Integrazione di funzioni

Il problema della integrazione di funzioni riveste un ruolo centrale in molti campi della Matematica. La nostra trattazione, estremamente superficiale e semplificata, ha come scopo l'introduzione agli aspetti computazionali del problema e fornisce un'altra serie di esempi dell'uso di Mathematica. *Purtroppo l'articolo è di difficile lettura per chi non conosce un minimo di analisi matematica. Lo scopo principale non è tanto fare una lezione di calcolo numerico, (esistono ottimi libri, anche in Italiano), quanto far vedere come con* Mathematica *molte sperimentazioni complesse possano essere implementate con poche righe di codice, Viene inoltre sottolineata la potenza di un uso coordinato di elaborazione simbolica e numerica*

Per distinguere i discorsi di tipo matematico dagli esempi di programmazione, questi ultimi sono riportati in corpo più piccolo ed in carattere corsivo.

di Francesco Romani

Introduzione

Data una funzione continua f(x), a valori reali non negativi, definita su un intervallo [a,b] della retta reale, l'integrale

$$\int_a^b f(x)dx$$

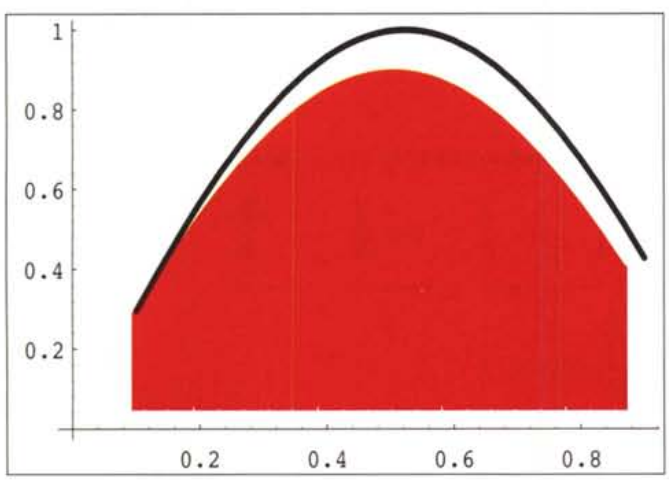
(1)

può essere definito come l'area della porzione di piano compresa tra la funzione e l'asse delle ascisse.

Con il Package Graphics`FilledPlot` è facile rappresentare geometricamente la definizione di cui sopra. (L'integrale definito è l'area della superficie rossa in Figura.1).

In[1]:=

```
<<"Graphics`FilledPlot`"
<<"Graphics`Colors`"
f[x_]:=Sin[3x];
FilledPlot[f[x],{x,0.1,0.9},
    PlotStyle->{Thickness[.01]},
    Fills->{Red}];
```



Una primitiva di f(x) è una qualunque funzione F(x) che soddisfa la relazione F'(x) = f(x). Il teorema fondamentale del calcolo integrale permette di calcolare l'integrale definito (1) come la differenza dei valori agli estremi dell'intervallo di integrazione di una funzione primitiva di f(x).

$$\int_a^b f(x)\,dx = F(b) - F(a).$$

(2)

Per alcune delle più semplici funzioni matematiche è possibile, attraverso le tecniche del calcolo simbolico dell'analisi, calcolare la forma esplicita di una primitiva e quindi ottenere una espressione matematica esatta per l'integrale definito.

Mathematica contiene un potente integratore simbolico, ovviamente meno astuto di un esperto matematico, ma certamente più abile del generico studente del liceo scientifico o dei primi anni di Università (o di un pasticcione come il sottoscritto).

