Interpolazione spline in Matlab

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica Pura e Applicata

7 maggio 2021

Splines

Sia $[a, b] \subset \mathbb{R}$ chiuso e limitato, e sia $a = x_0 < x_1 < \ldots < x_n = b$. Una spline di grado m (o *ordine* $m + 1$) è una funzione in $C^{m-1}([a, b])$ che in ogni intervallo $[x_i, x_{i+1}]$, con $i = 0, \ldots, n - 1$, è un polinomio di grado m.

Alcuni esempi:

- spline di grado 1 (lineari): una funzione in $C([a, b])$ che in ogni intervallo $[x_i, x_{i+1}]$, con $i = 0, \ldots, n - 1$, è un polinomio di grado 1.
- spline di grado 3 (cubiche): una funzione in $C^2([a, b])$ che in ogni intervallo $[x_i, x_{i+1}]$, con $i = 0, \ldots, n - 1$, è un polinomio di grado 3.

Interpolazione con splines

Si dimostra che

- la spline interpolante di grado 1 esiste ed è unica,
- la spline interpolante di grado 3 esiste ed è unica, qualora si aggiungano due opportune condizioni aggiuntive.

In quest'ultimo caso, classiche richieste aggiuntive sono

- Spline naturale: $s_3^{(2)}(a) = s_3^{(2)}(b) = 0$.
- Spline periodica: $s_3^{(1)}(a) = s_3^{(1)}(b)$, $s_3^{(2)}(a) = s_3^{(2)}(b)$.
- Spline vincolata: $s_3^{(1)}(a) = y'_a$, $s_3^{(1)}(b) = y'_b$ (con y'_a, y'_b assegnati).

Splines cubiche di tipo *not-a-knot*

La spline cubica s_3^{NAK} con vincolo not-a-knot è definita come segue:

- Interpola le coppie (x_0, y_0), ..., (x_n, y_n);
- E' un polinomio di grado 3 nell'intervallo $[x_0, x_2]$;
- E' un polinomio di grado 3 nell'intervallo $[x_{n-2}, x_n]$

Si dimostra che una tale spline di grado $n = 3$ esiste ed è unica.

Nota.

Si osservi che per le spline cubiche generiche s_3 non è detto che

- *la restrizione di s_3 a $[x_0, x_1]$ e $[x_1, x_2]$ siano lo stesso polinomio,*
- *similmente che la restrizione di s_3 a $[x_{n-2}, x_{n-1}]$ e $[x_{n-1}, x_n]$ siano lo stesso polinomio.*

Interpolanti splines in Matlab

In questa sezione vediamo come determinare le interpolanti splines lineari e cubiche in Matlab. La tentazione è usare l'help di Matlab relativamente al comando `spline` da cui otteniamo:

```
>> help spline
 spline Cubic spline data interpolation.
    PP = spline(X,Y) provides the piecewise polynomial form of the
    cubic spline interpolant to the data values Y at the data sites
         X,
    for use with the evaluator PPVAL and the spline utility UNMKPP.
    X must be a vector.
    ...
    Example:
    This illustrates the use of clamped or complete spline
         interpolation where
    end slopes are prescribed. In this example, zero slopes at the
         ends of an
    interpolant to the values of a certain distribution are enforced
         :
       x = -4:4; y = [0 .15 1.12 2.36 2.36 1.46 .49 .06 0];
       cs = spline(x,[0 y 0]);
       xx = linspace(-4,4,101);
       plot(x,y,'o',xx,ppval(cs,xx),'-');
    ...
    See also interp1, pchip, ppval, mkpp, unmkpp.
    ...
>>
```

Interpolanti splines in Matlab

Come suggerito proviamo anche qualche altro comando, quale `interp1`:

```
>> help interp1
 interp1 1-D interpolation (table lookup)

    Vq = interp1(X,V,Xq) interpolates to find Vq, the values of the
    underlying function V=F(X) at the query points Xq.
   ...
    Vq = interp1(X,V,Xq,METHOD) specifies the interpolation method.
    The available methods are:
      'linear'   - (default) linear interpolation
      'nearest'  - nearest neighbor interpolation
      'next'     - next neighbor interpolation
      'previous' - previous neighbor interpolation
      'spline'   - piecewise cubic spline interpolation (SPLINE)
      'pchip'    - shape-preserving piecewise cubic interpolation
      'cubic'    - same as 'pchip'
      'v5cubic'  - the cubic interpolation from MATLAB 5, which does
           not
                   extrapolate and uses 'spline' if X is not equally
                   spaced.
      'makima'   - modified Akima cubic interpolation
   ...
    For example, generate a coarse sine curve and interpolate over a
    finer abscissa:
        X = 0:10; V = sin(X); Xq = 0:.25:10;
        Vq = interp1(X,V,Xq); plot(X,V,'o',Xq,Vq,':.')
   ...
>>
```

Interpolanti splines in Matlab

Di conseguenza, il comando giusto per calcolare l'interpolante spline sembra essere `interp1`.

