FITTING DEI DATI

Dall'interpolazione polinomiale alle curve NURBS

Martedì Gaetano M63/1226

Salzillo Biagio M63/1227

A.A. 2020/21

Cos’è il fitting?

Il fitting è il processo di costruzione di una curva o di una funzione matematica, che meglio “spieghi” i dati raccolti, relativi ad un fenomeno. Il problema del fitting consiste nel costruire un corretto modello matematico che descriva i dati in modo attendibile. Abbiamo due tipi di modelli matematici:

- **Interpolante;**
- **Approssimante;**

Caso Studio

Verranno illustrate le varie tecniche di fitting dei dati, applicate ad un caso studio reale, ovvero al set di dati relativo alla temperatura media della città di Milano per un arco temporale di 22 anni: dal 1984 al 2006. I dati sono stati reperiti dal sito Web dell'ente statunitense National Centers for Environmental Information (NCEI).

Interpolazione vs. Approssimazione

INTERPOLAZIONE

Si costruisce una curva che passa per i punti assegnati, trascurando l'eventuale errore presente sui dati

APPROSSIMAZIONE

Si costruisce una curva che non passa necessariamente per i punti assegnati, considerando questi ultimi affetti da errore.

Condizioni di interpolazione

Sia F uno spazio di funzioni di variabile reale o complessa.

Assegnati:

- **n valori distinti reali o complessi** $\{x_i\}_{i=1,..,n}$
- **n valori distinti reali o complessi** $\{y_i\}_{i=1,..,n}$

Lagrange

Si cerca una funzione $f(x) \in F$ **tale che:**

$$f(x_i) = y_i \qquad i = 1, \dots, n$$

ovvero tali condizioni si traducono col fatto che la funzione deve passare per i punti assegnati.

Hermite

Si cerca una funzione interpolante che, nei punti x_i**, assuma gli stessi valori della funzione** $f(x)$ **originale e delle sue derivate, fino all'ordine q.**

Approssimazione

Dati n punti distinti $(x_i, y_i)_{i=1,..,n}$ si vuole costruire una funzione $f(x)$ tale che nei nodi $(x_i)_{i=1,..,n}$ non assuma i valori $(y_i)_{i=1,..,n}$ ma si scosti poco da essi.

Punti di controllo: **sono dei punti fissati, il cui scopo è fare da guida per la costruzione di una curva approssimante;**

Convex hull: **è il più piccolo poligono convesso che contiene i punti di controllo.**

Metodo dei minimi quadrati

Siano (x_i, y_i) con $i = 1, \dots, n$ i punti che rappresentano i dati in ingresso. Si vuole trovare una funzione f tale che approssimi la successione di punti. Questa funzione può essere determinata minimizzando la distanza tra le due successioni y_i e $f(x_i)$, ovvero la quantità d:

$$d = \sum_{i=1}^{n} [y_i - f(x_i)]^2$$

Interpolazione polinomiale

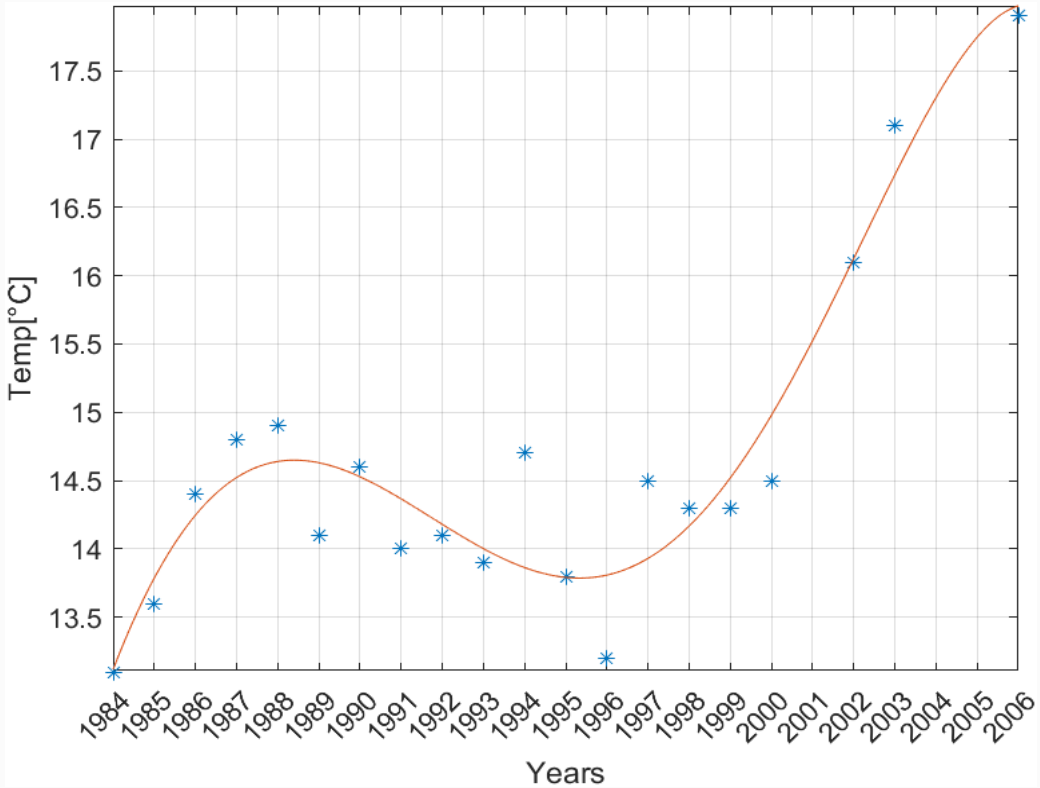
Nel caso dell'interpolazione polinomiale, la funzione $f(x)$ avere la forma di un polinomio. Il grado r del polinomio cercato può essere fissato a priori, purché sia pari almeno al numero di punti da interpolare meno 1, cioè $r < n - 1$;ciò affinché il problema sia *sovradeterminato*.

$$argmin_{(a_0,a_1)} d$$

$$d = \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i)]^2$$

Polyfit e *Polyval*

Per effettuare l'interpolazione polinomiale in MATLAB si utilizza la function *polyfit* per trovare il polinomio interpolante di grado r, e la function *polyval* per valutare il polinomio nei punti dati.



Le curve di Béziers

Le curve di Bézier sono delle particolari curve parametriche. Una curva di Bézier è il centro di massa del convex hull.

Centro di massa

Il centro di massa (CM) è il punto di un oggetto in cui è concentrato il suo peso, ovvero dove si può assumere che la forza di gravità sia applicata. Per un insieme finito di n punti può essere definito come:

$$CM = \frac{\sum_{i=1}^{n} m_i P_i}{\sum_{i=1}^{n} m_i}$$

Proprietà delle curve di Bézier

1. La curva deve interpolare il primo punto P_0 e l'ultimo punto P_n;
2. Tutti i punti della curva devono stare all'interno del convex hull;
3. La somma CM deve definire un punto del piano;
4. La curva deve essere indipendente dal sistema di riferimento cartesiano (cioè cambiando sistema ottengo sempre la stessa forma della curva).

Da queste prime proprietà si arriva alla conclusione che la somma delle m_i deve essere 1 e quindi l'equazione della curva di Bézier assume la forma:

$$CM = \sum_{i=1}^{n} m_i P_i$$

Altre proprietà

Per stabilire chi siano le m_i si analizzano altre proprietà delle curve relative alle derivate di diverso ordine calcolate nei punti di controllo: dati $n+1$ punti di controllo, la curva di Bézier di grado n è definita conoscendo la derivata:

- **zero in P_0**
- **zero in P_n**
- **prima in P_0**
- **prima in P_n**
- **seconda in P_0**
- **seconda in P_n**
- **…**
- **derivata di ordine $n-1$ in P_0**
- **derivata di ordine $n-1$ in P_n**

Curve di Béziers e i polinomi di Bernstein

L'equazione della curva di Bézier è così fatta:

$$C(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

con $P_0, \dots, P_n$ **punti di controllo e** $B_{i,n}(t)$ **polinomi di Bernstein di grado** n**,**

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad t \in [0,1]$$

Algoritmo di De Casteljau

L’algoritmo di De Casteljau è un metodo che permette di costruire in modo semplice e algoritmico le curve di Bézier.

