Interpolazione spline in Matlab per Ingegneria dell'Energia
Esercizi risolti. [1]

A. Sommariva[2]

Abstract

Interpolazione spline, esempi.

Ultima revisione: 15 dicembre 2018

1. Esercizi

La funzione di Runge $f(x) = 1/(1+x^2)$ appartiene a $C^\infty([-5,5])$ e di conseguenza, nonostante la convergenza puntuale dell'interpolazione polinomiale in nodi equispaziati non sia garantita, ciò non si può dire per l'interpolazione mediante splines lineari o cubiche (sappiamo sussistere la convergenza uniforme).

Nel caso lineare, se

- $\Delta = \{x_i\}$ è unaa suddivisione equispaziata di ampiezza h_Δ dell'intervallo $[-5,5]$,
- $x \in [x_i, x_{i+1}]$,

si dimostra che la spline $s_{1,\Delta}$ interpolante f nei nodi $\{x_i\}$ determinati dalla suddivisione è tale che

$$\|f - s_{1,\Delta}\|_\infty \leq \frac{h_\Delta^2}{4}.$$

Nel caso di splines cubiche vincolate si dimostra che la spline di questo tipo $s_{3,\Delta}$ interpolante f nei nodi $\{x_i\}$ determinati dalla suddivisione è tale che

$$\|f - s_{3,\Delta}\|_\infty \leq (120/384) h_\Delta^4.$$

Di conseguenza, qualora l'ampiezza della suddivisione tenda a 0, le interpolanti splines sono uniformemente convergenti alla funzione di Runge, in virtù del teorema dei due carabinieri.

1.1. Controesempio di Runge e le spline lineari

Aiutandosi con quanto visto in `demo_spline_lineare`, si definisca una function `errore_spline_lineare`, avente la seguente intestazione

```
% Oggetto:
% Errore della spline lineare "s1" interpolante "f" nei
% punti  x_k=a+(k-1)h, k=1,...,N+1, h=(b-a)/N
%
% Input:
% f: funzione da interpolare
% a,b: dominio di "f"
% N: numero di subintervalli equispaziati in cui e'
%    suddiviso [a,b]
%
% Output:
% maxerr: stima del massimo errore di interpolazione,
%         ovvero max(abs(f(x)-s1(x)))
```

Tale routine, data la funzione f, calcola l'interpolante lineare spline s_1 nelle coppie (x_k, y_k), dove

$$x_k = \frac{2\pi(k-1)}{N},\ k = 1, \ldots, N+1$$

e la valuta nei nodi test

$$t_k = \frac{2\pi(k-1)}{100000},\ k = 1, \ldots, 100001$$

ponendo tale risultato in z_k. Nella variabile `maxerr` venga assegnata una stima del massimo errore di interpolazione

$$\max_{x\in[a,b]} |f(x) - s_1(x)|$$

mediante

$$\max_{k=1,\ldots,100001} |f(t_k) - s_1(t_k)| = \max_{k=1,\ldots,100001} |f(t_k) - z_k|.$$

1.2. Svolgimento

```
function maxerr=errore_spline_lineare(f,a,b,N)

% Oggetto:
% Errore della spline lineare "s1" interpolante "f" nei ...
    punti
% x_k=a+(k-1)h, k=1,...,N+1, h=(b-a)/N
%
% Input:
% f: funzione da interpolare
% a,b: dominio di "f"
% N: numero di subintervalli equispaziati in cui e' ...
    suddiviso [a,b]
%
% Output:
% maxerr: stima del massimo errore di interpolazione, ...
    ovvero
%         max(abs(f(x)-s1(x)))


% punti equispaziati in cui interpolare "f".
```

```
hx=(b-a)/N;  x=a:hx:b;
y=feval(f,x); % valore funzione in "x"

% punti test in cui valutare gli errori forniti tra
% funzione e interpolante
ht=1/100000; t=a:ht:b;

% valori assunti dall'interpolante spline lineare
%  nei punti test "tx"
z = interp1(x,y,t,'linear');

% valore della funzione "f" nei nodi test
ftx=feval(f,t);

% valutazione errore di interpolazione nei nodi test
maxerr=max(abs(ftx-z));
```

