Quadratura

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

28 marzo 2017

Quadratura numerica

Problema.

Un classico problema dell'analisi numerica è quello di calcolare l'integrale definito di una funzione f in un intervallo avente estremi di integrazione a, b (non necessariamente finiti) cioè

$$I_w(f) := I_w(f, a, b) = \int_a^b f(x)\, w(x) dx$$

dove w è una funzione peso in (a, b) [1, p.206, p.270].

La nostra intenzione è di approssimare $I(f)$ come

$$I_w(f) \approx Q_N(f) := \sum_{i=1}^{N} w_i f(x_i) \qquad (1)$$

I termini w_i e $x_i \in [\alpha, \beta]$ sono detti rispettivamente pesi e nodi.

Formule di Newton-Cotes

Ricordiamo alcune formule di Newton-Cotes (chiuse).

Definizione (Regola del trapezio)

La formula

$$I(f) \approx S_1(f) := S_1(f, a, b) := \frac{(b-a)(f(a)+f(b))}{2}$$

si chiama regola del trapezio.

Definizione (Regola di Cavalieri-Simpson)

La formula

$$I(f) \approx S_3(f) := S_3(f, a, b) := \frac{b-a}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right]$$

si chiama regola di Cavalieri-Simpson.

Formule di Newton-Cotes composte

Visto che per $N \geq 8$ le formule risultano instabili, ci si domanda se sia possibile ottenere per $N \geq 8$ delle formule stabili.

Definizione (Formule composte)

Si suddivida l'intervallo (chiuso e limitato) $[a, b]$ *in* N *subintervalli* $T_j = [x_j, x_{j+1}]$ *tali che* $x_j = a + jh$ *con* $h = (b-a)/N$. *Dalle proprietà dell'integrale*

$$\int_a^b f(x)\,dx = \sum_{j=0}^{N-1} \int_{x_j}^{x_{j+1}} f(x)\,dx \approx \sum_{j=0}^{N-1} S(f, x_j, x_{j+1}) \qquad (2)$$

dove S *è una delle regole di quadratura finora esposte (ad esempio* $S_3(f)$*). Le formule descritte in (2) sono dette* **composte**.

Formule di Newton-Cotes composte

Definizione (Formula dei trapezi)

Siano $x_i = a + i - 1h$, $i = 1, \ldots, N$ *con* $h = (b-a)/N$. *La formula*

$$S_1^{(c)} := h \left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{N-1}) + \frac{f(x_N)}{2} \right] \qquad (3)$$

si chiama dei trapezi (o del trapezio composta).

Definizione (Formula di Cavalieri-Simpson composta)

Fissati N subintervalli, sia $h = \frac{b-a}{N}$. *Siano inoltre* $x_k = a + kh/2$, $k = 0, \ldots, 2N$. *La formula*

$$I(f) \approx S_3^{(c)}(f) := \frac{h}{6} \left[f(x_0) + 2 \sum_{r=1}^{N-1} f(x_{2r}) + 4 \sum_{s=0}^{N-1} f(x_{2s+1}) + f(x_{2N}) \right]$$

è nota come di Cavalieri-Simpson composta.

Implementazione Matlab di alcune formule composte

Mostriamo in Matlab/Octave un'implementazione trapezi_composta.m della formula composta dei trapezi

$$S_1^{(c)} := h \left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{N-1}) + \frac{f(x_N)}{2} \right] \tag{4}$$

nel caso $\{x_k\}$ siano equispaziati.

```
function [x,w]=trapezi_composta(N,a,b)

% FORMULA DEI TRAPEZI COMPOSTA.
% INPUT:
% N: NUMERO SUBINTERVALLI.
% a, b: ESTREMI DI INTEGRAZIONE.
% OUTPUT:
% x: NODI INTEGRAZIONE.
% w: PESI INTEGRAZIONE (INCLUDE IL PASSO!).

h=(b-a)/N;                  % PASSO INTEGRAZIONE.
x=a:h:b; x=x';              % NODI INTEGRAZIONE.
w=ones(N+1,1);              % PESI INTEGRAZIONE.
w(1)=0.5; w(N+1)=0.5;
w=w*h;
```

Implementazione Matlab di alcune formule composte

(Commento)

La funzione `trapezi_composta` *appena esposta calcola i nodi e i pesi della omonima formula composta.*
L'unica difficoltà del codice consiste nel calcolo dei pesi `w`*. Da*

$$I(f) \approx S_1^c(f) := \sum_{i=0}^{N} w_i f(x_i) \tag{5}$$

come pure per (3)

$$S_1^{(c)} := h \left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{N-1}) + \frac{f(x_N)}{2} \right] \tag{6}$$

deduciamo che $w_0 = w_N = h/2$ *mentre* $w_1 = \ldots = w_{N-1} = h$*, così giustificando le ultime linee della function* `trapezi_composta`*.*

