# Quadratura numerica

*30 giugno 2007*

## 1 Introduzione

Un classico problema dell'analisi numerica è quello di calcolare l'*integrale definito* di una funzione $f$ nell'intervallo $[a,b]$ cioè

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

dove supporremo per il momento che

1. $[a,b] \subset [\alpha,\beta] \subseteq \mathbb{R}$ sia un intervallo chiuso e limitato;
2. la funzione $f : [\alpha,\beta] \to \mathbb{R}$ sia sufficientemente regolare.

Successivamente indeboliremo queste condizioni.
La nostra intenzione è di di approssimare $I(f)$ come

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

I termini $w_i$ e $x_i \in [\alpha,\beta]$ sono detti rispettivamente pesi e nodi. Notiamo che consideriamo $[a,b] \subset [\alpha,\beta] \subseteq \mathbb{R}$: questo perchè in alcune formule si possono considerare nodi esterni al dominio di integrazione.

### 1.1 Formule di Newton-Cotes

Le prime *regole* di questo tipo sono dette di *Newton-Cotes* chiuse (cf. [**?**, p.336]) e si ottengono integrando l'interpolante di $f$ in nodi equispaziati

$$x_i = a + \frac{(i-1)\,(b-a)}{N-1},\ i = 1,\ldots,N.$$

Alcune classiche regole sono:

1. *regola del trapezio*

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

avente *grado di precisione* 1, cioè esatta per polinomi di grado inferiore o uguale a 1; si può dimostrare (con un po' di fatica) dal teorema del resto per l'interpolazione polinomiale (cf. [**?**, p.132]) che l'errore della regola del trapezio è

$$E_1(f) := I(f) - S_1(f) = \frac{-h^3}{12} f^{(2)}(\xi)$$

per qualche $\xi \in (a,b)$;

2. *regola di Cavalieri-Simpson*

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

avente *grado di precisione* 3, cioè esatta per polinomi di grado inferiore o uguale a 3; si può dimostrare (non facile!) che l'errore della regola di Cavalieri-Simpson è

$$E_3(f) := I(f) - S_3(f) = \frac{-h^5}{90} f^{(4)}(\xi),\ h = \frac{b-a}{2}$$

per qualche $\xi \in (a,b)$;

Per ulteriori dettagli si confronti [**?**, p.252-258], [**?**, p.333-336]. Qualora le funzioni da integrare non siano sufficientemente derivabili, una stima dell'errore viene fornita dalle formule dell'errore via nucleo di Peano ([**?**, p.259]). Ricordiamo che per $N > 8$ le formule di Newton-Cotes chiuse hanno pesi di segno diverso e sono instabili dal punto di vista della propagazione degli errori (cf. [**?**, p.196]).

## 1.2 Formule di Newton-Cotes 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

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

dove $S$ è una delle regole di quadratura finora esposte. Le formule descritte in (**??**) sono dette *composte*. Due casi particolari sono

1. *formula composta dei trapezi*

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

il cui errore è

$$E_1^{(c)}(f) := I(f) - S_1^{(c)}(f) = \frac{-(b-a)}{12} h^2 f^{(2)}(\xi),\ h = \frac{(b-a)}{N}$$

per qualche $\xi \in (a,b)$;

2. *formula composta di Cavalieri-Simpson* fissati il numero $N$ di subintervalli e i punti $x_k = a + kH/2$ dove

$$H = \frac{b-a}{N}$$

sia

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

il cui errore è

$$E_3^{(c)}(f) := I(f) - S_3^{(c)}(f) = \frac{-(b-a)}{180}(\frac{H}{2})^4 f^{(4)}(\xi)$$

per qualche $\xi \in (a,b)$.