Vediamo alcuni esempi dell'uso di **Integrate** *e come si può generare una tavola di primitive di funzioni elementari.*

In[2]:=

```
Integrate[a x+b,x]
```

Out[2]=

$$b\,x + \frac{a\,x^2}{2}$$

In[3]:=

```
Integrate[3/2 x^2,{x,0,1}]
```

Out[3]=

$$\frac{1}{2}$$

In[4]:=

```
Integrate[3/2 x^2,{x,a,b}]
```

Out[4]=

$$\frac{-a^3}{2} + \frac{b^3}{2}$$

In[5]:=

```
l={x^a, 1/x, E^x, Log[x], Sin[x], Cos[x]};
L=Integrate[l,x];
TableForm[Transpose[{l,L}]]
```

Out[5]=

x^a	$\frac{x^{1+a}}{1+a}$

```
1
-        Log[x]
x

E^x          E^x

Log[x]    -x + x Log[x]

Sin[x]    -Cos[x]

Cos[x]    Sin[x]
```

Purtroppo, frequentemente, si presentano due tipi di problemi di diversa natura ma di pari difficoltà:

- la funzione primitiva non esiste in forma esplicita. Ciò può accadere anche per funzioni molto semplici come per esempio e^{-x2};
- la funzione primitiva esiste in forma esplicita ma questa è molto complicata. L'utilizzazione diretta della primitiva porta a problemi di efficienza nel calcolo dell'integrale o di precisione numerica o di difficile verifica del programma corrispondente.

Per esempio vediamo la primitiva di arctan[x3] :

In[6]:=

```
Integrate[ArcTan[x^3],x]
```

Out[6]=

```
Sqrt[3] ArcTan[Sqrt[3]-2x]
------------------------- + x ArcTan[x^3] +
            2

  Sqrt[3] ArcTan[Sqrt[3]+2x]   Log[1+x^2]
  ------------------------- + ---------- -
              2                    2

  Log[1 - Sqrt[3] x + x^2]
  ------------------------ -
             4

  Log[1 + Sqrt[3] x + x^2]
  ------------------------
             4
```

Anche se alcune semplificazioni possono essere fatte a mano, il calcolo di una tale espressione non risulta né agevole né incoraggiante.

Integrazione approssimata

Un metodo per calcolare approssimativamente l'area compresa sotto la funzione, e quindi l'integrale (1) consiste nel suddividere l'intervallo di integrazione [a,b] in molti sotto-intervalli e quindi considerare i rettangoli aventi come base ciascun intervallo e come altezza il valore della funzione nel punto medio dell'intervallo.

L'uso combinato di **Plot** *e* **ListFilledPlot**` *permette di rappresentare graficamente anche questa definizione . (vedi Figura. 2).*

In[7]:=

```
l=Flatten[Table[{{x,0},{x,f[x+.1]},
     {x+.2,f[x+.1]}},{x,0.1,0.9,.2}],1]
```

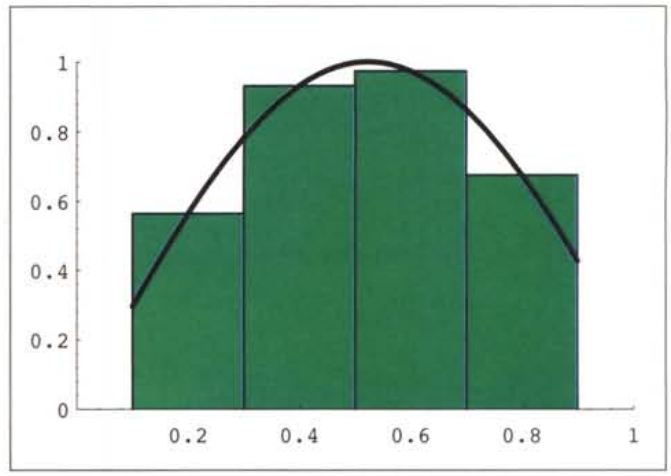
Out[7]=

```
{ {0.1, 0}, {0.1, 0.564642}, {0.3, 0.564642},
  {0.3, 0}, {0.3, 0.932039}, {0.5, 0.932039},
  {0.5, 0}, {0.5, 0.973848}, {0.7, 0.973848},
  {0.7, 0}, {0.7, 0.675463}, {0.9, 0.675463},
  {0.9, 0}, {0.9, 0.14112},  {1.1, 0.14112}}
```