La forma di una curva intermedia in generale è data da:

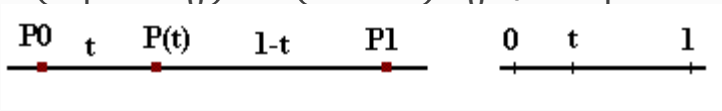
$$P_j^r(t) = (1-t)P_j^{r-1}(t) + tP_{j+1}^{r-1}(t)$$

Algoritmo di De Casteljau

GRADO 1

Un punto qualunque della curva $C(t)$ compreso tra P_0 e P_1 (due punti del piano) sarà dato da:

$$C(t) = P_0 + t(P_1 - P_0) = (1-t)P_0 + tP_1 \quad \textbf{con } t \in [0,1]$$



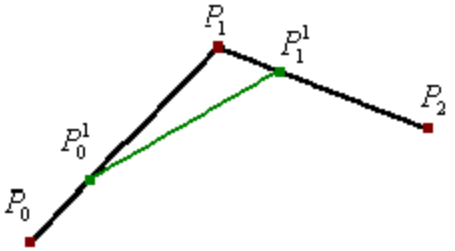
GRADO 2

L'equazione generale dei punti della curva è quindi ottenuta per sostituzione come segue:

$$P_0^1(t) = (1-t)P_0^0 + tP_1^0$$

$$P_1^1(t) = (1-t)P_1^0 + tP_2^0$$

$$P_0^2(t) = (1-t)P_0^1(t) + tP_1^1(t)$$



L'equazione della curva di Bézier allora è data:

$$C(t) = P_0^2(t) = (1-t)^2 P_0^0 + 2(1-t)tP_1^0 + t^2 P_2^0$$

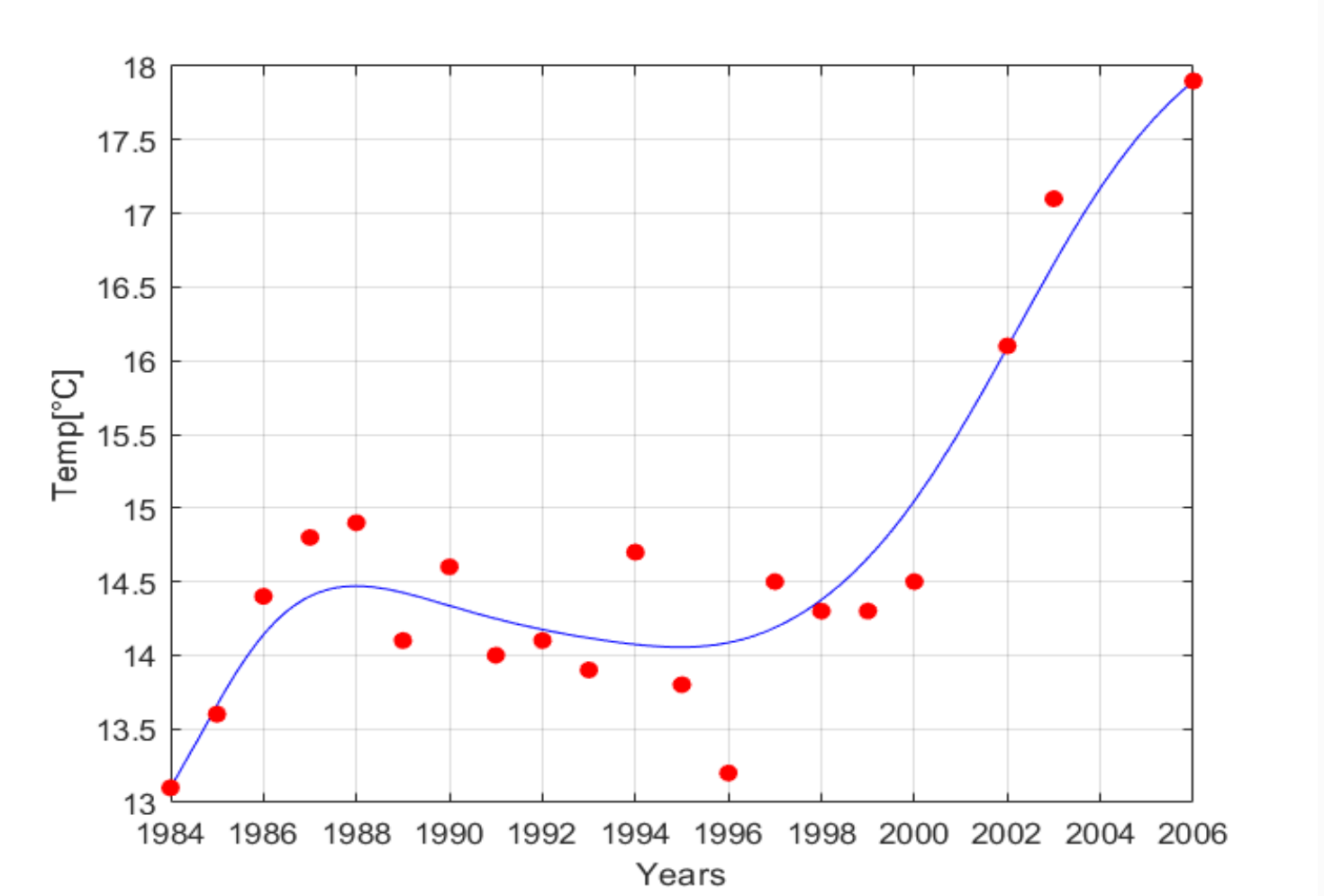
Curva di Bézier

MATLAB mette a disposizione una funzione che permette di generare i polinomi di Bernstein, ovvero:

B = bernsteinMatrix(k,t)