## 1.3 Implementazione Matlab di alcune formule composte

Mostreremo di seguito un'implementazione in Matlab/Octave della formula composta dei trapezi e di Cavalieri-Simpson.

```
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;
```

La funzione *trapezi_composta* appena esposta calcola i nodi e i pesi della omonima formula composta. Si potrebbe usare il comando Matlab *trapz* nella sua implementazione

```
>> help trapz

 TRAPZ  Trapezoidal numerical integration.
    Z = TRAPZ(Y) computes an approximation of the integral of Y via
    the trapezoidal method (with unit spacing).  To compute the integral
```

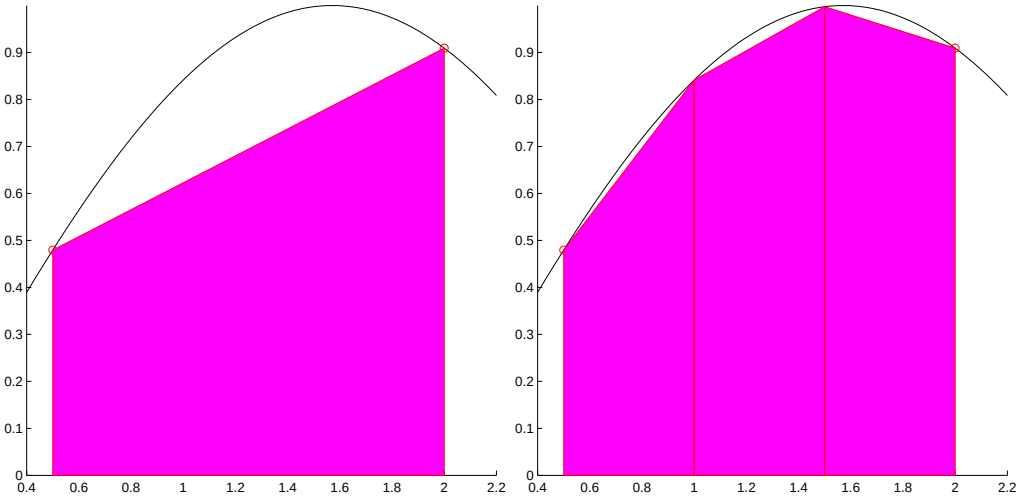


Figure 1: Formula dei trapezi e dei trapezi composta per il calcolo di $\int_{0.5}^{2} \sin(x)\,dx$ (area in magenta).

```
    for spacing different from one, multiply Z by the spacing increment.

    For vectors, TRAPZ(Y) is the integral of Y.
    ... ... ...
```

e sostituire la parte relativa al calciolo dei pesi con

```
w=h*trapz(ones(N+1,1));
```

Evitiamo il suo utilizzo perchè non presente in alcune vecchie versioni di Octave (ma non nella più recente 2.1.73).

Per quanto riguarda la formula di Cavalieri-Simpson composta

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

% FORMULA DI SIMPSON COMPOSTA.

% INPUT:
% N: NUMERO SUBINTERVALLI.
% a, b: ESTREMI DI INTEGRAZIONE.

% OUTPUT:
% x: NODI 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;
```

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 quadratura composta.

## 1.4 Formule gaussiane

Nelle formule interpolatorie di Newton-Cotes (come ad esempio la regola del Trapezio o di Cavalieri-Simpson) i nodi $x_1, \dots, x_N$ sono equispaziati e il grado di precisione $\delta$ è al massimo uguale al numero di nodi $N$. Ci si domanda se sia possibile scegliere nodi $x_1, \dots, x_N$ e pesi $w_1, \dots, w_N$ per cui le relative formule di quadratura abbiano grado di precisione $\delta > N$ (come ad esempio per la regola di Cavalieri-Simpson).

D'altro canto, se $w : [a,b] \to \mathbb{R}$ (non necessariamente limitato) è una funzione peso, cioè (cf. [**?**, 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 $(a,b)$.

Tra gli esempi più noti

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 $(-\infty, \infty)$;

Si supponga ora di dover calcolare per qualche funzione $f : (a,b) \to \mathbb{R}$

$$I(f,w) := \int_a^b f(x)w(x)\,dx.$$

Il problema è evidentemente più generale di quello in (**??**), visto che l'integranda $fw$ non é necessariamente continua in $[a,b]$ (si consideri ad esempio il peso di Chebyshev) oppure può succedere che l'intervallo sia illimitato come nel caso del peso di Laguerre o Hermite.

Esistono nuovamente $x_1,\ldots,x_N$ e pesi $w_1,\ldots,w_N$ (detti di *Gauss-nome funzione peso*) per cui le relative formule di quadratura abbiano grado di precisione $\delta > N$, cioè calcolino esattamente

$$\int_a^b p_m(x)w(x)\,dx$$

per $m > N$?

La risposta è affermativa, come si può vedere in [**?**, p.272]. Per ogni $N \leq 1$ esistono e sono unici dei nodi $x_1,\ldots,x_N$ e pesi $w_1,\ldots,w_N$ per cui il grado di precisione sia $2N-1$. 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
```

