Analisi degli errori

Corso di Calcolo Numerico, a.a. 2005/2006

Francesca Mazzia

Dipartimento di Matematica
Università di Bari

6 Marzo 2006

Errori Computazionali

- errori di arrotondamento: rappresentazione dei dati ed esecuzione delle operazioni in aritmetica finita
- errore di discretizzazione: approssimazione discreta di un problema continuo
- errore di convergenza: numero finito di passi in un procedimento iterativo

La storia di ARIANNE 5

- Fu lanciato il 4 Giugno 1996
- Tutto andò bene per 36 secondi
- Al secondo 37 cambiò rotta e si autodistrusse
- Il problema fu causato da un sistema di riferimento inerziale che cercò di convertire un numero a 64 bit a virgola mobile in un numero a 16 bit a virgola fissa
- Il numero era troppo grande e causò un errore di overflow, inviando un messaggio diagnostico al computer di bordo, il computer di bordo interpretò il messaggio diagnostico come un dato di volo.

Il missile Patriot

- Durante la guerra del golfo, il 25 Febbraio 1991, uno Scud Iracheno riusci a superare il sistema antimissile Patriot e colpì una caserma uccidendo 28 persone.
- Per seguire il movimento dello Scud il sistema doveva determinare l'intervallo fra due valori di misurazione sottraendo due valori di un timer.
- Il tempo in decimi di secondi era contenuto in registri di interi.
- Per calcolare l'intervallo, i valori nei registri dovevano essere convertiti in reali moltiplicandoli per 0.1. I registri utilizzati erano a virgola fissa a 24 bit.
- Un decimo non ha una rappresentazione binaria finita. Di conseguenza l'intervallo fu calcolato con errore.

Il Missile Patriot

- Il sistema era in funzione da cento ore, l'intervallo fu calcolato con una inaccuratezza di circa 0.34 sec, che corrisponde a un errore di posizione di 600 metri.
- Il numero 1/10 corrisponde a

$$(1/10)_2 = 0.0001100110011001100110011001100....$$

- Il registro a 24 bit nel Patriot memorizzava

$$0.00011001100110011001100$$

introducendo un errore di

$$0.00000000000000000000011001100...$$

$$\text{in decimale :} 0.000000095 = 9.5e-08.$$

- Moltiplicando con il numero in decimi di secondo dopo 100 ore abbiamo

$$0.000000095 \times 100 \times 60 \times 60 \times 10 = 0.34$$

Rappresentazione dei numeri.

- L'utilizzo in modo corretto del calcolatore per fare calcoli di tipo scientifico, richiede la conoscenza di come sono rappresentati i numeri e degli errori che derivano da questa rappresentazione.
- L'uso dei numeri reali richiede una attenzione particolare, essendo questi infiniti, mentre il calcolatore ci da la possibilità di rappresentarne solo un numero finito.
- La nostra notazione per rappresentare i numeri è una notazione posizionale a base 10. Ciò significa che se scriviamo 123 intendiamo esprimere il numero:

$$1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0.$$

Rappresentazione dei numeri.

Notazione scientifica:

$$\pm\gamma_0.\gamma_1\gamma_2\ldots\gamma_t\cdot\cdot\cdot\cdot N^q, \quad 0 \leq \gamma_i \leq N-1$$

mantissa: $m = \gamma_0.\gamma_1\gamma_2\ldots\gamma_t\ldots$
esponente: q
base : N
normalizzata se $\gamma_0 \neq 0$
I numeri di macchina o floating point sono del tipo:

$$\pm\gamma_0.\gamma_1\gamma_2\ldots\gamma_t \cdot N^q$$

con $\gamma_0 \neq 0$ e $M_1 \leq q \leq M_2$, più lo zero.

Standard IEEE

- costituisce un insieme di regole definito dall'istituto degli ingegneri elettrici e elettronici per la rappresentazione e l'elaborazione dei numeri floating point nei computer;
- specifica esattamente cosa sono i numeri floating point e come sono rappresentati nell'hardware e ha 4 scopi principali;
  - rendere l'aritmetica floating point il più accurata possibile;
  - produrre risultati sensati in situazioni eccezionali;
  - standardizzare le operazioni floating point fra i computer;
  - Dare al programmatore un controllo sulla manipolazione delle eccezioni;

Standard IEEE

- I due tipi di numeri rappresentati sono interi (fixed point) e reali (floating point).
- Un numero reale ha tipo float in c e real in Fortran e Matlab.
- Nella maggior parte dei compilatori c e in Matlab un float ha per default 8 byte invece di 4.
- Il formato IEEE sostituisce la base 10 con la base 2 per rappresentare il numero.

