}$:

$$\left(\frac{x-1}{x+1}\right)^3 \;;\; (x-1)^6 \;;\; (3-2x)^3 \;;\; (5x-7)^2 \;;\; 99-70x \;;\; \frac{1}{(x+1)^6} \;;\; \frac{1}{99+70x}$$

E' chiaro che la formula corrispondente alla funzione che possiede la derivata in x=$\sqrt{2}$ più grande, è quella che maggiormente propaga l'errore sul dato.

Supponiamo più in generale di avere dei dati $x_1, x_2,..,x_n$ ed una funzione di questi ultimi che indicheremo con

$$z=f(x_1, x_2,..,x_n).$$

Supponiamo ora di perturbare i dati e di calcolare la funzione f sui dati perturbati:

$$\bar{z} = f(\bar{x_1},\bar{x_2},...,\bar{x_n})$$

dove:

$$\bar{x_1}=x_1(1+\varepsilon_1)$$
$$\bar{x_2}=x_2(1+\varepsilon_2)$$
$$.....$$
$$\bar{x_n}=x_n(1+\varepsilon_n)$$

Per effetto di tali perturbazioni sui dati, otteniamo una perturbazione relativa sul risultato del tipo $\frac{\bar{z}-z}{z}$. Riguardo alla relazione tra queste perturbazioni, diamo la seguente definizione:

Definizione: Diremo che il problema $z=f(x_1, x_2,..,x_n)$ è **stabile** (o **ben posto**) se la perturbazione sul risultato è dello stesso ordine di grandezza delle perturbazioni sui dati.

Per una stima della quantità $\frac{\bar{z} - z}{z}$ consideriamo lo sviluppo di Taylor della funzione f troncato ai termini del prim'ordine:

$$\bar{z} = f(\bar{x_1},\bar{x_2},...,\bar{x_n}) = f(x_1, x_2,..,x_n) + x_1\varepsilon_1\frac{\partial f}{\partial x_1} + x_2\varepsilon_2\frac{\partial f}{\partial x_2} +...+ x_n\varepsilon_n\frac{\partial f}{\partial x_n}$$

$$\bar{z} = z + x_1\varepsilon_1\frac{\partial f}{\partial x_1} + x_2\varepsilon_2\frac{\partial f}{\partial x_2} +...+ x_n\varepsilon_n\frac{\partial f}{\partial x_n}$$

$$\frac{\bar{z} - z}{z} = \frac{x_1}{z}\frac{\partial f}{\partial x_1}\varepsilon_1 + \frac{x_2}{z}\frac{\partial f}{\partial x_2}\varepsilon_2 +... + \frac{x_n}{z}\frac{\partial f}{\partial x_n}\varepsilon_n$$

Otteniamo cosi la seguente stima dell'errore relativo su z in funzione degli errori relativi sui dati:

$$\frac{\bar{z} - z}{z} = K_1\varepsilon_1 + K_2\varepsilon_2 +... + K_n\varepsilon_n$$

dove i numeri $K_1= \frac{x_1}{z}\frac{\partial f}{\partial x_1}$, $K_2 = \frac{x_2}{z}\frac{\partial f}{\partial x_2}$ ... $K_n = \frac{x_n}{z}\frac{\partial f}{\partial x_n}$ sono detti **indici di condizionamento del problema** .

Gli indici K_i rappresentano i fattori di amplificazione delle perturbazioni relative sui dati. Se $|K_i| \approx 1$ diremo che il problema è stabile o ben posto, mentre se qualche $|K_i| >> 1$ diremo che il problema è instabile o mal posto.
Consideriamo ancora la formula 5 dell'esempio precedentemente trattato ed osserviamo che la corrispondente funzione 99 -70x ha, in $x=\sqrt{2}$, la derivata uguale a -70 e quindi l'indice di condizionamento è

$$|K| \approx \frac{\sqrt{2}\ 70}{0.00505...} \approx 20000$$

che corrisponde esattamente all'amplificazione dell'errore che avevamo già osservato. Anche per la formula 7 la precedente analisi differenziale fornisce un indice di condizionamento perfettamente adeguato ai risultati della tabella ($|K|=0.5$).
Il lettore verifichi che la moltiplicazione e la divisione sono operazioni ben poste per ogni coppia di operandi, mentre la somma è ben posta per operandi di segno uguale. Per quando riguarda invece la somma di addendi discordi, cioè la sottrazione, essa può essere mal condizionata.