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

```
ab=r_jacobi(N,a,b)
```

Come si vede dai commenti al file

```
% R_JACOBI Recurrence coefficients for monic Jacobi polynomials.
%
%    ab=R_JACOBI(n,a,b) generates the first n recurrence
%    coefficients for monic Jacobi polynomials with parameters
%    a and b. These are orthogonal on [-1,1] relative to the
%    weight function w(t)=(1-t)^a(1+t)^b. The n alpha-coefficients
%    are stored in the first column, the n beta-coefficients in
%    the second column, of the nx2 array ab. The call ab=
%    R_JACOBI(n,a) is the same as ab=R_JACOBI(n,a,a) and
%    ab=R_JACOBI(n) the same as ab=R_JACOBI(n,0,0).
%
%    Supplied by Dirk Laurie, 6-22-1998; edited by Walter
%    Gautschi, 4-4-2002.
```

$a$, $b$ corrispondono rispettivamente all'$\alpha$ e $\beta$ delle formule di Jacobi (e non agli estremi di integrazione!). I coefficienti di ricorrenza sono immagazzinati nella variabile $ab$. A questo punto si chiama la funzione *gauss.m* (reperibile nuovamente presso lo stesso sito di W. Gautschi)

```
% GAUSS Gauss quadrature rule.
%
%    Given a weight function w encoded by the nx2 array ab of the
%    first n recurrence coefficients for the associated orthogonal
%    polynomials, the first column of ab containing the n alpha-
%    coefficients and the second column the n beta-coefficients,
%    the call xw=GAUSS(n,ab) generates the nodes and weights xw of
%    the n-point Gauss quadrature rule for the weight function w.
%    The nodes, in increasing order, are stored in the first
%    column, the n corresponding weights in the second column, of
%    the nx2 array xw.
%
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];
```

che per un certo $N$, uguale al massimo al numero di righe della matrice di coefficienti di ricorrenza $ab$, fornisce nodi $x$ e pesi $w$ immagazzinati in una matrice che ha quale prima colonna $x$ e quale seconda colonna $w$. 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.

Conseguentemente per trovare i nodi e pesi relativi alla formula di Gauss-Legendre in $(a,b)$, che è una formula di tipo Gauss-Jacobi per $\alpha = 0$ e $\beta = 0$, nonchè calcolare con essi degli integrali di una funzione $f$, si procede come segue.

## 1.5 Una demo di Gauss-Legendre

Salviamo nel file *demo_gauss_legendre.m*

```
a=-1; b=1;
N=10;
ftype=1;

ajac=0; bjac=0;
```

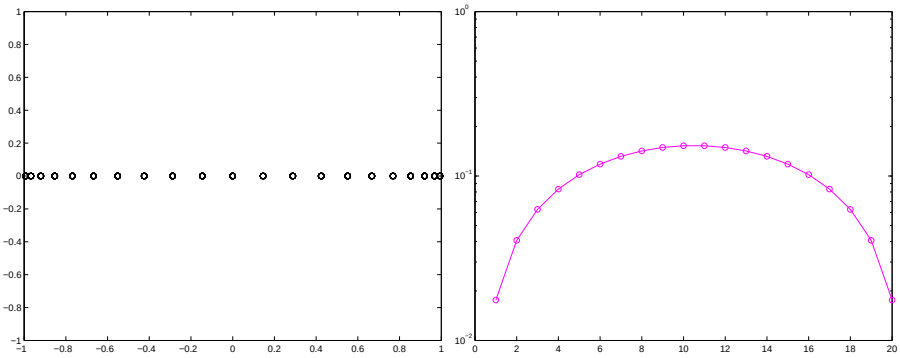

Figure 2: Grafico che illustra la distribuzione dei 20 nodi e i 20 pesi di Gauss-Legendre nell'intervallo $[1,1]$

