Esercitazione 1

Richiami di Teoria: Spline

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

$$p_N(x_i) = f_i,\ i = 0, \ldots, N. \tag{1}$$

Nel caso di nodi equispaziati

$$x_k = a + k\,\frac{(b-a)}{N},\ k = 0, \ldots, N; \tag{2}$$

al crescere di N, non si può garantire (in generale) che $||f(x) - p_n(x)||_\infty$ tenda a 0 (si ricordi il fenomeno di Runge!).

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, lo scopritore delle splines. Le spline sono particolari curve polinomiali a tratti, costituite da archi di curva che si succedono in sequenza, la cui espressione analitica si ottiene imponendo una serie di condizioni di continuità.
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 più 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 $C^{m-1} = C^2$, quindi con stessa tangente e curvatura.
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 proprietà della regolarità 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 proprietà aggiuntive a s_3:

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

Le spline in Matlab

La routine di interpolazione polinomiale a tratti in Matlab si chiama `interp1` La sintassi è la seguente:

```
yi = interp1(x,y,xi,tipo)
```

`x,y` specificano le coordinate dei punti di interpolazione.
`xi` sono i punti di campionamento per la valutazione dell'interpolante.
`tipo` è una stringa di caratteri che specifica il metodo di interpolazione:
`'nearest'` : interpolazione con polinomio costante a tratti
`'linear'` : polinomio lineare a tratti (default)
`'cubic'` : polinomio cubico a tratti (con derivate continue)
`'spline'` : interpolazione con spline cubica

Si consideri il seguente

```
clear all, close all
f=inline('exp(x).*cos(4*x)');
x=linspace(0,3,10);     % punti di interpolazione
y=f(x);
xi=linspace(0,3,300);   % punti di campionamento per la valutazione
fxi=f(xi);              % dell'interpolante

yinear=interp1(x,y,xi,'nearest');
yilin=interp1(x,y,xi,'linear');
yicub=interp1(x,y,xi,'cubic');
yispl=interp1(x,y,xi,'spline');

subplot(2,2,1);
plot(xi,fxi,'Linewidth',2); hold on; plot(xi,yinear,'m','Linewidth',2); ...
    plot(x,y,'r*','Linewidth',2); title('int. costante a tratti');
subplot(2,2,2);
plot(xi,fxi,'Linewidth',2); hold on; plot(xi,yilin,'m','Linewidth',2); ...
    plot(x,y,'r*','Linewidth',2); title('int. lineare a tratti');
subplot(2,2,3);
plot(xi,fxi,'Linewidth',2); hold on; plot(xi,yicub,'m','Linewidth',2); ...
    plot(x,y,'r*','Linewidth',2); title('int. cubica a tratti');
subplot(2,2,4);
plot(xi,fxi,'Linewidth',2); hold on; plot(xi,yispl,'m','Linewidth',2); ...
    plot(x,y,'r*','Linewidth',2); title('int. spline cubica');
```