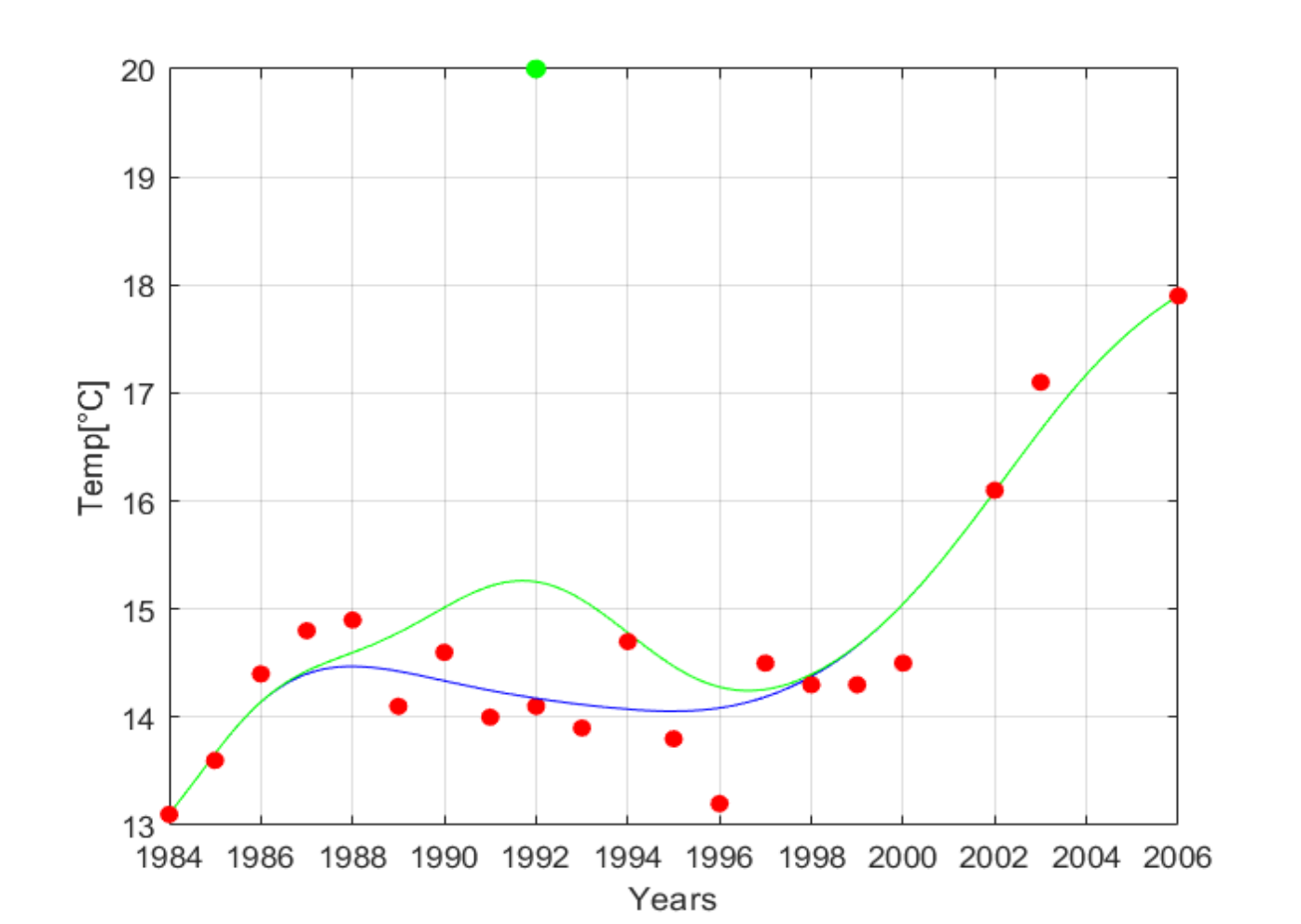
Tale funzione restituisce un vettore n-dimensionale, contenente gli n polinomi di Bernstein di grado $k = n - 1$ (dove n è il numero dei punti di controllo)

```
%Calcolo della matrice di Bernstein
syms t
B = bernsteinMatrix(length(years)-1,t)
BézierCurve = simplify(B*tavg)
fplot(BézierCurve(1),BézierCurve(2),[0,1])
hold on
scatter(tavg(:,1),tavg(:,2),'filled')
```



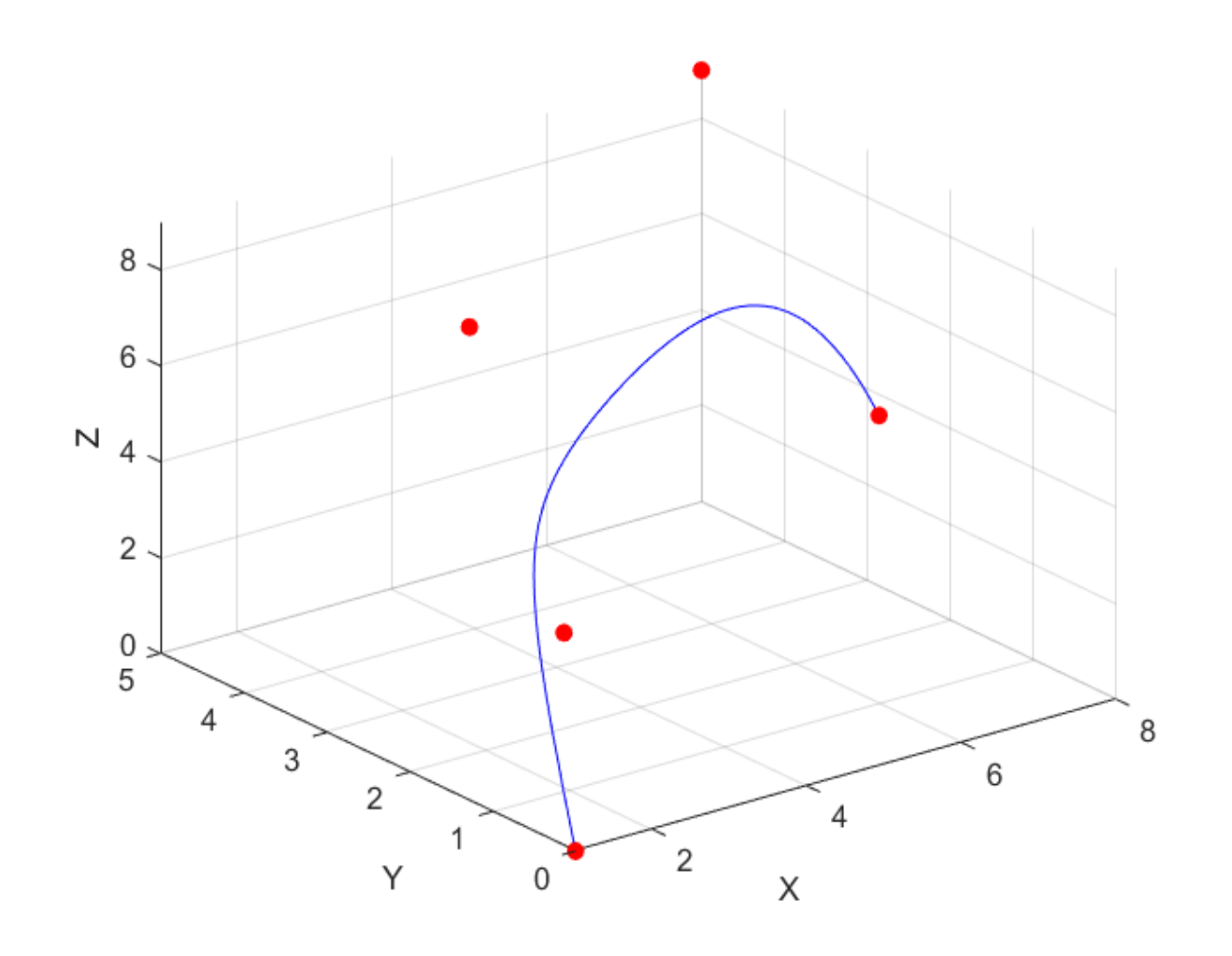
Supporto globale

Cambiando un punto di controllo, osserviamo che l'andamento della curva è cambiato, rispetto alla precedente. Quindi ciò che si osserva è che non si possono apportare modifiche "locali" alla curva, infatti modificando un solo punto di controllo viene modificata la curva "globalmente", questo perché le funzioni di base (polinomi di Bernstein) sono definite e sono non nulle in tutto l'intervallo di definizione della curva



Curve di Bézier 3D

In maniera analoga, in MATLAB è possibile costruire le curve di Bézier a partire da punti di controllo fissati nello spazio

