Quadratura numerica

Laboratorio di Calcolo Numerico

Federico Piazzon

24 Maggio 2022

Richiamo Trapezi e Parabole

Sia $f \in \mathcal{C}^0([a,b])$ con $[a,b]$ intervallo chiuso e limitato, la quadratura di f su $[a,b]$ consiste nell'approssimazione

$$\int_a^b f(x)dx = \sum_{i=1}^{N} \int_{a_i}^{b_i} f(x)dx \approx I_N^w(f) := \sum_{j=0}^{M(N,n)} w_j f(x_j).$$

I due casi più classici di formule *interpolatorie* su nodi equispaziati $x_j := a + jh, j = 0, 1, \dots, \frac{b-a}{h}$ sono:

$$w^{(trapezi)} = h(1/2, 1, 1, \dots, 1, 1/2), \quad h := \frac{b-a}{N}$$
$$w^{(parabole)} = h(1/3, 2/3, 4/3, 2/3, \dots\dots, 4/3, 1/3), \; h := \frac{b-a}{2N}.$$

Convergenza di Trapezi e Parabole

Convergenza Trapezi

Se $f \in \mathcal{C}^2([a,b])$ allora $\left|\int_a^b f(x)dx - I_N^{(trap)}(f)\right| \leq K\left(\frac{b-a}{N}\right)^2$.

Convergenza Parabole

Se $f \in \mathcal{C}^4([a,b])$ allora $\left|\int_a^b f(x)dx - I_N^{(parab)}(f)\right| \leq K\left(\frac{b-a}{2N}\right)^4$.

Trapezi e parabole con Matlab

```
function [x,w]=Trapezi(a,b,N)
h=(b-a)/N;
x=(a:h:b)';
w=h*[1/2,ones(1,N-1),1/2];
```

```
function [x,w]=Parabole(a,b,N)
h=(b-a)/(2*N);
x=(a:h:b)';
w=h.*[1,repmat([4 2],1,N-1),4,1]./3;
```

```
f=@(x) sin(x);
a=0;
b=pi/2;
I=w*f(x)
```

NB: pesi in riga e nodi in colonna per uso del prodotto scalare!

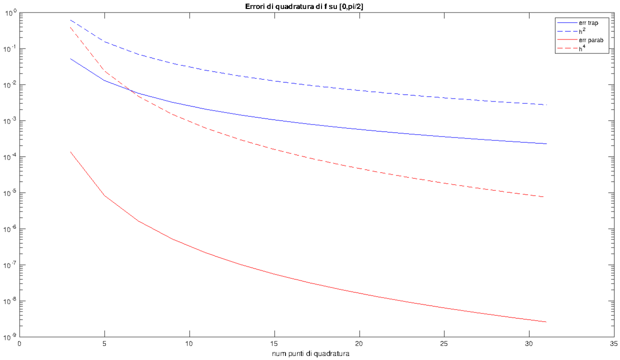
Testare `Trapezi` e `Parabole`

Esercizio 1a

Scrivere uno script `sinquad.m` che approssimi $\int_0^{\pi/2} \sin(x)dx$ utilizzando `Trapezi` e `Parabole` rispettivamente con $2N$ ed N sottointervalli e $N = 1, 2, \ldots, 15$.
Per ogni valore di N si calcolino gli errori assoluti e si crei un grafico semilogaritmico dei due errori rispetto ai punti utilizzati, di h^2 e di h^4.

Test di convergenza `sinquad.m`

Eseguiamo `sinquad.m` che implementa la quadratura su $2k+1$ punti equispaziati in $[0, \pi/2]$ con i trapezi e le parabole della funzione $f(x) = \sin(x)$. Vediamo il profilo di convergenza degli errori:



Test 2

Esercizio 1b

Si ripeta il test precedente nei casi

1. $f = |x|^{1.1}$, $a = -1, b = 1$
2. $f = |x|^{1.1}$, $a = -\sqrt{2}, b = \sqrt{3}$.

Si motivino i risultati ottenuti.