Aritmetica con gli interi

- Un intero viene rappresentato in 4 byte.
- Vi sono quindi $2^{32} \approx 4 \cdot 10^9$ interi a 32 bit che coprono l'intervallo da $-2 \cdot 10^9$ a $2 \cdot 10^9$.
- Addizione, sottrazione e moltiplicazione sono fatte esattamente se la risposta e compresa nell'intervallo.
- La maggior parte dei computer danno risultati imprevedibili se il risultato è fuori dal range (overflow).
- Lo svantaggio dell'aritmetica con gli interi è che non possono essere rappresentate le frazioni e l'intervallo dei numeri è piccolo.

Numeri Reali

Parola a 32 bit interpretata come un numero floating point

- il primo bit è il bit del segno, $s = +$ o $s = -$.
- I successivi 8 bit formano l'esponente.
- I rimanenti 23 bit determinano la mantissa.
- Vi sono 2 possibili segni, 256 esponenti (che variano da 0 a 255) e $2^{23} \approx 8.4$ milioni di possibili mantisse.

Numeri Reali

- Esponenti negativi - convenzione: lo zero è nella posizione 127, dopo ci sono i numeri positivi e prima i numeri negativi.
- In memoria viene rappresentato $q^* = q + 127$.
- Il primo bit della mantissa, che rappresenta γ_0, è sempre uguale a 1, quindi non vi è necessità di memorizzarlo esplicitamente.
- Nei bit assegnati alla mantissa viene memorizzato m^* e $m = 1.m^*$. Un numero floating point positivo ha quindi il valore $x = +(1.m^*)_2 2^{q^*-127}$ e la notazione $(1.m^*)_2$ indica che $1.m^*$ è interpretata in base 2.

Esempio

Il numero $2.752 \cdot 10^3 = 2752$ può essere scritto:

$$\begin{aligned} 2752 &= 2^{11} + 2^9 + 2^7 + 2^6 = \\ &= 2^{11}(1 + 2^{-2} + 2^{-4} + 2^{-5}) = \\ &= 2^{11}(1 + (0.01)_2 + (0.0001)_2 + (0.00001)_2)) = \\ &= 2^{11}(1.01011)_2 \end{aligned}$$

allora la rappresentazione di questo numero avrebbe segno $+$ esponente $q^* = q + 127 = 138 = (10001010)_2$ e $m^* = (010110\ldots0)_2$.

Eccezioni

- Il caso $q^* = 0$ (che corrisponde a 2^{-127}) e il caso $q^* = 255$ (che corrisponde a 2^{128}) hanno una interpretazione differente e complessa che rende la IEEE diversa dagli altri standard.
- Se $q^* = 0$ il valore del numero memorizzato è $x = \pm(0.m^*)_2 2^{-126}$.
- Questo è chiamato underflow graduale (l'underflow è la situazione in cui il risultato di una operazione è diversa da zero ma è più vicina a zero di qualsiasi numero floating point).
- I numeri corrispondenti vengono chiamati denormalizzati.
- L'underflow graduale ha la conseguenza che due numeri floating point sono uguali se e solo se sottraendone uno dall'altro si ha esattamente zero.

Eccezioni

- Se esludiamo i numeri denormali allora il più piccolo numero positivo floating point è $a = realmin = q^{-126}$, il successivo numero positivo b ha $q^* = 1$ e $m^* = 0.0 \cdots 01$, la distanza fra a e b è 2^{23} volte più piccola, della distanza fra a e zero. Senza l'underflow graduale ci sarebbe un gap fra zero ed il numero floating point più vicino.
- L'altro caso è $q^* = 255$ che ha due sottocasi, Inf se $m^* = 0$ e NaN se $m^* \neq 0$.
- Sia il c che il Matlab stampano Inf o NaN quando si stampa una variabile floating point che contiene questo risultato.
- Il computer produce Inf se il risultato di una operazione è più grande del più grande numero rappresentabile.
- Operazioni invalide che producono come risultato NaN sono: Inf/Inf, $0/0$, $Inf - Inf$. Operazioni con Inf hanno il significato usuale: $Inf + finito = Inf$, $Inf/Inf = NaN$, $Finito/Inf = 0$, $Inf - Inf = NaN$.

