SPLINES *

A. SOMMARIVA † E M.VENTURIN ‡

1. Introduzione. Si è visto che nel caso dell'interpolazione polinomiale, dati $N+1$ punti $a = x_0 < \ldots < x_N = b$, e i valori $y_0, \ldots, y_N$ ivi assunti da una funzione $y = f(x)$, esiste uno ed un solo polinomio p_N di grado N tale che

$$p_N(x_i) = f_i, \; i = 0, \ldots, N. \tag{1.1}$$

Nel caso di nodi equispaziati

$$x_k = a + k \, \frac{(b-a)}{N}, \; k = 0, \ldots, N; \tag{1.2}$$

al crescere di N, non si puó garantire che $f(x) - p_n(x)$ tenda a 0 (si ricordi il fenomeno di Runge!).
Piú in generale per un teorema di Faber (cf. [2, p.132], [6]), qualsiasi sia l'insieme di nodi relativi all'intervallo limitato [a,b] esiste una funzione continua f tale che l'interpolante P_n in tale set di punti non converge uniformemente a f (per $n \to \infty$). In altre parole al tendere di $n \to +\infty$, non si ha

$$\lim_{N \to \infty} \max_{x \in [a,b]} |f(x) - P_N(x)| \to 0.$$

Sorge spontaneo chiedersi se qualora si possegga un gran numero di punti, anche equispaziati, sia possibile calcolare un'approssimante di tipo polinomiale per cui al crescere di N si abbia $p_N \to f$.
Una risposta è stata data nel 1946 da Schoenberg (cf. [16], [7]), lo scopritore delle splines (cf. [12], [13], [14]). Ricordiamo che il termine spline deriva dal termine inglese dato all'omonimo strumento utilizzato da designers [5].
Il primo caso è quello delle splines di grado 1, cioè funzioni che in ogni intervallo $[x_i, x_{i+1}]$ (per $i = 0, \ldots, N-1$) sono polinomi di grado $m = 1$ e globalmente funzioni di classe $C^{m-1} = C^0$, cioè continue.
Il caso generale di splines di grado m risulta piú complicato. Un esempio notevole è quello delle splines cubiche s_3, cioè funzioni che in ogni intervallo $[x_i, x_{i+1}]$ (per $i = 0, \ldots, N-1$) siano polinomi di grado $m = 3$ e globalmente funzioni di classe $C^{m-1} = C^2$.
L'unicitá dell'interpolante è legata (ma non solo!) all'aggiungere alcune proprietá della spline agli estremi x_0, x_N.
Osserviamo infatti che in ogni intervallo $[x_i, x_{i+1}]$ le spline si possano rappresentare come

$$s_3(x) = c_{1,i} + c_{2,i}(x - x_i) + c_{3,i}(x - x_i)^2 + c_{4,i}(x - x_i)^3, \; i = 0, \ldots, N-1$$

e quindi per determinare s_3 in $\{x_i\}_{i=0,\ldots,N}$ servano $4N$ valori $c_{i,j}$. Da ragionamenti sulle proprietá della regolaritá della spline interpolante si vede che sono disponibili solo $4N - 2$ condizioni (di cui $N+1$ dal fatto che $s_3(x_i) = f_i$). Si procede richiedendo quindi una delle seguenti proprietá aggiuntive a s_3:

---

*Ultima revisione: 23 giugno 2010

†DIPARTIMENTO DI MATEMATICA PURA ED APPLICATA, UNIVERSITÀ DEGLI STUDI DI PADOVA, VIA TRIESTE 63, 35121 PADOVA, ITALIA (ALVISE@MATH.UNIPD.IT)

‡DIPARTIMENTO DI INFORMATICA, UNIVERSITÀ DEGLI STUDI DI VERONA, STRADA LE GRAZIE 15, 37134 VERONA, ITALIA (MANOLO.VENTURIN@GMAIL.COM)

FIGURA 1.1. *Isaac Jacob Schoenberg (1903-1990) e un esempio di una spline per designers.*

- *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*: $s_3^{(1)}(a) = f^{(1)}(a),\ s_3^{(1)}(b) = f^{(1)}(b)$.

La spline con vincolo *not-a-knot* forza $s_3^{(3)}$ ad essere continua nel secondo e nel penultimo nodo (e di conseguenza impone che i polinomi nei primi e negli ultimi due intervalli siano uguali). In altre parole, la suddivisione è data dai subintervalli $[x_0, x_2]$, $[x_2, x_3]$, ..., $[x_{N-2}, x_N]$ e si interpolano i dati $y_0, \ldots, y_N$ nei nodi $x_0, \ldots, x_N$. Per convincersi della definizione, dall'help online di Matlab

```
The "not-a-knot" end condition means that, at the first and last
interior break, even the third derivative is continuous (up to
round-off error).
```

Si osservi che, nel caso della condizione *knot-a-knot* se i punti da interpolare sono $x_0, \ldots, x_N$ e gli $N-2$ intervalli della suddivisione sono $[x_0, x_2]$, $[x_2, x_3]$, ..., $[x_{N-2}, x_N]$ allora necessitano $4(N-2)$ condizioni per determinare la spline cubica. Le condizioni dovute all'interpolazione sono $N+1$, mentre quelle dovute alla regolarità sono $3(N-3)$, poichè gli $N-3$ punti che sono estremi di subintervalli *interni* sono $x_2, \ldots, x_{N-2}$.

Per altri tipi di splines si veda [1], p. 170-171 oppure [9].