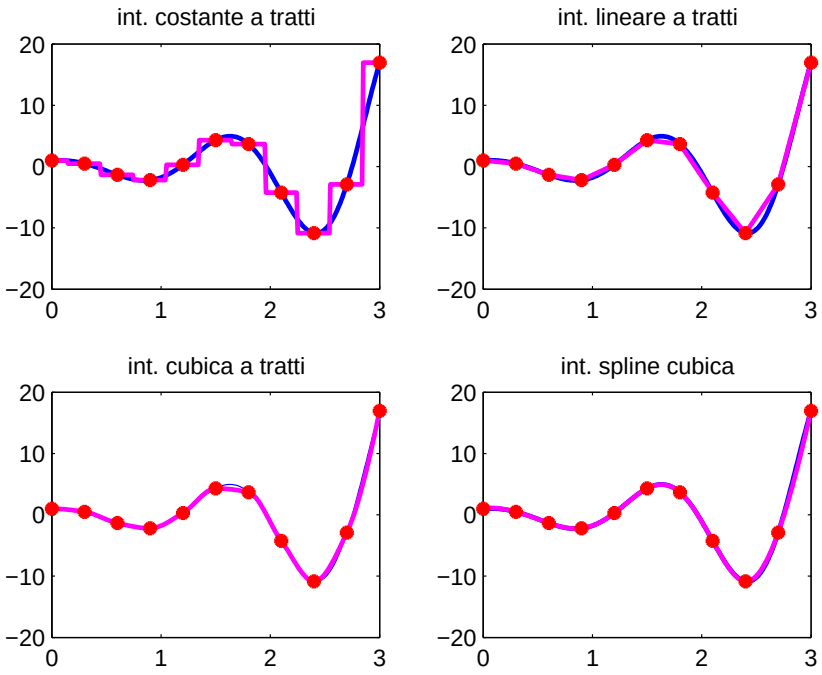


Figura 1: Interpolazione polinomiale con `interp1`

Abbiamo visto che esistono diversi tipi di spline cubiche interpolanti: naturali, vincolate, periodiche, not-a-knot. Con Matlab si calcolano facilmente le spline cubiche interpolanti attraverso la funzione predefinita `spline`:

```
s=spline(x,y,t)
```

dove x,y sono i vettori contenenti le coordinate dei punti da interpolare mentre t è un vettore di punti di campionamento nell'intervallo $[x_1, x_n]$ in cui valutare la spline cubica interpolante ad esempio per farne il grafico. La spline cubica implementata è quella *not-a-knot*, ovvero imponendo la continuità della derivata terza della spline nei punti: x_2 e x_{n-1}.

```
x = [−2 1 2 3]; y = [ 1 2 4 3];  % coordinate punti da interpolare
t=linspace(−2,3,50)              % punti di campionamento per la valutazione
st=spline(x,y,t);                % costruzione e valutazione della spline
plot(x,y,'ro',t,st,'LineWidth',2);
title('Spline cubica interpolante')
```

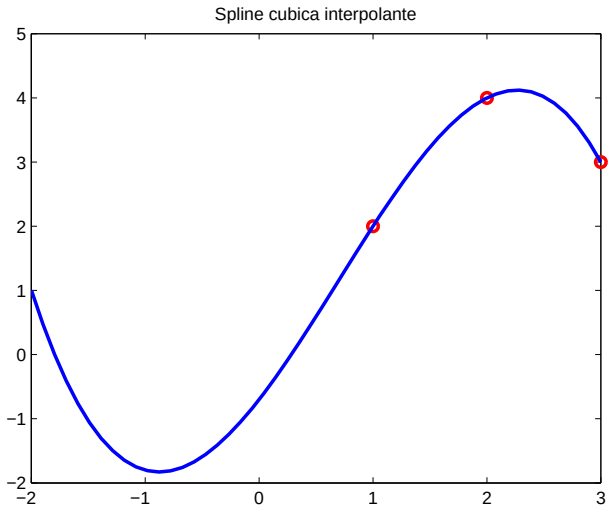


Figura 2: Interpolazione polinomiale con `spline`

Esercizio 1: Compressione utilizzando spline di grado 1

Dato un vettore di valori di una funzione, e.g. y=f(t) dove t=linspace(0,1,m) e m grande e una tolleranza tol, si desidera estrarre un sottoinsieme di nodi $x \subset t$ con $x(1) = 0$ e $x(n) = 1$, tale che l'errore $max(abs(s - y)) < tol$ dove s è la spline di grado 1 con nodi x. Una soluzione può essere ottenuta nel seguente modo. Si comincia con il porre con $x(1) = t(1)$ il primo nodo dell'insieme. Si cicla sui punti y successivi al primo nodo e si scartano quelli in cui l'errore è maggiore della tolleranza. Trovato il punto per cui questo errore nei punti interni è maggiore, diciamo $t(n)$ si inserisce nell'insieme miminale $t(n-1)$ che è il punto per cui al passo precedente l'errore era minore della tolleranza. Successivamente si riparte considerando $t(n-1)$ come nuovo punto iniziale.