Implementazione Matlab di alcune formule composte

Mostriamo in Matlab/Octave un'implementazione simpson_composta.m della formula composta di Cavalieri-Simpson

$$I(f) \approx S_3^{(c)}(f) := \frac{h}{6}\left[f(x_0) + 2\sum_{r=1}^{N-1} f(x_{2r}) + 4\sum_{s=0}^{N-1} f(x_{2s+1}) + f(x_{2N})\right]$$

nel caso $\{x_k\}$ siano equispaziati.

```
function [x,w]=simpson_composta(N,a,b)
% FORMULA DI SIMPSON COMPOSTA.
% INPUT:
% N: NUMERO SUBINTERVALLI.
% a, b: ESTREMI DI INTEGRAZIONE.
% OUTPUT:
% x: INTEGRAZIONE.
% w: PESI INTEGRAZIONE (INCLUDE IL PASSO!).
h=(b-a)/N;                  % AMPIEZZA INTERVALLO.
x=a:(h/2):b; x=x';          % NODI INTEGRAZIONE.
w=ones(2*N+1,1);                % PESI INTEGRAZIONE.
w(3:2:2*N-1,1)=2*ones(length(3:2:2*N-1),1);
w(2:2:2*N,1)=4*ones(length(2:2:2*N),1);
w=w*h/6;
```

Implementazione Matlab di alcune formule composte

Una volta noti il vettore (colonna) x dei nodi e w dei pesi di integrazione, se la funzione f è richiamata da un m-file $f.m$, basta

```
fx=f(x);                    % VALUT. FUNZIONE.
I=w'*fx;                    % VALORE INTEGRALE.
```

per calcolare il risultato fornito dalla formula di quad. composta.

Ricordiamo che se $\mathbf{w} = (\mathbf{w}_k)_{k=1,\ldots,M}$, $\mathbf{fx} = (f(x_k))_{k=1,\ldots,M}$ sono due vettori colonna allora il prodotto scalare

$$\mathbf{w} * \mathbf{fx} := \sum_{k=1}^{M} \mathbf{w}_k \cdot \mathbf{fx}_k := \sum_{k=1}^{M} \mathbf{w}_k \cdot f(x_k)$$

si scrive in Matlab/Octave come `w'*fx` (dimensionalmente il prodotto di un vettore $1 \times M$ con un vettore $M \times 1$ dà uno scalare (cioè un vettore 1×1)).

Implementazione Matlab di alcune formule composte

Esempio

Calcolare in Matlab

$$\begin{aligned}\int_0^1 \sin(x)dx &= (-\cos(1)) - (-\cos(0)) = -\cos(1) + 1 \\ &\approx 0.45969769413186.\end{aligned}$$

utilizzando la formula dei trapezi composta, suddividendo $[0,1]$ *in* 10 *subintervalli equispaziati.*

```
>> format long;
>> [x,w]=trapezi_composta(10,0,1);
>> fx=sin(x);
>> I_trapezi_composta=w'*fx
I_trapezi_composta =
   0.45931454885798
>> length(x)
ans =
    11
>>
```

Implementazione Matlab di alcune formule composte

Esempio

Calcolare in Matlab

$$\begin{aligned}\int_0^1 \sin(x)dx &= (-\cos(1)) - (-\cos(0)) = -\cos(1) + 1 \\ &\approx 0.45969769413186.\end{aligned}$$

utilizzando la formula di Cavalieri-Simpson composta, suddividendo $[0,1]$ *in* 5 *subintervalli equispaziati.*

```
>> format long;
>> [x,w]=simpson_composta(5,0,1);
>> fx=sin(x);
>> I_simpson=w'*fx
I_simpson =
   0.45969794982382
>> length(x)
ans =
    11
>>
```

Implementazione Matlab di alcune formule composte

Facciamo ora un altro esempio. Calcoliamo numericamente utilizzando le formule composte sopra citate

$$\int_{-1}^{1} x^{20} dx = (1^{21}/21) - (-1)^{21}/21 = 2/21 \approx 0.09523809523810.$$

A tal proposito scriviamo il seguente codice demo_composte.m.

```
a=-1; b=1; N=11;    %SCEGLIERE N DISPARI.
N_trap=N-1;
[x_trap,w_trap]=trapezi_composta(N_trap,a,b);
fx_trap=x_trap.^20;                    % VALUT. FUNZIONE.
I_trap=w_trap'*fx_trap;                % TRAPEZI COMPOSTA.
N_simpson=(N-1)/2;
[x_simp,w_simp]=simpson_composta(N_simpson,a,b)
fx_simp=x_simp.^20;                    % VALUT. FUNZIONE.
I_simp=w_simp'*fx_simp;                % SIMPSON COMPOSTA.
fprintf('\n\t [TPZ COMP][PTS]:%4.0f',length(x_trap));
fprintf('\n\t [TPZ COMP][RIS]:%14.14f',I_trap);
fprintf('\n\t [SIMPS.COMP][PTS]:%4.0f',length(x_simp));
fprintf('\n\t [SIMPS.COMP][RIS]:%14.14f',I_simp);
```

Implementazione Matlab di alcune formule composte

ottenendo

```
>> demo_composte
  [TPZ.COMP.][PTS]:    11
  [TPZ.COMP.][RIS]:  0.20462631505024
  [CS.COMP.][PTS]:    11
  [CS.COMP.][RIS]:  0.13949200364447
>>
```

Si può vedere che usando formule di derivazione più complessa, del tipo gaussiano o di tipo Clenshaw-Curtis (cf. [10], [11]) a parità di valutazioni della funzione f avremmo ottenuto

```
    [GAUSS-LEGENDRE   ]: 0.095238095238095649
    [CLENSHAW-CURTIS  ]: 0.094905176204004307
```

col costo aggiuntivo di dover calcolare tramite complicati algoritmi i pesi e i nodi di Gauss o i nodi di Clenshaw-Curtis via un IFFT.