2. Splines in Matlab ed Octave. Tra le toolbox di Matlab/Octave, si spera di trovare una procedura `spline` che esegua l'interpolante lineare spline. Quindi è naturale digitare dalla shell di Matlab/Octave `help spline`. Si ottiene un'aiuto del tipo

```
SPLINE Cubic spline data interpolation.
    YY = SPLINE(X,Y,XX) uses cubic spline interpolation to find YY,
    the values of the underlying function Y at the points in the
    vector XX.
    The vector X specifies the points at which the data Y is given.
    If Y is a matrix, then the data is taken to be vector-valued and
    interpolation is performed for each column of Y and YY will be
    length(XX)-by-size(Y,2).

    PP = SPLINE(X,Y) returns the piecewise polynomial form of the
    cubic spline interpolant for later use with PPVAL and the
    spline utility UNMKPP.

    Ordinarily, the not-a-knot end conditions are used. However,
    if Y contains two more values than X has entries, then the
```

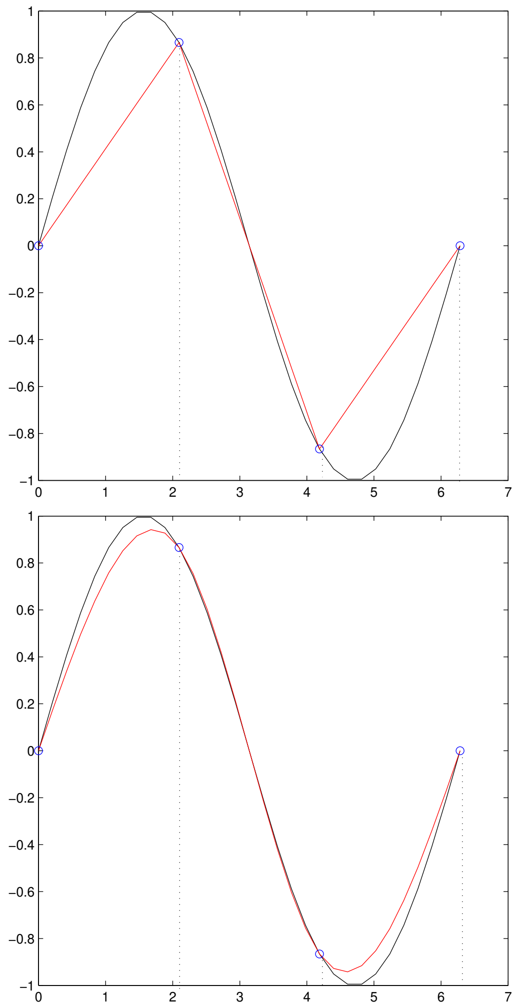

FIGURA 1.2. *Grafico che illustra l'interpolazione su nodi equispaziati nell'intervallo* $[0, 2\pi]$ *della funzione* $\sin(x)$ *mediante una spline lineari a tratti e una spline cubica con vincolo naturale.*

```
first and last value in Y are used as the endslopes for the
cubic spline.  Namely:
   f(X) = Y(:,2:end-1),
   df(min(X)) = Y(:,1),
   df(max(X)) = Y(:,end)

Example:
This generates a sine curve, then samples the spline over a
finer mesh:

    x = 0:10;  y = sin(x);
    xx = 0:.25:10;
    yy = spline(x,y,xx);
    plot(x,y,'o',xx,yy)
```

```
Example:
This illustrates the use of clamped or complete spline inter-
polation where end slopes are prescribed. 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, PPVAL, SPLINES (The Spline Toolbox).
```

L'help è interessante per diversi motivi.

1. Si capisce che la function `spline` non è adatta ai nostri propositi, perchè serve per il calcolo di spline cubiche (e non lineari). Le utilizzeremo nella prossima lezione.

2. Il primo esempio ci dice come svolgere parte dell'esercizio. Consideriamo la porzione di codice

   ```
   x = 0:10;  y = sin(x);
   xx = 0:.25:10;
   yy = spline(x,y,xx);
   plot(x,y,'o',xx,yy)
   ```

   Si capisce subito che valuta su una griglia di numeri naturali $x = 0, \ldots, 10$ la funzione $\sin(x)$, la approssima con spline (cubiche!) interpolanti in tale insieme di ascisse $(x_i, \sin(x_i))$. Proprio quello che vogliamo fare con le spline lineari a tratti (se solo avessimo la routine adatta e cambiassimo il set di ascisse, adattandolo al nostro caso).

3. Suggerisce di guardare `interp1`. Proviamo su Matlab/Octave `help interp1`. Con un po' di fatica capiamo che pare essere la routine giusta.

In una versione abbastanza recente di Matlab troviamo la seguente descrizione

```
INTERP1 1-D interpolation (table lookup).
   YI = INTERP1(X,Y,XI) interpolates to find YI, the values of
   the underlying function Y at the points in the vector XI.
   The vector X specifies the points at which the data Y is
   given. If Y is a matrix, then the interpolation is performed
   for each column of Y and YI will be length(XI)-by-size(Y,2).

   YI = INTERP1(Y,XI) assumes X = 1:N, where N is the length(Y)
   for vector Y or SIZE(Y,1) for matrix Y.

   Interpolation is the same operation as "table lookup".
   Described in "table lookup" terms, the "table" is [X,Y]
   and INTERP1 "looks-up" the elements of XI in X, and, based
   upon their location, returns values YI interpolated within
   the elements of Y.

   YI = INTERP1(X,Y,XI,'method') specifies alternate methods.
   The default is linear interpolation.  Available methods are:
```

```
  'nearest'   - nearest neighbor interpolation
  'linear'    - linear interpolation
  'spline'    - piecewise cubic spline interpolation (SPLINE)
  'pchip'     - piecewise cubic Hermite interpolation (PCHIP)
  'cubic'     - same as 'pchip'
  'v5cubic'   - the cubic interpolation from MATLAB 5, which
                does not extrapolate and uses 'spline' if X
                is not equally spaced.

YI = INTERP1(X,Y,XI,'method','extrap') uses the specified
method for extrapolation for any elements of XI outside the
interval spanned by X.
Alternatively, YI = INTERP1(X,Y,XI,'method',EXTRAPVAL) replaces
these values with EXTRAPVAL.  NaN and 0 are often used for
EXTRAPVAL.
The default extrapolation behavior with four input arguments is
'extrap' for 'spline' and 'pchip' and EXTRAPVAL = NaN for the
other methods.

For example, generate a coarse sine curve and interpolate over a
finer abscissa:
    x = 0:10; y = sin(x); xi = 0:.25:10;
    yi = interp1(x,y,xi); plot(x,y,'o',xi,yi)

See also INTERP1Q, INTERPFT, SPLINE, INTERP2, INTERP3, INTERPN.
```

