Ottavio Serra

Elementi di analisi numerica e metodi "Monte Carlo"

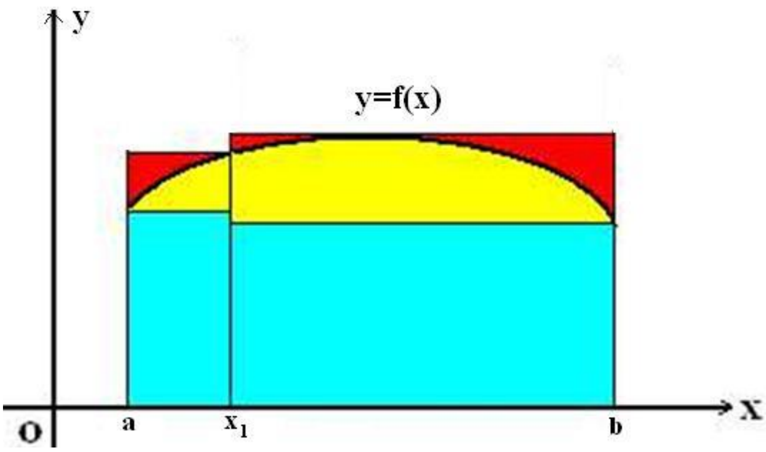
Metodi approssimati per l'integrazione: aree, volumi (e lunghezze). Formula dei trapezi ***chiusa*** e formula ***aperta***.

Metodo di bisezione per gli zeri (preventivamente isolati) di una funzione continua f(x).

Metodo di Newton per gli zeri (preventivamente isolati) di una funzione continua e derivabile.

Generazione di numeri ***random e*** Metodi *Montecarlo* per l'integrazione.

Aree. Data la funzione (continua) y=f(x) in [a, b], l'*area* del rettangoloide sottostante tra il grafico di f e l'asse x si può approssimare con rettangoli, suddividendo [a, b] in n parti con punti intermedi $x_1, x_2, \ldots x_{n-1}$. ($a=x_0$, $b=x_n$).



La somma dei rettangoli aventi altezza pari al minimo di f(x) in $[x_{i-1}, x_i]$ dà il valore dell'***area*** del rettangoloide per difetto, la corrispondente somma dei rettangoli con altezza pari al massimo la dà per eccesso. (Area celeste per difetto, area celeste + gialla esatta, area celeste + gialla più rossa per eccesso).

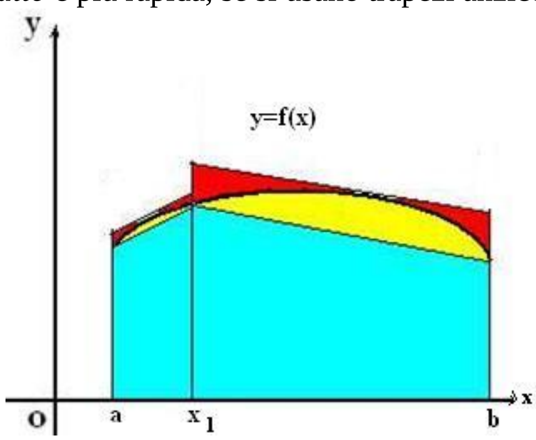
$$S_{difetto} = \sum_{i=1}^{n} m_i.\Delta x_i \le \int_a^b f(x)dx \le \sum_{i=1}^{n} M_i \Delta x_i = S_{eccesso}$$

essendo m_i ed M_i rispettivamente il minimo e il massimo di f(x) in $[x_{i-1}, x_i]$, $\Delta x_i = x_i - x_{i-1}$.

Infittendo la decomposizione di [a, b], $S_{difetto}$ cresce, $S_{eccesso}$ decresce e le due somme convergono all'***area esatta*** $\int_a^b f(x)dx$ del rettangoloide .

Il limite va fatto per $Max(\Delta x_i) \to 0$ e non per $n \to \infty$: perchè?

La convergenza al valore *esatto* è più rapida, se si usano trapezi anzicchè rettangoli.



Di solito [a, b[si decompone in parti uguali (è più comodo per i calcoli) $\Delta x_i = h = (b-a)/n$.