Riportare in uscita:

1. il grafico della funzione interpolante ed interpolata;
2. il grafico dell'errore tra funzione interpolante ed interpolata;
3. il grafico del tasso di compressione al variare della tolleranza.

Una linea guida per il codice (da completare è) è la seguente:

```
% Inizializzazioni

m = % numero punti
```

```
t = linspace();
y = % funzione da interpolare mediante compressione

tol = % Tolleranza del metodo


% caso base
x  = t(1); % Salvo il punto iniziale
fx = y(1); % Salvo il suo valore associato

% numero nodi della funzioni interpolante
numnodi = length(t);

% Nodo in cui parte l'interpolazione lineare
% corrisponde al primo nodo dell'insieme x
startnode = 1;

% nodo corrente preso in considerazione
currnode = 2;

% Nodo finale di interpolazione
endnode = 3;

% ciclo dal nodo 2 fintantoche il nodo corrente non sia l'ultimo dell'interpolazione
while(endnode ≠ numnodi),
    % Calcolo il valore dell'interpolazione tra i nodi [startnode e endnode]
    % punti dell'interpolazione lineare
    xa =
    fa =
    xb =
    fb =
    % nodi dove necessaria interpolazione per confronto
    % interni all'intervallo startnode+1 a currnode
    ind = startnode+1:currnode;
    % x della funzione originaria
    xv =
    % y della funzione originaria
    fv =

    % valori calcolati mediante interpolazione
    % nei nodi interni all'intervallo preso in considerazione
    % utilizzo routine interp1
    fv_int =

    % Test di aggiornamento dell'insieme x
    % Se esiste almento un valore fuori dalla tolleranza specificata
    % bisogna aggiornare x, mi aiuto con il calcolo del massimo in valore
    % assoluto tra valori della interpolante ed interpolata.
    errore =

    % se errore maggiore della tolleranza aggiorno l'insieme x
```

```
    if( errore>tol )
        % aggiorno x e fx con valore dell'estermo di b
        x  =
        fx =
        % aggiorno start node come nodo corrente
        startnode = currnode;
    end


    % Aggiorno nodi da considerare
    currnode = currnode + 1;
    endnode = currnode + 1;
    % se ultimo nodo lo aggiungo all'insieme finale
    if(endnode == numnodi)
        x =
        fx =
    end
end

% stampo risultati a video

% 1) confronto tra funzione interpolante ed interpolata

% 2) grafico dell'errore massimo commesso

% numero punti della funzione interpolata ed interpolante
```

Risultati

Esempio $f(x) = x^2$ nell'intervallo $(0, 1)$ con $m = 101$ punti di valutazione. Tolleranza utilizzata per la compressione 0.005.

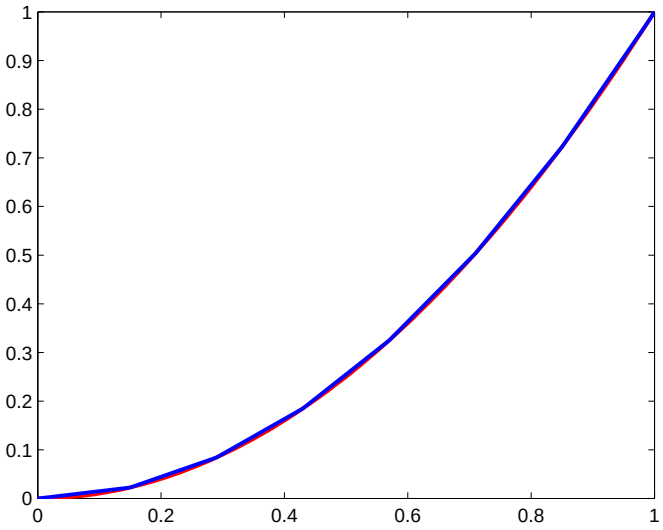


Figura 3: Risultati esercizio 1: Grafico della funzione interpolante ed interpolata.