.

```
ab_jac=r_jacobi(N,ajac,bjac);          % TERM. RICORSIVI.
xw_jac=gauss(N,ab_jac);                % NODI E PESI IN MATRICE.
x_jac=xw_jac(:,1);                     % NODI GAUSS-LEGENDRE [-1,1].
x_jac_ab=((a+b)/2)+((b-a)/2)*x_jac;    % NODI GAUSS-LEGENDRE [a,b].
w_jac=xw_jac(:,2);                     % PESI GAUSS-LEGENDRE [-1,1].
w_jac_ab=((b-a)/2)*w_jac;              % PESI GAUSS-LEGENDRE [a,b].
fx_jac_ab=f(x_jac_ab,ftype);           % VALUTAZIONE FUNZIONE.
I_jac=w_jac_ab'*fx_jac_ab;             % VALORE INTEGRALE.


% RISULTATI.
fprintf('\n \t [RISULTATO][GAUSS-LEGENDRE  ]: %18.18f',I_jac);


% RISULTATI.
res=exact_results(ftype);
fprintf('\n \t [GAUSS-LEGENDRE  ][ERR.ASS.]: %2.2e',abs(I_jac-res));

% SOMMA ABS. PESI.
sw_jac=sum(abs(w_jac));
fprintf('\n \t [SOMMA ABS. PESI][GAUSS - LEGENDRE]: %2.2e',sw_jac);

% PLOT PESI.
semilogy(1:length(w_jac),w_jac,'mo-'); hold on;
```

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

```
function fx=f(x,ftype)

% ALCUNE FUNZIONI CHE FANNO PARTE DEL SET STUDIATO NELL'ARTICOLO:
% "IS GAUSS QUADRATURE BETTER THAN CLENSHAW-CURTIS?"
```

```
% DI L.N. TREFETHEN.

switch ftype
case 1
    fx=x.^20;
case 2
    fx=exp(x);
case 3
    fx=exp(-x.^2);
case 4
    fx=1./(1+16*(x.^2));
case 5
    fx=exp(-x.^(-2));
case 6
    fx=abs(x); fx=fx.^3;
end
```

le cui prime 6 funzioni test sono ricavate dall'articolo [**?**].

Scaricando dalla pagina web del corso i files *demo_gauss_legendre.m*, *f.m* e dalla pagina di W. Gautschi *r_jacobi.m* e *gauss.m*, nonchè il file *exact_results.m* contenente i risultati esatti di tali integrali con almeno 10 cifre decimali *dopo la virgola*, possiamo fare alcuni esperimenti.

Alcune note prima di cominciare.

1. Si osservi che per ottenere i nodi e pesi di Gauss-Legendre in $(a,b)$ da quelli di Gauss-Jacobi in $(-1,1)$ abbiamo effettuato uno *scaling*: se $x_{i,[-1,1]}$ sono i nodi di Gauss-Jacobi in $[-1,1]$ allora i nodi di Gauss-Jacobi $x_{i,[a,b]}$ in $[a,b]$ sono

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

2. 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

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

   L'idea è la seguente. Dovendo le formule essere esatte per le costanti come la funzione $f \equiv 1$, nel caso della funzione peso di Legendre $w\ equiv1$

$$\sum_i w_{i,[a,b]} = \int_a^b w(x)\,dx = b-a$$

   mentre nel caso di $a=-1$, $b=1$ derivante dalla funzione peso di Jacobi abbiamo

$$\sum_i w_{i,[-1,1]} = \int_{-1}^1 w(x)\,dx = 2;$$

   si ha quindi l'intuizione che i pesi in $(a,b)$ siano quelli in $(-1,1)$ moltiplicati per $\frac{b-a}{2}$.

3. Nonostante l'introduzione riguardante nodi e pesi in $[a,b]$ non necessariamente uguale a $[-1,1]$, alla fine eseguiremo test esclusivamente in quest'ultimo intervallo. Qualora necessario, basta aggiungere nuove funzioni matematiche al file *f.m*, modificare adeguatamente $a$, $b$, fornire il relativo risultato esatto in *exact_results.m* e testare la formula gaussiana. Per avere il risultato con alta precisione si usi la funzione *quadl.m* o *quad8.m* di Matlab oppure *demo_gauss_legendre.m* con $N = 500$;

4. l'intervallo tipico di Gauss-Legendre è $[a,b]$ (chiuso), mentre per Gauss-Jacobi l'intervallo è tipicamente $(a,b)$ (aperto),poichè in generale gli esponenti della funzione peso di Jacobi possono essere negativi; per Gauss-Legendre il problema non sussiste, visto che la funzione peso è continua, e i punti $a$, $b$ sono un insieme *trascurabile*.