Accuratezza

- L'accuratezza delle operazioni Floating Point è determinata dalla grandezza degli errori di arrotondamento.
- Questo errore di arrotondamento è determinato dalla distanza di un numero dal numero floating point più vicino.
- Eccetto che per i numeri denormalizzati, i numeri floating point differiscono di un bit, l'ultimo bit di m^*. Cioè $2^{-23} \approx 10^{-7}$ in singola precisione. Questo è l'errore relativo e non l'errore assoluto.

IEEE - singola precisione

Occupa 4 byte = 32 bits

	$\longleftarrow$ 8 $\longrightarrow$	$\longleftarrow$ 23 $\longrightarrow$ —
s	q	m
segno	esponente	mantissa

Esso rappresenta $(-1)^s \cdot 2^{q-127} \cdot (1.m)$ Si noti che $\gamma_0 = 1$ nella mantissa non deve essere esplicitamente memorizzato, quindi $m = \gamma_1\gamma_2 \ldots \gamma_t$.
Range di numeri positivi normalizzati da 2^{-126} a $2^{127} \times 1.11 \cdots 1 \approx 2^{128}$ (da 1.2×10^{-38} a 3.4×10^{38})

IEEE - doppia precisione

Occupa 8 byte = 64 bits

	$\longleftarrow$ 11 $\longrightarrow$	$\longleftarrow$ 52 $\longrightarrow$ —
s	q	m
segno	esponente	mantissa

Esso rappresenta $(-1)^s \cdot 2^{q-1023} \cdot (1.m)$

Range di numeri positivi normalizzati da 2^{-1022} a $2^{1023} \times 1.11 \cdots 1 \approx 2^{1024}$ (da 2.2×10^{-308} a 1.8×10^{308})

IEEE - Eccezioni aritmetiche e risultati di default

tipo di eccezione	esempio	risultato di default
operazione invalide	$0/0 \quad 0 \times \infty$ $\sqrt{-1}$	NaN (Not a number)
Overflow		$\pm\infty$
Divisione per zero	Numero finito non nullo/0	$\pm\infty$
Underflow		numeri denormali

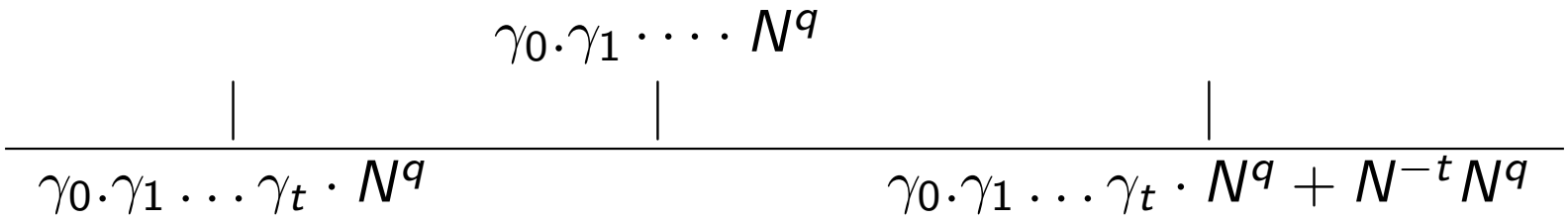
Troncamento e Arrotondamento

Consideriamo il numero reale:

$$x = \pm\gamma_0 . \gamma_1\gamma_2 \ldots \gamma_t \cdot \cdots \cdot N^q$$

compreso fra il massimo e il minimo numero rappresentabile (Se si cerca di rappresentare un numero fuori da questo range si ha il problema dell'*underflow* o dell'*overflow*)
Vi sono due modi diversi per approssimare questo numero mediante un numero floating point:

1) Troncamento:
   si trascurano le cifre successive a γ_t ;
2) Arrotondamento:
   se $\gamma_{t+1} < N/2$ trascurare tutte le cifre dopo γ_t ;
   se $\gamma_{t+1} \geq N/2$, aggiungere 1 a γ_t e trascurare le rimanenti cifre.



