ERRORI DI CALCOLO (ovvero **Nessuno è Perfetto)**

E' possibile che i computer commettano errori? Proviamo ad indagare il loro funzionamento da un punto di vista matematico con qualche semplice esempio.

La derivazione numerica

Consideriamo il caso interessante della derivazione numerica dal punto di vista degli errori. La derivata di una funzione è un'operazione semplice da eseguire e da apprendere, ma assolutamente non banale. Il caso numerico è ancora più complesso, ma anche estremamente interessante ed adatto ad una esposizione didattica approfondita.

Come sappiamo bene, la derivata di una funzione $f(x)$ nel punto x_0 è definita come il limite per $h \to 0$ dell'espressione

$$(f(x_0+h)-f(x_0))/h$$

che è conosciuta come **rapporto incrementale** e fornisce un'approssimazione della derivata tanto più precisa quanto più è piccolo h.

L'errore che si commette con questa approssimazione può essere stimato sviluppando la differenza tra $f'(x_0)$ e il rapporto incrementale $(f(x_0+h)-f(x_0))/h$ in serie di potenze di h.

Per fare un esempio concreto, consideriamo la funzione $f(x)=cos(x)$ e la sua derivata $f'(x)=-sin(x)$ nel punto $x_0=1.1$. Se indichiamo con $fd1(x_0,h)$ l'approssimazione di $f'(x_0)=-sin(x_0)$ data dal rapporto incrementale, si ha $fd1(x_0,h)=(cos(x_0+h)-cos(x_0))/h$ e l'errore $Err1(h)$ che si commette è dato dallo sviluppo in serie di Taylor di $f'(x_0)-fd1(x_0,h)$, cioè dall'espressione

$$Err1(h)=-(1/2)\ cos(x_0)\ h + (1/6)\ sin(x_0)\ h^2 + (1/24)\ cos(x_0)\ h^3 + o(h^4).$$

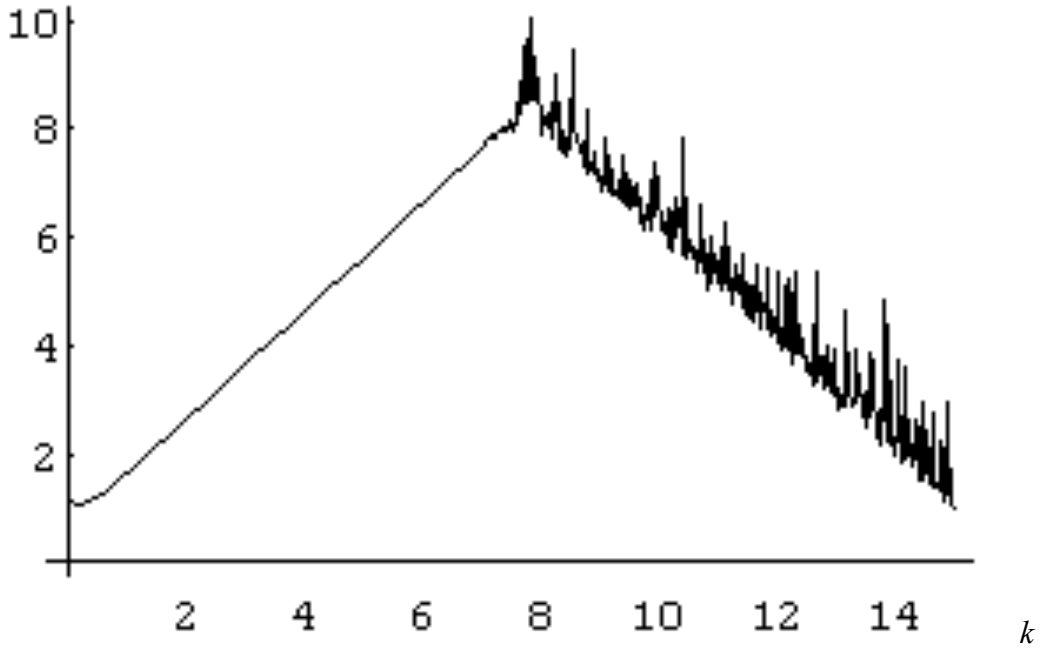
Per valori piccoli di h, il primo termine dell'errore, che dipende da h, è molto più grande degli altri, che dipendono da h^2, h^3, ... e così si usa dire che l'errore è proporzionale ad h a meno di termini di ordine superiore. Sembra quindi vero che sia sufficiente ridurre h sempre di più per avere una precisione sempre più grande del valore approssimato della derivata.