Se ora testiamo il primo esempio, cioè il calcolo di

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

otteniamo

```
>> demo_gauss_legendre

          [RISULTATO][GAUSS-LEGENDRE  ]: 0.095235169647764462
          [GAUSS-LEGENDRE  ][ERR.ASS.]: 2.93e-006
          [SOMMA ABS. PESI][GAUSS - LEGENDRE]: 2.00e+000
>>
```

## 1.6 Una implementazione di Clenshaw-Curtis

Per quanto riguarda la celebrata formula di Clenshaw-Curtis in $[a,b]$, usiamo una implementazione di Waldvogel

```
function [x,w]=clenshaw_curtis(n,a,b)

N=[1:2:n-1]'; l=length(N); m=n-l; K=[0:m-1]';

g0=-ones(n,1); g0(1+l)=g0(1+l)+n; g0(1+m)=g0(1+m)+n;
g=g0/(n^2-1+mod(n,2));
end_N=length(N);
v0=[2./N./(N-2); 1/N(end_N); zeros(m,1)];
end_v0=length(v0);
v2=-v0(1:end_v0-1)-v0(end_v0:-1:2);
wcc=ifft(v2+g); weights=[wcc;wcc(1,1)];
k=0:n; nodes=(cos(k*pi/n))';

x=(a+b)/2+((b-a)/2)*nodes;
w=weights*((b-a)/2);
```

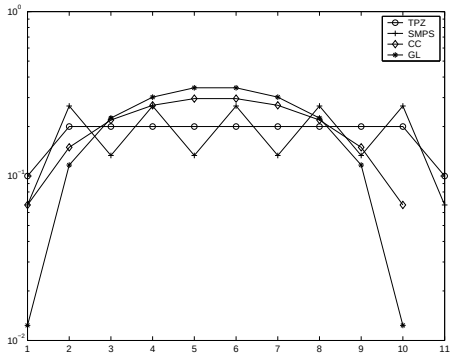


Figure 3: Grafico che illustra la distribuzione dei nodi della formula dei trapezi composta, Simpson composta, Clenshaw-Curtis e Gauss-Legendre nell'intervallo $[1,1]$

.

## 1.7 Una suite di confronto tra le varie formule

Una suite più completa che effettua un confronto tra le formule sopracitate (quasi a parità di numero di valutazioni della funzione) nonchè il *metodo di Romberg* e quello di *Clenshaw-Curtis* (cf. [**?**]) è presente in *demo_quadratura.m*, che necessita di

```
demo_block.m
romberg.m
exact_results.m
trapezi_composta.m
f.m
simpson_composta.m
r_jacobi.m
gauss.m
clenshaw_curtis.m
```

Tale programma *principale* di Matlab inoltre fornisce il tempo di CPU impiegato da ogni formula, il numero di valutazioni, la somma dei pesi e plotta i pesi corrispettivi (che sono positivi!).