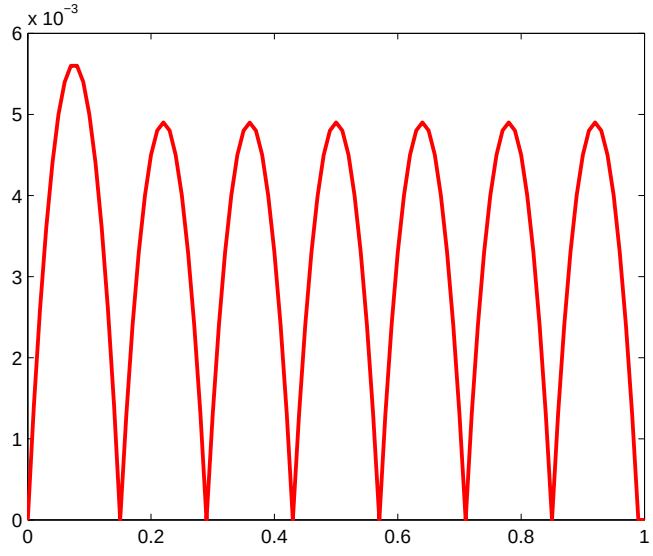


Figura 4: Risultati esercizio 1: Grafico dell'errore.

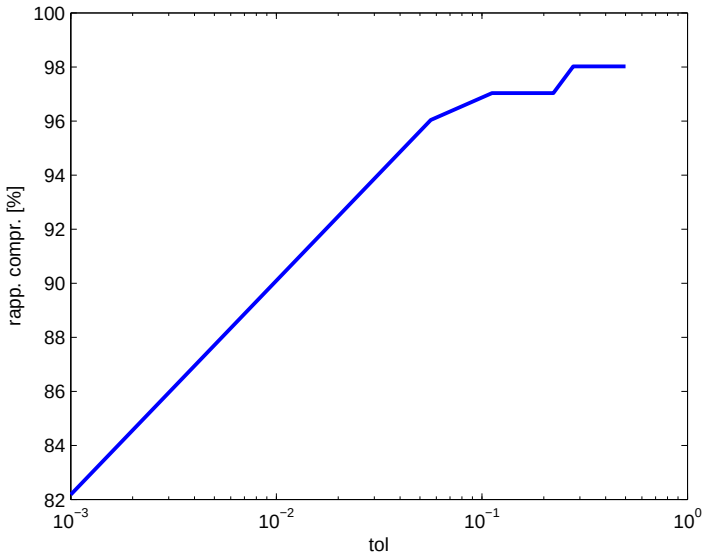


Figura 5: Risultati esercizio 1: Andamento del rapporto di compressione al variare della toleranza.

Esercizio 2: Ordine di una spline

Trovare l'ordine dell'errore uniforme fra una spline di grado 3 e una funzione f(x) liscia (ad esempio $exp(x)/cos(x)$ nell'intervallo $(0, 1)$) come una potenza di h per nodi equispaziati e condizioni di bordo del tipo:

1. naturali
2. vincolate
3. not-a-knot

Per valutare una spline naturale in MatLab si utilizza la seguente sintassi:

```
pp = csape(x,y,'variational');
v = ppval(pp,xval);
```

Per valutare una spline vincolata/complete in MatLab si utilizza la seguente sintassi:

```
pp = csape(x,y,'complete');
v = ppval(pp,xval);
```

Per valutare una spline not-a-knot in MatLab si utilizza la seguente sintassi:

```
v = spline(x,y,xval);
```

Da una suddivione dell'intervallo di interpolazione (a, b) in n suddivisioni l'ordine dell'errore uniforme può essere valutato in un numero maggiori di suddivisioni ad esempio $4n$.