Esercizio

Esercizio

Usando la formula composta dei trapezi e di Cavalieri-Simpson, con $n = 2, 4, 8, \ldots, 512$ *suddivisioni equispaziate dell'intervallo di integrazione, calcolare*

- $\int_0^{\pi} \exp(x) \cdot \cos(4x) dx = \frac{\exp(\pi)-1}{17}$;
- $\int_0^1 x^{5/2} dx = \frac{2}{7}$;
- $\int_{-\pi}^{\pi} \exp(\cos(x)) dx = 7.95492652101284$;
- $\int_0^{\pi/4} \exp(\cos(x)) dx = 1.93973485062365$;
- $\int_0^1 x^{1/2} dx = \frac{2}{3}$.

Descrivere come decresce l'errore in scala semilogaritmica e quale sia l'andamento del rapporto $E_n(f)/E_{2n}(f)$ *con* $n_1 = 2$, *dove* $E_n(f) = |I(f) - I_n(f)|$ *con* $I(f)$ *integrale esatto e* $I_n(f)$ *valore fornito dalla formula di quadratura. Sembra asintoticamente costante?*

Nota.

Si osservi che se $E_n(f) \approx C/n^k$ *allora* $E_n(f)/E_{2n}(f) \approx 2^k$.

Formule gaussiane

Sia $w : (a, b) \to \mathbb{R}$ (non necessariamente limitato) è una funzione peso, cioè tale che (cf. [1, p.206, p.270])

1. w è nonnegativa in (a, b);
2. w è integrabile in $[a, b]$;
3. esista e sia finito
$$\int_a^b |x|^n w(x)\, dx$$
per ogni $n \in \mathbb{N}$;
4. se
$$\int_a^b g(x) w(x)\, dx$$
per una qualche funzione nonnegativa g allora $g \equiv 0$ in (a, b).

Formule gaussiane

Tra gli esempi più noti ricordiamo

1. *Legendre*: $w(x) \equiv 1$ in $[a, b]$ limitato;
2. *Jacobi*: $w(x) = (1-x)^{\alpha}(1+x)^{\beta}$ in $(-1, 1)$ per $\alpha,\ \beta \geq -1$;
3. *Chebyshev*: $w(x) = \frac{1}{\sqrt{1-x^2}}$ in $(-1, 1)$;
4. *Laguerre*: $w(x) = \exp(-x)$ in $[0, \infty)$;
5. *Hermite*: $w(x) = \exp(-x^2)$ in $(-\infty, \infty)$;

Formule gaussiane

Teorema (Formule gaussiane)

Per ogni $n \geq 1$ esistono e sono unici dei nodi $x_1, \ldots, x_n$ e pesi $w_1, \ldots, w_n$ per cui la formula

$$\int_a^b f(x)w(x) \approx \sum_{k=1}^{n} w_k f(x_k)$$

ha grado di precisione almeno $2n-1$.

I nodi sono gli zeri del polinomio ortogonale di grado n,

$$\phi_n(x) = A_n \cdot (x - x_1) \cdot \ldots \cdot (x - x_n)$$

e i corrispettivi pesi sono

$$w_i = \int_a^b L_i(x)w(x)dx = \int_a^b L_i^2(x)w(x)dx,\ i = 1, \ldots, n.$$

Formule gaussiane in Matlab

Eccetto che in pochi casi (come ad esempio per la funzione peso di Chebyshev), non esistono formule esplicite per l'individuazione di tali nodi e pesi.

Una volta venivano tabulati, oggi si consiglia di applicare del software che si può trovare ad esempio nella pagina di Walter Gautschi

$$http://www.cs.purdue.edu/people/wxg$$

Formule gaussiane in Matlab

Fissato un peso, ad esempio quello di Jacobi, si cercano per prima cosa i *coefficienti di ricorrenza*, che possono essere calcolati con l' m-file r_jacobi.m nel sito di W. Gautschi.
La sintassi è la seguente

$$\boxed{\texttt{ab} = \texttt{r\_jacobi}(\texttt{n}, \texttt{a}, \texttt{b})}$$

Come si vede dai commenti al file, genera i primi n coeff. di ricorrenza dei polinomi ortogonali monici relativamente alla funzione peso di Jacobi

$$w(x) = (1-x)^a (1+x)^b.$$

Formule gaussiane in Matlab: sulla ricorrenza

Quindi

- a, b corrispondono rispettivamente all'α e β delle formule di Jacobi (e non agli estremi di integrazione!).
- I coefficienti di ricorrenza sono immagazzinati nella variabile ab.
- Nel caso di `r_jacobi` si usa la formula ricorsiva dei polinomi ortogonali (monici!) [7, p.216]

$$\begin{aligned} &\phi_{n+1}(x) = (x - \overline{\alpha}_n)\phi_n(x) - \overline{\beta}_n\phi_{n-1}(x), \ k = 1, 2, \ldots \\ &\phi_{-1}(t) = 0, \ \phi_0(t) = 1. \end{aligned} \tag{7}$$