Il programma *demo_quadratura.m* consiste in

```
N=10;
a=-1; b=1;
ftype=1;

[x_tpz,w_tpz,I_tpz,x_simp,w_simp,I_simp,x_jac_ab,w_jac_ab,I_jac,x_cc,w_cc,...
I_cc,timing]=demo_block(N,a,b,ftype);

I_romb=romberg(a,b,N,ftype);

% RISULTATI.
fprintf('\n \t [RISULTATO][TRAPEZI COMPOSTA]: %18.18f',I_tpz);
```

```
fprintf('\n \t [RISULTATO][SIMPSON COMPOSTA]: %18.18f',I_simp);
fprintf('\n \t [RISULTATO][GAUSS-LEGENDRE  ]: %18.18f',I_jac);
fprintf('\n \t [RISULTATO][CLENSHAW-CURTIS ]: %18.18f',I_cc);
fprintf('\n \t [RISULTATO][ROMBERG         ]: %18.18f',I_romb);

% RISULTATI.
res=exact_results(ftype);
fprintf('\n \n \t [TRAPEZI COMPOSTA][ERR.ASS.]: %2.2e',abs(I_tpz-res));
fprintf(' [ERR.REL.]: %2.2e',abs(I_tpz-res)/abs(res) );
fprintf('\n \t [SIMPSON COMPOSTA][ERR.ASS.]: %2.2e',abs(I_simp-res));
fprintf(' [ERR.REL.]: %2.2e',abs(I_simp-res)/abs(res) );
fprintf('\n \t [GAUSS-LEGENDRE  ][ERR.ASS.]: %2.2e',abs(I_jac-res));
fprintf(' [ERR.REL.]: %2.2e',abs(I_jac-res)/abs(res));
fprintf('\n \t [CLENSHAW-CURTIS ][ERR.ASS.]: %2.2e',abs(I_cc-res));
fprintf(' [ERR.REL.]: %2.2e',abs(I_cc-res)/abs(res));

% CPUTIMES.
fprintf('\n \n \t [CPUTIME] [TRAPEZI COMPOSTA]:
%2.2e',timing(2)-timing(1));
fprintf('\n \t [CPUTIME] [SIMPSON COMPOSTA]: %2.2e',timing(4)-timing(3));
fprintf('\n \t [CPUTIME] [GAUSS - LEGENDRE]: %2.2e',timing(6)-timing(5));
fprintf('\n \t [CPUTIME] [CLENSHAW-CURTIS ]: %2.2e',timing(8)-timing(7));

% VALUTAZIONI FATTE.
fprintf('\n \n \t [VALUTAZIONI] [TRAPEZI COMPOSTA]: %4.0f',length(x_tpz));
fprintf('\n \t [VALUTAZIONI] [SIMPSON COMPOSTA]: %4.0f',length(x_simp));
fprintf('\n \t [VALUTAZIONI] [GAUSS - LEGENDRE]: %4.0f',length(x_jac_ab));
fprintf('\n \t [VALUTAZIONI] [CLENSHAW-CURTIS ]: %4.0f',length(x_cc));

% SOMMA ABS. PESI.
sw_tpz=sum(abs(w_tpz)); sw_simp=sum(abs(w_simp));
sw_jac=sum(abs(w_jac_ab)); sw_cc=sum(abs(w_cc));
fprintf('\n \n \t [SOMMA ABS. PESI][TRAPEZI COMPOSTA]: %2.2e',sw_tpz);
fprintf('\n \t [SOMMA ABS. PESI][SIMPSON COMPOSTA]: %2.2e',sw_simp);
fprintf('\n \t [SOMMA ABS. PESI][GAUSS - LEGENDRE]: %2.2e',sw_jac);
fprintf('\n \t [SOMMA ABS. PESI][CLENSHAW-CURTIS ]: %2.2e',sw_cc);

% PLOT PESI.
semilogy(1:length(w_tpz),w_tpz,'ro-'); hold on;
semilogy(1:length(w_simp),w_simp,'ko-'); hold on;
semilogy(1:length(w_jac),w_jac,'mo-'); hold on;
semilogy(1:length(w_cc),real(w_cc),'go-'); hold on;
```

Una volta lanciato fornisce

```
[RISULTATO][TRAPEZI COMPOSTA]: 0.204626315050238370
[RISULTATO][SIMPSON COMPOSTA]: 0.139492003644474840
[RISULTATO][GAUSS-LEGENDRE  ]: 0.095235169647764462
[RISULTATO][CLENSHAW-CURTIS ]: 0.094066801525297616
[RISULTATO][ROMBERG         ]: 0.095238095238095080
```

```
[TRAPEZI COMPOSTA][ERR.ASS.]: 1.09e-001 [ERR.REL.]: 1.15e+000
[SIMPSON COMPOSTA][ERR.ASS.]: 4.43e-002 [ERR.REL.]: 4.65e-001
[GAUSS-LEGENDRE  ][ERR.ASS.]: 2.93e-006 [ERR.REL.]: 3.07e-005
[CLENSHAW-CURTIS ][ERR.ASS.]: 1.17e-003 [ERR.REL.]: 1.23e-002

[CPUTIME] [TRAPEZI COMPOSTA]: 1.60e-002
[CPUTIME] [SIMPSON COMPOSTA]: 0.00e+000
[CPUTIME] [GAUSS - LEGENDRE]: 0.00e+000
[CPUTIME] [CLENSHAW-CURTIS ]: 0.00e+000

[VALUTAZIONI] [TRAPEZI COMPOSTA]:   11
[VALUTAZIONI] [SIMPSON COMPOSTA]:   11
[VALUTAZIONI] [GAUSS - LEGENDRE]:   10
[VALUTAZIONI] [CLENSHAW-CURTIS ]:   10

[SOMMA ABS. PESI][TRAPEZI COMPOSTA]: 2.00e+000
[SOMMA ABS. PESI][SIMPSON COMPOSTA]: 2.00e+000
[SOMMA ABS. PESI][GAUSS - LEGENDRE]: 2.00e+000
[SOMMA ABS. PESI][CLENSHAW-CURTIS ]: 2.00e+000
```