Lo schema del codice può essere il seguente:

```
% Inizializzazioni

f = % funzione da interpolare
a = % estermo sinistro
b = % estremo destro


% Tipo di interpolazione
tipo = 'naturali';
% tipo = 'not-a-knot';
% tipo = 'complete';

% Inizializzazioni errore e passo h
errinf=[];
h = [];

% Ciclo al variare delle suddivisione (ad esempio da 2 a 10 in potenza di 2)
for i=2:10,
```

```
    % Numero di suddivisioni
    ndiv = 2^i;

    % punti equispaziati di interpolazione e relativa funzione
    x =
    y =

    % punti su cui valutare l'errore e relativa funzione
    xval =
    fval =

    % costruisco spline in base al tipo
    switch tipo
        case{'naturali'}

        case{'not-a-knot'}

        case{'complete'}

    end

    % valuto errore ed aggiorno relativo vettore
    errinf =
    % valuto h del passo ed aggiorno relativo vettore
    h =

end

% disegno errore in scala log-log
loglog(h,errinf)
grid on
```

Risultati

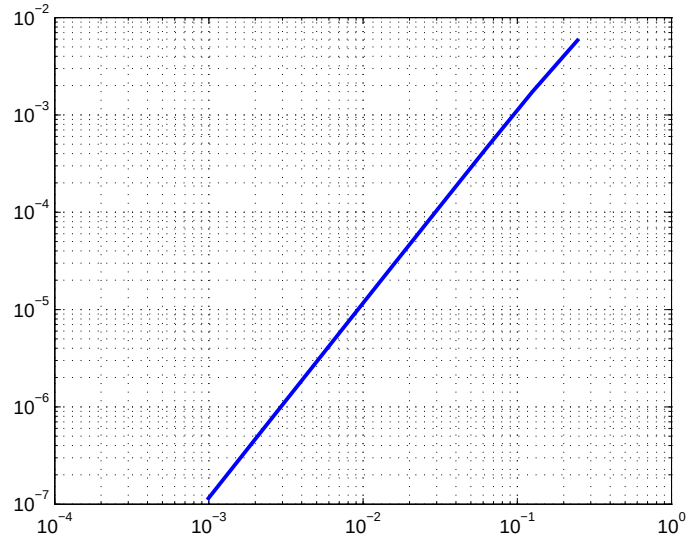


Figura 6: Risultati esercizio 2: Andamento dell'errore.

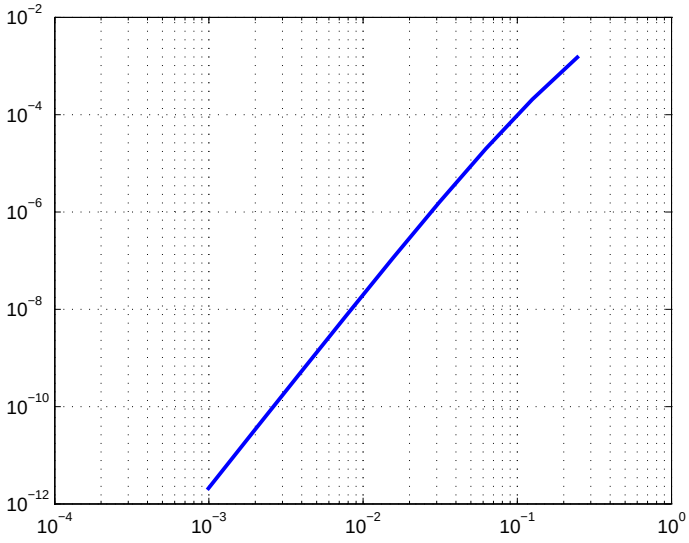


Figura 7: Risultati esercizio 2: Andamento dell'errore.

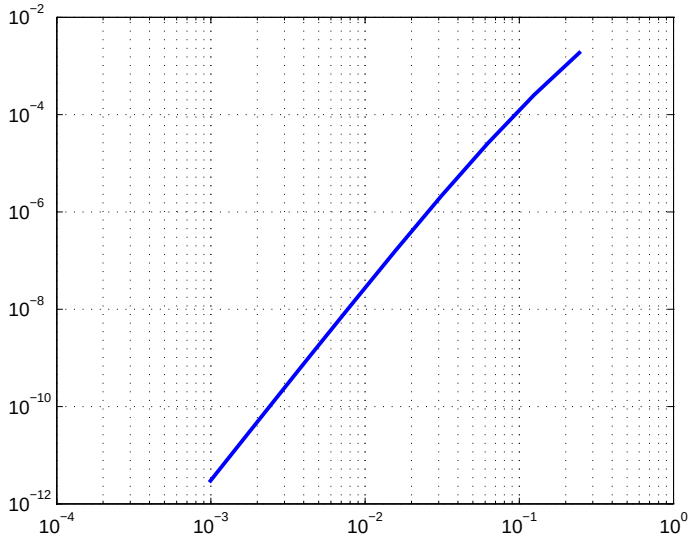


Figura 8: Risultati esercizio 2: Andamento dell'errore.