Lo si capisce in particolare quando l'help dice

```
Vq = interp1(X,V,Xq,METHOD) specifies the interpolation method.
```

ovvero che con ulteriori specifiche `METHOD`, quali `linear` o `spline`, permette di valutare la spline interpolante (rispettivamente lineare e cubica con condizioni not-a-knot) nelle coppie (X_k, V_k), $k = 1, \ldots, m$ nei punti Xq_s, $s = 1, \ldots, M$ e assegna tale risultato a Vq_s.

Interpolanti spline lineari in Matlab

Registriamo nel file demo_spline_lineare.m:

```
function demo_spline_lineare

% Oggetto:
% Interpolazione della funzione "sin" in [0,2*pi],
% mediante splines lineari

a=0; b=2*pi;
f=@(x) sin(x);

% numero subintervalli
N=7;

% 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=1/10000; t=a:ht:b;
```

Interpolanti spline lineari in Matlab

```
% 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));

% stampa risultati a video
fprintf('\n \t massimo errore di interpolazione: %1.2e',maxerr);
fprintf('\n \n');

%---------------- plot ----------------
clf;
plot(t,ftx,t,z,'LineWidth',2);
hold on;
% grafico delle coppie da interpolare
plot(x,y,'ko','LineWidth',1,'MarkerFaceColor','c','MarkerSize',8);
% legenda e titolo
title('Esempio interpolante spline lineare');
legend('sin(x)','intp. spline lineare','coppie da intp.');
hold off
```

Interpolanti spline lineari in Matlab

La lettura del codice dice che

- Data la funzione $f(x) = \sin(x)$ si calcola l'interpolante lineare spline s_1 nelle coppie (x_k, y_k), dove

$$x_k = \frac{2\pi(k-1)}{7}, \quad k = 1, \ldots, 8$$

  e la si valuta nei nodi test

$$t_k = \frac{2\pi(k-1)}{10000}, \quad k = 1, \ldots, 10001$$

  ponendo tale risultato in z_k.
- A video viene scritto il massimo errore di interpolazione

$$\max_{k=1,\ldots,10001} |f(t_k) - s_1(t_k)| = \max_{k=1,\ldots,10001} |f(t_k) - z_k|.$$

- Di seguito si esegue il grafico sia di f che di s_1, evidenziando le coppie da interpolare.

Da command-window:

```
>> demo_spline_lineare

   massimo errore di interpolazione: 9.66e-02

>>
```

Interpolanti spline lineari in Matlab

Quale risultato otteniamo il grafico in figura.