Infatti, detti

$$z = x + y$$

e

$$\bar{z}=x(1+\varepsilon_x) + y(1+\varepsilon_y),$$

si ha:

$$\bar{z}=x+x\varepsilon_x + y + y\varepsilon_y = z + x\varepsilon_x + y\varepsilon_y$$

$$\frac{\bar{z}-z}{z} = \frac{x}{z}\varepsilon_x + \frac{y}{z}\varepsilon_y$$

Se x ed y sono di segno opposto almeno uno degli indici di condizionamento $x/z(=K_1)$ e $y/z(=K_2)$ è maggiore di 1 e possono essere grandi quando la somma z è piccola rispetto agli addendi. La sottrazione tra addenti quasi uguali è dunque l'unica operazione elementare che risulta instabile
Si osservi che questo è esattamente ciò che accade nella formula 5 degli esempi precedenti.

Consideriamo più in generale un **algoritmo** cioè una sequenza ordinata di operazioni elementari con le quali si ottiene la soluzione di un determinato problema. Siccome ogni operazione della sequenza verrà eseguita come operazione di macchina,

ogni passo di questa sequenza da origine a due componenti di errore. La prima è dovuta all'approssimazione delle operazioni di macchina e, in accordo con la nostra ipotesi di lavoro, non supera l'unità di arrotondamento. In qualche caso questa componente di errore può essere nulla. La seconda, più importante e "pericolosa", è dovuta alla propagazione dell'errore accumulato fino al passo precedente. Essa dipende esclusivamente dalla natura *stabile* o *instabile* dell'operazione che si compie nel passo considerato.

Per meglio afferrare questo punto consideriamo il seguente esempio.

Sia dato il problema

$$z=a + b + c$$

per il quale consideriamo due possibili algoritmi:

1° algoritmo: $z= (a + b) + c$

2° algoritmo: $z= a + (b + c)$

L'analisi dell'errore nel primo algoritmo fornisce:

$$\xi = fl(a + b) = (a + b)(1 + \varepsilon_1) \qquad |\varepsilon_1| \leq \varepsilon$$
$$\bar{z} = fl(\xi + c) = (\xi + c)(1 + \varepsilon_2) \qquad |\varepsilon_2| \leq \varepsilon$$

da cui si ottiene, trascurando il termine $\varepsilon_1\varepsilon_2$:

$$\bar{z} = ((a + b)(1 + \varepsilon_1) + c)(1 + \varepsilon_2)=z + (a + b)\varepsilon_1 + z\,\varepsilon_2 \qquad (0.3)$$
$$\frac{\bar{z} - z}{z} = \frac{a + b}{z}\varepsilon_1 + \varepsilon_2$$

Per il secondo algoritmo si ottiene una analoga stima dell'errore relativo del tipo:

$$\frac{\bar{z}' - z}{z} = \frac{b + c}{z}\varepsilon'_1 + \varepsilon'_2 \qquad \text{con} \qquad |\varepsilon'_1| \leq \varepsilon \quad \text{e} \quad |\varepsilon'_2| \leq \varepsilon.$$

E' evidente che tra i due algoritmi è più stabile quello corrispondente al più piccolo tra gli indici di condizionamento $\frac{a + b}{z}$ e $\frac{b + c}{z}$.

Si eseguano i calcoli e l'analisi dell'errore dei due algoritmi sui seguenti dati:

$$a = 0{,}14254679 \cdot 10^{-2}$$
$$b = -0{,}18437533 \cdot 10^{2}$$

$c = 0{,}18436111 \cdot 10^2$.