Esercizio 3: Ordine di una spline

Ripetere l'esercizio nel caso di una funzione non liscia come $\sqrt{x}$ nell'intervallo $(0, 1)$.

Definizione e proprità delle funzioni base delle B-Spline

Prendiamo $X = \{x_1, x_2, \ldots, x_N\}$ una sequenza non decrescente di numeri reali ($x_i \leq x_{i+1}$). Gli x_i sono chiamati nodi "knots", e X è il vettore dei nodi. Definiamo la i-esima funzione base di ordine k, indicata con $B_{i,k}$, come

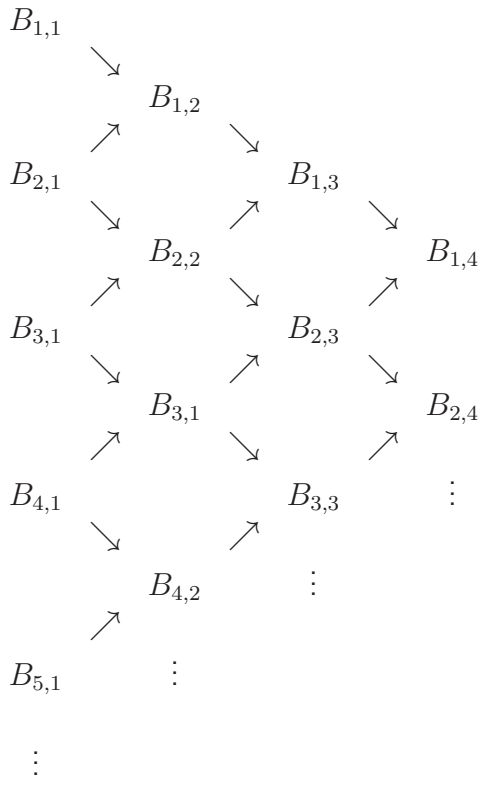
$$B_{i,1}(t) = \begin{cases} 1 & x_i \leq x < x_{i+1} \\ 0 & \text{altrimenti} \end{cases}$$

$$B_{i,k}(t) = \frac{t - x_i}{x_{i+k-1} - x_i} B_{i,k-1}(t) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} B_{i+1,k-1}(t)$$

Questa formula è nota come formuala di Cox–DeBoor.

Osserviamo che:

- $B_{i,1}(t)$ è una funzione a tratti costante uguale a zero ovunque tranne nell'intervallo $[x_i, x_{i+1})$;
- per ogni k, $B_{i,k}(t)$ è una combinazione lineare di due funzioni di base di ordine $k-1$;
- il calcolo della funzione di ordine k genera una tabella del tipo



- se nel calcolo delle funzioni di base si determina un quoziente del tipo $0/0$ questo viene definito uguale a zero;

Elenchiamo alcune proprietà delle funzioni base delle B-Spline:

- Proprietà di supporto locale, $B_{i,k}(t) = 0$ se $t \notin [x_i, x_{i+k})$;
- In ogni sotto-intervallo $[x_j, x_{j+1})$ al massimo k delle $B_{i,k}(t)$ possono essere diverse da zero, nominalemente $B_{j-k}(u), \ldots, B_{j,k}(t)$;
- $B_{i,k}(t) \geq 0$ per ogni i, k e x (non-negativa);
- per ogni sotto-intervallo $[x_i, x_{i+1})$, la somma di tutte le funzioni non nulle in quel intervallo è uguale a 1 per qualunque t (parizione dell'unità).