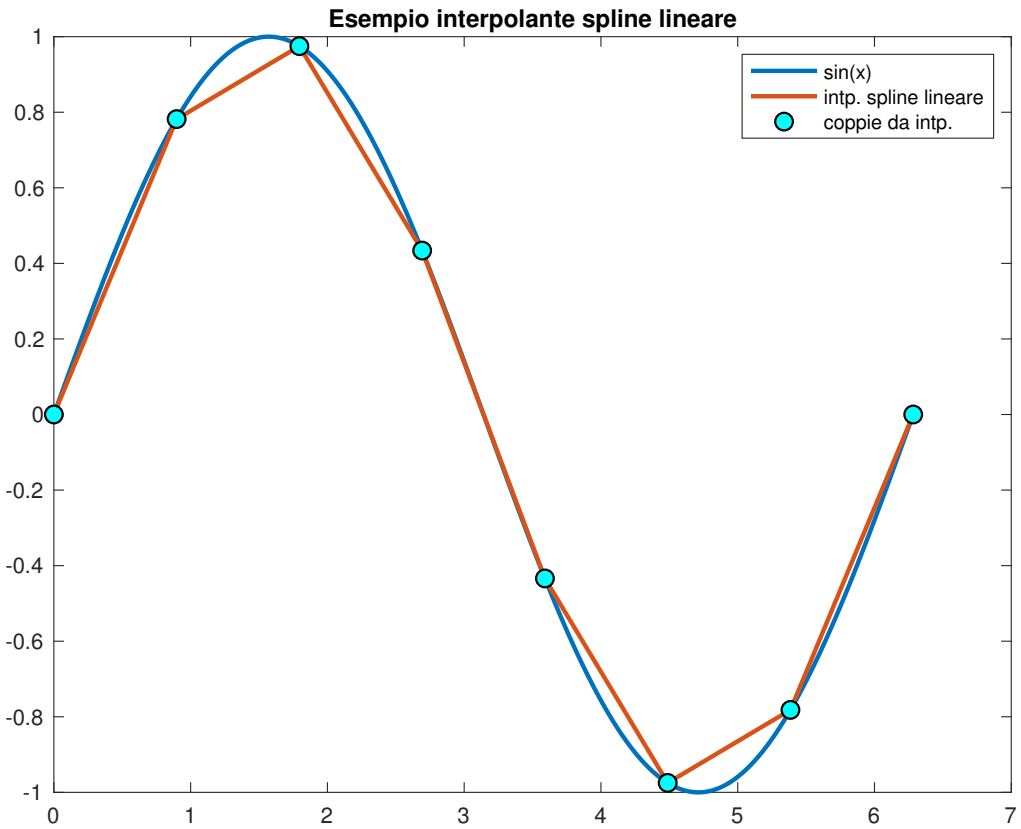

Figura: Grafici della funzione $f(x) = \sin(x)$ e dell'interpolante spline lineare s_1.

Interpolanti spline cubiche in Matlab

Modifichiamo la demo precedente in una nuova function

demo_spline_cubica.m,

sostituendo

```
ty = interp1(x,y,tx,'linear');
```

con

```
ty = interp1(x,y,tx,'spline');
```

Data la funzione $f(x) = \sin(x)$, la routine

- calcola l'interpolante spline cubica (con condizioni *not-a-knot*) $s_7^{(3)}$ nelle coppie (x_k, y_k), dove $x_k = \frac{2\pi(k-1)}{7}$, $k = 1, \ldots, 8$,
- valuta $s_7^{(3)}$ nei nodi test $tx_k = \frac{2\pi(k-1)}{10000}$, $k = 1, \ldots, 10001$, ponendo tale risultato in ty_k.

Da command-window:

```
>> demo_spline_cubica
   massimo errore di interpolazione: 1.45e-02
>>
```

Interpolanti spline cubiche in Matlab

Quale risultato otteniamo il grafico in figura.