Con un po' di fatica capiamo che la routine che fa al caso nostro è

```
v = interp1(x,y,u,'linear')
```

dove

1. `x`, `y` sono rispettivamente i vettori di ascisse e ordinate dei punti in cui si desidera interpolare; in altre parole si cerca la funzione lineare a tratti s_1 tale che $s_1(x_i) = y_i$.

2. se `u` un insieme di ascisse prescelto dall'utente (ad esempio per plottare s_1) allora $v_i = s_1(u_i)$.

Per quanto visto deduciamo che in Matlab/Octave, i problemi dell'interpolazione spline sono semplificati dall'utilizzo, dei comandi `spline` e `interp1`.
Più precisamente:

1. Per quanto riguarda la spline cubica (not a knot) che interpola le coppie (x_i, y_i), il suo valore v_i nei punti u_i è dato da

   ```
   v=spline(x,y,u).
   ```

2. Simili risultati possono essere raggiunti dal comando `interp1`. Cosí

   ```
   v = interp1(x,y,u,'linear');
   ```

   ```
   v = interp1(x,y,u,'spline');
   ```

```
v = interp1(x,y,u,'cubic');
```

forniscono i valori v_i dell'interpolante lineare, cubica not a knot e cubica Hermite nei punti u_i.
Rimandiamo i dettagli sul loro utilizzo all'help in linea e agli esempi allegati al corso.

2.0.1. Errore dell'interpolante spline lineare e cubica. Sia $s_1 : [a, b] \to \mathbb{R}$ una spline di grado 1, che interpola le coppie $(x_i, f(x_i))$ dove

$$x_0 = a < x_1 < \ldots < x_i < x_{i+1} < \ldots < x_N = b.$$

Dal teorema dell'errore dell'interpolazione polinomiale nel caso $N = 1$ abbiamo che *puntualmente*

$$f(x) - p_N(x) = f^{(2)}(\xi)\frac{(x - x_0)(x - x_1)}{2},\ \xi \in (a, b) \tag{2.1}$$

da cui è immediato avere una stima d'errore (*uniforme*!) per funzioni $f \in C^2([a, b])$

$$|f(x) - s_1(x)| \leq h_i^2 \frac{M_i}{8} \tag{2.2}$$

per

$$h_i = (x_{i+1} - x_i),\ i = 0, \ldots, N - 1,$$

$$M_i = \max_{x \in (x_i, x_{i+1})} |f^{(2)}(x)|.$$

Infatti, se $x \in [x_i, x_{i+1}]$, da (2.1)

$$|f(x) - s_1(x)| = \left| f^{(2)}(\xi)\frac{(x - x_i)(x - x_{i+1})}{2} \right| \tag{2.3}$$

$$= \left| f^{(2)}(\xi) \right| \left| \frac{(x - x_i)(x - x_{i+1})}{2} \right| \tag{2.4}$$

$$\leq \max_{x \in [x_i, x_{i+1}]} |f^{(2)}(x)| \cdot \max_{x \in [x_i, x_{i+1}]} \left| \frac{(x - x_i)(x - x_{i+1})}{2} \right| \tag{2.5}$$

da cui l'asserto in quanto il massimo di

$$\left| \frac{(x - x_i)(x - x_{i+1})}{2} \right|$$

si ha per $c = (x_i + x_{i+1})/2$ e quindi

$$\left| \frac{(x - x_i)(x - x_{i+1})}{2} \right| \leq \left| \frac{(c - x_i)(c - x_{i+1})}{2} \right| = \frac{1}{2} \left| \frac{(x_{i+1} - x_i)}{2} \frac{(x_i - x_{i+1})}{2} \right| = \frac{h_i^2}{8}.$$

Analizziamo ora l'errore effettuato da una spline cubica $s_3 : [a, b] \to \mathbb{R}$ che interpola le coppie $(x_i, f(x_i))$ dove

$$x_0 = a < x_1 < \ldots < x_i < x_{i+1} < \ldots < x_N = b.$$

E' utile ricordare che se $f : [a, b] \to \mathbb{R}$ è una funzione continua, allora

$$\|f\|_{\infty} := \max_{x \in [a,b]} |f(x)|.$$

Per quanto riguarda l'errore, si può provare (non facile!) quanto segue (cf. [2], p. 163, [1], p. 163)

TEOREMA 2.1. *Supponiamo $f \in C^4([a, b])$. Posto $h = \max h_i$, si consideri una successione di suddivisioni tale che esista una costante K tale che*

$$\frac{h}{h_j} \leq K, \ j = 0, \ldots, N-1.$$

Allora esiste una costante c_k indipendente da h tale che

$$\|f^{(k)} - s^{(k)}_{3,\Delta_N}\|_{\infty} \leq c_k K h^{(4-k)} ||f^{(4)}||_{\infty}, \ k = 0, 1, 2, 3 \tag{2.6}$$

dove $\|f\|_{\infty} = \max_{x \in [a,b]} |f(x)|$, $\Delta_N = \{x_i\}_{i=0,\ldots,N}$.