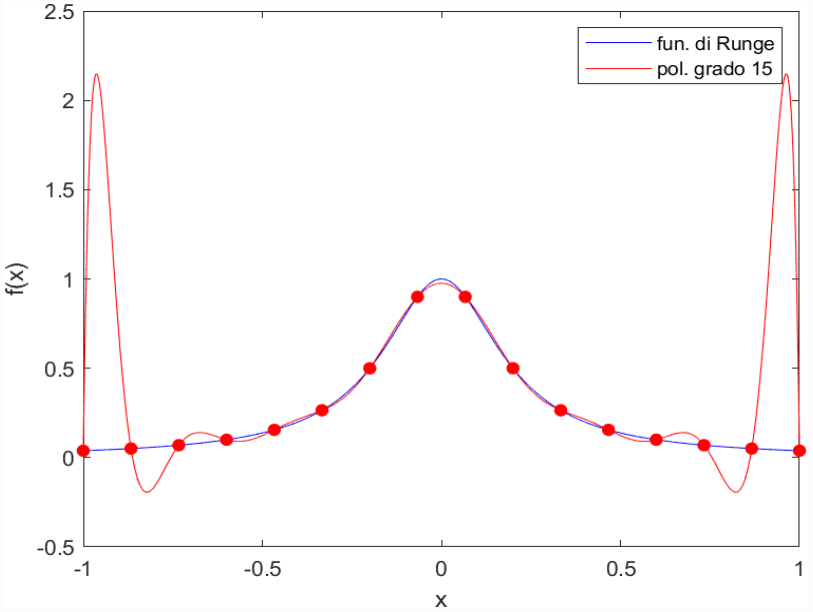


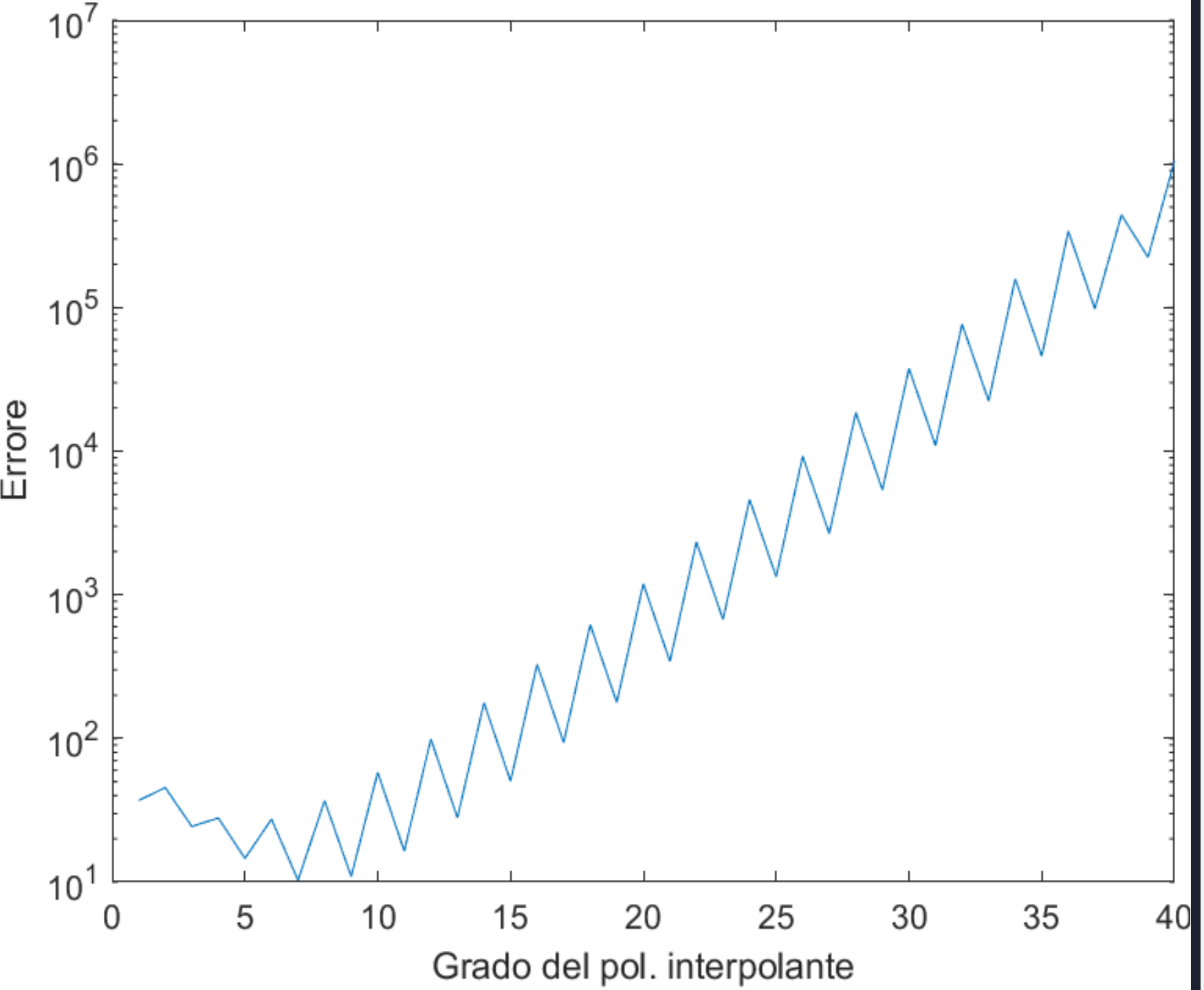
Il fenomeno di Runge

$$f(x) = \frac{1}{1 + 25x^2}$$

Runge scoprì che interpolando questa funzione su un insieme di punti x_i , con $i = 1, \dots, n$, equidistanti nell'intervallo $[-1,1]$, con un polinomio $P_r(x)$ di grado $r \leq n$, l'interpolazione risultante oscilla in ampiezza verso gli estremi dell'intervallo. Inoltre, si dimostra che l'errore tra $f(x)$ e $P_n(x)$ tende all'infinito all'aumentare del grado del polinomio:

$$\lim_{n \to \infty} \left(\max_{x \in [-1,1]} |f(x) - P_n(x)| \right) = +\infty$$