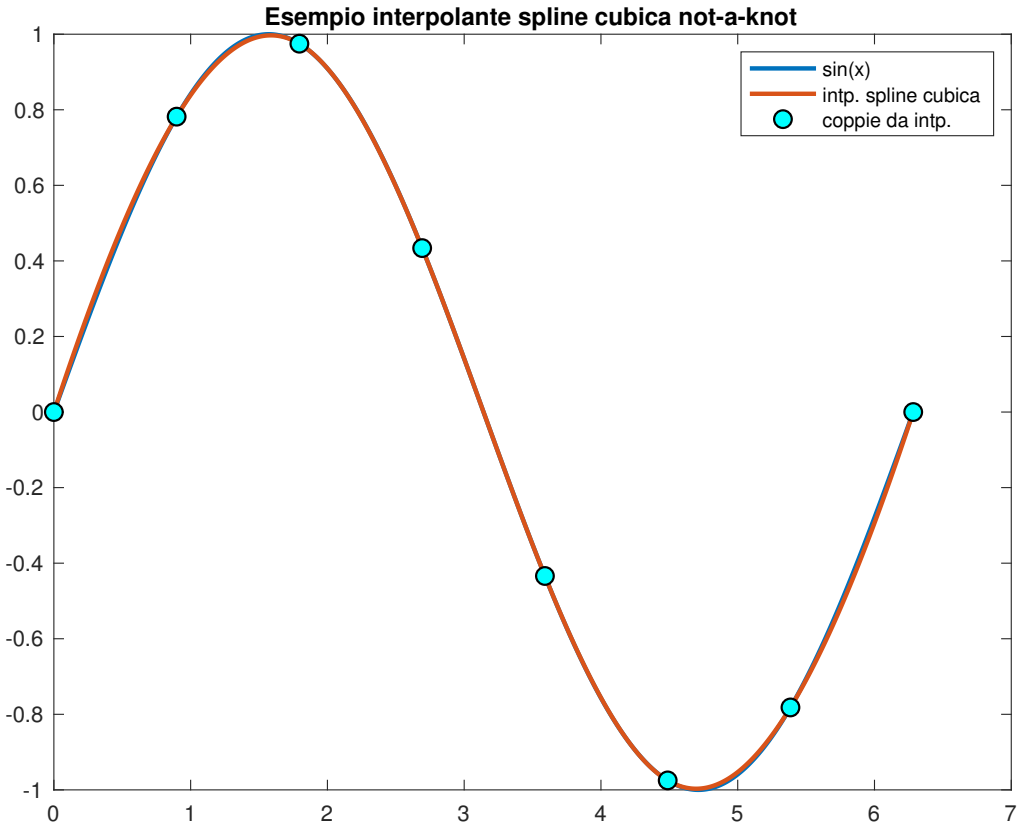

Figura: Grafici della funzione $f(x) = \sin(x)$ e dell'interpolante spline cubica s_3 con condizioni *not-a-knot*.

Interpolanti spline cubiche in Matlab

Un paragone visivo e geometrico mostra con evidenza la miglior qualità dell'approssimazione mediante spline cubica (con condizione *not-a-knot*).

Qualora intendiamo determinare la spline cubica *naturale* (ovvero con $s_3^{(2)}(0) = s_3^{(2)}(2\pi) = 0$), che sia interpolante nei nodi prefissati, basta sostituire

```
ty = interp1(x,y,tx,'spline');
```

con

```
pp=csape(x,y,'variational'); ty=ppval(pp,tx);
```

Le routines csape.m e ppval.m richiedono sia installata la toolbox Matlab spline. Salvata la modifica nel file `demo_spline_cubica_naturale.m` abbiamo

```
>> demo_spline_cubica_naturale
   massimo errore di interpolazione: 2.00e-03
>>
```

Quale risultato otteniamo il grafico in figura, non molto diverso dal precedente ottenuto mediante una spline cubica interpolante con condizione *not-a-knot*.

Per altri tipi di splines cubiche, si veda [1].

Interpolanti spline cubiche in Matlab

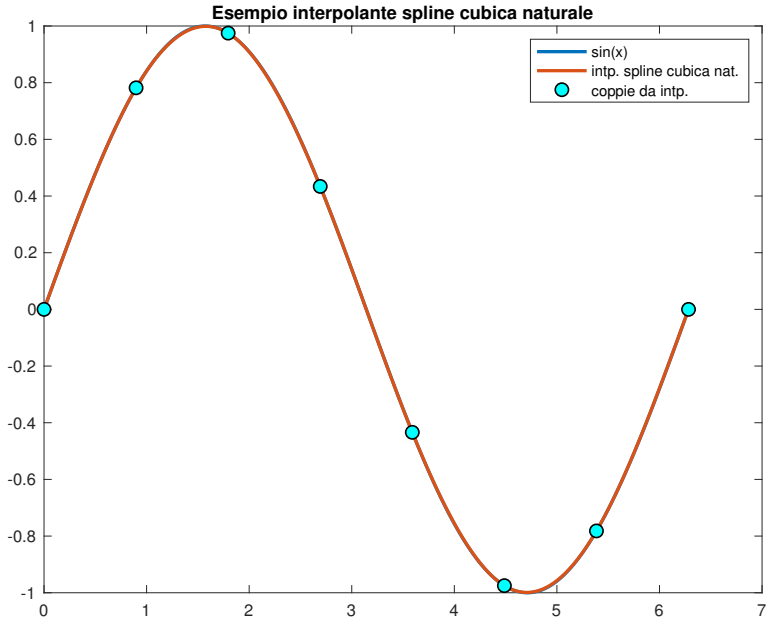

Figura: Grafici della funzione $f(x) = \sin(x)$ e dell'interpolante spline cubica s_3 con condizioni *naturali*.

Interpolanti spline cubiche in Matlab: esercizio errore_spline_lineare

Si definisca una function

errore_spline_lineare

avente la seguente intestazione

```
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)))
```

Tale function

- definisca il vettore `x` in cui $x_k = a + (k-1)\cdot h$, $h = (b-a)/N$, $k = 1, \ldots, N+1$ (punti equispaziati, non serve utilizzare il ciclo for!);
- definisca il vettore `y` in cui $y_k = f(x_k)$;
- definisca il vettore `t` in cui $t_k = a + (k-1)\cdot h$, $h = (b-a)/M$, $k = 1, \ldots, M+1$ in cui $M = 100000$ (punti equispaziati!);
- valuti la spline lineare s interpolante le coppie $(x_k, f(x_k))$, $k = 1, \ldots, N+1$ nei punti test t_k, $k = 1, \ldots, M+1$ e ponga il risultato nel vettore `z` (si utilizzi il comando `interp1`);
- valuti la funzione f nei punti test t_k, $k = 1, \ldots, M+1$ e ponga il risultato nel vettore `ftx`;
- calcoli il massimo errore `maxerr`, ovvero $\max_{k=1,\ldots,M+1} |f(t_k) - s(t_k)|$.