## 1.8 Altri confronti

Il programma principale *demo_quadratura_2.m*

```
a=-1; b=1;
ftype=1;

N_vett=2:40;
LN=length(N_vett);
res=exact_results(ftype);

for index=1:LN;
    N=N_vett(index);
    [x_tpz,w_tpz,I_tpz,x_simp,w_simp,I_simp,x_jac,w_jac,I_jac,...
                  x_cc,w_cc,I_cc,timing]=demo_block(N,a,b,ftype);
    relerr_tpz(index)=abs(I_tpz-res)/abs(res);
    relerr_simp(index)=abs(I_simp-res)/abs(res);
    relerr_jac(index)=abs(I_jac-res)/abs(res);
    relerr_cc(index)=abs(I_cc-res)/abs(res);
end

semilogy(N_vett,relerr_tpz,'r-'); hold on;
semilogy(N_vett,relerr_simp,'k-'); hold on;
semilogy(N_vett,relerr_jac,'m-'); hold on;
semilogy(N_vett,relerr_cc,'g-');
```

permette di fare confronti tra gli errori relativi delle formule di quadrature sopra

indicate relativamente ad ognuna funzione di test. Proviamo quale esempio

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

In virtù del grado di precisione dei metodi, ci si aspetta che per $N$ sufficientemente alto, le formule di Gauss-Legendre (e Clenshaw-Curtis) approssimino tale integrale a meno della precisione di macchina.

Nelle figure successive, illustriamo tutti i test possibili, per le funzioni presenti nel file *f.m*.

## 1.9 Un'integrale con funzione peso

Consideriamo ora l'integrale

$$\int_{-1}^{1} \exp(x)\sqrt{1-x}\,dx = 1.77914365469190979259117902999 41. \tag{3}$$

Tale risultato è stato ottenuto usando il comando simbolico di Matlab 6.5 (non funziona in Octave, vedere in alternativa il programma Maxima!!)

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

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

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

Si ottiene da *demo_quadratura_2.m* il grafico in figura **??**, che mostra come pure la formula di Gauss-Legendre non abbia una grande performance.

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

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

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

Per testare questa ultima formula gaussiana, osserviamo che l'integrale richiesto può essere calcolato modificando *demo_gauss_legendre.m*. Infatti se $w(x) = \sqrt{1-x}$, allora $g(x) = \exp(x)\,w(x)$ il che corrisponde a usare Gauss-Jacobi con $f(x) = \exp(x)$ che è proprio la seconda funzione in *f.m*. Quindi modificheremo *demo_gauss_legendre.m* imponendo i settings