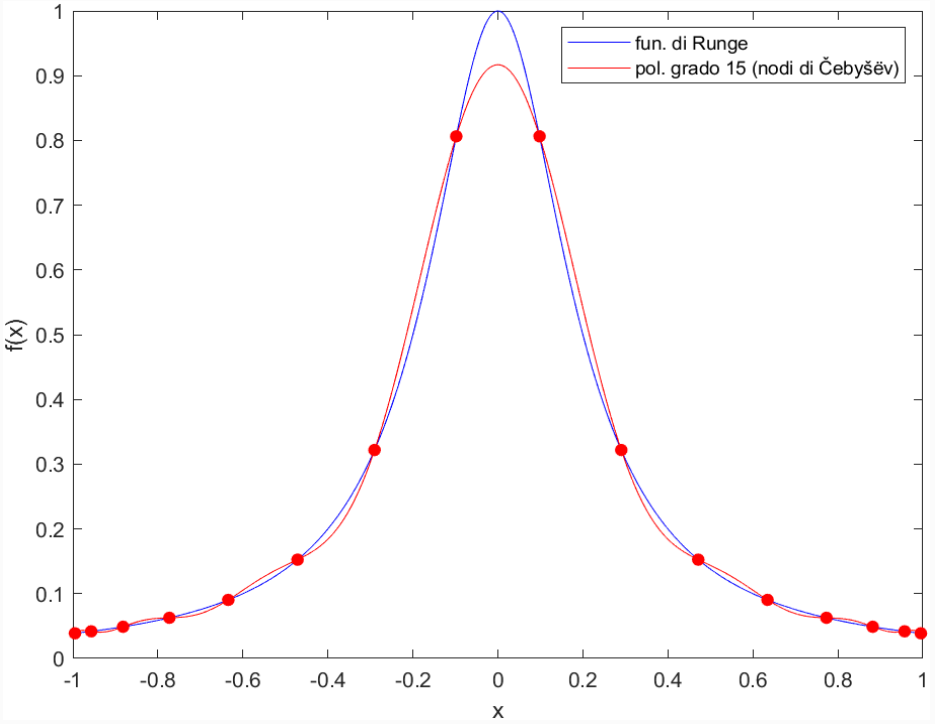
Errore di interpolazione

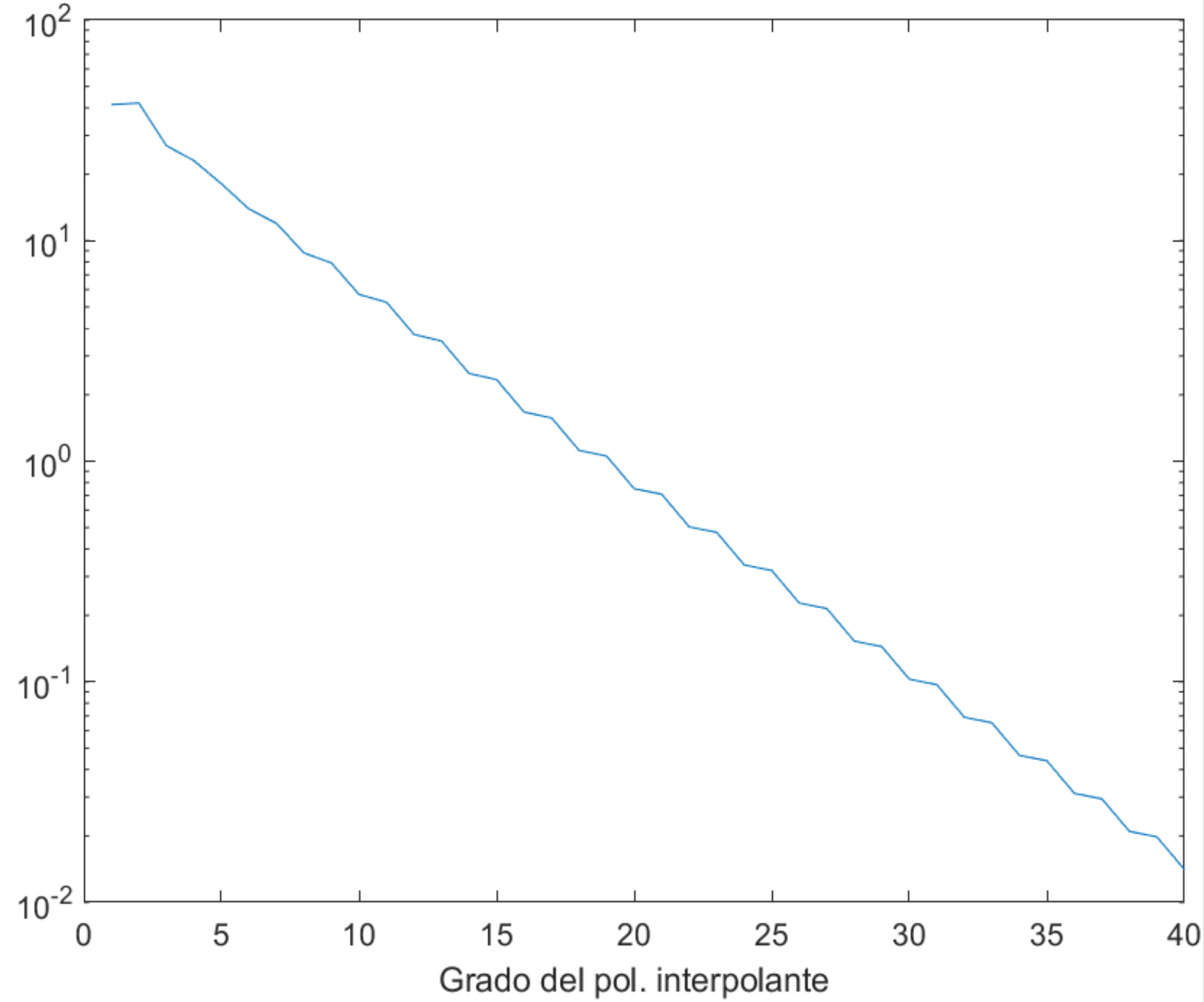
Interpolazione ottenuta considerando punti di controllo equispaziati nell'intervallo $[-1,1]$

I nodi di Čebyšëv

È possibile ottenere uno schema di interpolazione il cui errore diminuisca all'aumentare del numero di nodi considerati nell'intervallo $[-1,1]$, utilizzando i nodi di Čebyšëv.

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), con\ i = 1, \dots, n$$





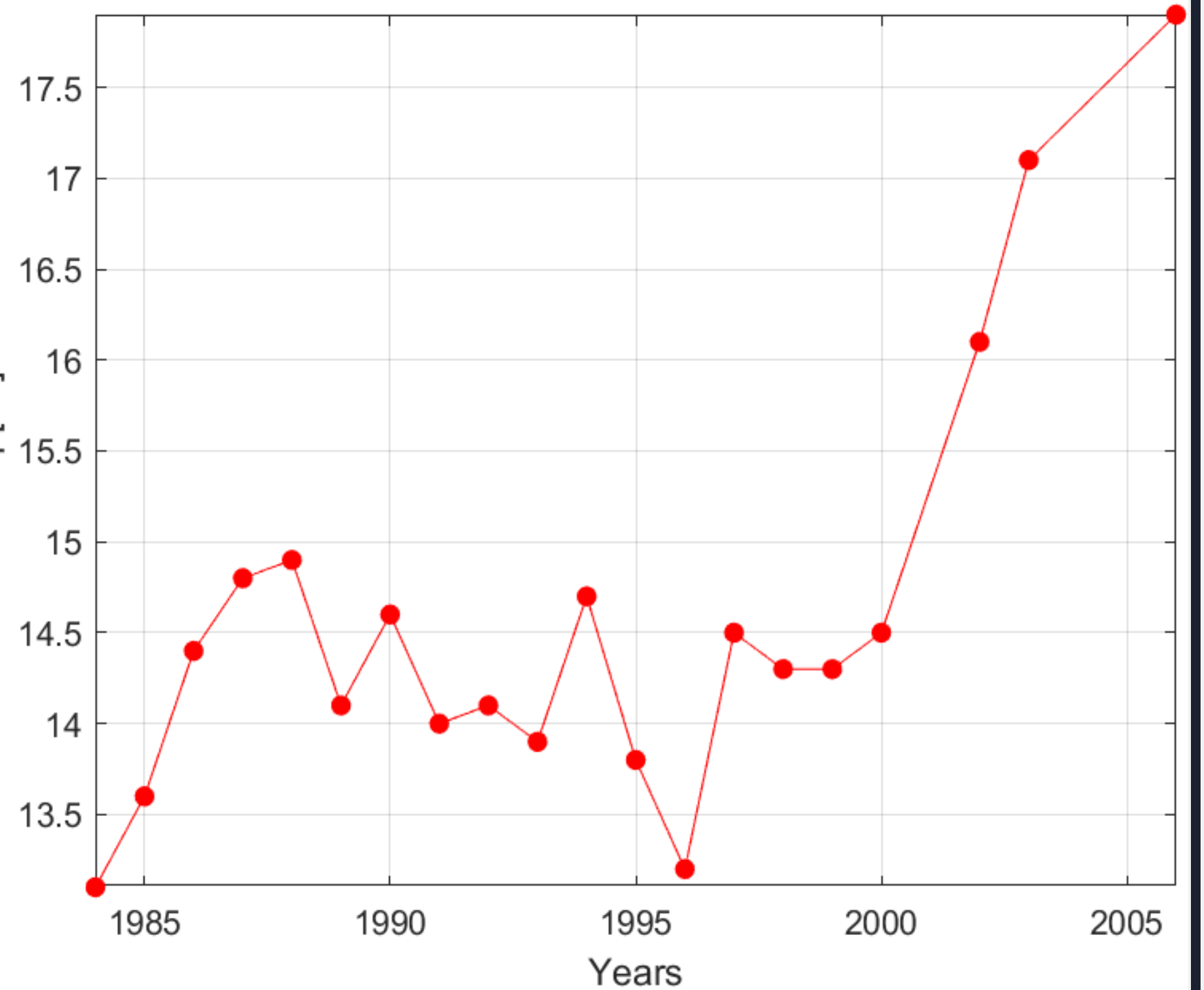
Errore di interpolazione

Interpolazione ottenuta considerando i nodi di Čebyšëv.