COROLLARIO 2.2. *Supponiamo $f \in C^4([a, b])$. Fissata un'ampiezza $h = (b-a)/N$, si consideri la suddivisione di $[a, b]$ data da $\cup_{i=0}^{N-1} [a + ih, a + (i+1)h]$.*
Allora esistono delle costanti c_0, c_1, c_2, c_3 indipendenti da h tale che

$$\|f - s_{3,\Delta_N}\|_{\infty} \leq c_0 \, h^4 ||f^{(4)}||_{\infty}, \tag{2.7}$$

$$\|f^{(1)} - s^{(1)}_{3,\Delta_N}\|_{\infty} \leq c_1 \, h^3 ||f^{(4)}||_{\infty}, \tag{2.8}$$

$$\|f^{(2)} - s^{(2)}_{3,\Delta_N}\|_{\infty} \leq c_2 \, h^2 ||f^{(4)}||_{\infty}, \tag{2.9}$$

$$\|f^{(3)} - s^{(3)}_{3,\Delta_N}\|_{\infty} \leq c_3 \, h^1 ||f^{(4)}||_{\infty}, \tag{2.10}$$

dove $\|f\|_{\infty} = \max_{x \in [a,b]} |f(x)|$, $\Delta_N = \{x_i\}_{i=0,\ldots,N}$.

Nel caso di spline interpolanti e vincolate si ha $c_0 = 5/384$, $c_1 = 1/24$, $c_2 = 3/8$. Nel caso di splines di tipo *Hermite* si ha $c_0 = 1/384$ (cf. [11], p. 301).

La condizione su h nel teorema è chiaramente verificata per suddivisioni uniformi, ed in particolare esclude che alcuni intervalli possano degenerare ad un punto. Si noti non solo come la spline approssima la funzione, ma come pure le sue derivate convergano alle rispettive derivate della funzione f.

3. Esercizi sull'interpolazione spline lineare a tratti. Fissato $N \in \mathcal{N}$, $N \geq 2$, si calcoli l'interpolante spline lineare s della funzione `runge` nei nodi

$$x_k = -5 : h_x : 5, \ k = 0, \ldots, N$$

dove $h_x = 10/N$. Confrontando il valore dell'interpolante spline con quello della funzione di Runge nei nodi

$$u_k = -5 : h_u : 5, \ k = 0, \ldots, 4\,N$$

con $h_u = hx/4$ e utilizzando il comando `norm` (si consulti l'help!) per il calcolo della norma $\|\cdot\|_{\infty}$ di un vettore, si valuti l'errore compiuto in norma infinito. Ricordiamo che se $v = \{v_i\}$ allora

$$\|v\|_{\infty} = \max_i |v_i|.$$

Nel caso della spline lineare, é precisa la classica stima dell'errore della spline lineare a tratti?

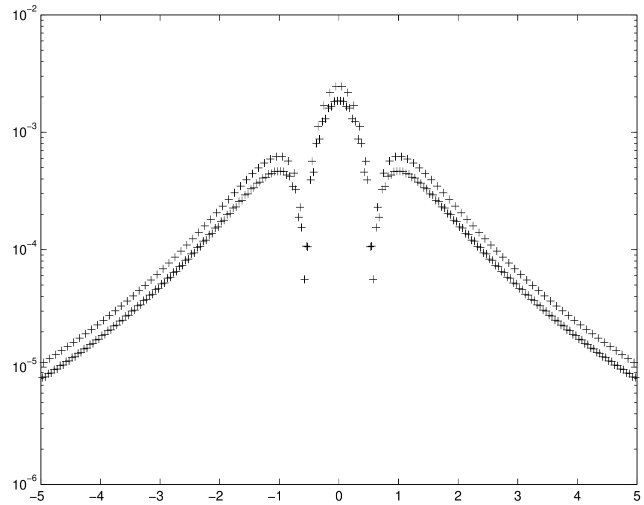

FIGURA 3.1. *Grafico che illustra l'errore della spline lineare a tratti nell'approssimare la funzione di Runge nell'intervallo* $[-5, 5]$*, con intervalli di ampiezza* $1/10$*.*

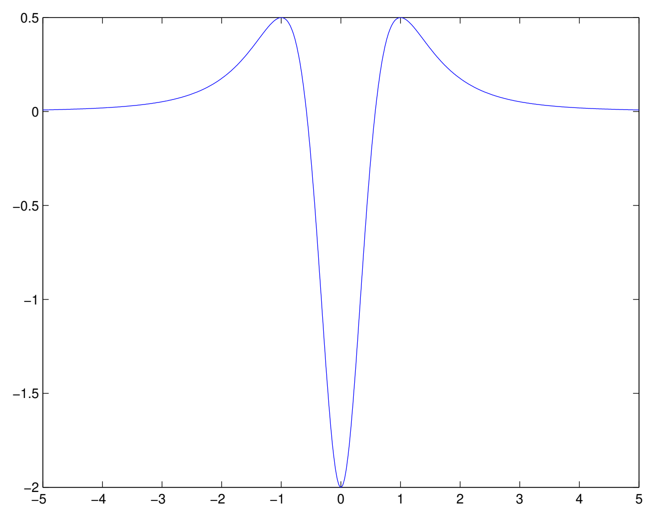

FIGURA 3.2. *Grafico che illustra il grafico della derivata seconda della funzione di Runge.*