Sia $fl(x)$ l'approssimazione di x.

Troncamento:

$$|fl(x) - x| = 0.\gamma_{t+1} \cdots \cdot N^{q-t} \leq N^{q-t}.$$

Arrotondamento

$$|fl(x) - x| \leq \frac{1}{2} N^{q-t}$$

Errore Assoluto

$$\Delta x = |fl(x) - x|$$

Errore relativo

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

Teorema

Se $x \neq 0$ e usiamo t cifre per rappresentare la mantissa, allora:

$$\left| \frac{fl(x) - x}{x} \right| \leq u \tag{1}$$

dove:

$$u = \begin{cases} N^{-t} & \text{troncamento} \\ \frac{1}{2} N^{-t} & \text{arrotondamento} \end{cases}$$

Dimostrazione

Sia $x = \gamma_0.\gamma_1\gamma_2 \ldots \gamma_t \cdots N^q$, allora $|x| \geq N^q$ e, nel caso dell'arrotondamento, si ha:

$$\frac{|fl(x) - x|}{|x|} \leq \frac{\frac{1}{2} N^{q-t}}{N^q} \leq \frac{1}{2} N^{-t},$$

Il numero u è detto unità di arrotondamento o precisione di macchina.

Secondo lo standard IEEE utilizzando la 8 byte per rappresentare i numeri la precisione di macchina risulta essere $u = 2^{-52} \approx 2.22 \cdot 10^{-16}$.
L'errore relativo, piuttosto che l'errore assoluto, è legato alle cifre esatte di un numero.

Poniamo:

$$\epsilon = \frac{fl(x) - x}{x}$$

Sappiamo che $|\epsilon| \leq u$ *e*

$$fl(x) = x(1 + \epsilon),$$

Operazioni con i numeri di macchina

Sia $o \in \{+, -, \times, /\}$ e siano x e y due numeri floating point. È improbabile che l'esatto valore di xoy sia un numero floating point.
Esempio t=3, N=10

$$1.111 \cdot 10^3 \times 1.111 \cdot 10^2 = 1.234321 \cdot 10^3$$

che richiede più di tre cifre decimali.
Il computer dovrebbe eseguire le operazioni aritmetiche di base in modo che il risultato finale sia il risultato esatto arrotondato al più vicino numero floating point. Cioè

Modello delle operazioni aritmetiche:

$$fl(xoy) = (xoy)(1+\epsilon), \qquad |\epsilon| \leq u$$

o

$$fl(xoy) = (xoy)/(1-\epsilon), \qquad |\epsilon| \leq u$$

L'aritmetica floating point IEEE richiede questo.

Esercizio

Consideriamo i seguenti numeri di macchina
$\pm\gamma_0.\gamma_1\gamma_2 10^{\pm e_0 e_1}$:

- qual è il valore di realmin? qual è il valore di realmax?
- se usiamo l'arrotondamento qual è il valore della precisione di macchina?

Utilizzando i numeri di macchina appena definiti calcolare:

$$\frac{(x+2)^2 - 4}{x}$$

per $x = 6.00 \cdot 10^{-3}$ e $x = 2.00 \cdot 10^{-3}$.
Calcolare l'errore relativo in entrambi i casi e spiegare i risultati ottenuti.

Soluzione

Lavorando in base 10 le cifre della mantissa possono assumere i seguenti valori $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$, poichè i numeri di macchina sono rappresentati utilizzando la notazione esponenziale normalizzata $\gamma_0 \neq 0$. L'esponente può assumere i valori $\pm 0, 1, \cdots, 99$.

- Realmin è il più piccolo numero positivo rappresentabile. Realmin $= 1.00 \cdot 10^{-99}$
- Realmax è il più grande numero positivo rappresentabile.Realmax $= 9.99 \cdot 10^{99}$.
- Il valore della precisione di macchina è: $10^{-2}/2$.

