Laboratorio di Analisi Numerica
Lezione 10

Gianna Del Corso <delcorso@di.unipi.it>
Federico Poloni <f.poloni@sns.it>

11 Dicembre 2012

Quantità di esercizi: in questa dispensa ci sono *più esercizi* di quanti uno studente medio riesce a farne durante una lezione di laboratorio, specialmente tenendo conto anche degli esercizi facoltativi. Questo è perché sono pensate per "tenere impegnati" per tutta la lezione anche quegli studenti che già hanno un solido background di programmazione.
Quindi fate gli esercizi che riuscite, partendo da quelli *non* segnati come facoltativi, e non preoccupatevi se non li finite tutti!

1 Formule di quadratura

Una formula di quadratura è un'approssimazione di $S = \int_a^b f(x)\,dx$ della forma

$$S_{n+1} = \sum_{i=1}^{n} \omega_i f(x_i),$$

dove gli $n+1$ punti $x_i \in [a,b]$ per $i = 0, 1, \ldots, n$sono i *nodi* della formula e ω_i, $i = 0, \ldots, n$ sono i pesi. I pesi sono *indipendenti* dalla funzione, ma non dai nodi.

Una formula di quadratura ha *grado di precisione* k se integra perfettamente polinomi di grado minore o uguale a k.

1.1 Formule interpolatorie

Alla $f(x)$ si sostituisce il suo polinomio di interpolazione sui noi $x_i \in [a,b]$, $p(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$. Abbiamo

$$S_{n+1} = \sum_{i=0}^{n} \omega_i f(x_i), \quad \omega_i = \int_a^b L_i(x)\,dx = \int_a^b \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}\,dx.$$

Formule di questo tipo sono dette interpolatorie.

Una formula di quadratura interpolatoria S_{n+1} ha grado di precisione compreso tra n e $2n+1$.

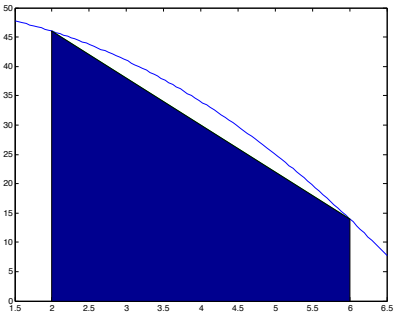

Figura 1: Approssimazione con la formula dei due punti

- Nel caso di nodi equidistanti si ottengono le formule di *Newton-Cotes*. Il grado di precisione di queste formule è n o $n+1$, i pesi sono numeri razionali, ma per $n \geq 8$ i pesi non sono tutti dello stesso segno e possiamo avere errori di cancellazione quando si sommano tra loro i termini.
- Si possono determinare i nodi in modo da massimizzare il grado di precisione. So ottengono le formule gaussiane, che hanno sempre pesi positivi anche se non sempre sono numeri razionali.

1.2 Formule di Newton-Cotes

Posto $h = (b-a)/n$, i nodi $x_i = a + i * h$, $i = 0, 1, \ldots, n$ sono equidistanti di passo h. Le formule di quadratura risultano

$$S_{n+1} = h \sum_{i=0}^{n} \alpha_i f(x_i),$$

dove

$$\alpha_i = \int_0^1 \prod_{j=0, j\neq i}^{n} \frac{t-j}{i-j}\, dt, \quad i = 0, 1, \ldots, n$$

dipendono da i e da n ma non dai nodi per cui i valori possono essere precomputati.

Formula dei due punti. Per $n = 1$, otteniamo $S_2 = \frac{h}{2}\left[f(a) + f(b)\right]$.

Esercizio 1. Si scriva una funzione `function S=duepunti(f, a, b)` che data una funzione f e un intervallo $[a, b]$ approssimi $\int_a^b f(x)\, dx$ con la formula dei due punti. Si aggiunga anche la rappresentazione grafica della funzione e del trapezio la cui area rappresenta l'approssimazione dell'integrale.