In virtù di precedenti osservazioni, scriviamo in Matlab nel file `splinelineare.m`

```
N=100; a=-5; b=5;
hx=(b-a)/N;        % PASSO INTP. PTS.
x=a:hx:b;          % ASCISSE (INTP).
y=runge(x);        % ORDINATE (INTP).
h1=hx/4;           % PASSO TEST PTS.
u=a:h1:b;          % ASCISSE (TEST).
v = interp1(x,y,u,'linear');  % ORDINATE (TEST).
vv=runge(u);       % VALORE ESATTO IN NODI TEST.
err=norm(vv-v,inf);    % ERRORE ASSOLUTO IN
                       % NORMA INFINITO.
```

```
fprintf('\n \t [ERR]: %2.2e',err);
plot(u,v,'k-',u,vv,'r-');  % PLOT SPLINE VS. RUNGE.
fprintf('\n \t [PAUSE]');
pause(5);
semilogy(u,abs(vv-v),'+');  % PLOT ERRORE.
```

Osserviamo dall'esperimento che la funzione è ricostruita così bene, che l'interpolante e la funzione di Runge si sovrappongono in modo indistinguibile. Il plot dell'errore assoluto, che in virtù del comando di pausa viene eseguito dopo 5 secondi, mostra che il massimo errore è compiuto nell'origine.
La stima d'errore delle splines lineari a tratti per funzioni $f \in C^2([a,b])$ è

$$|f(x) - s_1(x)| \leq h_i^2 \frac{M_i}{8} \tag{3.1}$$

dove

$$h_i = (x_{i+1} - x_i),\ i = 0, \ldots, N-1,$$

$$M_i = \max_{x \in (x_i, x_{i+1})} |f^{(2)}(x)|.$$

Osserviamo ora che

1. Nel caso della funzione di Runge f si ha

$$f^{(2)}(x) = (8\,x^2)/(x^2+1)^3 - 2/(x^2+1)^2.$$

2. Nel nostro esempio, l'ampiezza dell'intervallo è costante e vale $h_i = h$.

Quindi se $x \in [x_i, x_{i+1}]$, allora

$$|f(x) - s_1(x)| \leq h^2 \frac{M_i}{8} \tag{3.2}$$

Si esegua un grafico di $f^{(2)}$ in $[-5, 5]$ e trovato il suo massimo in valore assoluto, diciamo M, si verifichi che `err` del programma sopra introdotto sia inferiore a $h^2 \frac{M}{8}$. É una buona stima?

Facoltativo: se

$$M_i = \max_{x \in (x_i, x_{i+1})} |f^{(2)}(x)|$$

la stima sopracitata è buona in ogni subintervallo $[x_i, x_{i+1}]$? Verificarlo mediante un plot di $h^2 \frac{M_i}{8}$ relativamente alla funzione di Runge f, eseguendo un confronto con l'errore `abs(vv-v)`. Motivare qualitativamente perchè il massimo errore compiuto dalla spline è nell'origine. A tal proposito si consideri il grafico della derivata seconda della funzione di Runge e (3.2).

4. Confronto tra interpolazione spline lineare e cubica. In questa sezione faremo un breve confronto tra una spline lineare e una cubica. Consideriamo la funzione $f(x) := \log(x)$ nell'intervallo $[a,b] := [1,2]$ e suddividiamo $[a,b]$ in N subintervalli. Quindi calcoliamo per $N = 3, 4, \ldots, 10$ le interpolanti splines lineari a tratti, cubiche con vincoli *naturali* o *not a knot*. Qualora possibile stimiamo l'errore commesso.
Introduciamo quindi il seguente codice, che salveremo in un file *splinecf.m*:

```
a=1; b=2; int_type=1;

switch int_type
case 1
    fprintf('\n \t [SPLINE LINEARE] \n');
case 2
    fprintf('\n \t [SPLINE CUBICA NATURALE] \n');
case 3
    fprintf('\n \t [SPLINE CUBICA NOT A KNOT] \n');
end

for N=3:10
    h=(b-a)/N;
    x=a:h:b;
    y=log(x);
    s=a:(h/10):b;

    switch int_type
    case 1 % LINEARE.
        t=interp1(x,y,s,'linear');
        err(N)=norm(t-log(s),inf);
        err_ext_min=(2/(b^2))*(h^2)/8;
        err_ext_max=(2/(a^2))*(h^2)/8;
        fprintf('\n [N]: %2.0f [ERR. (INF)] %2.2e',N, err(N));
        fprintf(' [EST.][MIN]: %2.2e [MAX]: %2.2e',err_ext_min,err_ext_max);
        if (err(N) > err_ext_min) & (err(N) < err_ext_max)
          fprintf(' [OK]');
      else
          fprintf(' [KO]');
        end
    case 2 % CUBICA NATURALE.
        pp=csape(x,y,'variational');
        t = ppval(pp,s);
        err(N)=norm(t-log(s),inf);
        fprintf('\n [N]: %2.0f [ERR. (INF)] %2.2e',N,err(N));
    case 3 % CUBICA NOT A KNOT.
        t=spline(x,y,s);
        err(N)=norm(t-log(s),inf);
        fprintf('\n [N]: %2.0f [ERR. (INF)] %2.2e',N,err(N));
    end

    plot(s,log(s),'k-'); hold on;
    plot(s,t,'r-');


    pause(2);
    hold off;
end
```