Usando i trapezi inscritti si ottiene la formula *chiusa* :

$$T_c = h[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(a+i.h)].$$

Usando i trapezi circoscritti, quelli aventi il lato obliquo tangente al grafico di f, il cui punto di tangenza approssimativamente ha per ascissa il punto medio di $[x_{i-1}, x_i]$ si ha la formula *aperta* (aperta nel senso che non utilizza i valori estremi f(a) ed f(b)):

$$T_a = h.\sum_{i=1}^{n} f(a-h/2+i.h) = h\sum_{i=1}^{n} f(a+\frac{2i-1}{2}h).$$

T_c e T_a danno il valore dell'***area*** una per difetto e l'altra per eccesso (se il graficodi f è concavo come nella figura, T_c lo dà per difetto, T_a per eccesso; il contrario, se f è convessa).

Dal disegno si nota, e si potrebbe dimostrare rigorosamente che, detti E_c ed E_a i corrispondenti errori, risulta $E_c \approx -2 E_a$, perciò da ***area*** = $T_c+E_c = Ta+E_a$, segue $T_c -2E_a \approx Ta+E_a \rightarrow E_a \approx (T_c -T_a)/3$ e pertanto

$$area = \int_a^b f(x)dx \approx\approx \frac{2T_a + T_c}{3}$$

Con un'approssimazione molto maggiore di quella di T_c o di T_a (formula di Simpson).

Nota Se f(x) rappresenta l'area della sezione di un solido ottenuta con piani perpendicolari a una retta(asse x), l'integrale misura il volume di un solido.

Lunghezza di una curva. Si approssima la curva con una poligonale di vertici P_0= A(a,f(a)), $P_1(x_1,f(x_1))$,… $P_i(x_i,f(x_i))$,… P_n=B(b,f(b)). Dividendo [a, b] in n parti uguali e posto $\Delta x_i = \Delta x = (b-a)/n$, $x_i = a+i.\Delta x$ (per i =0..n) , $\Delta y_i = f(x_{i+1})-f(x_i)$, la lunghezza della poligonale che approssima la lunghezza della curva (per $\Delta x \rightarrow 0$) sarà $L = \sum_{i=1}^{n} \sqrt{(\Delta x_i)^2 + (\Delta y_i)^2} = \sum_{i=1}^{n} \sqrt{1+\left(\frac{\Delta y_i}{\Delta x_i}\right)^2}.\Delta x_i$ e qui ci si potrebbe contentare. Ma se si osserva che, per $\Delta x_i \rightarrow 0$, $\Delta y_i / \Delta x_i \rightarrow f'(x_i)$, sempre che f sia derivabile, si conclude che

$$L = \int_a^b \sqrt{1+f'^2(x)}dx$$

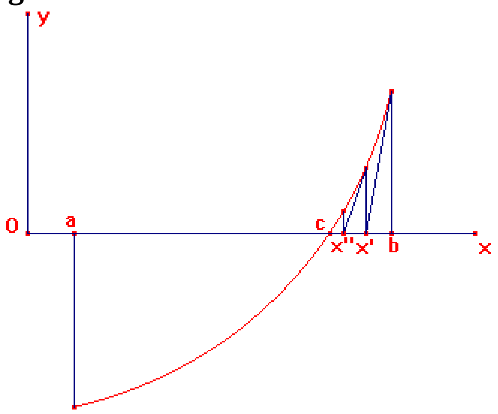
e si possono usare i metodi numerici per l'integrazione.

Zero di una funzione. Se f(x) è continua in [a, b], con f(a).f(b)<0, esiste almeno un punto interno c ad [a, b], detto *zero* di f, in cui f(c)=0. Se in [a, b] f è monotòna , questo *zero* è unico.

Metodo di bisezione. Posto S:=f(a), si **RIPETA il ciclo:** sia c:=(a+b)/2; se f(c) è concorde con S, si ponga a:=c (mi sposto a destra), altrimenti si ponga b:=c (mi sposto a sinistra),

FINCHE' (b-a) < ε (prefissato).

Metodo di Newton o della tangente. La funzione f deve essere *liscia* (derivabile).



Basta un solo punto x_0 per innescare l'iterazione; nel disegno x_0=b. La tangente al grafico in $P(x_0, f(x_0))$ è $y=f(x_0)+f'(x_0).(x-x_0)$, che taglia l'asse x nel punto x_1 (x' nella figura), risultando $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$. Iterando il calcolo, la tangente nel punto di ascissa x_1 interseca l'asse x in un punto x_2 (x'' nella figura) ancora più vicino allo *zero* ***c***. L'iterazione si arresta quando $\left|\frac{f(x_n)}{f'(x_n)}\right| < \varepsilon$.