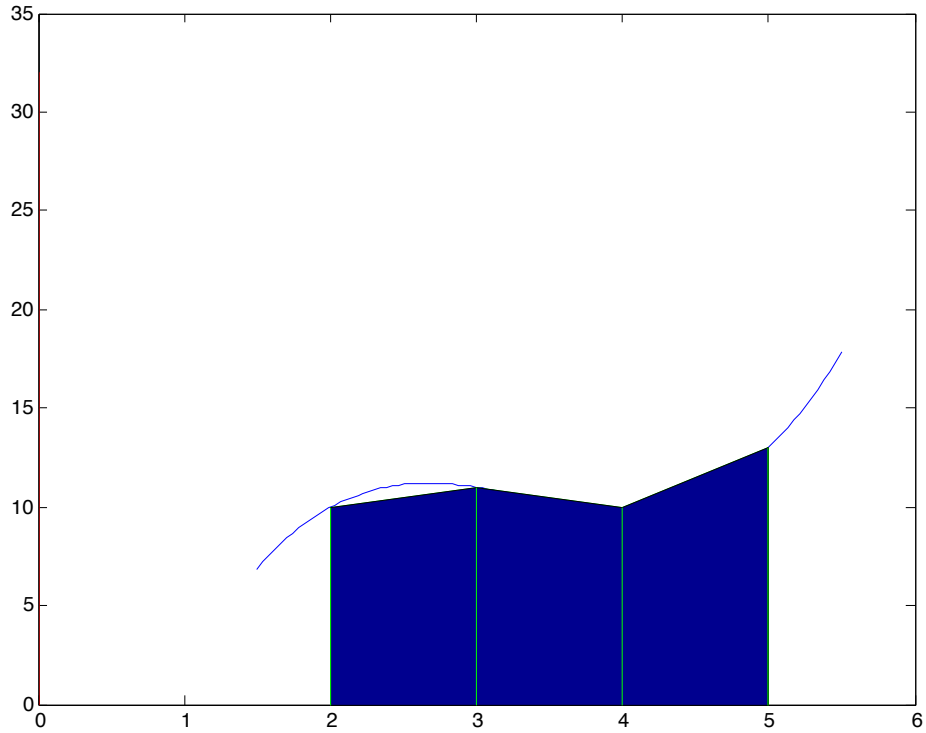

Figura 2: Approssimazione con la formula dei trapezi

1.3 Formule Newtoniane composte

Le formule di Newton-Cotes con $n > 7$ in generale non vengono utilizzate perchè i coefficienti non sono tutti dello stesso segno e possiamo ottenere errori di cancellazione. Si preferisce utilizzare formule composte mediante la ripetuta applicazioni di formule interpolatori su particolare sottointervalli.

L'intervallo $[a, b]$ è suddiviso in N sottointervalli con i punti equidistanti z_k, $k = 0, 1, \ldots, N$, tali che $a = z_0$ e $b = z_N$. Abbiamo

$$\int_a^b f(x)\,dx = \sum_{k=0}^{N-1} \int_{z_k}^{z_{k+1}} f(x)\,dx.$$

Si approssima poi ognuno degli integrali sull'intervallo $[z_k, z_{k+1}]$ con la formula inter-

polatoria $S_{n+1}^{(k)}$ su n punti, ottenendo

$$J_{n+1}^{(N)} = \sum_{k=0}^{N-1} S_{n+1}^{(k)}.$$

Formula dei trapezi. Si utilizza su ogni intervallo $[z_k, z_{k+1}]$ la formula dei due punti, $S_2^k = \frac{(b-a)}{2N}[f(z_k)+f(z_{k+1}]$, otteniamo la formula dei trapezi la cui interpretazione grafica è in Figura 2

$$J_2^{(N)} = \sum_{k=0}^{N-1} (\frac{b-a}{2N}\,[f(z_k) + f(z_k+1)] = \frac{b-a}{2N}\left[f(a) + 2\sum_{k=1}^{N-1} f(z_k) + f(b)\right].$$

Esercizio 2. Si scriva una funzione `function S=trapezi(f, a, b, N)` che data una funzione f e un intervallo $[a, b]$ approssima $\int_a^b f(x)\,dx$ con la formula dei trapezi su N punti. Si aggiunga anche la rappresentazione grafica della funzione e delle aree che contribuiscono all'integrale.

Esercizio 3. Si testi la funzione precedente per approssimare alcuni integrali, quali $\int_0^1 \sqrt(x)\,dx$, $\int_{-1}^1 \frac{dx}{x^2-\alpha}$, per $\alpha = 1.01$ e $\alpha = 2$, $\int_0^1 e^{-x^2}\,dx$.

Utilizzando l'espressione del resto del polinomio di interpolazione, si può dimostrare che il resto della formula composta dei trapezi è dato da

$$R_2^{(N)} = S - J_2^{(N)} = -\frac{(b-a)^3}{12N^2} f''(\xi), \quad \xi \in (a, b),$$

cioè $S - J_2^{(N)} \approx \frac{\delta_N}{N^2}$, dove δ_N dipende da $f''(\xi)$. Da questo, nell'ipotesi che $f''(x)$ vari poco nell'intervallo $[a, b]$, poichè $S - J_2^{(2N)} \approx \frac{\delta_{2N}}{4\,N^2}$, abbiamo che, assumendo $\delta_N \approx \delta_{2N} \approx \delta$, $J_2^{(N)} - J_2^{(2N)} \approx \frac{\delta}{4\,N^2}$, da cui

$$S - J_2^{(2N)} \approx \frac{J_2^{(N)} - J_2^{(2N)}}{3},$$

cioè la differenza tra l'approssimazione dell'integrate con la regola dei trapezi applicata su N e su $2N$ punti fornisce una stima del resto dell'approssimazione. Si può quindi procedere con successivi raddoppi di N fino a quando

$$\left|J_2^{(N)} - J_2^{(2N)}\right| < \varepsilon. \tag{1}$$

Si ricortda che per definire una funzione si può usare il comando `nomefun=@(x)\ldots`