Per avere un'idea più precisa di questa approssimazione possiamo definire una funzione $precisione(t)=-Log_{10}(|t|)$, che permette di stimare il numero di cifre significative in un risultato approssimato. Qui t rappresenta la differenza tra il valore vero ed l'approssimazione della grandezza considerata. Per esempio, se il valore corretto è 1.239999998 ed il valore approssimato è 1.239999997, la differenza vale $0.000000001=10^{-9}$ e $precisione(10^{-9})=9$ sta a significare che l'errore cade sulla nona cifra decimale e l'approssimazione possiede quindi 8 cifre decimali esatte.

Se applichiamo la funzione *precisione* alla differenza $f'(x_0)-fd1(x_0,h)$, cioè alla differenza tra la derivata e la sua approssimazione data dal rapporto incrementale, e poniamo $h=10^{-k}$, dove k è un numero compreso, per esempio, tra 0 e 15, possiamo costruire il grafico di *precisione* in funzione di

k, che ci dice come varia la precisione di $fd1(x_0,h)$ man mano che *h* diviene sempre più piccolo. Il risultato sorprendente è il seguente :

precisione(f'-fd1)



Si nota come il numero di cifre significative dell'approssimazione data dal rapporto incrementale dapprima salga fino a raggiungere un massimo verso $k \approx 8$, cioè $h \approx 10^{-8}$, per poi diminuire di nuovo. Qual' è la spiegazione di questo strano comportamento ? Il fatto è che la nostra approssimazione non è affetta soltanto dall'errore, proporzionale ad *h*, che abbiamo analizzato fino ad ora e che chiameremo **errore analitico**, ma anche da un **errore di arrotondamento** che si verifica nel calcolo di $(f(x_0+h)-f(x_0))$ e che, contrariamente al primo, tende a crescere al diminuire di *h*.

Nel grafico si vede come i due errori contrapposti (analitico e di arrotondamento) si bilancino fornendo il migliore risultato per $h \approx 10^{-8}$. Si nota pure che, mentre l'errore analitico ha un andamento regolare, l'errore di arrotondamento è frastagliato, essendo presente una componente casuale.

Come possiamo migliorare la nostra approssimazione ? Dallo sviluppo in serie, calcolato precedentemente, possiamo notare che il termine più importante dell'errore analitico (quello lineare) è una funzione dispari di *h*. Perciò se riscriviamo la stessa approssimazione sostituendo -*h* al posto di *h*, l'errore avrà valore uguale e segno opposto. Possiamo allora combinare le due approssimazioni (quella ottenuta con *h* e quella ottenuta con -*h*) in modo da eliminare il termine

lineare dell'errore analitico. Questo risultato si ottiene con la semplice combinazione *(fd1(h)+fd1(-h))*/2.

Chiamiamo *fd2* questa nuova approssimazione della derivata di una funzione in un punto. Nel caso che stiamo considerando della funzione $f(x)=cos(x)$, si ha :

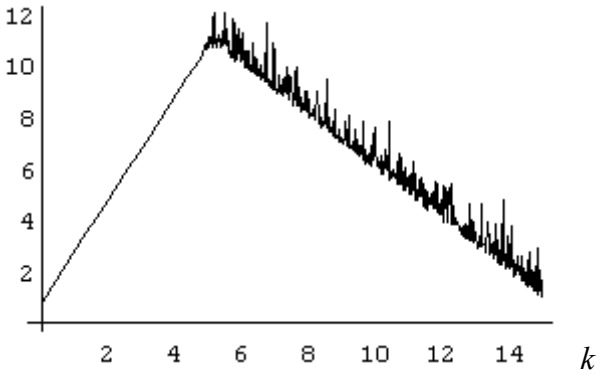
$$fd2(h)=(cos(x_0+h)-cos(x_0-h))/2h.$$

L'errore analitico, che in questo caso indichiamo con *Err2(h)*, ora vale :

$$Err2(h)= (1/6)\ sin(x_0)\ h^2 + o(h^4).$$

Se applichiamo ora la funzione *precisione* alla differenza $f'(x_0)-fd2(x_0,h)$, cioè alla differenza tra la derivata e la sua nuova approssimazione, possiamo costruire un grafico simile al precedente :

precisione(f'-fd2)



La nuova approssimazione presenta un errore di arrotondamento che è dello stesso tipo del precedente, ma, siccome la pendenza dell'errore analitico è doppia (infatti $Log(h^2)=2\ Log(h)$) l'incontro tra i due grafici avviene per valori di h più grandi ($h\approx10^{-5}$) e il numero di cifre significative sale quasi a 12.