Interpolazione polinomiale a tratti

Si scelgono come funzioni interpolanti dei polinomi, solitamente di grado basso per evitare i fenomeni di oscillazione. L'intervallo di rappresentazione $[x_1, x_n]$ viene suddiviso in k sotto-intervalli, ognuno dei quali contenente m ascisse relative ad m punti di controllo, e su di ognuno di questi sotto-intervalli viene costruito un polinomio interpolante di grado $m-1$, per un totale di k polinomi.

Il caso più semplice di interpolazione polinomiale a tratti consiste nell'utilizzare unicamente polinomi di primo grado, ed è detta ***interpolazione lineare a tratti***.



Interpolazione lineare a tratti

```
t=linspace(min(years),max(years),1e4)';

f = interp1(years,temp,t);

plot(t,f,'r')

hold on

scatter(years,temp,'or','filled')

xlabel('Years')

ylabel('Temp[°C]')
```

Curve B-Spline

Le curve Spline sono delle curve composte da altre curve, e sono per questo dette curve composite. Una curva Spline è costruita congiungendo curve polinomiali mantenendone continuità e regolarità. Le B-Spline sono Spline linearmente indipendenti.

Curve B-Spline

L'equazione delle curve B-Spline è la seguente:

$$C(t) = \sum_{i=0}^{n} P_i B_{i,h}(t)$$

dove i P_i **sono gli** $n+1$ **punti di controllo e** $B_{i,h}(t)$ **è l'i-esima funzione B-Spline di grado** h **definita su un intervallo di nodi** $[t_i\ ,\ t_{i+h+1}]$**, dove la funzione ha supporto compatto (ovvero il sottoinsieme dei punti del dominio in cui la funzione assume valori non nulli).**

Curve B-Spline

Vettore dei nodi:

$T = (t_0, t_1, \dots, t_{n+h+1}), \quad t_i < t_{i+1}$

Il numero dei nodi allora è così ottenuto:

$$num\_nodi = num\_punti + grado + 1$$

dove

$$grado = h$$

$$num\_punti = n + 1 \ (numero\ dei\ punti\ di\ controllo)$$

I termini $B_{i,h}(t)$: sono dei polinomi a tratti, di grado h, calcolati attraverso le formule ricorsive di De Boor:

$con\ h = 1 \qquad B_{i,h}(t) = \begin{cases} 1 & se\ t_i \le t < t_{i+1} \\ 0 & altrimenti \end{cases}$

$con\ h > 1 \qquad B_{i,h}(t) = \frac{t - t_i}{t_{i+h-1} - t_i} B_{i,h-1}(t) + \frac{t_{i+h} - t}{t_{i+h} - t_{i+1}} B_{i+1,h-1}(t)$

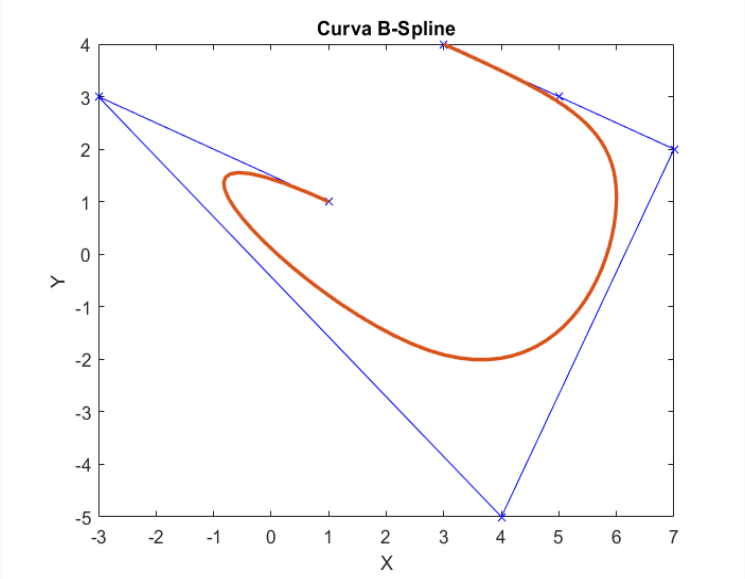
B-Spline MATLAB

```
pp = bspline(t)
```

È una funzione utilizzata per plottare una curva B-Spline e le sue funzioni di base a partire dal vettore dei nodi.

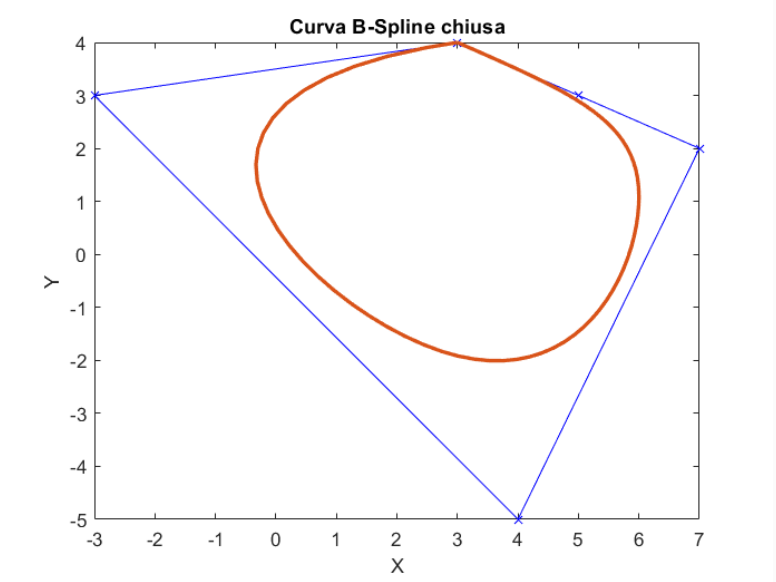

```
[q, qd, qdd, pp] = bsplinepolytraj(controlpoints,tinterval,tsamples)
```

È una funzione che permette di rappresentare una curva B-Spline cubica, a partire da dei fissati punti di controllo (in questo esempio, i punti di controllo sono (3,4),(5,2),(7,2),(4,-5),(-3,3) e (1,1))