In[8]:=

```
lp=ListFilledPlot[Drop[l,-2],
    Fills->{Green}];
pl=Plot[f[x],{x,0.1,0.9},
    PlotStyle->{Thickness[.01]}];
Show[lp,pl,PlotRange->{{0,1},{0,1}}]
```

Prendendo un numero sempre maggiore di intervalli aventi



lunghezze sempre più piccole l'area totale dei rettangoli si avvicina sempre di più all'integrale cercato. Questo modo di procedere, che rispecchia la definizione matematica dell'integrale come limite delle *somme di Riemann*, si può formalizzare come segue. Supponiamo di scegliere n sotto-intervalli tutti di uguale lunghezza h=(b-a)/n, allora l'integrale (1) viene approssimato con

$$I_n = h\sum_{i=0}^{n-1} f(a+hi+\frac{h}{2}). \qquad (3)$$

Si può inoltre dimostrare che il limite di I_n per $n\to\infty$ è l'integrale (1).

Costruiamo con Mathematica una tabella che mostra l'andamento dell'errore commesso nella approssimazione del calcolo dell'integrale di **Sin[3x]** *in [0.1,0.9] con il metodo (3). I calcoli sono stati eseguiti con 35 cifre decimali al fine di poter trascurare i problemi derivanti dall'uso di una aritmetica finita.*

In[9]:=

```
f[x_]:=Sin[3x];
a=1/10;
b=9/10;
Iesatto=Integrate[f[x],{x,a,b}]
```

Out[9]=

```
Cos[3/10]    Cos[27/10]
--------- - ----------
    3            3
```

In[10]:=

```
N[Iesatto,35]
```

Out[10]=

```
0.61980287704755572254161250317046001
```

In[11]:=

```
Iappr[n_]:=Module[{h,nodi},
  h=(b-a)/n;
  nodi=Table[a + h i + h/2,{i,0,n-1}];
  N[h Plus@@Map[f,nodi],35]];
```

In[12]:=
```
Do[Print[n=10^i," ",
        N[Abs[Iesatto-Iappr[n]]]],{i,1,10}]
```

```
10     0.00149003
100    0.0000148755
1000   1.48753 10^-7
10000  1.48753 10^-9
```

Formule Interpolatorie

Dall'esempio della Tabella 2 si nota come l'errore diminuisce in modo quadratico rispetto al crescere di n (ovvero se n viene moltiplicato per 10 l'errore viene diviso per 10^2). Tale comportamento rende estremamente costoso ottenere precisioni elevate. Nell'esempio sopra riportato, per ottenere 15 cifre esatte dell'integrale cercato sarebbero necessarie circa un miliardo di valutazioni della funzione integranda (in pratica, non potendosi trascurare l'errore dovuto alla aritmetica finita, la situazione è ancora peggiore). Una soluzione alternativa, semplice ed efficace, consiste nel cercare una funzione, facile da integrare esattamente, che assuma valori molto vicini alla funzione da integrare. Di solito si sceglie un polinomio p(x) che *interpola* f(x) (vedi MC n. 133).
L'integrale del polinomio approssimante di Lagrange si può scrivere:

$$\int_a^b p(x)\,dx = \sum_{i=1}^{n} f(x_i)\left(\int_a^b \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\,dx\right)$$

Gli integrali tra parentesi non dipendono dai valori assunti dalla funzione f(x) ma solo dall'intervallo di integrazione e dalla scelta dei punti x_i; l'integrale del polinomio p(x) è una approssimazione dell'integrale cercato e si può scrivere nel modo seguente:

$$I_n = \sum_{i=1}^{n} w_i f(x_i). \qquad (4)$$

Questo tipo di formula di quadratura è detta *interpolatoria*. I punti $\{x_1,x_2,...,x_n\}$ formano un insieme di n punti distinti interni all'intervallo di integrazione detti *nodi* e i valori $\{w_1,w_2,...,w_n\}$ un insieme di n valori reali detti *pesi* della formula di quadratura. È evidente che una formula interpolatoria su n punti è esatta (a meno di eventuali errori di arrrotondamento) se la funzione da integrare è anch'essa un polinomio di grado al più n-1.
Utilizzando Mathematica, *il programma per il calcolo dei pesi di una formula interpolatoria si basa sulla formula vista sopra ovvero sulla inegrazione simbolica del polinomio di Lagrange e peresenta una ccomplessità di ordine* n^2.
In[13]:=
```
Lagrange[x_,i_]:=Product[
  If[i=!=j,(t-x[[j]])/(x[[i]]-x[[j]]),1],
     {j,Length[x]}]
```