Formule gaussiane in Matlab

A questo punto si chiama la funzione gauss.m (reperibile nuovamente presso lo stesso sito di W. Gautschi)

$$\boxed{\texttt{xw} = \texttt{gauss}(\texttt{N}, \texttt{ab})}$$

definita da

```
function xw=gauss(N,ab)
N0=size(ab,1); if N0<N, error('input array ab too short'), end
J=zeros(N);
for n=1:N, J(n,n)=ab(n,1); end
for n=2:N
  J(n,n-1)=sqrt(ab(n,2));
  J(n-1,n)=J(n,n-1);
end
[V,D]=eig(J);
[D,I]=sort(diag(D));
V=V(:,I);
xw=[D ab(1,2)*V(1,:)'.^2];
```

Formule gaussiane in Matlab

- Tale routine, che per un certo N, uguale al massimo al numero di righe della matrice di coefficienti di ricorrenza `ab`, fornisce **N nodi x e i corrispettivi pesi w** immagazzinati in una matrice `xw` che ha quale prima colonna x e quale seconda colonna w.

  Osserviamo che in tale codice i polinomi ortogonali sono monici e quindi compatibili con quelli *forniti* da `r_jacobi`.
- La descrizione di perchè tale software fornisca il risultato desiderato è complicata ma può essere trovata nella monografia di W. Gautschi sui polinomi ortogonali, per Acta Numerica.

Formule gaussiane in Matlab

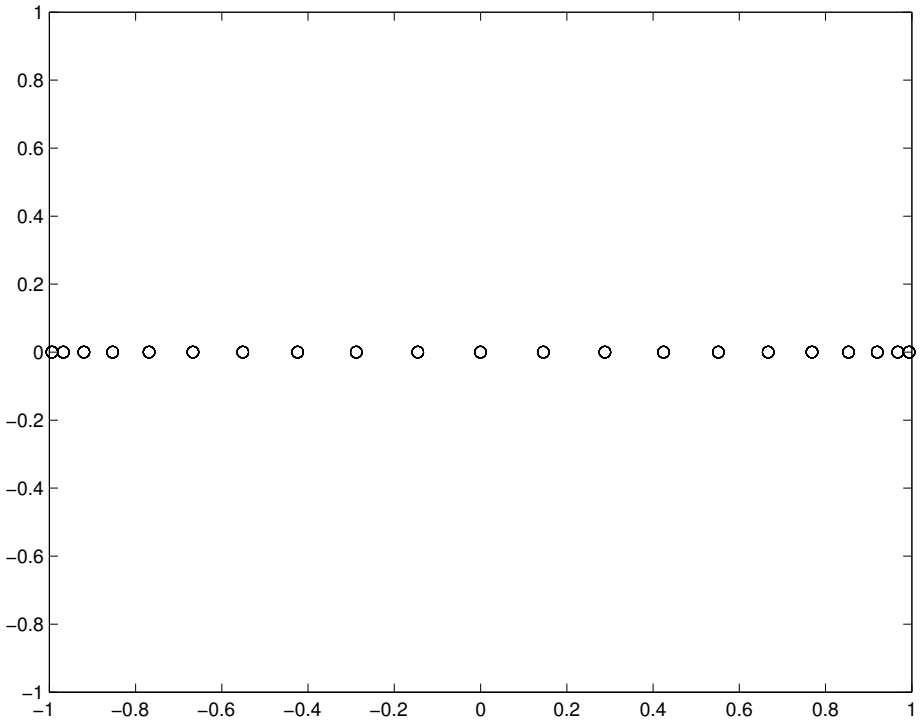

Figura : Grafico che illustra la distribuzione dei 20 nodi di Gauss-Legendre nell'intervallo $[1, 1]$

.