Complementi alla teoria degli errori

Un approccio generale per l'analisi dell'errore di un algoritmo applicato ad un assegnato problema è fornito dalla seguente **analisi all'indietro**. Essa si basa sul fatto che un algoritmo per il calcolo di $z=f(x_1, x_2,..,x_n)$ eseguito con operazioni di macchina fornisce un risultato effettivo $\bar{z}$ corrispondente al valore *esatto* della funzione f calcolata su dati perturbati $\bar{x}_i=x_i(1+\varepsilon_i)$:

$$\bar{z} = f(\bar{x_1},\bar{x_2},...,\bar{x_n}) .$$

L'analisi all'indietro consiste nello stimare i valori ε_i in termini dell'unità di arrotondamento:

$$\varepsilon_i=c_i\varepsilon .$$

I numeri c_i sono chiamati **indici di condizionamento dell'algoritmo**.

Definizione: Un algoritmo si dirà **stabile** o **ben condizionato** se fornisce la soluzione esatta di un problema perturbato con errori relativi ε_i dell'ordine di grandezza dell'unità di arrotondamento.

Una volta eseguita l'analisi all'indietro si esegue l'analisi della stabilità del problema in funzione delle perturbazioni ε_i sui dati. Si ottiene così la stima dell'errore:

$$\frac{\bar{z} - z}{z} = K_1\varepsilon_1 + K_2\varepsilon_2 +... + K_n\varepsilon_n=(K_1c_1 + K_2c_2 +... + K_nc_n)\varepsilon$$

Con questa analisi si può distinguere la stabilità o instabilità del problema da quella dell'algoritmo.

Nell'esempio precedente si vede, dalla (0.3), che il primo algoritmo equivale al calcolo di :

$$\bar{z} = a(1 + \sigma) + b(1 + \sigma) + c(1+\varepsilon_2) \qquad \text{con} \quad |\sigma|\le 2\varepsilon \qquad \text{e } |\varepsilon_2| \le \varepsilon$$

mentre il secondo equivale a:

$$\bar{z}' = b(1 + \sigma') + c(1 + \sigma') + a(1+\varepsilon'_2) \qquad \text{con} \quad |\sigma'|\le 2\varepsilon \qquad \text{e } |\varepsilon'_2| \le \varepsilon$$

Gli algoritmi appaiono entrambi stabili inquanto forniscono la somma esatta di tre addendi affetti da una perturbazione non superiore a 2ε . Quando si va però ad analizzare l'effetto di tali perturbazioni sul problema z=a+b+c si ottiene:

$$\frac{\bar{z}-z}{z}=\frac{a+b}{z}\sigma+\frac{c}{2}\varepsilon_2 \qquad \text{per il primo algoritmo}$$

$$\frac{\bar{z}-z}{z}=\frac{b+c}{z}\sigma'+\frac{a}{z}\varepsilon_2' \qquad \text{per il secondo algoritmo.}$$

Rispetto a tali perturbazioni sui dati, il problema può essere ben o mal condizionato, dipende dagli indici di condizionamento $\frac{a+b}{z}$, $\frac{b+c}{z}$, $\frac{c}{z}$ e $\frac{a}{z}$.

Per i valori proposti di a,b e c il problema è mal condizionato ed a ciò è dovuta la disastrosa propagazione dell'errore che si riscontra. In particolare si osservi che il primo algoritmo è peggiore del secondo.

Come applicazione della teoria svolta, supponiamo di voler calcolare il valore del polinomio:

$$p(x)=a_n x^n+a_{n-1}x^{n-1}+.....+a_1 x+a_0$$

Ciò può essere fatto sostanzialmente in due modi, attraverso lo **schema naturale** e lo **schema di Horner**.

Schema naturale:

dati: x, a_0 , ..., a_n
s=1
p=a_0
per i=1,2,...,n
    s=s∗x
    p=p+a_i∗s
fine
scrivi p

Il costo computazionale è di n somme e 2n moltiplicazioni.