I numeri coinvolti sono tutti rappresentabili esattamente nella nostra macchina. Supponiamo che ogni operazione aritmetica dia come risultato il valore esatto arrotondato.

Soluzione

Eseguiamo le operazioni con $x = 6.00 \cdot 10^{-3}$:

- $fl(x + 2) = fl(2.006 \cdot 10^0) = 2.01 \cdot 10^0$
- $fl((2.01 \cdot 10^0)^2 = fl(4.0401 \cdot 10^0) = 4.04 \cdot 10^0$
- $fl(4.04 \cdot 10^0 - 4.00 \cdot 10^0) = fl(0.04 \cdot 10^0) = 4.00 \cdot 10^{-2}$
- $fl(4.00 \cdot 10^{-2}/6.00 \cdot 10^{-3}) = fl(6.6\ldots6 \cdot 10^0) = 6.67 \cdot 10^0$

Il valore esatto è:

$$4.005999\ldots10^0$$

L'errore relativo è:

$$\frac{|6.67 \cdot 10^0 - 4.00599\cdots\cdot 10^0|}{4.00599\cdots\cdot 10^0} \approx 1.47 \cdot 10^{-2}$$

Soluzione

Esaguiamo le operazioni con $x = 2.00 \cdot 10^{-3}$:

- $fl(x + 2) = fl(2.002 \cdot 10^0) = 2.00 \cdot 10^0$
- $fl((2.00 \cdot 10^0)^2 = fl(4.0000 \cdot 10^0) = 4.00 \cdot 10^0$
- $fl(4.00 \cdot 10^0 - 4.00 \cdot 10^0) = fl(0.00 \cdot 10^0) = 0.00$
- $fl(0.00/2.00 \cdot 10^{-3}) = 0.00$

Il valore esatto è:

$$4.001999999999 \cdots \cdot 10^0$$

L'errore relativo è:

$$\frac{|0.00 - 4.001999999999 \cdots \cdot 10^0|}{4.001999999999 \cdots \cdot 10^0} = 1.00 \cdot 10^0$$

Commenti Se $x \approx y$ allora $|x - y|$ ha un errore relativo molto grande che si chiama

errore di cancellazione

Vedremo più avanti una spiegazione di questo errore.
Provare a ripetere l'esercizio usando $x = 6.00 \cdot 10^{-1}$ e $x = 6.00 \cdot 10^{-2}$.

Esempio cancellazione

Eseguiamo $\sqrt{x+1} - \sqrt{x}$, utilizzando il sistema floating point dell'esercizio precedente e $x = 1.00 \cdot 10^3$:
$fl(x+1) = fl(1.00 \cdot 10^3 + 0.001 \cdot 10^3) = fl(1.001 \cdot 10^3) = 1.00 \cdot 10^3$
quindi $\sqrt{1.00^3} - \sqrt{x}$ sarà uguale a zero.
Possiamo utilizzare delle formule equivalenti per calcolare la stessa quantità, il sistema floating point darà risultati diversi.
Esempio:

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

L'errore relativo è $\approx 4.7 \cdot 10^{-4}$.
Provate ad eseguire i calcoli per esercizio.

Le operazioni con i numeri di macchina non godono di tutte le proprietà di cui godono le corrispondenti operazioni con i numeri reali. Per esempio non valgono più la proprietà associativa e quella distributiva.
Esempio $N = 10$ e $t = 2$:

$$5.24 \cdot 10^{-2} + 4.04 \cdot 10^{-2} + 1.21 \cdot 10^{-1}$$

$$5.24 \cdot 10^{-2} + (4.04 \cdot 10^{-2} + 1.21 \cdot 10^{-1})$$

- $= 5.24 \cdot 10^{-2} + (0.404 + 1.21)10^{-1}$
- $= 5.24 \cdot 10^{-2} + 1.61 \cdot 10^{-1}$
- $= (0.524 + 1.61)10^{-1} = 2.13 \cdot 10^{-1}$;

$$(5.24 \cdot 10^{-2} + 4.04 \cdot 10^{-2}) + 1.21 \cdot 10^{-1}$$

- $= 9.28 \cdot 10^{-2} + 1.21 \cdot 10^{-1}$
- $= (0.928 + 1.21)10^{-1}$
- $= 2.14 \cdot 10^{-1}$.