Si può ancora migliorare la nostra approssimazione ? Si, con un piccolo sforzo. Possiamo valutare *fd2* con un passo ridotto a $h/2$ e considerare una combinazione lineare di *fd2(h)* e *fd2(h/2)*, cioè *fd3(h)* = a *fd2(h)* + b *fd2(h/2)*. L'errore analitico *Err3(h)*, dato dallo sviluppo in serie di Taylor di *f'-fd3*, in questo caso vale :

$$Err3(h) = (sin(x_0)\text{-a } sin(x_0)\text{-b } sin(x_0)) + (\text{a } sin(x_0)/6 + \text{b } sin(x_0)/24)\ h^2 -$$
$$(\text{a } sin(x_0)/120 + \text{b } sin(x_0)/1920)\ h^4 + o(h^6).$$

Per eliminare i primi due termini di *Err3*, occorre soddisfare le condizioni 1-a+b=0 e a/6+b/24=0. Le soluzioni di questo sistema sono a=-1/3 e b=4/3, per cui

$$fd3(h) = (8\ cos(h/2 + x_0) - cos(h + x_0) - 8\ cos(h/2 - x_0) + cos(h - x_0))\ /\ 6h.$$

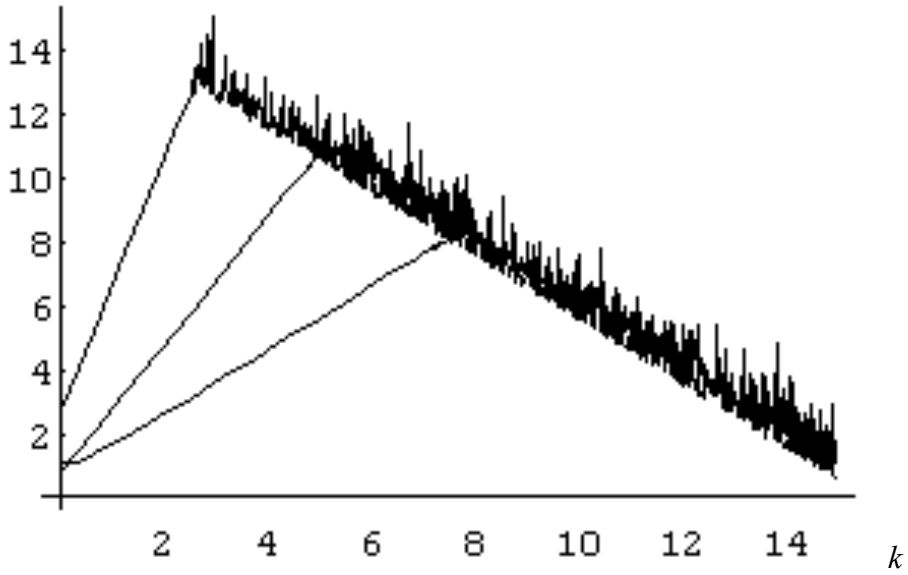
L'approssimazione risultante ha ora errore di ordine h^4. Si ha infatti

$$Err3(h)=(sin(x_0)\ h^4)/480 + o(h^6).$$

Il grafico seguente mostra l'andamento della funzione *precisione* nei tre casi considerati corrispondenti agli errori analitici *Err1*, *Err2*, *Err3*. L'ultima approssimazione presenta una pendenza dell'errore analitico che è quadrupla della prima : il caso ottimale si ha per ($h \approx 10^{-3}$) e il numero di cifre significative è arrivato quasi a 14.