2. Esercizio `demo_runge_spline_lineare`

Utilizzando la funzione

`errore_spline_lineare`

si definisca una funzione

`demo_runge_spline_lineare`

che

1. non abbia variabili di input, né di output;
2. calcoli il valore assunto dalle variabili `maxerr`, definendo il vettore `eev`, aventi lunghezza 9, tale che la k sima componente di `eev` corrisponda al valore `maxerr` fornito tramite `errore_spline_lineare` per $N = 2^k$,
3. esegua in una figura i grafici in scala semilogaritmica delle coppie $(2^k$,`eev`$(k))$,
4. utilizzi quale titolo della figura la stringa

   `'Errori di interpolazione spline lineare:  nodi equispaziati'`

   ed il plot abbia la preferenza `'LineWidth',2`;
5. salvi su un file

   `errori_interpolazione_spline_lineare.txt`

   i valori del tipo 2^k utilizzati, gli errori `eev`, cosicchè la tabella risultante abbia alla k-sima riga,

   - la quantità 2^k con 5 cifre prima della virgola e nessuna dopo la virgola, in notazione decimale,
   - l'errore `eev(k)`, ovvero la k-sima componente del vettore `eev`, con 1 cifra prima della virgola, una dopo la virgola, in notazione esponenziale.

2.1. Svolgimento

```
function demo_runge_spline_lineare

% Oggetto:
% Interpolazione della funzione di Runge in [-5,5],
% mediante splines lineari. Massimi errori in 2^k
% suddivisioni equispaziate con k=1,...,11.

a=-5; b=5;
f=@(x) 1./(1+x.^2); % funzione vettoriale
```

```
for k=1:11
    N=2^k;
    maxerr=errore_spline_lineare(f,a,b,N);
    eev(k)=maxerr;
end

% ------------ plot ------------

clf;
semilogy(1:11,eev,'LineWidth',2);
hold on;
% legenda e titolo
title('Errori di interpolazione spline lineare: nodi ...
    equispaziati');
hold off;

% ---------- salvataggio risultati su file ----------

% creazione del file con facolta' di scrittura.
fid=fopen('errori_interpolazione_spline_lineare.txt','w'...
    );
% dati immagazzinati nella matrice A (si immagazzinino ...
    come vettori riga,
% e bisogna ricordare che "eev", "ecv" sono riga.
A=[1:11; eev];
% scrittura dei dati su file.
fprintf(fid,'\n %3.0f  %1.15e',A);
% chiusura file
fclose(fid);
```

Lanciando tale routine nella command-window, otteniamo il grafico in figura, e il file di testo

`'errori_interpolazione_spline_lineare.txt'`

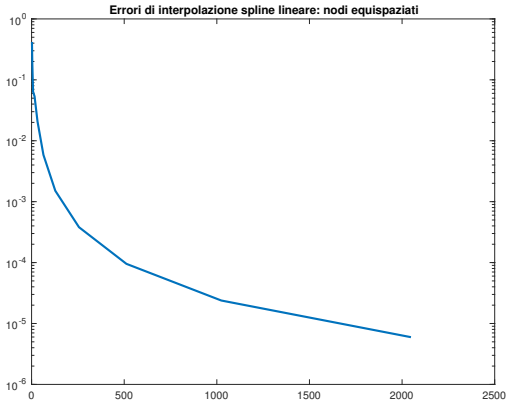

Figura 1: Errore compiuto interpolando la funzione di Runge mediante spline lineari, in intervalli equispaziati. L'ascissa corrisponde al numero di subintervalli della suddivisione, l'ordinata ad una stima del massimo errore compiuto.

```
     2  4.2e-01
     4  1.8e-01
     8  6.4e-02
    16  5.4e-02
    32  2.1e-02
    64  5.9e-03
   128  1.5e-03
   256  3.8e-04
   512  9.5e-05
  1024  2.4e-05
  2048  6.0e-06
```

Dalla figura e dalla tabella, deduciamo che numericamente, all'aumentare del numero delle suddivisioni, il massimo errore di interpolazione tende a 0, come indicato dalla teoria.