Addizione e Sottrazione

Siano x ed y numeri reali tali che $x + y$ è diverso da zero e calcoliamo la somma $fl(fl(x) + fl(y))$.
$fl(fl(x) + fl(y)) = (x(1 + \epsilon_x) + y(1 + \epsilon_y))(1 + \epsilon)$
con $|\epsilon|, |\epsilon_x|, |\epsilon_y| \leq u$
Calcoliamo l'errore relativo trascurando i termini contenenti $\epsilon\epsilon_x$ ed $\epsilon\epsilon_y$,

$$\frac{|(x + y) - (fl(x) + fl(y))(1 + \epsilon)|}{|x + y|}$$
$$= \frac{|x + y - (x + x\epsilon_x + y + y\epsilon_y)(1 + \epsilon)|}{|x + y|} \approx$$
$$\approx |\epsilon| + |\epsilon_x| \frac{|x|}{|x + y|} + |\epsilon_y| \frac{|y|}{|x + y|}.$$

Se $x + y$ non è piccolo l'errore relativo è dello stesso ordine degli errori relativi $|\epsilon|$, $|\epsilon_x|$ ed $|\epsilon_y|$.

Se $x + y$ è molto piccolo l'errore relativo è grande. Esempio
$x = 0.147554326$, $y = 0.147251742$, $t = 5$ ed $N = 10$.
Usando l'arrotondamento
$fl(x) = 1.47554 \cdot 10^{-1}$
$fl(y) = 1.47252 \cdot 10^{-1}$
$fl(fl(x) - fl(y)) = 3.02000 \cdot 10^{-4}$,
$x - y = 3.02584 \cdot 10^{-4}$.
Le ultime tre cifre della mantissa risultano errate. L'errore relativo risultante è:

$$\frac{3.02584 - 3.02000}{3.02584} \approx 10^{-3}$$

che è piuttosto elevato.

Fenomeno della cancellazione numerica.

Prodotto

Allo stesso modo si può analizzare il prodotto:

$$fl(fl(x) * fl(y)) = fl(x(1+e_x) * y(1+e_y)) = x(1+e_x)y(1+e_y)(1+e_*)$$

l'errore relativo è:

$$|x(1+e_x)y(1+e_y)(1+e_*) - xy|/|xy|$$

cioè

$$|(1+e_x)(1+e_y)(1+e_*) - 1|$$

semplificando ed eliminando i termini che contengono i prodotti di pi'**u**errori si ottiene:

$$\text{errore relativo nel prodotto } \approx |e_x| + |e_y| + |e_*|$$

Calcolo del valore di un polinomio: algoritmo 1

$$p(x) = a_0 x^N + a_1 x^{N-1} + \cdots + a_N$$

Come possiamo valutarlo al variare di x?
Un algoritmo standard è:

```
px = a(N)
for j=N-1:-1:0
   px = px + a(j) * x^(N-j)
end
```

Contiamo le operazioni aritmetiche:
addizioni : N
moltiplicazioni : $1 + 2 + 3 + \cdots N = \frac{N(N+1)}{2}$
Ogni termine $a_j x^{N-j}$ è stato calcolato indipendentemente dagli altri termini.

Calcolo del valore di un polinomio: algoritmo 2

Possiamo modificare l'algoritmo calcolando ricorsivamente $x_j = x * x^{j-1}$.
L'algoritmo diventa:

```
px = a(N) + a(N-1)*x
xp =  x
for j=N-2:-1:0
   xp =  x * xp
   px = px + a(j) * xp
end
```

Le operazioni aritmetiche sono:
addizioni: N
moltiplicazioni : $N + N - 1 = 2N - 1$
Il costo è molto inferiore rispetto al primo algoritmo. Esempio: N=20
primo algoritmo: 210 moltiplicazioni
secondo algoritmo: 39 moltiplicazioni

Calcolo del valore di un polinomio: algoritmo 3

Un algoritmo ancora più efficiente è la regola di Ruffini-Horner, che esegue le moltiplicazioni in modo innestato