Come appare evidente, il caso in apparenza non difficoltoso dell'approssimazione numerica della derivata è in realtà irto di insidie, con risultati che non sono quelli che l'intuito, a prima vista, ci avrebbe suggerito.

precisione



Quanto abbiamo visto sul calcolo approssimato della derivata è sostanzialmente dovuto al fatto che, data una funzione reale di variabile reale *f(x)*, il suo calcolo nella aritmetica approssimata di un computer non è possibile tranne che in alcuni casi particolari. Quella che in generale viene calcolata è una approssimazione affetta da vari tipi di errori.

L'errore analitico (o di troncamento)

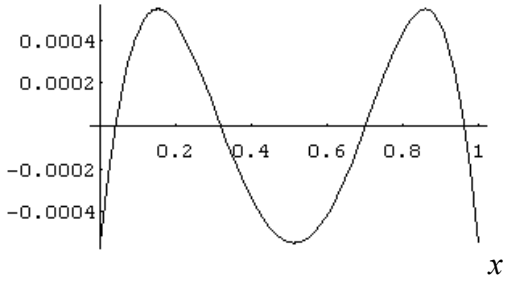
Se *f(x)* non è una funzione razionale, ovvero non è calcolabile come combinazione delle quattro operazioni aritmetiche, quello che viene calcolato non è *f(x)* ma una sua approssimazione razionale *g(x)*.

Per esempio, per approssimare e^x nell'intervallo (0,1) si può usare il polinomio di terzo grado

$$p(x) = 0.2799765\, x^3 + 0.421703\, x^2 + 1.016602\, x + 0.9994552$$

commettendo un errore minore in modulo a $0.5 \cdot 10^{-3}$, come mostrato nel seguente grafico :

$p(x)-e^x$



L'errore di arrotondamento (o algoritmico)

Una funzione razionale viene calcolata con una opportuna sequenza delle quattro operazioni aritmetiche ; ogni operazione, effettuata in aritmetica finita, è affetta da errore. La combinazione di tutti gli errori da origine al cosiddetto errore di arrotondamento.

Un esempio di errore di arrotondamento molto vistoso è il fenomeno della cancellazione, ovvero la perdita di cifre significative dovute alla differenza di numeri molto vicini. Calcoliamo, per esempio, la differenza tra $diff=cos(1.1)-cos(1.1+10^{-10})$ prima con 19 e poi con 10 cifre di precisione ; i risultati sono rispettivamente :

0.4535961214255773887 -
0.4535961213364566516 =
$diff_{19}$ = 0.0000000000891207371

e

0.4535961214 -
0.4535961213 =
$diff_{10}$ = 0.0000000001

Pur operando con 19 cifre, $diff_{19}$ ha solo 9 cifre significative, $diff_{10}$ addirittura ha solo una cifra significativa.

L'errore di arrotondamento è detto anche **errore algoritmico** perché può dipendere dal particolare algoritmo scelto per il calcolo. Per esempio, consideriamo la somma delle potenze settime dei numeri da 1 a 1000 che è il seguente numero a 24 cifre : 125500583333041666750000.

Operando con 19 cifre di precisione e sommando gli addendi in ordine crescente si ottiene un errore di *SommaCrescente-SommaEsatta*=8192. Sommando invece in ordine decrescente si ottiene un errore 4 volte maggiore : *SommaDecresente-SommaEsatta*=-32768. Il motivo di questa differenza risiede nel fatto che nell'eseguire la somma, gli ultimi addendi, che sono numeri di poche cifre, sono completamente "cancellati" dal fatto di aggiungerli ad un numero che possiede più cifre di quelle rappresentabili nel computer. Nella parte finale dell'algoritmo, gli addendi della somma decrescente sono più piccoli che nel caso della somma crescente e ciò produce un errore maggiore.

L'errore inerente (o di rappresentazione interna)

Ritornando al problema della rappresentazione di una funzione reale, va ancora notato che, se il valore di x non è rappresentabile nella aritmetica del computer o comunque è affetto da un errore Δx, quello che viene calcolato non è $f(x)$, ma l'approssimazione $f(x+\Delta x)=f(x)+f'(x)\cdot\Delta x$ a meno di termini di ordine superiore in Δx.