La codifica della routine MatLab che implementa il calcolo delle funzioni base B-Spline è la seguente (versione ricorsiva per semplicità)

```
function value = BSplineEval(i,k,X,t)
% INPUT
% i indice della B-Spline (o intervallo) da 1 a length(X)-k
% k ordine della B-Spline
% X vettore dei nodi
% t parametro in cui viene valutata la B-Spline
%
% OUTPUT
% value rappresenta il valore della B-Spline in t


% Caso speciale (t==X(end))
if t == X(end)
    if i == length(X) - k
        value = 1;
        return;
    else
        value = 0;
        return;
    end
end

if k == 1,
    if X(i) <= t && t < X(i+1)
        value = 1;
    else
        value = 0;
    end
else
    if X(i+k-1) == X(i) && X(i+k) == X(i+1)
        value = 0;
    elseif X(i+k) == X(i+1)
        value = (t-X(i))/(X(i+k-1)-X(i)) * BSplineEval(i,k-1,X,t);
    elseif X(i+k-1) == X(i)
```

```
        value = (X(i+k)-t)/(X(i+k)-X(i+1)) * BSplineEval(i+1,k-1,X,t);
    else
        value = (t-X(i))/(X(i+k-1)-X(i)) * BSplineEval(i,k-1,X,t) + ...
            (X(i+k)-t)/(X(i+k)-X(i+1)) * BSplineEval(i+1,k-1,X,t);
    end
end
```

Esercizio 4

Mostrare come si comporta la B-spline relative al seguente vettore dei "knots":

$$X = \{0\,,\,0\,,\,0\,,\,0\,,\,1/4\,,\,1/2\,,\,3/4\,,\,1\,,\,1\,,\,1\,,\,1\}\,.$$

per $k = 4$.

Risultati

```
%Suggerimento per la soluzione:
close all
clear all
clc

uvalue = linspace(0,1,101);
k =


X =
kmax =

Nip = zeros(length(uvalue),kmax);
for kind=1:kmax,
    for tind=1:length(uvalue),
        t = uvalue(tind);
        Nip(tind,kind) =
    end
end

figure(1);
h = plot(uvalue,Nip);
set(h,'LineWidth',2);
set(gca,'XTick',linspace(0,1,5));
```

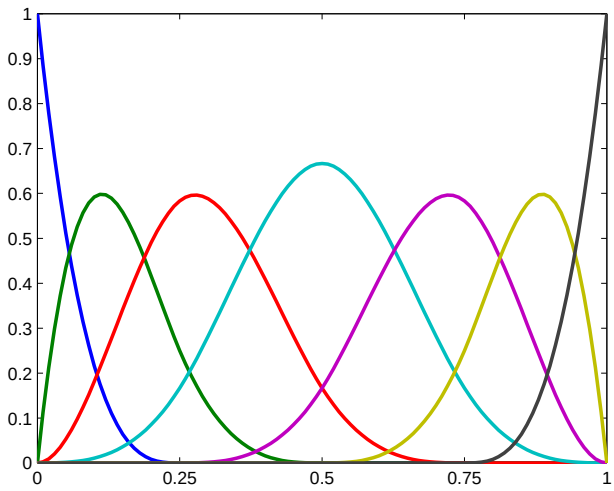


Figura 9: Funzioni base per le B-Spline relative all'esercizio 4.

Esercizio 5

Mostrare come si comporta la B-spline relative al seguente vettore dei "knots":

$$X = (0\,,\, 1/4\,,\, 1/4\,,\, 1/2\,,\, 3/4\,,\, 1\,,\, 1\,,\, 1)$$

per $k = 3$.