ESEMPI:
$N = 2 : p(x) = a_2 + x(a_1 + a_0 x)$
$N = 3 : p(x) = a_3 + x(a_2 + x(a_1 + x a_0))$
$N = 4 : p(x) = a_4 + x(a_3 + x(a_2 + x(a_1 + x a_0)))$
Il numero di operazioni è, rispettivamente, 2, 3 e 4 moltiplicazioni. Il secondo algoritmo ne richiedeva 3,5 e 7.
In generale:
$p(x) = a_N + x(a_{N-1} + \cdots + x(a_1 + a_0 x)) \cdots)$

Calcolo del valore di un polinomio: algoritmo 3

L'algoritmo è:

```
px = a(0)
for j=1:N
  px = a(j) + px*x
end
```

Le operazioni aritmetiche sono:
addizioni: N
moltiplicazioni : N

Approssimazione della derivata prima.

Sappiamo che per definizione la derivata prima di una funzione $f(x)$ è data da

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

e utilizzando il polinomio di Taylor possiamo dire che:

$$\frac{f(x+h) - f(x)}{h} = \frac{f(x) + hf'(x) + \frac{h^2}{2} f''(\xi) - f(x)}{h}$$

con $x \leq \xi \leq x + h$.
Da qui deriva che

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{h}{2} f''(\xi)$$

La quantità $\tau(h) = \frac{h}{2} f''(\xi)$ si chiama errore di troncamento o errore di discretizzazione e dipende da h. Possiamo dire che l'errore va a zero come $O(h)$

Approssimazione della derivata prima.

Consideriamo il rapporto

$$\frac{f(x+h)-f(x-h)}{2h}$$

Utilizzando lo sviluppo in serie di Taylor abbiamo:

$$f(x+h)-f(x-h)=$$

$$=f(x)+hf'(x)+\frac{h^2}{2}f''(x)+\frac{h^3}{6}f'''(x)+\frac{h^4}{24}f^{(4)}(\xi_1)$$

$$-f(x)+hf'(x)-\frac{h^2}{2}f''(x)+\frac{h^3}{6}f'''(x)-\frac{h^4}{24}f^{(4)}(\xi_2)=$$

$$=2hf'(x)+\frac{h^3}{3}f'''(x)+\frac{h^4}{24}(f^{(4)}(\xi_1)-f^{(4)}(\xi_2))$$

Approssimazione della derivata prima.

In definitiva abbiamo

$$\frac{f(x+h)-f(x-h)}{2h} =$$

$$= f'(x) + \frac{h^2}{6} f'''(x) + \frac{h^3}{24}(f^{(4)}(\xi_1) - f^{(4)}(\xi_2)) =$$

$$= f'(x) + O(h^2)$$

CONDIZIONAMENTO DI UN PROBLEMA

L' algoritmo è la sequenza di istruzioni per risolvere un problema

Problema e Algoritmo

$$x \xrightarrow{P} y \qquad x \xrightarrow{A} y$$

In realtà poiché l'algoritmo è eseguito al calcolatore risulta

$$x \xrightarrow{\overline{A}} y + \delta y$$

Esistono vari algoritmi per poter risolvere un particolare problema.
Esistono problemi tali che, qualunque algoritmo venga utilizzato per risolverli, se eseguito in aritmetica di macchina, genera nel risultato un errore molto elevato.
Questo fenomeno è una particolarità intrinseca del problema e non dipende dagli algoritmi utilizzati.

PROBLEMI BEN POSTI

$$x \xrightarrow{P} y$$

Il problema P si dice ben posto se

- $\forall x \qquad \exists | y$
- i risultati variano con continuità rispetto ai dati. Cioè $\exists K > 0$ tale che

$$x_1 \xrightarrow{P} y_1 \qquad\qquad x_2 \xrightarrow{P} y_2$$

allora

$$\frac{\|y_1 - y_2\|}{\|y_1\|} \leq K \frac{\|x_1 - x_2\|}{\|x_1\|}$$

Se K è grande un piccolo errore nei dati comporta un grande errore nei risultati e il problema si dice

MAL CONDIZIONATO

Per i problemi MAL CONDIZIONATI la perturbazione nei dati dovuta alla rappresentazione finita genera errori elevati sul risultato, qualunque sia l'algoritmo utilizzato.
Se K è piccolo il problema si dice

BEN CONDIZIONATO

K si dice INDICE DI CONDIZIONAMENTO DEL PROBLEMA.