Formule gaussiane in Matlab

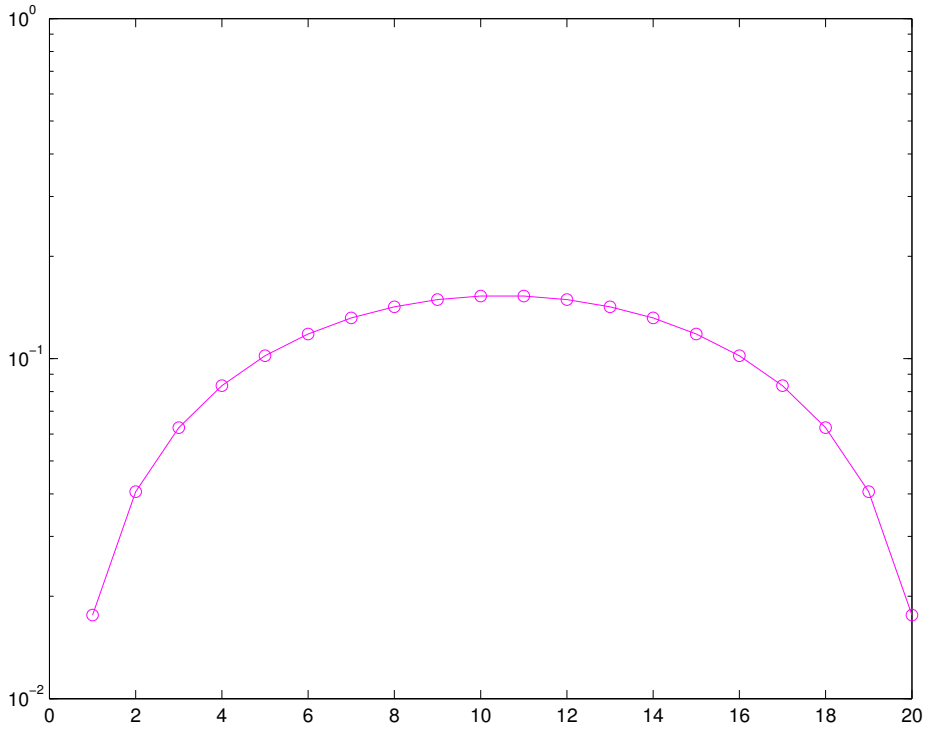

Figura : Grafico che illustra la distribuzione dei 20 pesi di Gauss-Legendre nell'intervallo $[1, 1]$

.

Una demo di Gauss-Jacobi

Salviamo nel file integrazione_gauss_jacobi.m

```
function [I,x,w]=integrazione_gauss_jacobi(N,alpha,beta,a,b,f)
% INPUT:
% N: NUMERO NODI.
% alpha,beta: w(x)=((1-x)^alpha)*((1+x)^beta)
% a,b: SE alpha=0, beta=0, SONO ESTREMI DI INTEGRAZIONE,
%    ALTRIMENTI a=-1, b=1.
% f: INTEGRANDA (RISPETTO w).
% OUTPUT:
% I: APPROSSIMAZIONE DI \int_a^b f(x) w(x) dx
% x,w: NODI/PESI UTILIZZATI.
ab=r_jacobi(N,alpha,beta); %TERM.RICORSIVI
xw=gauss(N,ab); %NODI/PESI IN MATRICE .
x=xw(:,1);w=xw(:,2);%NODI/PESI GAUSS JAC.[-1,1].
if (alpha == 0) & (beta == 0) & ((a~=-1)|(b~=1))
 x=((a+b)/2)+((b-a)/2)*x;%NODI GAUSS-LEG.[a,b].
 w=((b-a)/2)*w;%PESI GAUSS-LEG.[a,b].
end
fx=feval(f,x);%VALUTAZIONE FUNZIONE.
I=w'*fx;%VALORE INTEGRALE.
```

Una demo di Gauss-Jacobi

e nel file f.m delle funzioni su cui effettueremo dei test. Un esempio è

```
function fx=f(x)
fx=x.^20;
% ALCUNE FUNZIONI TRATTE DA
% "IS GAUSS QUADRATURE BETTER THAN CLENSHAW–CURTIS?"
% DI L.N. TREFETHEN.
% fx=exp(x);
% fx=exp(-x.^2);
% fx=1./(1+16*(x.^2));
% fx=exp(-x.^(-2));
% fx=abs(x); fx=fx.^3;
% fx=x.^(0.5);
% fx=exp(x).*(sqrt(1-x));
```

Una demo di Gauss-Jacobi

Le funzioni test [10]

1. $f(x) = x^{20}$;
2. $f(x) = \exp(x)$;
3. $f(x) = \exp(-x^2)$;
4. $f(x) = 1./(1 + 16 * x^2)$;
5. $f(x) = \exp(-x^{-2})$;
6. $f(x) = |x|^3$;
7. $f(x) = x^{0.5}$;
8. $f(x) = \exp(x) * (\sqrt{1 - x})$;

con $a = -1$, $b = 1$ possono essere valutate a scelta cencellando % riponendolo altrove.

- Scaricando dalla pagina web del corso
  - `integrazione_gauss_jacobi.m`, `f.m`
- Scaricando dalla pagina web di W. Gautschi
  - `r_jacobi.m`,
  - `gauss.m`

possiamo fare quindi alcuni esperimenti.

Facoltativo. Una demo di Gauss-Jacobi: scalatura

L'idea della scalatura consiste nel modificare la formula di quadratura relativa ad un intervallo $[a, b]$ di riferimento così da poterla utilizzare in un intervallo generico $[\alpha, \beta]$, cambiando esclusivamente nodi e pesi.

Si osservi che per ottenere i nodi e pesi di Gauss-Legendre in (a, b) da quelli di Gauss-Legendre in $(-1, 1)$ abbiamo effettuato uno **scalatura**.

- se $x_{i,[-1,1]}$ sono i nodi di Gauss-Legendre in $[-1, 1]$ allora i nodi di Gauss-Legendre $x_{i,[a,b]}$ in $[a, b]$ sono

$$x_{i,[a,b]} = ((a+b)/2) + ((b-a)/2) \cdot x_{i,[-1,-1]};$$

- se $w_{i,[-1,1]}$ sono i pesi di Gauss-Jacobi in $[-1, 1]$ allora i pesi di Gauss-Jacobi $w_{i,[a,b]}$ in $[a, b]$ sono

$$w_{i,[a,b]} = ((b-a)/2) \cdot w_{i,[-1,1]}.$$

Facoltativo. Formule gaussiane in Matlab: scalatura

Specialmente per le formule con peso $w(x) \equiv 1$ si ha spesso da effettuare l'integrale $\int_a^b f(x)dx$ con (a, b) diverso da $(-1, 1)$.