In[14]:=
```
Pesi[nodi_]:=Integrate[
Table[Lagrange[nodi,i],
                {i,Length[nodi]}], {t,-1,1}]
```
In[15]:=
```
Pesi[{-1,1}]
```
Out[15]=
```
{1, 1}
```
In[16]:=
```
Pesi[{-1,0,1}]
```
Out[16]=
```
 1  4  1
{-, -, -}
 3  3  3
```
È importante notare che con un linguaggio di programmazione che non riunisca le capacità di calcolo simbolico con quelle di calcolo numerico l'approccio immediato visto sopra non è possibile. Il calcolo dei pesi per via puramente numerica di solito viene effettuato risolvendo un sistema lineare con una complessità di ordine n^3*, affrontando nel contempo, problemi non banali di stabilità numerica.*
La scelta più immediata per i nodi di una formula interpolatoria è quella di prendere n punti equidistanti nell'intervallo di integrazione (x_i = a+(i-1) h, i=1,2,..,n, h=(b-a)/(n-1)). Le formule risultanti sono dette di *Newton-Cotes* . Di solito si calcolano i pesi per l'intervallo prefissato [-1,1].
Definiamo i nodi e i pesi di Newton-Cotes su [-1,1].
In[17]:=
```
NCnodi[n_]:=
      Table[-1+2/(n-1) i,{i,0,n-1}];
NCpesi[n_]:= Pesi[NCnodi[n]];
```
In[18]:=
```
Do[Print[n," ",
        InputForm[NCpesi[n]]],{n,2,10}]
```

```
2 {1, 1}
3 {1/3, 4/3, 1/3}
4 {1/4, 3/4, 3/4, 1/4}
5 {7/45, 32/45, 4/15, 32/45, 7/45}
6 {19/144, 25/48, 25/72, 25/72, 25/48,
19/144}
7 {41/420, 18/35, 9/140, 68/105, 9/140,
18/35, 41/420}
8 {751/8640, 3577/8640, 49/320, 2989/8640,
   2989/8640, 49/320, 3577/8640, 751/8640}
9 {989/14175, 5888/14175, -928/14175,
10496/14175,
   -908/2835, 10496/14175, -928/14175,
5888/14175,
   989/14175}
10 {2857/44800, 15741/44800, 27/1120,
1209/2800,
    2889/22400, 2889/22400, 1209/2800,
27/1120,
    15741/44800, 2857/44800}
```
Si possono fare alcune osservazioni sulle formule di Newton-Cotes:
• $w_i = w_{n-i+1}$, i=1,2,...n, ovvero i pesi sono simmetrici rispetto al centro dell'intervallo (questa proprietà è dimostrabile per ogni formula interpolatoria con nodi simmetrici rispetto al centro dell'intervallo di integrazione). La somma dei pesi è uguale alla lunghezza dell'intervallo.
• i pesi sono tutti positivi fino ad un certo punto poi compaiono anche pesi negativi. Questo è un fatto molto spiacevole, infatti le formule con pesi negativi presentano maggiori problemi dal punto di vista della accuratezza numerica dei risultati.