Questo caso si verifica molto più comunemente di quanto si creda. Consideriamo per esempio il seguente programma BASIC :

```
10 A1=0:A2=0
20 B1=0.1:B2=1
30 FOR I=1 TO 10
40 A1=A1+B1:A2=A2+B2
50 NEXT I
60 PRINT A1,A1-1
70 PRINT A2,A2-10
99 END
```

Questo programma somma dieci volte il valore 0.1 e lo pone nella variabile A1, somma dieci volte il valore 1 e lo pone nella variabile A2, poi stampa A1, A1-1, A2, A2-10. Se eseguito su un usuale PC con il Microsoft BASIC, i risultati in singola precisione sono :

```
A1=1       A1-1=1.192093E-7
A2=10      A2-10=0.
```

Si vede come il calcolo con i numeri interi sia stato eseguito esattamente, mentre il valore contenuto nella variabile A1, che dovrebbe essere 1, in realtà è differente, con un errore sulla settima cifra significativa.

Riproviamo con un altro esempio per convincerci che non si è trattato di un caso :

```
80 A1=2
90 B1=0.1
92 FOR I=1 TO 10
94 A1=A1-B1
96 NEXT I
98 PRINT A1
99 END
```

In questo caso al valore di partenza 2 viene sottratto per dieci volte il valore 0.1. Il risultato dovrebbe essere 1, in realtà il computer fornisce 0.9999998.

Cosa sta succedendo ? Questi numeri non sono molto grandi e sono tutti vicini tra loro, qual'è allora la sorgente dell'errore ? La risposta è semplice, ma sottile : sebbene i numeri utilizzati sembrino "innocui", alcuni tra loro non sono rappresentabili con un numero di cifre finito nel sistema di numerazione binario (che è quello usato dal computer). La quantità decimale 0.1 è infatti

periodica e nel sistema binario si scrive 0.00011001100110011....., ciò introduce ad ogni passo dei calcoli degli algoritmi precedenti un piccolo errore che si ripercuote sul risultato finale.

Nota : come si converte in binario un numero decimale reale

Per convertire un numero decimale reale in binario, si può convertire separatamente la sua parte intera e quella frazionaria.

La parte intera viene divisa per 2 tante volte quanto più è possibile, prendendo nota dei resti, che in ordine inverso a come sono stati ottenuti nel calcolo, rappresentano la parte intera binaria.

La parte frazionaria viene moltiplicata per due, prendendo nota della parte intera del prodotto.

Consideriamo per esempio il numero 109.78125, la sua pare intera diviene :

Divisioni	**Quozienti**	**Resto**
109/2	54	1
54/2	27	0
27/2	13	1
13/2	6	1
6/2	3	0
3/2	1	1
1/2	0	1

Il quoziente zero indica la fine del calcolo. Il resto può ovviamente essere solo 0 o 1, dato che la divisione è per 2. La sequenza dei resti letta dal basso verso l'alto fornisce l'equivalente binario richiesto : 109 decimale = 1101101 binario.

La parte decimale diviene :

Moltiplicazioni	**Parti intere**
$0.78125 \cdot 2 = 1.56250$	1
$0.56250 \cdot 2 = 1.1250$	1
$0.125 \cdot 2 = 0.250$	0
$0.25 \cdot 2 = 0.50$	0
$0.5 \cdot 2 = 1.00$	1

La parte frazionaria uguale a zero indica la fine del calcolo. Si noti che la parte intera di ogni prodotto può essere solo 0 o 1 in quanto raddoppiamo sempre numeri inferiori ad 1. La sequenza delle cifre parte-intera letta dall'alto verso il basso fornisce l'equivalente binario cercato. 0.78125 decimale = 0.11001 binario. Dunque 109.78125 decimale = 1101101.11001 binario.

Riflessioni

E' ovvio che questa trattazione tocca solo la superficie di queste problematiche, lo scopo è che serva da stimolo per qualche riflessione basata su basi scientifiche su oggetti come i computer a cui ci affidiamo a volte con troppa fiducia.

Bibliografia

1) Matematica di base per il calcolatore, Seymour Lipschutz, Collana Schaum, Etas Libri.
2) Mcmicrocomputer n.131, luglio/agosto 1993, pag. 241 (articolo di Francesco Romani).
3) Introduzione alla Matematica Computazionale, R. Bevilacqua, D. Bini, M. Capovani, O. Menchi, Zanichelli (1992).
4) Elaborazione Statistica dei Dati Sperimentali, Hugh D. Young, Veschi Editore, Roma.
5) Programma "derivata_approssimazioni.nb" per Mathematica 5.2