2.2. *Controesempio di Runge e le spline cubiche*

Aiutandosi con quanto visto in

`demo_spline_cubica_naturale`,

si definisca una function

`errore_spline_cubica_naturale`,

avente la seguente intestazione

```
% Oggetto:
% Errore della spline cubica naturale "s3" interpolante
% "f" nei punti  x_k=a+(k-1)h, k=1,...,N+1, h=(b-a)/N
%
% Input:
% f: funzione da interpolare
% a,b: dominio di "f"
% N: numero di subintervalli equispaziati in cui e'
%    suddiviso [a,b]
%
% Output:
% maxerr: stima del massimo errore di interpolazione,
%         ovvero max(abs(f(x)-s3(x)))
```

Tale routine, data la funzione f, calcola l'interpolante cubica spline (con condizioni naturali) s_3 nelle coppie (x_k, y_k), dove

$$x_k = \frac{2\pi(k-1)}{N},\ k = 1, \ldots, N+1$$

e la valuta nei nodi test

$$t_k = \frac{2\pi(k-1)}{100000},\ k = 1, \ldots, 100001$$

ponendo tale risultato in z_k. Nella variabile `maxerr` venga assegnata una stima del massimo errore di interpolazione

$$\max_{x\in[a,b]} |f(x) - s_3(x)|$$

mediante

$$\max_{k=1,\ldots,100001} |f(t_k) - s_3(t_k)| = \max_{k=1,\ldots,100001} |f(tx_k) - z_k|.$$

2.3. *Svolgimento*

L'implementazione è la stessa di

`errore_spline_lineare`

eccetto che al posto di

```
z=interp1(x,y,'linear');
```

abbiamo

```
pp=csape(x,y,'variational');
z=ppval(pp,t);
```

Quindi:

```
function maxerr=errore_spline_cubica_naturale(f,a,b,N)

% Oggetto:
% Errore della spline lineare "s3" interpolante "f" nei ...
    punti
% x_k=a+(k-1)h, k=1,...,N+1, h=(b-a)/N
%
% Input:
% f: funzione da interpolare
% a,b: dominio di "f"
% N: numero di subintervalli equispaziati in cui e' ...
    suddiviso [a,b]
%
% Output:
% maxerr: stima del massimo errore di interpolazione, ...
    ovvero
%         max(abs(f(x)-s3(x)))


% punti equispaziati in cui interpolare "f".
hx=(b-a)/N;  x=a:hx:b;
y=feval(f,x); % valore funzione in "x"

% punti test in cui valutare gli errori forniti tra
% funzione e interpolante
ht=1/100000; t=a:ht:b;

% valori assunti dall'interpolante spline lineare
%  nei punti test "tx"
pp=csape(x,y,'variational'); z=ppval(pp,t);

% valore della funzione "f" nei nodi test
ftx=feval(f,t);

% valutazione errore di interpolazione nei nodi test
maxerr=max(abs(ftx-z));
```

3. **Esercizio** `demo_runge1`

Utilizzando la funzione

`errore_spline_cubica_naturale`,

si definisca una funzione

`demo_runge_spline_cubica_naturale`

che

1. non abbia variabili di input, né di output;
2. calcoli il valore assunto dalle variabili `maxerr`, definendo il vettore `eev`, aventi lunghezza 9, tale che la k sima componente di `eev` corrisponda al valore `maxerr` fornito tramite `errore_spline_cubica_naturale` per $N = 2^k$,
3. esegua in una figura i grafici in scala semilogaritmica delle coppie $(2^k$,`eev`$(k))$,
4. utilizzi quale titolo della figura la stringa

   `'Errori di interpolazione spline cubica naturale: nodi equispaziati'`

   ed il plot abbia la preferenza `'LineWidth',2`;
5. salvi su un file

   `errori_interpolazione_spline_cubica_naturale.txt`

   i valori del tipo 2^k utilizzati, gli errori `eev`, cosicchè la tabella risultante abbia alla k-sima riga,

- la quantità 2^k con 5 cifre prima della virgola e nessuna dopo la virgola, in notazione decimale,
- l'errore `eev(k)`, ovvero la k-sima componente del vettore `eev`, con 1 cifra prima della virgola, una dopo la virgola, in notazione esponenziale.