1. Dal punto di vista della stima dell'errore nel caso di spline lineari, ricordiamo che la derivata seconda di $f(x) := \log(x)$ è $f^{(2)}(x) = -1/x^2$ e quindi $|f^{(2)}(x)| = |-1/x^2| = 1/x^2$. Di conseguenza, essendo $|f^{(2)}(x)|$ decrescente, si vede subito

che

$$1/b^2 < |f^{(2)}(\xi)| < 1/a^2$$

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

2. Non c'è molto da commentare nel codice eccetto che per la implementazione delle spline cubiche naturali. A tal proposito abbiamo usato il comando

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

Per capire il significato di `csape` ci aiutiamo, come al solito, con l'help di Matlab/Octave relativamente a *csape*.

```
CSAPE Cubic spline interpolation with various end-conditions.

   PP  = CSAPE(X,Y)

   returns the cubic spline interpolant (in ppform) to the given
   data (X,Y) using Lagrange end-conditions (see default in table below).

   PP  = CSAPE(X,Y,CONDS) uses the end-conditions specified in CONDS, with
   default values (which depend on the particular conditions).

   PP  = CSAPE(X,Y,CONDS,VALCONDS) uses the end-conditions specified
   in CONDS, with particular values as specified in VALCONDS.

   CONDS may be a *string* whose first character matches one of the
   following: 'complete' or 'clamped', 'not-a-knot', 'periodic',
   'second', 'variational', with the following meanings:

   'complete'    : match endslopes (as given in VALCONDS, with
                   default as under *default*)
   'not-a-knot'  : make spline C^3 across first and last interior
                   break (ignoring VALCONDS if given)
   'periodic'    : match first and second derivatives at first data
                   point with those at last data point
                   (ignoring VALCONDS if given)
   'second'      : match end second derivatives (as given in VALCONDS,
                   with default [0 0], i.e., as in variational)
   'variational' : set end second derivatives equal to zero
                   (ignoring VALCONDS if given)
   The *default* : match endslopes to the slope of the cubic that
                   matches the first four data at the respective end.
   ... ...

     x = 0:4; y=-2:2; s2 = 1/sqrt(2);
      clear v
      v(3,:,:) = [0 1 s2 0 -s2 -1 0].'*[1 1 1 1 1];
      v(2,:,:) = [1 0 s2 1 s2 0 -1].'*[0 1 0 -1 0];
      v(1,:,:) = [1 0 s2 1 s2 0 -1].'*[1 0 -1 0 1];
      sph = csape({x,y},v,{'clamped','periodic'});
      values = fnval(sph,{0:.1:4,-2:.1:2});
    surf(squeeze(values(1,:,:)),squeeze(values(2,:,:)),squeeze(values(3,:,:)))
```

```
    % the previous two lines could have been replaced by: fnplt(sph)
    axis equal, axis off

 See also CSAPI, SPAPI, SPLINE.
```

L'uso di `fnval` crea problemi nelle vecchie releases di Matlab ed Octave ed è stato sostituito con `ppval`.

Pur non essendo chiarissimo, si capisce che

```
pp  = csape(x,y,conds)
```

serve per costruire la spline cubica con vincoli che devono essere descritti nella variabile *conds* come *'variational'* se si desidera la spline naturale. Dal breve esempio (che essendo bidimensionale, non ci interessa) si capisce che *fnval* serve per valutare la spline definita da *pp* nei nodi *s*.

Vediamo i risultati al variare del parametro *inttype* tra 1 e 3 (analizzando quindi nell'ordine le interpolanti splines lineari a tratti, cubiche con vincoli *naturali* o *not a knot*.).

```
[SPLINE LINEARE]

[N]:  3 [ERR. (INF)] 1.03e-002 [EST.][MIN]: 6.94e-003 [MAX]: 2.78e-002 [OK]
[N]:  4 [ERR. (INF)] 6.21e-003 [EST.][MIN]: 3.91e-003 [MAX]: 1.56e-002 [OK]
[N]:  5 [ERR. (INF)] 4.15e-003 [EST.][MIN]: 2.50e-003 [MAX]: 1.00e-002 [OK]
[N]:  6 [ERR. (INF)] 2.97e-003 [EST.][MIN]: 1.74e-003 [MAX]: 6.94e-003 [OK]
[N]:  7 [ERR. (INF)] 2.23e-003 [EST.][MIN]: 1.28e-003 [MAX]: 5.10e-003 [OK]
[N]:  8 [ERR. (INF)] 1.73e-003 [EST.][MIN]: 9.77e-004 [MAX]: 3.91e-003 [OK]
[N]:  9 [ERR. (INF)] 1.39e-003 [EST.][MIN]: 7.72e-004 [MAX]: 3.09e-003 [OK]
[N]: 10 [ERR. (INF)] 1.14e-003 [EST.][MIN]: 6.25e-004 [MAX]: 2.50e-003 [OK]

[SPLINE CUBICA NATURALE]

[N]:  3 [ERR. (INF)] 5.22e-003
[N]:  4 [ERR. (INF)] 2.93e-003
[N]:  5 [ERR. (INF)] 1.91e-003
[N]:  6 [ERR. (INF)] 1.34e-003
[N]:  7 [ERR. (INF)] 9.86e-004
[N]:  8 [ERR. (INF)] 7.57e-004
[N]:  9 [ERR. (INF)] 6.00e-004
[N]: 10 [ERR. (INF)] 4.86e-004

[SPLINE CUBICA NOT A KNOT]

[N]:  3 [ERR. (INF)] 8.28e-004
[N]:  4 [ERR. (INF)] 2.63e-004
[N]:  5 [ERR. (INF)] 1.29e-004
[N]:  6 [ERR. (INF)] 6.94e-005
[N]:  7 [ERR. (INF)] 4.07e-005
[N]:  8 [ERR. (INF)] 2.54e-005
[N]:  9 [ERR. (INF)] 1.67e-005
[N]: 10 [ERR. (INF)] 1.14e-005
```