ALGORITMI STABILI

Per risolvere un problema posso formulare un algoritmo che eseguito sui dati di input x dà il risultato y

$$x \xrightarrow{A} y$$

In realtà poiché l'algoritmo è eseguito al calcolatore risulta

$$x \xrightarrow{\overline{A}} y + \delta y$$

Un algoritmo si dice stabile se, applicato a un problema ben condizionato, l'errore relativo fra i valori ottenuti utilizzando l'aritmetica di macchina e quelli ottenuti con aritmetica esatta,

$$\frac{|\delta y|}{|y|},$$

è piccolo, cioè l'effetto degli errori di arrotondamento è trascurabile.

RADICI DI POLINOMI

$$(x-1)^4 = x^4 - 4x^3 + 6x^2 - 4x + 1$$

radice 1, molteplicità 4.
Perturbiamo il termine noto

$$(x-1)^4 - 1e-8 = x^4 - 4x^3 + 6x^2 - 4x + 1 - 1e-8$$

radici:

$$\begin{aligned} x_1 &= 1.01 \\ x_2 &= 0.99 \\ x_3 &= 1 + i0.01 \\ x_4 &= 1 - i0.01 \end{aligned}$$

quindi una pertubazione relativa di 1e-8 sul termine noto produce una perturbazione relativa di 1e-2 sul risultato.

Il risultato di una operazione di macchina può essere visto come l'operazione esatta su dati perturbati
Addizione:

$$fl(x+y) = (x+y)(1+\epsilon) = (x+\epsilon x) + (y+\epsilon y),$$

dati perturbati $(x+\delta x)$ e $(y+\delta y)$, $\delta x = \epsilon x$ e $\delta y = \epsilon y$.
Moltiplicazione:

$$fl(xy) = xy(1+\epsilon) = (x+\delta x)(y+\delta y)$$

con $\delta x = 0, \delta y = \epsilon y$.
La tecnica di considerare la perturbazione nei dati che porterebbe allo stesso risultato finale se le operazioni fossero eseguite con precisione infinita viene chiamata

ANALISI DEGLI ERRORI ALL'INDIETRO (BACKWARD)

.

ANALISI DEGLI ERRORI ALL'INDIETRO

Se la soluzione calcolata dall'algoritmo $y + \delta y$ è la soluzione esatta del problem P calcolato su dati perturbati $\mathbf{x} + \delta x$ cioè:

$$\begin{array}{ccc} & P & \\ x + \delta x & \longrightarrow & y + \delta y \end{array}$$

allora l'algoritmo si dice STABILE se l'errore relativo:

$$\frac{|\delta x|}{|x|}$$

dipende linearmente dal numero di operazioni. È instabile se dipende esponenzialmente dal numero di operazioni. Cioè se E_n è l'errore alla n-sima operazione dell'algoritmo si ha:
$E_n \approx c_0 n E_0$ crescita lineare, $c_0 > 0$
$E_n \approx c_1^n E_0$ crescita esponenziale, $c_1 > 0$

Propagazione degli errori

$y = f(x)$, $y \neq 0$
perturbiamo x con un errore Δx.
il risultato y avrà un errore Δy.
Sia f derivabile in un intorno di x i ha:
$y + \Delta y = f(x + \Delta x) \simeq f(x) + f'(x)\Delta x$

$$\frac{\Delta y}{y} \simeq x\frac{f'(x)}{f(x)}\frac{\Delta x}{x}.$$

L' errore relativo su x produce un errore relativo su y che dipende da:

$$K(x, f) = \left| x\frac{f'(x)}{f(x)} \right|$$

Questa quantità è l'indice di condizionamento del problema.

Esempio

Sia $f(x) = log(x)$.
Calcoliamo

$$K(x, f) = \left| x \frac{f'(x)}{f(x)} \right| = \left| x \frac{1/x}{log(x)} \right|$$

il calcolo del logaritmo è un problema mal condizionato se x è vicino a 1.
Provare a calcolare $K(x, f)$ per $cos(x), sin(x), e^x$.