Proprietà delle curve B-Spline

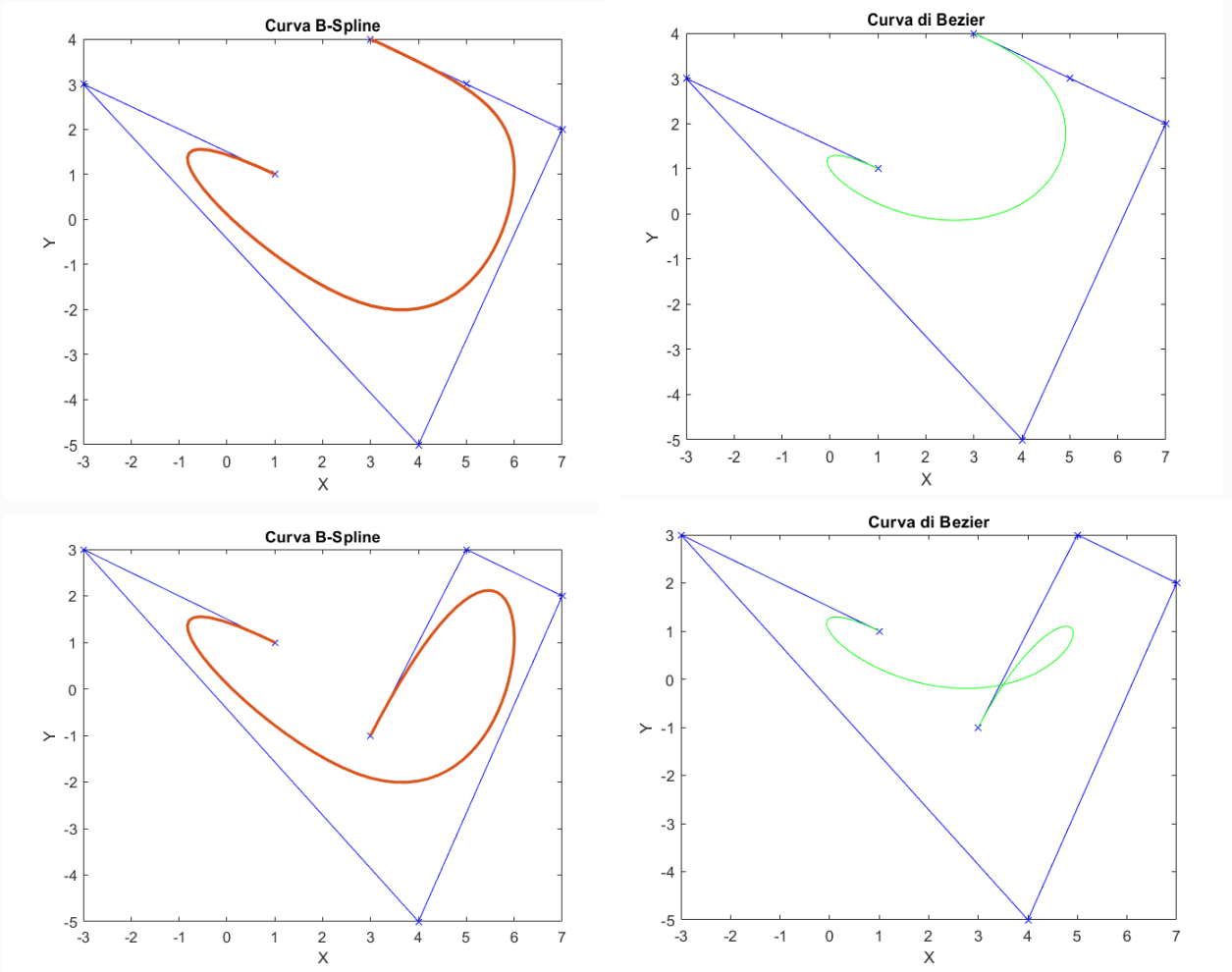
- Una curva B-Spline può essere chiusa se l'ultimo punto di controllo P_n coincide con il primo P_0;



- Proprietà di invarianza: possiamo traslare, ruotare e deformare la curva applicando, ad esempio, delle trasformazioni geometriche o affini solo ai punti di controllo e poi riapplicare la formula per rivalutare la curva (evitandoci di dover effettuare la trasformazione per ogni singolo valore della curva);

Proprietà delle curve B-Spline

- **Controllo locale: come conseguenza del supporto compatto delle B-Spline, se P_j cambia, $C(t)$ risulta modificata solo in corrispondenza dei punti di definizione della B-Spline $B_{j,h}$;**



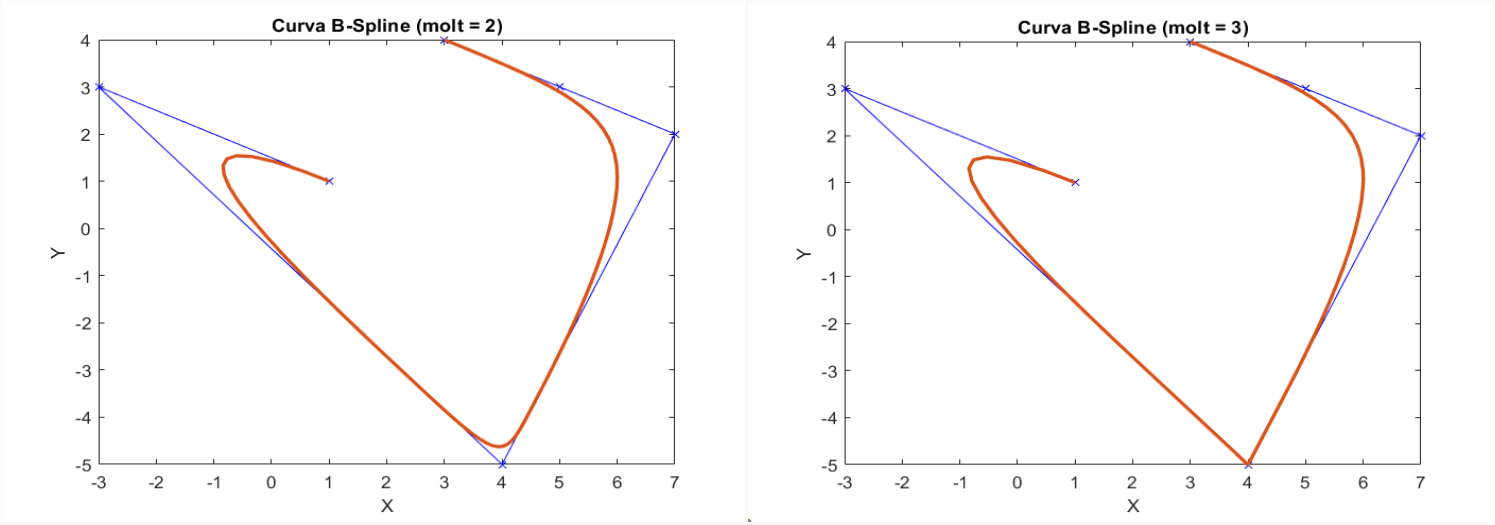
Proprietà delle curve B-Spline

- **È possibile modificare la curva in modo tale che sia tangente al primo e all'ultimo segmento nel primo e nell'ultimo punto di controllo, rispettivamente, imponendo che il primo e l'ultimo nodo abbiano molteplicità** $h + 1;$