3.1. Svolgimento

Lo svolgimento è praticamente uguale a quello di

`demo_runge_spline_lineare`

a parte la chiamata della funzione

`errori_interpolazione_spline_cubica_naturale`

Otteniamo:

```
function demo_runge_spline_cubica_naturale

% Oggetto:
% Interpolazione della funzione di Runge in [-5,5],
% mediante splines cubiche naturali. Massimi errori in ...
    2^k
% suddivisioni equispaziate con k=1,...,11.

a=-5; b=5;
f=@(x) 1./(1+x.^2); % funzione vettoriale

for k=1:11
    N=2^k;
    maxerr=errore_spline_cubica_naturale(f,a,b,N);
    eev(k)=maxerr;
end

% ------------- plot -------------

clf;
semilogy(2.^(1:11),eev,'LineWidth',2);
hold on;
% legenda e titolo
title('Errori di interpolazione spline cubica naturale: ...
    nodi equispaziati');
hold off;

% ---------- salvataggio risultati su file ----------

% creazione del file con facolta' di scrittura.
fid=fopen('errori_interpolazione_spline_cubica_naturale....
    txt','w');
% dati immagazzinati nella matrice A (si immagazzinino ...
    come vettori riga,
% e bisogna ricordare che "eev", "ecv" sono riga.
A=[2.^(1:11); eev];
% scrittura dei dati su file.
fprintf(fid,'\n %5.0f  %1.1e',A);
% chiusura file
fclose(fid);
```

Lanciata tale demo nella command-window otteniamo la figura e la tabella sotto illustrate come da file

`errori_interpolazione_spline_cubica_naturale.txt`.

```
    2  6.0e-01
    4  2.8e-01
    8  5.6e-02
   16  3.7e-03
   32  6.6e-04
   64  4.0e-05
  128  2.5e-06
  256  6.3e-07
  512  1.6e-07
```

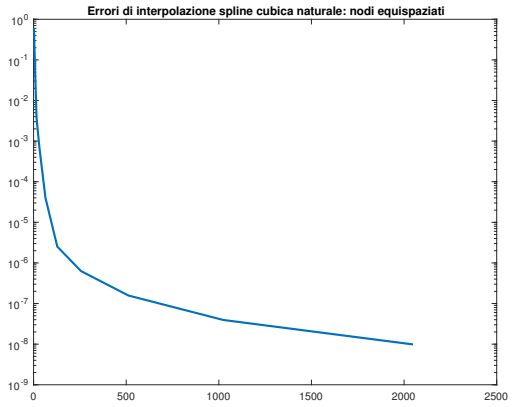

Figura 2: Errore compiuto interpolando la funzione di Runge mediante spline cubiche naturali, in intervalli equispaziati. L'ascissa corrisponde al numero di subintervalli della suddivisione, l'ordinata ad una stima del massimo errore compiuto.

```
 1024  3.9e-08
 2048  9.9e-09
```

Una rapida occhiata alle tabelle mostra la migliore performance della interpolazione mediante spline cubiche naturali. Ad esempio, con 2048 intervalli equispaziati, la spline lineare porge un massimo errore di interpolazione di $6.0 \cdot 10^{-6}$, mentre la spline cubica naturale di $9.9 \cdot 10^{-9}$.

Bibliografia

[1] Mathworks, Cubic spline interpolation with end conditions, https://www.mathworks.com/help/curvefit/csape.html