Risultati

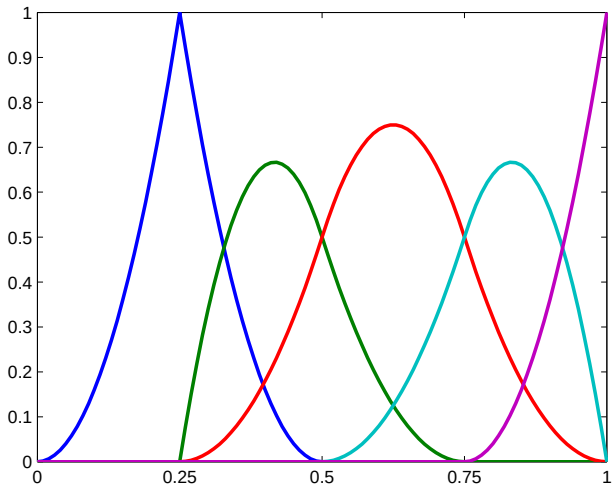


Figura 10: Funzioni base per le B-Spline relative all'esercizio 5.

Esercizio 6

Mostrare come si comporta la B-spline relative ai seguenti vettori dei "knots":

$$X = (0, 1)$$

$$X = (0, 0, 1, 1)$$

$$X = (0, 0, 0, 1, 1, 1)$$

$$X = (0, 0, 0, 0, 1, 1, 1, 1)$$

$$X = (0, 0, 0, 0, 0, 1, 1, 1, 1, 1)$$

$$X = (0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)$$

$$X = (0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1)$$

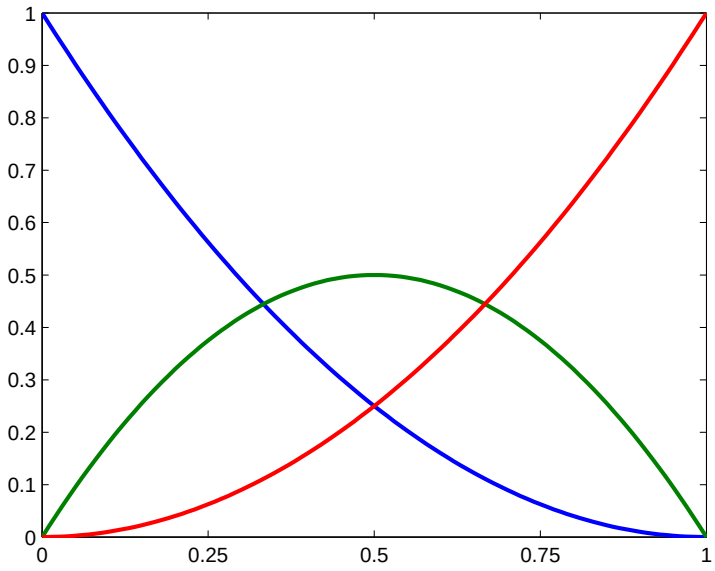


Figura 11: B-Spline di ordine 3.

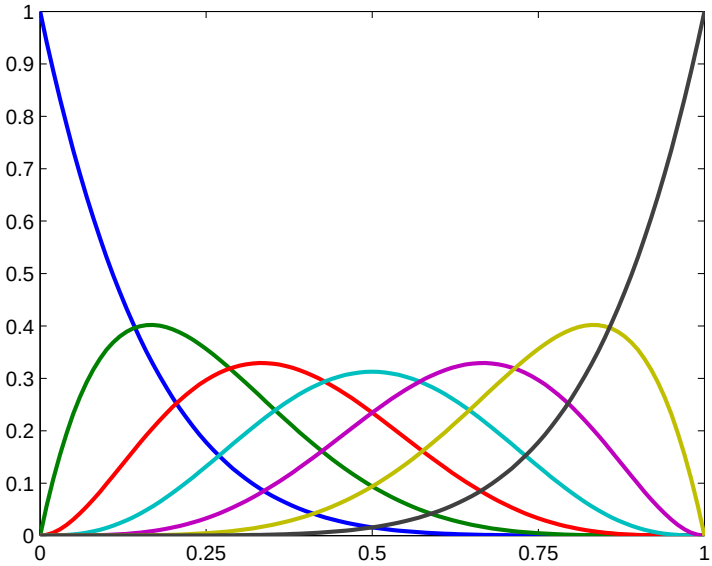


Figura 12: B-Spline di ordine 7.