In tal caso si osserva che se $\gamma(t) = (a(1-t)/2) + (b(1+t)/2)$, per $\{t_k\}$ e $\{w_k\}$ nodi e pesi della formula di Gauss-Legendre,

$$\int_a^b f(x)dx = \int_{-1}^{1} \frac{b-a}{2} f(\gamma(t))dt \approx \sum_{k=1}^{n} \frac{b-a}{2} w_k f(\gamma(t_k))$$

e quindi

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} w_k^* f(x_k^*)$$

con $w_k^* = \frac{b-a}{2} w_k$, $x_k = \gamma(t_k) = (a(1-t_k)/2) + (b(1+t_k)/2)$.

Questo processo si chiama *scalatura* dei nodi e dei pesi.

Chebfun

Nota.

Qualora si intenda effettuare quadratura via `Chebfun`*, i nodi/pesi di quadratura di classici pesi sono disponibili, anche nell'ordine di molte migliaia.*

- *nodi/pesi di Chebyshev:* *chebpts*,
- *nodi/pesi di Legendre:* *legpts*,
- *nodi/pesi di Jacobi:* *jacpts*,
- *nodi/pesi di Lobatto:* *lobpts*,
- *nodi/pesi di Radau:* *radaupts*,
- *nodi/pesi di Hermite:* *hermpts*,
- *nodi/pesi di Laguerre:* *lagpts*.

Chebfun

In Chebfun i nodi e i pesi di una formula gaussiana vengono calcolati in una frazione di secondo.

Per convincersene, osservando che il benchmark dipende dal computer e che varia di volta in volta,

```
>> tic; [x,w]=legpts(100); toc
Elapsed time is 0.012519 seconds.
>> tic; [x,w]=legpts(10000); toc
Elapsed time is 0.313394 seconds.
>> tic; [x,w]=chebpts(100); toc
Elapsed time is 0.002481 seconds.
>> tic; [x,w]=hermpts(100); toc
Elapsed time is 0.011739 seconds.
```

Esempio 1

Esempio

Calcolare con 11 *nodi di Gauss-Legendre*

$$\int_{-1}^{1} x^{20}\,dx \approx 0.09523809523809523300$$

```
>> format long e
>> N=11; ajac=0; bjac=0; a=-1; b=1;
>> [I_jac,x_jac,w_jac]=integrazione_gauss_jacobi(N,ajac,bjac,a,b,@f);
>> I_jac
I_jac =
     9.523809523809562e-02
>> I_jac-0.09523809523809523300
ans =
     3.885780586188048e-16
>>
```

Nota.

Si osservi che con 11 *nodi la formula di Gauss-Legendre integra esattamente* $\int_{-1}^{1} x^{20}dx$, *da cui l'errore assoluto pari a* $4.163336342344337e-016$.

Esempio 2

Esempio

Calcolare opportunamente l'integrale

$$\int_{-1}^{1} \exp(x)\sqrt{1-x}\,dx = 1.77914365469190979259117902999 41. \tag{8}$$

Tale risultato è stato ottenuto usando il comando simbolico di Matlab 7.6.

```
>> syms x
>> int('(exp(x)) * ( (1-x)^(0.5) )',-1,1)
ans =
1.77914365469190979259117902999 41
```

Il comando, è stato modificato nelle più recenti versioni di Matlab e non offre il risultato desiderato. Con Chebfun possiamo comunque approssimare tale valore.

```
>> f=chebfun('(exp(x))*((1-x)^(0.5))','splitting','on','vectorize');
```

Esempio 2

Si capisce che

1. `syms x` rende la variabile `x` di tipo simbolico (e non numerico!);
2. il termine

```
int('(exp(x)) * ( (1-x)^(0.5) )',-1,1)
```