Bibliografia

R.Bevilacqua, D. Bini, M. Capovani, O.Menchi. **Introduzione alla Matematica Computazionale.** Zanichelli ,1987.
R.Bevilacqua, D. Bini, M. Capovani, O.Menchi. **Metodi Numerici.** Zanichelli, 1992.
P.Davis and P. Rabinowitz. **Methods of Numerical Integration.** Academic Press, 1984. *(È la "Bibbia" dell'integrazione numerica consigliabile a tutti coloro che vogliono approfondire seriamente l'argomento)*
F.Romani. **Metodi di Approssimazione.** MCmicrocomputer, 133, ottobre 1993.
S. Wolfram, ***Mathematica.*** **A System for Doing Mathematics by Computer, II Edition**. Addison Wesley, 1991.

La trasformazione lineare

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)dt$$

permette di usare una formula calcolata per l'intervallo [-1,1] per integrare f in un qualsiasi intervallo [a,b].
`ApplyRule[nodi,pesi]` integra f tra a e b applicando una formula interpolatoria di nodi e pesi assegnati (calcolati per l'intervallo [-1,1]).

In[[19]:=
```
ApplyRule[nodi_,pesi_]:=N[(b-a)/2
   Map[f,(b-a)/2 nodi + (a+b)/2].pesi, 35]
```
In[20:=
```
ApplyRule[a,b,{-1,0,1},{1/3,4/3,1/3}]
```
Out[20]=
```
0.62838400444151829779043564214784764
```
Costruiamo la tabella dei risultati della applicazione delle formule di Newton-Cotes al calcolo del nostro integrale.
In[21]:=
```
Do[Print[n," ",N[Abs[Iesatto-
 ApplyRule[a,b,NCnodi[n],NCpesi[n]]]],
   {n,2,10}]
```
```
2  0.330643
3  0.00858113
4  0.00373937
5  0.0000748656
6  0.00004192
7  5.39411 10^-7
8  3.29754 10^-7
9  2.89399 10^-9
10 1.85039 10^-9
```
Si nota una netta superiorità rispetto alla applicazione del metodo delle somme di Riemann.

Formule di Gauss

Formule interpolatorie molto migliori possono essere ottenute grazie ad un complesso procedimento matematico basato sui *polinomi ortogonali*. Tali formule, dette formule *Gaussiane*, hanno le seguenti caratteristiche:

- i nodi sono ottenuti come radici di un particolare polinomio, essi sono numeri irrazionali, tutti all'interno dell'intervallo di integrazione e simmetrici rispetto al centro dell'intervallo; anche i pesi sono numeri irrazionali ma sempre positivi.
- le formule di Gauss con n punti integrano esattamente polinomi fino al grado 2n-1 e non esistono formule interpolatorie con n punti capaci di fare di meglio.

Costruire e applicare le formule di Gauss con un normale linguaggio richiede l'uso di numerosi programmi per il calcolo degli opportuni polinomi ortogonali, per la ricerca degli zeri e il calcolo dei pesi. Con i nostri potenti mezzi il compito è molto più facile. Una volta scoperto, dalla teoria, che i nodi per le formule di Gauss tra -1 e 1 sono le radici dei polinomi di Legendre troviamo questi ultimi già pronti nel nostro linguaggio:
In[22]:=
```
LegendreP[4,x]
```
Out[22]=
```
          2        4
3 - 30 x  + 35 x
-----------------
        8
```
I nodi si trovano risolvendo l'equazione `LegendreP[n,x]==0` *(con 35 cifre di precisione)*
In[23]:=
```
GLnodi[n_] :=
   x/.NSolve[LegendreP[n,x]==0,x,35];
GLpesi[n_]:= Pesi[GLnodi[n]];
```
La tabella seguente mostra i risultati della applicazione delle formule di GaussLegendre al calcolo del solito integrale. Il miglioramento rispetto alle formule di Newton-Cotes è strabiliante: bastano 11 punti per avere una precisione di 26 cifre decimali.
In[24]:=
```
Do[Print[n," ",N[Abs[Iesatto-
 ApplyRule[a,b,GLnodi[n],GLpesi[n]]]],
   {n,3,11,2}]
```
```
3  0.0000723804
5  1.93742  10^-9
7  1.06569  10^-14
9  1.9023   10^-20
11 1.41494  10^-26
```
In teoria non esiste un limite al numero di punti con cui può essere calcolata una formula di tipo Gaussiano; dal punto di vista pratico non conviene scegliere formule con più di 20-50 punti perché i pesi e nodi delle formule di Gauss devono essere precalcolati e memorizzati nel programma e inoltre, se la funzione è regolare, pochi punti sono più che sufficienti a raggiungere i limiti di precisione imposti dall'uso di un aritmetica finita dei linguaggi tradizionali (di solito circa 16 cifre decimali). Se la funzione presenta delle discontinuità è preferibile spezzare l'intervallo di integrazione e trattare separatamente i sottointervalli ottenuti.