5. Facoltativo: Un esempio sull'utilizzo delle spline per approssimare una curva. Sia una funzione $f : [a, b] \to \mathbb{R}^2$, definita come

$$f(t) = \left(\begin{array}{c} u(t) \\ v(t) \end{array} \right) \tag{5.1}$$

con $u, v : [a, b] \to \mathbb{R}$. Noti i valori $u(t_i)$, $v(t_i)$ per $i = 0, \ldots, N$ si possono calcolare le rispettive interpolanti splines s_u, s_v e quindi approssimare f con $\tilde{f}$ dove

$$\tilde{f}(t) = \left(\begin{array}{c} s_u(t) \\ s_v(t) \end{array} \right) \tag{5.2}$$

Un esempio interessante è l'approssimazione del cerchio con splines. Essendo l'equazione del cerchio

$$f(t) = \left(\begin{array}{c} \cos(t) \\ \sin(t) \end{array} \right) \tag{5.3}$$

per $t = [0, 2\,\pi]$. Se

$$t_k = \frac{2k\pi}{N},\ k = 0, \ldots, N,$$

l'interpolante spline puó essere facilmente calcolata via il comando Matlab `interp1` applicato alla coppia di vettori `t`, `u` e `t`, `v` dove al solito $t = (t_k)$, $u = (u(t_k))$, $v = (v(t_k))$.
Per testare il comportamento di tale ricostruzione si utilizzi il seguente codice `parmspline.m` per $N = 4, 8, 12, 16, 20$ (nel nostro caso eseguiamo il caso $N = 6$)

```
N=6;
a=-pi; b=pi;      % INTERVALLO.
h=(b-a)/N;        % PASSO PARAMETRI.

% PUNTI INTP.
t=a:h:b;          % PARAMETRI CURVA INTP.
xt=cos(t);        % ASCISSA PTO IN CURVA.
yt=sin(t);        % ORDINATA PTO IN CURVA.

% TEST POINTS.
htest=h/100;        % PASSO PARAMETRO t.
ttest=a:htest:b;    % PARAMETRI CURVA TEST.

% SPLINES INTP. DELLA CURVA PARAMETRICA.
xttest=interp1(t,xt,ttest,'spline');
yttest=interp1(t,yt,ttest,'spline');

% PLOT.
theta=0:htest:2*pi;
plot(xttest,yttest,'k-',cos(theta),sin(theta),'r-');
```

Osserviamo che il plot del cerchio può essere *ovalizzato* dalla parte grafica (come il plot di Matlab o il GNUPLOT per Octave). Aiutandosi con l'help di Matlab cercare quali parametri possono sostituire `'spline'` nel comando `'interp1'` e provarli, verificando le differenze nella ricostruzione del cerchio per $N = 8$.
Per ulteriori approfondimenti, si considerino [2], [10]. Le equazioni parametriche sono tratte da [8].

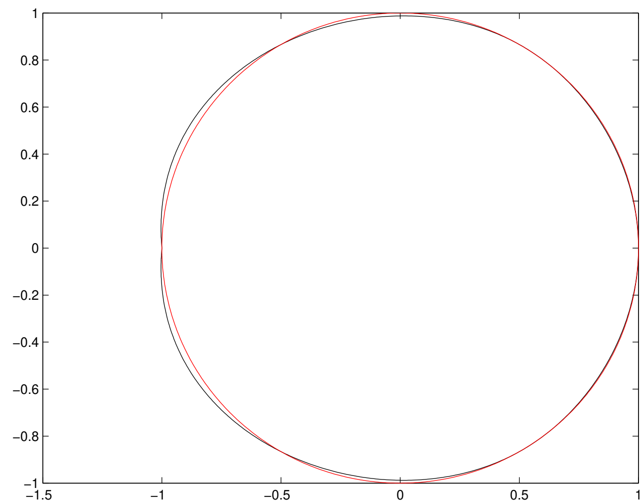

FIGURA 5.1. *Ricostruzione del cerchio con spline cubiche parametriche, per $N = 6$ in* `parmspline.m`.

6. Facoltativo: Calcolo della derivata con splines. Si verifichi, mediante grafici, che la derivata dell'interpolazione di tipo spline cubica della funzione $f(x) = \sin(10\,x)$ su $n = 11$ nodi equispaziati nell'intervallo $[0, 1]$ approssima, senza interpolarla, la derivata della funzione $f(x)$.

6.1. Commenti all'esercitazione. Occorre usare la function `spline`. Tanto in Matlab quanto in Octave, la sintassi del comando è la `pp=spline(X,Y)` dove `pp` è una struttura dati contenente, tra l'altro, `pp.P`, una matrice di ordine $(n-1) \times 4$ in cui alla riga k-sima sono immagazzinati i coefficienti $a_{k,3}, \ldots, a_{k,0}$ della restrizione della spline (polinomio $a_{k,3}x^3 + \ldots + a_{k,0}$) all'intervallo I_k. Occorre quindi creare una nuova struttura (diciamo `pp1`), del tipo di `pp`, contenente la matrice `pp.P1` di ordine $(n-1) \times 3$ dei coefficienti $b_{k,2} = 3 * a_{k,3}, \ldots, b_{k,0} = a_{k,1}$ delle derivate dei polinomi. Si osservi che la derivata di

$$a_{k,3}\,x^3 + \ldots + a_{k,0}$$

è

$$3 * a_{k,3}\,x^2 + \ldots + a_{k,1}$$

e quindi se la matrice `pp.P` è

$$pp.P = \begin{pmatrix} a_{1,3} & a_{1,2} & a_{1,1} & a_{1,0} \\ a_{2,3} & a_{2,2} & a_{2,1} & a_{2,0} \\ a_{3,3} & a_{3,2} & a_{3,1} & a_{3,0} \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

necessariamente

$$pp.P1 = \begin{pmatrix} 0 & 3 \cdot a_{1,2} & 2 \cdot a_{1,1} & a_{1,0} \\ 0 & 3 \cdot a_{2,2} & 2 \cdot a_{2,1} & a_{2,0} \\ 0 & 3 \cdot a_{3,2} & 2 \cdot a_{3,1} & a_{3,0} \\ 0 & \ldots & \ldots & \ldots \end{pmatrix}$$