calcola simbolicamente l'integrale

$$\int_{-1}^{1} \exp(x)\sqrt{1-x}\, dx.$$

Esempio 2

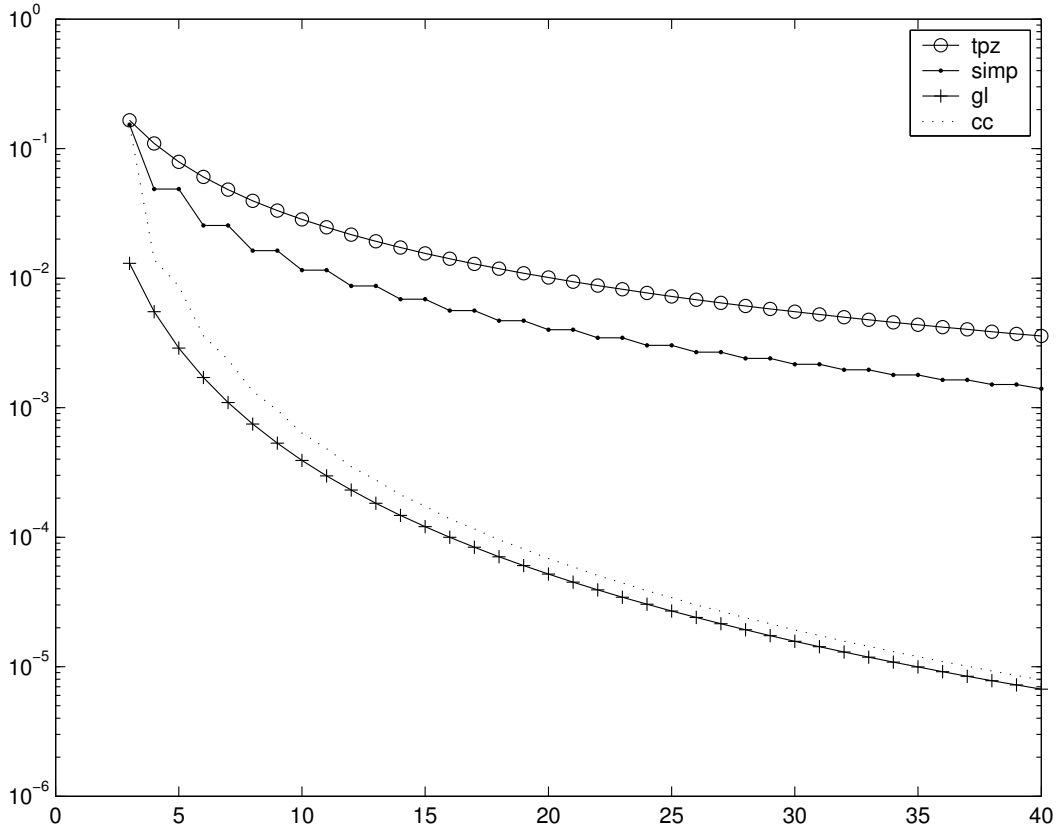

Figura : Grafico che illustra l'errore delle formule di quad. dei trap. composta, Cav.-Simpson comp., Gauss-Legendre e Clenshaw-Curtis relativamente a $\int_{-1}^{1} \exp(x)\sqrt{1-x}\,dx$.

Esempio 2

E' immediato osservare che $w(x) = \sqrt{1-x}$ è un peso di Gauss-Jacobi

$$w(t) = (1-t)^{\alpha}(1+t)^{\beta}$$

per $\alpha = 1/2$ e $\beta = 0$.

Di conseguenza, se $w(x) = \sqrt{1-x}$, $g(x) = \exp(x)\, w(x)$, $f(x) = \exp(x)$ abbiamo

$$\int_{-1}^{1} \exp(x)\sqrt{1-x}dx = \int_{-1}^{1} g(x)dx = \int_{-1}^{1} f(x)w(x)dx$$

Quindi, se intendo usare il codice sulle formule gaussiane con peso di Jacobi $w(x) = \sqrt{1-x}$ devo fornire la funzione $f(x) = \exp(x)$ (e non $g(x) = \exp(x)\, w(x)$

Esempio 2

```
>> a=-1; b=1;
>> [I_jac,x_jac,w_jac]=integrazione_gauss_jacobi(10,1/2,0,a,b,@exp);
>> I_jac
I_jac =
     1.779143654691909e+00
>> I_jac-1.779143654691909792591179029994 1
ans =
    -4.440892098500626e-16
>> length(x_jac)
ans =
    10
>> [I_jac,x_jac,w_jac]=...
 integrazione_gauss_jacobi(10,0,0,-1,1,inline('exp(x).*sqrt(1-x)'));
>> I_jac
I_jac =
     1.779841121014780e+00
>> I_jac-1.779143654691909792591179029994 1
ans =
     6.974663228704880e-04
>> length(x_jac)
ans =
    10
>>
```

Esempio 2

Entrambe le formule hanno lo stesso numero di nodi (e pesi), come si vede dalla riga di comando

```
>> length(x_jac)
ans =
    10
```

ma offrono risultati diversi, con un errore assoluto di

- circa $4.44 \cdot 10^{-16}$ per Gauss-Jacobi con $a = 1/2$, $b = 0$,
- circa $6.97 \cdot 10^{-4}$ per Gauss-Legendre (cioè Gauss-Jacobi con $a = 0$, $b = 0$).

Esercizio

Usando una opportuna formula gaussiana, con $n = 10, 20, 30$ punti, calcolare

- $\int_0^{\pi} \exp(x) \cdot \cos(4x) dx = \frac{\exp(\pi)-1}{17}$;
- $\int_0^1 x^{5/2} dx = \frac{2}{7}$;
- $\int_{-\pi}^{\pi} \exp(\cos(x)) dx = 7.95492652101284$;
- $\int_0^{\pi/4} \exp(\cos(x)) dx = 1.93973485062365$;
- $\int_0^1 x^{1/2} dx = \frac{2}{3}$.
- Descrivere come decresce l'errore in scala semilogaritmica.

Nota.

