Analisi degli errori

Corso di Calcolo Numerico, a.a. 2008/2009

Francesca Mazzia

Dipartimento di Matematica
Università di Bari

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

Errore

Se A è una quantità che vogliamo calcolare e A_h è un'approssimazione di A, allora l'errore commesso è la differenza fra i due valori:

errore

$$\text{errore} = A - A_h;$$

Errore Assoluto e Relativo

L'errore assoluto è il valore assoluto dell'errore:

errore assoluto

$$\text{errore assoluto} = |A - A_h|;$$

e l'errore relativo si ottiene normalizzando l'errore assoluto con il valore esatto, se $A \neq 0$:

errore relativo

$$\text{errore relativo} = \frac{|A - A_h|}{|A|}.$$

L'errore relativo è più significativo dell'errore assoluto. È ragionevole chiedere che l'errore relativo sia minore di un valore prefissato.

Se conosciamo una maggiorazione dell'errore assoluto, cioè:

$$|A - A_h| < tol.$$

possiamo fare una stima del valore esatto:

$$A_h - tol \leq A \leq A_h + tol.$$

Se conosciamo una maggiorazione dell'errore relativo, cioè:

$$\frac{|A - A_h|}{|A|} < tol.$$

possiamo fare una stima del valore esatto:

$$\frac{A_h}{1 + tol} \leq A \leq \frac{A_h}{1 - tol}.$$

Esempio

u	$\tilde{u}$	Errore Assoluto	Errore Relativo
1	0.99	0.01	0.01
1	1.01	0.01	0.01
-1.5	-1.2	0.3	0.2
100	99.99	0.01	0.0001
100	99	1	0.01

Se riteniamo accettabili approssimazioni in cui

$$\frac{|A - A_h|}{|A|} < 0.001,$$

allora siano $A = 123457$ e $A_h = 123500$, calcoliamo l'errore relativo:

$$\frac{43}{123457} = 0.00034,$$

e poichè l'errore è minore di 0.001, l'approssimazione è accettabile.

Siano invece $A = 341.5$ e $A_h = 300$, l'errore relativo è:

$$\frac{41.5}{341.5} = 0.121,$$

e quindi l'approssimazione non è accettabile.

Notazione: uguaglianza approssimata

Se due quantità sono approssimativamente uguali, useremo la notazione $\approx$ per indicare questa relazione.
Questa è una notazione ambigua. È vero che $0.99 \approx 1$? Forse si. È vero che $0.8 \approx 1$? Forse no. Sia h un parametro reale che tende a zero tale che $\lim_{h \to 0} A_h = A$ allora,

$$A_h \approx A$$

per ogni h "sufficientemente piccolo".
Sia n un parametro intero che tende all'infinito tale che $\lim_{n \to \infty} A_n = A$ allora,

$$A_n \approx A$$

per ogni n "sufficientemente grande".

esempio

Un modo per scrivere la derivata prima di una funzione è

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

Possiamo quindi concludere che per h sufficientemente piccolo

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

L'uguaglianza approssimata verifica le proprietà transitiva, simmetrica e riflessiva:

$$\begin{aligned} A \approx B, B \approx C &\rightarrow A \approx C \\ A \approx B &\rightarrow B \approx A \\ & A \approx A \end{aligned}$$

Notazione: ordine asintotico

Un'altra notazione è la notazione dell"'O grande", conosciuta come **ordine asintotico**. Supponiamo di avere un valore y e una famiglia di valori che lo approssimano y_h, Se esiste una costante $C > 0$, indipendente da h, tale che:

$$|y - y_h| \leq C|\beta(h)|,$$

per h sufficientemente piccolo, allora diciamo che:

$$y = y_h + O(\beta(h)) \qquad \text{per } h \to 0,$$

cioè $y - y_h$ è dell'ordine di $\beta(h)$, $\beta(h)$ è una funzione del parametro h tale che $\lim_{h\to 0} \beta_h = 0$. Ci concentriamo sul modo in cui l'errore dipende dal parametro h e ignoriamo dettagli meno importanti come il valore di C. L'utilizzo è analogo se abbiamo una successione x_n che approssima x per valori di n grandi:

$$|x - x_n| \leq C|\beta(n)|, \qquad x = x_n + O(\beta(n))$$

Teorema di Taylor

Teorema

Sia $f(x)$ una funzione avente $n+1$ derivate continue su $[a,b]$ per qualche $n \geq 0$, e siano x,$x_0 \in [a,b]$. Allora

$$f(x) = p_n(x) + R_n(x)$$

con

$$p_n(x) = \sum_{k=0}^{n} \frac{(x-x_0)^k}{k!} f^{(k)}(x_0)$$

e

$$R_n(x) = \frac{1}{n!} \int_{x_0}^{x} (x-t)^n f^{(n+1)}(t)dt.$$

Inoltre esiste un punto ξ_x tra x e x_0 tale che:

$$R_n(x) = \frac{(x-x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi_x)$$

esempio

Supponiamo di volere approssimare la derivata prima di una funzione. Sappiamo che:

$$f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h} = f'_h(x_0)$$

per h sufficiantemente piccolo. Vogliamo calcolare come $f'_h(x_0)$ si avvicina a $f'(x_0)$. Usiamo il Teorema di Taylor, con $n=2$, $x = x_0 + h$ per esprimere $f(x_0+h)$:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(\xi_h)$$

quindi

$$\begin{aligned} f'_h(x_0) = & \quad \frac{f(x_0+h)-f(x_0)}{h} \\ = & \quad \frac{f(x_0)+hf'(x_0)+\frac{h^2}{2}f''(\xi_h)-f(x_0)}{h} \\ = & \quad f'(x_0)+\tfrac{h}{2}f''(\xi_h) = f'(x_0)+O(h). \end{aligned}$$

Approssimazione della derivata prima.