Correzione esercizio errore_spline_lineare

```
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 funz. 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));
```

Interpolanti spline cubiche in Matlab: esercizio demo_runge_spline_lineare

Si definisca una funzione `demo_runge_spline_lineare` che

1. non abbia variabili di input, né di output;
2. definisca la funzione di Runge `f` ove $f(x) = 1/(1+x^2)$, in forma vettoriale;
3. ponga `a=-5` e `b=5`;
4. utilizzando la chiamata

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

   calcoli per $N = 2, 4, 8, 16, 32, 64, 128, 256, 512 = 2^9$ il valore assunto da `maxerr` e immagazzini il valore ottenuto per $N = 2^k$ nella k-sima componente del vettore `eev`; in altri termini si calcoli per ogni `N(k)=2^k`, $k = 1, \ldots, 9$, la quantitá `maxerr` relativa a tale scelta di `N(k)` e la si salvi in `eev(k)` (attenzione, la funzione `errore_spline_lineare` non é vettoriale, quindi necessita un `ciclo for`);
5. esegua in una figura i grafici in scala semilogaritmica delle coppie $(2^k, \texttt{eev}(k))$,
6. utilizzi quale titolo della figura la stringa

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

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

   `errori_interpolazione_spline_lineare.txt`

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

Correzione esercizio demo_runge_spline_lineare

```
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=1:9
    N=2^k;
    maxerr=errore_spline_lineare(f,a,b,N);
    eev(k)=maxerr;
end

%--------------- plot ---------------
clf;
semilogy(2.^(1:9),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=[2.^(1:9); eev];
% scrittura dei dati su file.
fprintf(fid,'\n %5.0f  %1.1e',A);
% chiusura file
fclose(fid);
```

Interpolanti spline cubiche in Matlab: esercizio errore_spline_cubica

Si definisca una function

errore_spline_cubica

avente la seguente intestazione

```
function maxerr=errore_spline_cubica(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)))
```

Tale function

- definisca il vettore `x` in cui $x_k = a + (k-1)\cdot h$, $h = (b-a)/N$, $k = 1, \ldots, N+1$ (punti equispaziati, non serve utilizzare il ciclo for!);
- definisca il vettore `y` in cui $y_k = f(x_k)$;
- definisca il vettore `t` in cui $t_k = a + (k-1)\cdot h$, $h = (b-a)/M$, $k = 1, \ldots, M+1$ in cui $M = 100000$ (punti equispaziati, non serve utilizzare il ciclo for!);
- valuti la spline cubica s_3 con condizioni *not-a-knot* interpolante le coppie $(x_k, f(x_k))$, $k = 1, \ldots, N+1$ nei punti test t_k, $k = 1, \ldots, M+1$ e ponga il risultato nel vettore `z` (si utilizzi il comando `spline`);
- valuti la funzione f nei punti test t_k, $k = 1, \ldots, M+1$ e ponga il risultato nel vettore `ftx`;
- calcoli il massimo errore `maxerr`, ovvero $\max_{k=1,\ldots,M+1} |f(t_k) - s_3(t_k)|$.

Interpolanti spline cubiche in Matlab: esercizio demo_runge_spline_cubica

Si definisca una funzione `demo_runge_spline_cubica` che

1. non abbia variabili di input, né di output;
2. definisca la funzione di Runge `f` ove $f(x) = 1/(1+x^2)$, in forma vettoriale;
3. ponga `a=-5` e `b=5`;
4. utilizzando la chiamata

   ```
   maxerr=errore_spline_cubica(f,a,b,N)
   ```

   calcoli per $N = 2, 4, 8, 16, 32, 64, 128, 256, 512 = 2^9$ il valore assunto da `maxerr` e immagazzini il valore ottenuto per $N = 2^k$ nella k-sima componente del vettore `eev`;
5. esegua in una figura i grafici in scala semilogaritmica delle coppie $(2^k, \text{eev}(k))$,
6. utilizzi quale titolo della figura la stringa

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

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

   ```
   errori_interpolazione_spline_cubica.txt
   ```

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

Interpolanti spline in Matlab: esercizi

Gli esercizi proposti e la loro soluzione sono descritti nel documento:

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

Bibliografia I

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