Esercizio 4. Seguendo il seguente schema

```
function [S, err, nval]=doubletrap(f,a, b, tol, nmax)
% disegno della funzione
clearplot;
```

```
h= % passo di discritzzazione per avere un grafico soddisfacente
x=a:h:b;
y=f(x);
hold on;
plot(x,y, 'b');


JN= %approssimazione con la formula dei due punti;

% disegnare le rette verticali (a, f(a)) ad (a, 0), e (b, f(b)) e (b, 0)

if nmax>=3 % trapezi con 3 punti
    c=(a+b)/2;
    fc=f(c);
    plot([c,c], [fc,0], 'g');
    J2N=JN/2+fc*(b-c);
  else
    S=JN;
endif

 N=2; %denota il numero di intervalli
errore=abs(J2N-JN);
while (errore>tol & N+1<nmax)

       JN=...
       N=...

       J2N=affinatr(f,a, b, JN, N); % funzione da scrivere
       % che affina la stima JN raddoppiando il numero
       % dei punti, valuta la funzione solo sui nuovi punti
     errore=... ;
endwhile
  S=J2N;
  err=errore;
  nval=N+1;
  holf off;
endfunction
```

si scriva una funzione `function [S, err, nval]=doubletrap(f, a, b, tol, nmax)` che data una funzione f e un intervallo $[a, b]$ applichi alla funzione `f` la formula dei trapezi nell'intervallo `[a,b]`, raddoppiando il numero di intervalli ogni volta, fintanto che la differenza tra due stime successive non sia più piccola della tolleranza `tol` o non siano state fatte `nmax` iterazioni. Si aggiunga anche la rappresentazione grafica in cui si evidenziano i punti sui quali è stata valutata la funzione.

Esercizio 5. Si applichi la funzione doubletrap per stimare $\int_{-1}^{1} \frac{dx}{x^2-\alpha}$ con $\alpha = 1.01$ e

$\alpha = 2$. Si veda come si comporta l'errore in questi due casi e si cerchi di capire il motivo.

Esercizio 6 (facoltativo). Alla luce di quanto osservato nell'esercizio precedente, si nota che se una funzione ha intervalli sui quali è più difficile stimare l'integrale, potrebbe essere più coinveniente infittire i nodi di integrazione solo deve ce ne sia realmente bisogno.

Riuscite a trovare un modo di affinare la stima ricorsivamente solo sugli intervalli dove le stime precedenti non erano sufficienti a garantire un'approssimazione data (in base alla formula per stimare l'errore con $\frac{J_{2N}-J_N}{3}$)?

```
function [s,numval, errore] = trapad(f,a,b,tol,nmax)

clf;
                              % controlla estremi
                              % valuta la funzione nei primi tre punti
c = (a+b)/2;
fa = f(a);
fb =f(b);
fc = f(c);
numval = 3;

h= %passo di discretizzaziopne sufficientemente piccolo

% si disegna il grafico della funzione
x=a:h:b;
y=f(x);
hold on;
plot(x,y);
plot([a a],[0 fa],'g');
plot([b b],[0 fb],'g');
plot([c c],[0 fc],'g');

[s,numval, errore] = trap35(fn,a,c,b,fa,fc,fb,tol,nmax, numval);
endfunction

% funzione ricorsiva che calcola l'integrale in [a,b] con
% la formula dei trapezi a 3 punti e (se altre valutazioni di f
% sono possibili) anche con la formula a 5 punti.
% se la differenza fra queste due valutazioni e' minore della tolleranza
% OK, altrimenti si richiama ricorsivamente trap35
% Si assume che a <c <b e che i punti siano equispaziati.

function [s, numval, err] = trap35(fn,a,c,b,fa,fc,fb,tol,nmax, numval)
```

```
val3 = ...; % trapezi con 3 punti
if(numval>nmax)
  s = val3; % non sono possibili altre valutazioni
  err=0;
  return;
end

% -- possiamo calcolare qualche nuovo punto
a1 =... % punto medio [a,c]
b1 = ... % punto medio [c,b]
fa1 = f(a1);
fb1 = f(b1);
numval = ...;
plot .... % si disegnano i segmenti verticali (a1, f(a1)) (a1, 0) e (b1, f(b1)), (b1, 0)

% -- calcola stima piu' accurata
val5 =... ;% trapezi con 5 punti
s=val5;
err=abs(val5-val3)/3;
if err>tol & numval<nmax

  %chiamata ricorsiva su intervallo [a,c]
   [s1, numval1, err1]=trap35(...)'

   %chiamata ricorsiva su intervallo [c, b]

   [s2, numval2, err2] = trap35(...);

   s=...; %stima dell'integrale
   numval=....;
   err=....;

else
    err;
endif
endfunction
```