Schema di Horner:

Il metodo di Horner si basa sul fatto che il polinomio p(x) può essere scritto nella seguente forma:

$$p(x)=(..(((a_n)x+a_{n-1})x+a_{n-2})x+.....+a_1)x+a_0$$

da ciò si ricava lo schema:

dati: x, a_0 , ..., a_n
p=a_n
per i=1,2,...,n
    p=p∗x+a_{n-i}
fine
scrivi p

Un primo vantaggio dell'algoritmo di Horner consiste nel minore costo computazionale che si riduce a n somme ed n moltiplicazioni.
Il principale vantaggio però sta nella sua maggiore stabilità rispetto allo schema naturale. Per vedere ciò eseguiamo le analisi all'indietro.

Analisi all'indietro dello schema naturale:

Indicando come al solito con $\overline{f(w)}$ il risultato di macchina della funzione f(w), si vede che l'ultimo passo dell'algoritmo naturale è:

$$\overline{\sum_{i=0}^{n} a_i x^i} = fl\left(\overline{\sum_{i=0}^{n-1} a_i x^i} + fl\left(a_n \cdot fl\left(x \cdot \overline{x^{n-1}}\right)\right)\right) =$$

$$\left\{\overline{\sum_{i=0}^{n-1} a_i x^i} + fl\left(a_n \cdot fl\left(x \cdot \overline{x^{n-1}}\right)\right)\right\}(1+\varepsilon_n) =$$

$$\left\{\overline{\sum_{i=0}^{n-1} a_i x^i} + \left(a_n \cdot fl\left(x \cdot \overline{x^{n-1}}\right)\right)(1+\mu_n)\right\}(1+\varepsilon_n) =$$

$$\left\{\overline{\sum_{i=0}^{n-1} a_i x^i} + \left(a_n \cdot x \cdot \overline{x^{n-1}}\right)(1+\rho_n)(1+\mu_n)\right\}(1+\varepsilon_n)$$

Poichè :

$$\overline{x^{n-1}} = x\,\overline{x^{n-2}}(1-\rho_{n-1}) = \ldots = x^{n-1}(1-\rho_2)\ldots(1-\rho_{n-1})$$

si ottiene:

$$\overline{\sum_{i=0}^{n} a_i x^i} = \left\{\overline{\sum_{i=0}^{n-1} a_i x^i} + a_n x^n (1+\rho_2)(1+\rho_3)\ldots(1+\rho_n)(1+\mu_n)\right\}(1+\varepsilon_n)$$

dove tutte le perturbazioni presenti sono di modulo inferiore all'unità di arrotondamento ε. Per semplificare la notazione ed i calcoli, supponiamo che per ogni prodotto di k perturbazioni di questo tipo si abbia:

$$\prod_{i=0}^{k-1}(1+\lambda_i) \cong (1+k\varepsilon).$$

Si ottiene quindi:

$$\overline{\sum_{i=0}^{n} a_i x^i} \approx \left\{\overline{\sum_{i=0}^{n-1} a_i x^i}\right\}(1+\varepsilon) + a_n x^n (1+(n+1)\varepsilon)$$

Applicando ricorsivamente la formula appena trovata si ha:

$$\overline{\sum_{i=0}^{n-1} a_i x^i} \approx \left\{\overline{\sum_{i=0}^{n-2} a_i x^i}\right\}(1+\varepsilon) + a_{n-1} x^{n-1}(1+n\varepsilon)$$

dalla quale si ottiene:

$$\overline{\sum_{i=0}^{n} a_i x^i} \approx \left\{\overline{\sum_{i=0}^{n-2} a_i x^i}\right\}(1+2\varepsilon) + a_{n-1} x^{n-1}(1+(n+1)\varepsilon) + a_n x^n (1+(n+1)\varepsilon)$$

Continuando con le sostituzioni si trova:

$$\overline{\sum_{i=0}^{n} a_i x^i} \approx a_0(1+n\varepsilon) + a_1 x(1+(n+1)\varepsilon) + a_2 x^2(1+(n+1)\varepsilon) + \ldots + a_n x^n (1+(n+1)\varepsilon).$$