La convergenza è quadratica, cioè ad ogni passo l'errore è proporzionale al quadrato dell'errore precedente. Ciò significa che se a un certo punto l'errore è 1/10, dopo due passi l'errore scende a 10^{-4} e al passo successivo l'errore scende a 10^{-8}.

L'iterazione è la seguente: x:=b; RIPETI Correzione:= f(x)/Derivata[f(x)] ; x:=x - Correzione

FINCHE' |Correzione|<ε.

C'è perciò bisogno di una Function che calcola la derivata: **Function** Derivata(x);

Costante h= 10^{-4} {per esempio};

Derivata:=[f(x+h)-f(x-h)]/(2h);

N.B. In quali casi il metodo di Newton non funziona?

Generazione di numeri *Random.* I numeri random o pseudo-casuali sono numeri generati con un algoritmo che simula un evento aleatorio. In un linguaggio di programmazione come *Pascal* o *Delphi* esistone delle funzioni che che generano numeri random : Random(n) restituisce un numeo random nell'intervallo di numeri interi [0, n-1], random (*senza argomento*) restituisce un numero random reale x ≥0 e <1. [0≤x<1]. Se ci servono numeri random interi n tra a e b inclusi, occorre porre n:=a+Random(b-a+1). Se ci servono numeri reali x ≥ a e < b, occorre porre x:=a+(b-a).random.

Principio dei metodi Monte Carlo. Se una figura *F* è interna a un rettangolo I di base [a, b] e altezza [min, max], un punto *random* interno al rettangolo è un punto P(x,y) con x=a+(b-a).random e y=min+(max – min).random. La probabilità che un punto P cada dentro la figura *F* è area(*F*)/area(I) e quindi area(*F*)=Probabilità.area(I). Se si generano n punti random e k cadono dentro F, avremo Area(*F*)=(k/n).(b-a).(max-min). Questo è il metodo Monte Carlo più semplice, detto *hit or miss* (colpito o mancato).Esso va bene per integrali di funzioni di una o due variabili. Per integrare funzioni di più variabili, integrali tripli o di dimensione maggiore, occorrono metodi Monte Carlo più raffinati, per esempio il metodo *Sample mean* (metodo della *media campionaria*). (Vedi nel mio sito digilander.libero.it/ottavioserra0)

Riporto il metodo di integrazione con i trapezi.

Program Integrale_Trapezi;

```
 const eps=5e-6;  Type reale=Extended; Naturale=1..MaxLongInt;
 var a,b,Ta,Tc:reale;

 FUNCTION F(x:reale):reale;
 Begin
  F:=2*Sqrt(1-x*x)   (* ESEMPIO DI FUNZIONE DA INTEGRARE*)
 End;

 Procedure trapezi;
  var i,n:Naturale;h,x,sa,sc:reale;
 begin h:=b-a;sa:=f((a+b)/2);sc:=(f(a)+f(b))/2;
  ta:=sa*h;Tc:=sc*h;n:=1;
  repeat n:=2*n; h:=h/2;x:=a-h/2;
   sc:=sc+sa;sa:=0;
   for i:=1 to n do
   begin x:=x+h;sa:=sa+f(x)
   end;
   Tc:=sc*h; Ta:=sa*h;
```

```
  writeln(Tc:30:5,Ta:30:5)
 until Abs(Tc-Ta)<eps;
 WRITELN('                              N° iterazioni:',n)
 end;
BEGIN  writeln;
 writeln('                 OTTAVIO SERRA');
 writeln('         INTEGRALE DEFINITO DI UNA FUNZIONE.');
 writeln('                (Metodo dei trapezi). ');
 writeln('    Devi avere già inserito la f(x) nella FUNCTION');
 write('      Estremo sinistro della x: a=');readln(a);
 write('      Estremo destro   della x: b=');readln(b);
 TRAPEZI(a,b,ta,tc);
 Write('    FINE, premi IMVIO:');readln
END.
```

Esercizio. Volendo generare in modo casuale i numeri interi da 1 ad **n** (per esempio n=90 come nella tombola) con sole **n** *estrazioni*, cioè senza dover ripetere l'estrazione se un numero è già uscito, come procedere? **Suggerimento** per un possibile algoritmo: porre i numeri da 1 ad n in un *vettore* e poi generare un *indice* random da 1 a k, dove k va decrescendo da n a 1 e procedere a un opportuno scambio.