Il vettore *pp* ottenuto dal comando `pp=spline(X,Y)` contiene varie informazioni. Un tipico esempio:

```
      form: 'pp'
    breaks: [1x11 double]
     coefs: [10x4 double]
    pieces: 10
     order: 4
       dim: 1
```

Per disporre dei breaks e dei coefficienti, si esegue il comando

```
[breaks,coeffs]=unmkpp(pp)
```

Se il file `unmkpp` non è disponibile tra le functions di Matlab, si copi il seguente codice

```
function [x, P, n, k, d] = unmkpp(pp)
  if nargin == 0
    usage("[x, P, n, k, d] = unmkpp(pp)")
  endif
  if !isstruct(pp)
    error("unmkpp: expecting piecewise polynomial structure");
  endif
  x = pp.x;
  P = pp.P;
  n = pp.n;
  k = pp.k;
  d = pp.d;
endfunction
```

o lo si scarichi dal sito http://users.powernet.co.uk/kienzle/octave/matcompat/scripts/datafun/

Alternativamente provare un comando del tipo

```
coeffs = pp.P;
```

Una volta modificata la matrice di coefficienti `coeffs` in `dcoeffs` per il calcolo delle derivate bisogna ricostruire una struttura dati adatta alle spline e quindi simile a `coeffs`. A tal proposito basta eseguire il comando

```
pp1=mkpp(breaks,dcoeffs)
```

Per valutare i polinomi a tratti, si usa la funzione `ppval` valutata su `pp1`. Si può produrre un grafico di tutto il dominio $[0, 1]$ per valutare la bontà dell'approssimazione della derivata e di un intorno di un nodo di interpolazione per valutare il carattere non interpolatorio.
Un codice Matlab che svolge quanto chiesto è il seguente

```
a=0; b=1;

% Interpolazione con splines.
h=(b-a)/10;
x=a:h:b;
y=sin(10*x);
pp=spline(x,y);
```

```
[breaks,coeffs]=unmkpp(pp);

% Derivata con splines.
N=size(coeffs,1);
dcoeffs=[zeros(N,1) 3*coeffs(:,1) 2*coeffs(:,2) coeffs(:,3)];
pp1=mkpp(breaks,dcoeffs);
u=a:0.005:b;
dsu=ppval(pp1,u);
dfu=10*cos(10*u);

% Plot errore assoluto.
semilogy(u,abs(dsu-dfu),'-');
```

7. Online. Qualora si vogliano ulteriori delucidazioni si confrontino i seguenti links

1. http://it.wikipedia.org/wiki/Funzione_spline
2. http://en.wikipedia.org/wiki/Spline_%28mathematics%29
3. http://it.wikipedia.org/wiki/Interpolazione_spline
4. http://www.dmi.units.it/~bellen/calcolo_numerico/CAPIT-5A.PDF

Per chi volesse approfondire l'uso delle splines con Matlab, si suggerisce la monografia gratuita *Spline Toolbox User's guide*, scaricabile al sito

http://www.mathworks.com/access/helpdesk/help/pdf_doc/splines/splines.pdf

Molto interessante la demo Matlab

http://www.mathworks.com/access/helpdesk/help/toolbox/splines/splines.shtml

consultabile in molte versioni di Matlab digitando

```
>> demo toolbox splines
>>
```

RIFERIMENTI BIBLIOGRAFICI

[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] Carl DeBoor, *Spline Toolbox User's guide*,
http://www.mathworks.com/access/helpdesk/help/pdf_doc/splines/splines.pdf.
[5] Carl DeBoor, *personal homepage*,
http://pages.cs.wisc.edu/~deboor/draftspline.html.
[6] Mac Tutor, Georg Faber,
http://www-history.mcs.st-andrews.ac.uk/Biographies/Faber.html.
[7] Mac Tutor, Isaac Jacob Schoenberg,
http://www-history.mcs.st-andrews.ac.uk/Biographies/Schoenberg.html.
[8] The MathWorks Inc., *Numerical Computing with Matlab*,
http://www.mathworks.com/moler
[9] The MathWorks Inc., *Spline Toolbox*,
http://www.mathworks.com/products/splines/demos.html?file=/products/demos/shipping/splines/csapidem.html
[10] A. Quarteroni e F. Saleri, *Introduzione al calcolo scientifico*, Springer Verlag, 2006.
[11] A. Suli e D. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.
[12] Wikipedia, Funzione Spline,
http://it.wikipedia.org/wiki/Funzione_spline.

[13] Wikipedia, Spline Device,
http://en.wikipedia.org/wiki/Spline_%28device%29.
[14] Wikipedia, Spline Mathematics,
http://en.wikipedia.org/wiki/Spline_%28mathematics%29.
[15] Wikipedia, Interpolazione Spline,
http://it.wikipedia.org/wiki/Interpolazione_spline.
[16] Wikipedia, Isaac Jacob Schoenberg,
http://en.wikipedia.org/wiki/Isaac_Jacob_Schoenberg.