Raffinamento del passo di integrazione

Una strategia è raffinare fin quando il valore dell'integrale "si stabilizza"

Esercizio 2

Si scriva una function matlab intestata

```
[int,I,step,flag]=MyQuad(a,b,f,formula,toll,maxN)
```

che, presa in input il puntatore a funzione (regola di quadratura) `formula` (chiamata `formula(a,b,N)`, calcoli la quadratura di f con $1, 2, 4, \ldots$ sottointervalli fermandosi qualora il nuovo valore di quadratura differeisca dal precedente di meno di `toll` o il numero di sottointervalli abbia raggiunto `maxN`.
La function deve restituire in output l'ultima approssimazione `integrale`, la successione delle approssimazioni `I`, l'ultima differenza tra approssimazioni `step` e `flag`=1 in caso in cui la tolleranza sia stata raggiunta, o `flag`=0 in caso in cui la tolleranza non sia stata raggiunta.

Test di raffinamento

Testiamo la function `MyQuad` con lo script `TestMyQuad.m` disponibile su moodle

```
Quadratura con trapezi e parabole
trapezi non ha raggiunto la tolleranza
parabole ha raggiunto la tolleranza
intervalli       punti   integrale               errore
512                      513     1.331656139
    6.590297463e-06
256              513     1.331649576     2.776926399e-08
```

NB: per passare la formula di quadratura nella chiamata di `MyQuadrature` usiamo l'operatore di conversione a function handle. Perchè?

Formule di quadratura interpolatorie semplici

Approssimare formule interpolatorie di grado alto

Formula interpolatoria

$$\int_a^b f(x)dx \approx \int_a^b \sum_{j=0}^{n} f(x_j)\ell_{j,n}(x)dx = \sum_{j=0}^{n} f(x_i)\int_a^b \ell_{j,n}(x)dx$$

Dunque si ha

$$w_j := \int_a^b \ell_{j,n}(x)dx,$$

i pesi sono *i momenti della base di Lagrange*.

Come possiamo approssimare numericamente i w_j?

Approssimazione dei w_j

Consideriamo la matrice rettangolare (Vandermonde nella base di Lagrange)

$$L_{i,j} = \ell_{j,n}(z_i)$$

dove z_i, $i = 1, 2, dots, M >> n + 1$ sono nodi di quadratura per una formula "accurata" avente pesi ω_i.
Allora

$$\omega^t L = (\sum_{i=1}^{M} \omega_i \ell_{j,n}(z_i))_{j=0,1,\dots,n} \approx w^t.$$

Ad esempio potremmo usare una formula di Cavalieri Simpson, raffinando finchè i momenti calcolati non si stabilizzano.

Implementazione

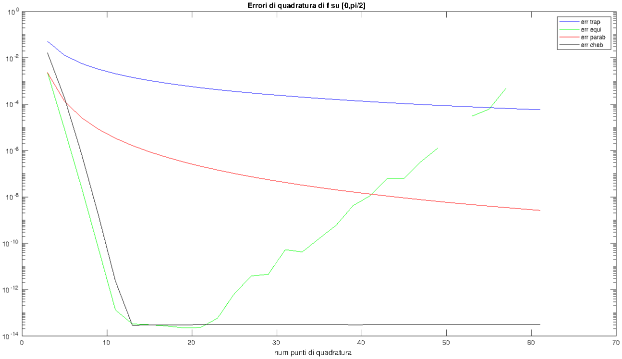
```
function [w,W,flag]=FormulaInterpolatoria(xinterp,a,b,toll
    ,Nmax)
N=1;
[xeval,wp]=Parabole(a,b,N);
L=LagrangePoly(xinterp,xeval);
w=wp*L;
W=w;flag=0;
step=toll+1;
while step>toll && N<Nmax/2
    N=2*N;
    [xeval,wp]=Parabole(a,b,N);
    L=LagrangePoly(xinterp,xeval);
    W=[W;wp*L];
    step=norm(W(end,:)-W(end-1,:));
end
w=W(end,:);
if step<toll
    flag=1;
end
```

Test di `FormulaInterpolatoria`

Esercizio 3

Si modifichi lo script `sinquad.m` affinchè approssimi l'integrale di $\sin(x)$ in $[0, \pi/2]$ anche tramite quadratura interpolatoria (con lo stesso numero di punti) con nodi equispaziati `xequi=linspace(a,b,n+1)'` o di Chebyshev `xcheb=(a+b)/2+(b-a)/2*`
`cos((2*(n-1:-1:0)+1)./(2*n+1)*pi)'`

Risultati del test



Cosa è successo? E’ una sorta di "fenomeno di Runge"?