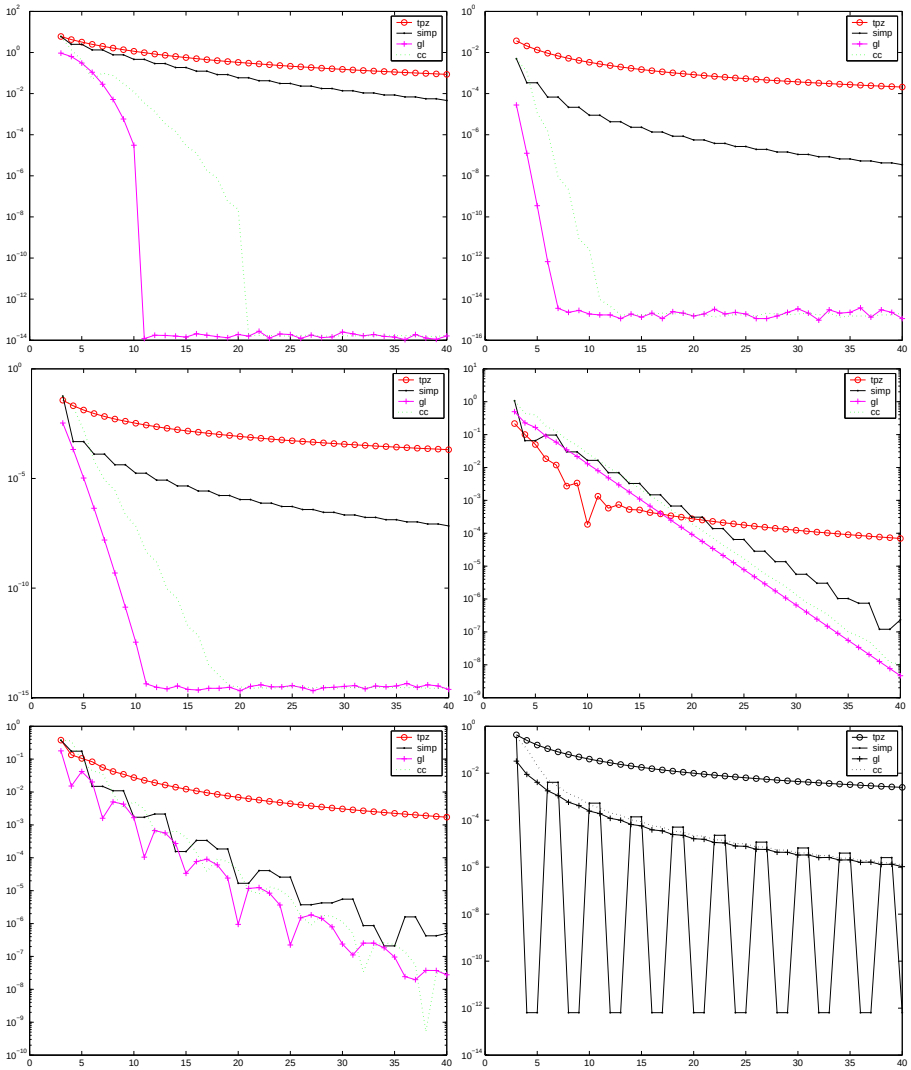


Figure 4: Grafico che illustra l'errore delle formule di quadratura dei trapezi composta, Cavalieri-Simpson composta, Gauss-Legendre e Clenshaw-Curtis su 6 funzioni test.

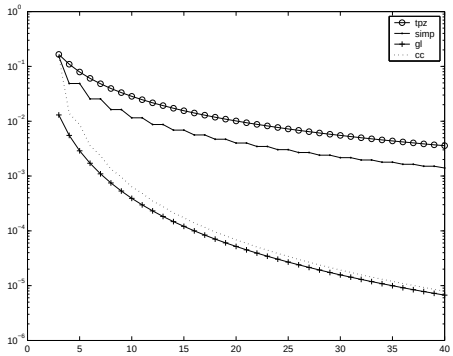


Figure 5: Grafico che illustra l'errore delle formule di quadratura dei trapezi composta, Cavalieri-Simpson composta, Gauss-Legendre e Clenshaw-Curtis sulla funzione test (**??**).

```
a=-1; b=1;
N=10;
ftype=2;

ajac=1/2; bjac=0;
```

e imporremo il risultato esatto non

```
res=exact_results(ftype);
```

bensì

```
res=1.77914365469190979259117902999 41;
```

Ai fini del corso, ciò è salvato nel file *demo_gauss_legendre_2.m*. Con $N = 5$ abbiamo

```
>> demo_gauss_legendre_2

          [RISULTATO][GAUSS-JACOBI  ]: 1.779143654120646200
          [GAUSS-JACOBI  ][ERR.ASS.]: 5.71e-010
          [SOMMA ABS. PESI][GAUSS - JACOBI]: 1.89e+000

>>
```

mentre con $N = 10$ nodi otteniamo

```
>> demo_gauss_legendre_2

          [RISULTATO][GAUSS-JACOBI  ]: 1.779143654691909300
          [GAUSS-JACOBI  ][ERR.ASS.]: 4.44e-016
          [SOMMA ABS. PESI][GAUSS - JACOBI]: 1.89e+000

>>
```

# References

[1] K. Atkinson, *Introduction to Numerical Analysis*, Wiley, 1989.

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

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

[4] The MathWorks Inc., *Numerical Computing with Matlab*, `http://www.mathworks.com/moler`.

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

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

[7] L.N. Trefethen, *"Is Gauss quadrature better than Clenshaw-Curtis?"*, SIAM Reviews, to appear (2007).

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