Abbiamo quindi scoperto che gli errori di arrotondamento nel metodo naturale hanno l'effetto di produrre il valore, nello stesso punto x, di un polinomio

$$p'(x)=a'_n x^n + a'_{n-1} x^{n-1} + \ldots\ldots\ldots + a'_1 x + a'_0$$

con coefficienti perturbati secondo le stime:

$$a'_0 \approx a_0(1+n\varepsilon)$$

$$a'_i \approx a_i(1+(n+1)\varepsilon) \qquad i=1,2,\ldots,n$$

Analisi all'indietro dello schema di Horner:

Analogamente a quanto fatto per lo schema naturale, l'ultimo passo del ciclo dello schema di Horner è:

$$\overline{p_n} = fl\left(a_0 \cdot fl\left(x \cdot \overline{p_{n-1}}\right)\right)$$

con le stesse semplificazioni del caso precedente, si ottengono le stime:

$$\begin{aligned}\overline{p_n} &\approx \left(a_0 \cdot fl\left(x \cdot \overline{p_{n-1}}\right)\right)(1+\varepsilon) \\ &\approx \left(a_0 \cdot \left(x \cdot \overline{p_{n-1}}\right)(1+\varepsilon)\right)(1+\varepsilon) \\ &\approx a_0(1+\varepsilon) + x \cdot \overline{p_{n-1}}(1+2\varepsilon).\end{aligned}$$

In modo analogo si trova:

$$\overline{p_{n-1}} \approx a_1(1+\varepsilon) + x \cdot \overline{p_{n-2}}(1+2\varepsilon)$$

da cui:

$$\overline{p_n} \approx a_0(1+\varepsilon) + x \cdot a_1(1+3\varepsilon) + x^2 \overline{p_{n-2}}(1+4\varepsilon).$$

Procedendo in modo ricorsivo si ottiene infine che lo schema di Horner equivale al calcolo esatto, nello stesso punto x, di un polinomio

$$p'(x)=a'_n x^n + a'_{n-1} x^{n-1} + \ldots\ldots\ldots + a'_1 x + a'_0$$

con coefficienti perturbati secondo le stime:

$$a'_i \approx a_i(1+(2i+1)\varepsilon) \qquad i=0,1,\ldots,n-1$$

$$a'_n \approx a_n(1+2n\varepsilon).$$

Per concludere l'analisi della stabilità e per poter fare un confronto tra i due schemi occorre studiare la stabilità del valore del polinomio p(x) rispetto alle perturbazione sui coefficienti.
L'analisi differenziale per il polinomio p, visto come funzione dei suoi coefficienti:

$$p(x)= f(x,a_0,a_1,\ldots,a_n)$$

rispetto al polinomio perturbato

$$\overline{p(x)}=f(x,a_0(1+\varepsilon_0),a_1(1+\varepsilon_1),...,a_n(1+\varepsilon_n))$$

fornisce:

$$\frac{\bar{p}-p}{p}=\frac{a_0}{p}\frac{\partial f}{\partial a_0}\varepsilon_0+\frac{a_1}{p}\frac{\partial f}{\partial a_1}\varepsilon_1+...+\frac{a_n}{p}\frac{\partial f}{\partial a_n}\varepsilon_n$$

$$=\frac{1}{p}(a_0\varepsilon_0+a_1\varepsilon_1 x+....+a_{n-1}\varepsilon_{n-1}x^{n-1}+a_n\varepsilon_n x^n)$$

dove ε_i sono le perturbazioni relative sui coefficienti.
Per lo schema naturale si ha:

$$\frac{\bar{p}-p}{p}=\frac{\varepsilon}{p}(na_0+(n+1)(a_1x+....+a_nx^n))$$

mentre per lo schema di Horner si ha:

$$\frac{\bar{p}-p}{p}=\frac{\varepsilon}{p}(a_0+3a_1x+5a_2x^2....+(2n-1)a_{n-1}x^{n-1}+2na_nx^n)).$$

Dal confronto si osserva che, sebbene per valori grandi di x lo schema naturale appare in generale leggermente più stabile, lo schema di Horner è molto più stabile per valori di x piccoli. Inoltre accade che nelle comuni applicazioni, quali le approssimazioni in serie di potenze, i coefficienti dei termini di grado alto sono piccoli e ciò riduce l'eventuale vantaggio dello schema naturale.