- *Per gli esempi con le radici quadrate, trasformare l'integrale nell'intervallo* $[-1, 1]$ *e vedere se si può utilizzare una opportuna funzione peso.*
- *Non serve vedere quale sia l'andamento del rapporto* $E_n(f)/E_{2n}(f)$*, dove* $E_n = |I(f) - I_n(f)|$ *con* $I(f)$ *integrale esatto e* $I_n(f)$ *valore fornito dalla formula di quadratura gaussiana a n punti.*

Esercizio (facoltativo)

Si calcolino per $N = 10, 20$ con la formula composta dei trapezi, di Cav.-Simpson e un'appropriata formula gaussiana

$$\int_{-1}^{1} x^{20}\,dx = 2/21 \approx 0.095238095238096801 \tag{9}$$

$$\int_{-1}^{1} e^{x}\,dx = e - e^{-1} \approx 2.3504023872876032 \tag{10}$$

$$\int_{-1}^{1} e^{-x^2}\,dx = \mathrm{erf}(1)\cdot\sqrt{\pi} \approx 1.4936482656248538 \tag{11}$$

$$\int_{-1}^{1} 1/(1+16x^2)\,dx = 1/2 \cdot \mathrm{atan}(4) \approx 0.66290883183401628 \tag{12}$$

$$\int_{-1}^{1} e^{-1/x^2}\,dx \approx 0.17814771178156086 \tag{13}$$

$$\int_{-1}^{1} |x|^3\,dx = 1/2 \tag{14}$$

$$\int_{0}^{1} \sqrt{x}\,dx = 2/3 \tag{15}$$

$$\int_{-1}^{1} e^{x}\sqrt{1-x}\,dx \approx 1.7791436546919095 \tag{16}$$

Esercizio (facoltativo)

1. Quali delle due formule ha errori relativi inferiori?
2. Quali funzioni risultano più difficili da integrare numericamente?
3. La scelta $N = 10, 20$ nei singoli codici a quanti nodi corrisponde?
4. Suggerimento: ricordarsi che il peso $w(x) \equiv 1$ corrisponde ad un'opportuna scelta del peso di Jacobi.

Esercizio (facoltativo)

1. Quali delle due formule ha errori relativi inferiori?
2. Quali funzioni risultano più difficili da integrare numericamente?
3. La scelta $N = 10, 20$ nei singoli codici a quanti nodi corrisponde?
4. Suggerimento: ricordarsi che il peso $w(x) \equiv 1$ corrisponde ad un'opportuna scelta del peso di Jacobi.

Esercizio (facoltativo)

Calcolare con una opportuna formula gaussiana basata su 11 nodi:

1. $$\int_{-1}^{1} x^{20}\sqrt{1-x^2}dx$$

2. $$\int_{-1}^{1} \exp(x^2)\cdot(1-x)^{0.5}\cdot(1+x)^{1.5}dx$$

Il primo integrale, può essere calcolato *esattamente* con Mathematica, Maple (da Matlab, aiutarsi con Google) o `quadgk` di Matlab, una toolbox di Matlab specifica per l'integrazione numerica, con un errore assoluto di 10^{-15}

```
>> Q = quadgk(@(x)(x.^20).*(1-x.^2).^(0.5) ,-1,1,'AbsTol',10^(-15))
Q =
   0.025160880188796
>>
```

Esercizio (facoltativo)

In alternativa usando online il Wolfram Integrator, e ricordando che devo integrare in $(-1, 1)$, dal teorema fondamentale del calcolo si vede che si tratta di valutare

$$\frac{14549535}{1816657920} \cdot (\arcsin(1) - \arcsin(-1)).$$

Utilizzando la shell di Matlab

```
>> c=14549535/1816657920;
>> s=asin(1)-asin(-1);
>> I=s*c;
>> format long;
>> I
I =
   0.025160880188796
>>
```

Esercizio (facoltativo)

Il secondo integrale non è integrabile col Wolfram integrator e nemmeno col calcolo simbolico di Matlab.

```
>> syms x
>> int((exp(-x.^2)).*((1-x).^(0.5)).*((1+x).^(1.5)),-1,1)
Warning: Explicit integral could not be found.
> In sym.int at 58
ans =
int(exp(-x^2)*(1-x)^(1/2)*(1+x)^(3/2),x = -1 .. 1)
```

Esercizio (facoltativo)

Utilizzando la shell di Matlab e quadgk con un errore assoluto di 10^{-15}

```
>> format long;
>> I=quadgk(@(x)(exp(x.^2)).*((1-x).^(0.5)).*((1+x).^(1.5)),-1,1,...
        'AbsTol',10^(-15))
I =
   2.086318843895086
>>
```

Bibliografia


K. Atkinson, *Introduction to Numerical Analysis*, Wiley, 1989.


K. Atkinson e W. Han, *Theoretical Numerical Analysis*, Springer, 2001.


V. Comincioli, *Analisi Numerica, metodi modelli applicazioni*, Mc Graw-Hill, 1990.


S.D. Conte e C. de Boor, *Elementary Numerical Analysis, 3rd Edition*, Mc Graw-Hill, 1980.


P.J. Davis, *Interpolation and Approximation*, Dover, 1975.


W. Gautschi, personal homepage, http://www.cs.purdue.edu/people/wxg.


W. Gautschi, , *Orthogonal polynomials (in Matlab)*, JCAM 178 (2005) 215–234


A. Quarteroni e F. Saleri, *Introduzione al calcolo scientifico*, Springer Verlag, 2006.


A. Suli e D. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.


L.N. Trefethen, *Is Gauss quadrature better than Clenshaw-Curtis?*, SIAM Reviews, (2007).


J. Waldvogel, *Fast Construction of the Fejer and Clenshaw-Curtis Quadrature Rules*,BIT 46 (2006),no. 1,195–202.