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 $O(h)$

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h} = f'_h(x_0)$$

Questa formula per approssimare la derivata si chima metodo alle differenze in avanti.
Per vedere numericamente come si comporta eseguiamo in R il file script derivata _ dem.R

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

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

$$= f'(x) + \frac{h^2}{6} f'''(x) + \frac{h^3}{48}(f^{(4)}(\xi_1) - f^{(4)}(\xi_2)) =$$

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

Questa tecnica per approssimare la derivata prima si chiama metodo delle differenze centrali.
Per confrontarla con il metodo delle differenze in avanti eseguiamo in R la funzione derivata.R.

Il seguente esempio fa vedere che eseguiendo istruzione al calcolatore non sempre otteniamo il risultato che ci aspettiamo.
Il seguente file di tipo script dal nome `radice.R` . I commenti nel file vengono preceduti dal

```
 #

radice = function(xt,n){
# function radice
# esempio errori di arrotodamento
# eseguiamo per n volte la radice quadrata di x e
# per n volte il quadrato del risultato.
x=xt
for (i in 1:n)
  x = sqrt(x)
for (i in 1:n)
  x = x^2
return(x)
}
```

Adesso eseguiamo il file con diversi valori di n con la seguente istruzione ponendo `xt = 100`.

```
>source("radice.R")
> x=radice(100,10)
> x
[1] 100
> err <- abs(x-100)/100
> err
[1] 6.366463e-14

> x=radice(100,20)
> x
[1] 100

> err <- abs(x-100)/100
> err
[1] 1.155499e-10
```

```
> x=radice(100,30)
> err <- abs(x-100)/100
> err
[1] 2.299032e-07
> x=radice(100,30)
> x
[1] 99.99998
> err <- abs(x-100)/100
> err
[1] 2.299032e-07
> x=radice(100,40)
> x
[1] 100.0054
> err <- abs(x-100)/100
> err
[1] 5.436917e-05
> x=radice(100,60)
> x
[1] 1
```

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, implementato in R, è:

```
pol1 = function(x,a){
N=length(a)
px <- a[N]
for (j in seq(N-1,1,-1)){
   px <- px + a[j] * x^(N-j)
}
return(px)
}
```

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, implementato in R, diventa:

```
pol2 = function(x,a){
N=length(a)
px = a[N] + a[N-1]*x
xp =  x
for (j in seq(N-2,1,-1)){
   xp=x*xp
   px = px + a[j] * xp
}
return(px)
}
```

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

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, implementato in R, è:

```
horn = function(x,a){
N=length(a)
px = a[1]
for (j in seq(2,N)){
   px = a[j] + px * x
}
```

Le operazioni aritmetiche sono:
addizioni: N
moltiplicazioni : N

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

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

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 stabile dovrebbe generare una soluzione numerica che puo' essere considerata la soluzione esatta di un problema vicino a quello di partenza.

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

ALGORITMI STABILI

Se un algoritmo stabile è applicato ad un problema ben condizionato, allora la soluzione $y + \delta y$ è vicina alla soluzione esatta y, cioè l'errore relativo:

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

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

$$(x-1)^4 - 10^{-8} = x^4 - 4x^3 + 6x^2 - 4x + 1 - 10^{-8}$$

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 10^{-8} sul termine noto produce una perturbazione relativa di 10^{-2} sul risultato.

Errore in avanti

L'errore in avanti (forward) è l'errore fra la soluzione esatta di un problema e la soluzione calcolata, cioè:

errore assoluto in avanti

$$|\delta y|$$

errore relativo in avanti

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

Errore all'indietro

L'errore all'indietro (backward) è l'errore fra i dati di input del problema e quelli che generano la soluzione perturbata in aritmetica esatta, cioè:

errore assoluto all'indietro

$$|\delta x|$$

errore relativo all'indietro

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

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, $c_1 > 0$

Numero di condizione di una funzione

$y = f(x)$, $y \neq 0$
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 $cos(x), sin(x), e^x$.

Esercizio

Data la funzione $f(x) = x^2 - 6x + 9$ fare il grafico del numero di condizione, $K(x, f)$, utilizzando 500 punti nell'intervallo $[3 - 10^{-2}, 3 - 20 * eps]$ e nell'intervallo $[3 + 20 * eps, 3 + 10^{-2}]$ con eps la precisione di macchina. Commentare i risultati. Costruire un vettore chiamato dx che contenga 500 punti dell'intervallo $[-10^{-8}, 10^{-8}]$, calcolare $x = 3 + dx$ e fare il grafico della funzione $f(x)$ rispetto a dx. Commentare i risultati.