Stime dell'errore

In generale non serve disporre di una formula di integrazione molto precisa se poi non si possiedono metodi per valutare l'errore commesso nel calcolo dell'integrale. Si possono distinguere due fonti di errore: l'*errore analitico* pari alla differenza tra l'integrale esatto e la sua approssimazione teorica, e l'*errore di arrotondamento,* pari alla differenza tra l'approssimazione teorica e quella effettivamente calcolata con un

numero fini to di cifre di precisione.
Nella pratica le stime teoriche dell'errore risultano non applicabili e si preferisce effettuare stime empiriche basate sul fatto che se I' e I" sono due approssimazioni per l'integrale I, e inoltre I" è molto più precisa di I' allora | I'-I" | è una buona stima di | I'-I | e (spesso ma non sempre) una stima largamente pessimistica di | I-I" |. In base a queste considerazioni un algoritmo per il calcolo di una approssimazione dell'integrale I e di una stima dell'errore può essere strutturato come segue:

- si calcola I' con una formula di precisione h;
- si calcola I" con una formula di precisione k>h;
- si stima l'errore con la formula $E_{stimato}$ = | I'-I" |;
- si restituisce la coppia di valori (I",$E_{stimato}$).

Tecniche più raffinate tengono conto anche dell'errore di arrotondamento.
Il lettore si domanderà cosa fare se, data un funzione da integrare e due regole di integrazione, la stima empirica della precisione del risultato ci mostra che siamo lontani dal soddisfare le necessità dell'utente.
Ad esempio, sia f(x)=-√(,x) la funzione da integrare nell'intervallo [0,2] utilizzando la formula di Gauss con 9 punti per calcolare I" e la formula con 7 punti per calcolare I', abbiamo:

In[25]:=
```
f[x_]:=Sqrt[x];
a=0;
b=2;
Iesatto=Integrate[f[x],{x,0,2}]
```
Out[25]=
```
4 Sqrt[2]
---------
    3
```
In[26]:=
```
Iesatto=N[Iesatto,35]
```
Out[26]=
```
1.8856180831641267317355849656129308
```
In[27]:=
```
I1=ApplyRule[GLnodi[7],GLpesi[7]]
```
Out[27]=
```
1.8863150597131536907422045907967
```
In[28]:=
```
I2=ApplyRule[GLnodi[9],GLpesi[9]]
```
Out[28]=
```
1.8859597795066652761272285941 66
```
In[29]:=
```
Estimato = I2-I1
```
Out[29]=
```
-0.0003552802064884146149759966 3
```
Ls stima prevede quindi un errore sulla terza cifra decimale.
Per la cronaca,
In[30]:=
```
Evero = Iesatto - I2
```
Out[30]=
```
-0.0003416963425385443916436285 54
```
ma questo l'utente non lo sa.
Se necessitano 12 cifre di precisione, tutto il lavoro sembra inutile e si presentano due possibilità:
1) procurarsi una coppia di formule di precisione maggiore e ritentare, sperando di essere più fortunati;
2) ridurre il problema del calcolo dell'integrale originario alla somma di più integrali su intervalli più piccoli per i quali le nostre formule lavorano meglio.
Il secondo approccio, certamente più scientifico, è la base di una serie di tecniche per la *approssimazione automatica di integrali numerici* di cui parleremo in una prossima puntata.

MC

Francesco Romani è raggiungibile tramite Internet all'indirizzo romani@di.unipi.it