- **Le curve B-Spline in generale non interpolano alcun punto di controllo;**

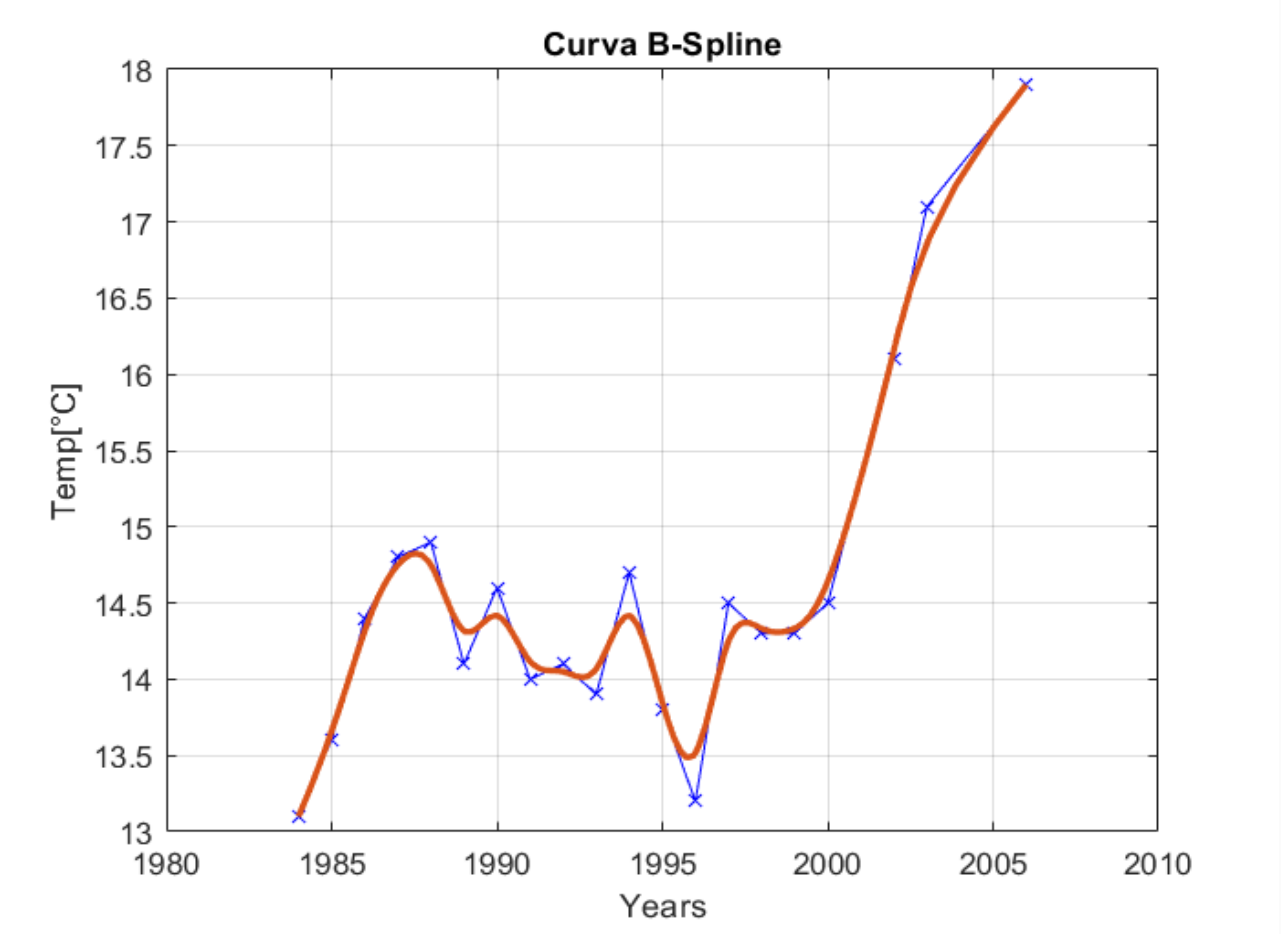
Proprietà delle curve B-Spline

- **Fissato il grado h, all'aumentare della molteplicità ($molt$) di un nodo, la curva si avvicina al nodo multiplo, fino ad interpolare il punto di controllo corrispondente ($molt = h + 1$), questo perché più si aumenta la molteplicità di un nodo t_i tanto più si riduce il supporto della B-Spline definita su quel nodo e quindi il risultato è che tende più rapidamente al suo valore massimo;**



Curva B-Spline

```
%Selezioniamo i punti di controllo del caso studio
cpts = tavg';
%Scelgo    l'intervallo    temporale    selezionando
%l'istante iniziale e finale
tpts = [0 21];
%Campioni temporali per la traiettoria, specificati
come vettore
tvec = linspace(0,21,1e4);

[q,~,~,pp] = bsplinepolytraj(cpts,tpts,tvec);
plot(cpts(1,:),cpts(2,:),'xb-');
title('Curva B-Spline')
xlabel('Years')
ylabel('Temp[°C]')
grid on
hold on
fnplt(pp)
```



NURBS

Nel 1975 circa, presso la BOEING Corporation, per ampliare la varietà di curve rappresentabili (come le coniche o le curve trigonometriche), si decise di estendere le B-Spline tramite l'impiego di polinomi razionali, ottenendo così, le curve **NURBS** (Non Uniform Rational B-Spline).

NURBS

Una curva NURBS è definita da quattro caratteristiche:

- grado
- punti di controllo
- vettore dei nodi
- pesi

Il grado

Il grado è un numero intero positivo h, solitamente variabile tra 1 e 5, per evitare il fenomeno delle oscillazioni, anche se teoricamente esso può assumere un qualunque valore intero positivo. A volte, si può far riferimento all'ordine di una curva NURBS, ossia un numero intero positivo pari al grado della curva più uno.

$$ordine = h + 1$$

I punti di controllo e i pesi

I **punti di controllo** sono una serie di punti $P_0, \dots, P_n$ in numero almeno pari a $h + 1$, che hanno la funzione di fare da guida alla curva da rappresentare. Ad ogni vertice è associato un **peso** $w_i \geq 0$ $(con\ i = 0, \dots, n)$ che indica la capacità di tale punto di attrarre a sé la curva. Quando i punti di controllo di una curva hanno tutti lo stesso peso (solitamente 1), la curva viene denominata **«non razionale»**; in caso contrario, è detta **«razionale»**.

Il vettore dei nodi

Il vettore dei nodi è un vettore numerico $T = (t_0, t_1, \dots, t_m)$ costituito da $m + 1$ elementi. In particolare, il valore di m deve essere pari ad:

$$m = n + h + 1$$

dove $n + 1$ è il numero di vertici di controllo, ed h è il grado della curva.

Il vettore dei nodi

Si dice che un nodo ha **molteplicità piena** se il suo valore si ripete tante volte quante il grado h della curva, mentre un nodo si dice **semplice** se esso ha molteplicità pari a 1.

Se una sequenza di nodi inizia con un nodo a molteplicità piena, segue con dei nodi semplici, termina con un nodo a molteplicità piena e tutti i valori sono ugualmente spaziati, i suoi nodi sono detti **uniformi**. Viceversa, il vettore dei nodi è detto **non uniforme**.

Se $t_0 = 0 \; e \; t_m = 1$ il vettore dei nodi T è detto **"standard"**.

Il vettore dei nodi

$h = 3$

$T = [0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 4 \quad 4]$ vettore dei nodi uniforme

$T = [0 \quad 0 \quad 1 \quad 2 \quad 5 \quad 6 \quad 7 \quad 8 \quad 8]$ vettore dei nodi non uniforme

$T = [0 \quad 0 \quad 0 \quad 0.25 \quad 0.5 \quad 0.75 \quad 1 \quad 1 \quad 1]$ vettore dei nodi in forma standard

NURBS

Dati i punti di controllo $P_0, \dots, P_n$, con i relativi pesi $w_0, \dots, w_n$, ed il vettore di nodi $T = (t_0, \dots, t_m)$, con $m = n + h + 1$, la curva NURBS parametrica $C(t)$ sarà pari a:

$$C(t) = \frac{\sum_{i=0}^{n} w_i * P_i * \ B_{i,h}(t)}{\sum_{i=0}^{n} w_i * B_{i,h}(t)} \ , con \ t \in [t_0, t_m]$$

dove con $B_{i,h}(t)$ si indicano le funzioni di base B-Spline di grado h.

Si può utilizzare anche una forma equivalente quale:

$$C(t) = \sum_{i=0}^{n} P_i * R_{i,h}(t) \ , con \ t \in [t_0, t_{n+h+1}]$$

dove:

$$R_{i,h}(t) = \frac{w_i * B_{i,h}(t)}{\sum_{i=0}^{n} w_i * B_{i,h}(t)} \ , con \ t \in [t_0, t_{n+h+1}]$$

Proprietà

Le curve NURBS possiedono le seguenti proprietà:

- **La funzione $R_{i,h}(t)$ è una funzione razionale di grado h in t;**
- **La funzione $R_{i,h}(t)$ è non negativa $\forall\ i, h, t$;**
- **Supporto locale: la funzione $R_{i,h}(t)$ assume valori non nulli solo nell'intervallo $[t_i, t_{i+h+1}]$;**
- **La somma di tutte le funzioni di base $B_{i,h}(t)$ su ogni intervallo $[t_i, t_{i+h+1}]$ è pari a 1;**
- **In presenza di nodi multipli, con molteplicità pari a k , la curva di NURBS $C(t)$ è una funzione di classe C^{h-k};**

Proprietà

- **Dato un nodo di molteplicità $h+1$, la curva NURBS passa esattamente per il punto di controllo associato a tale nodo;**
- **Sia P_i un punto di controllo di molteplicità pari a k, allora la curva NURBS $C(t)$ risultante sarà una funzione di classe C^{h-k};**
- **Dato un punto di controllo P_i di molteplicità $k = h+1$, allora la curva $C(t)$ passerà esattamente per il punto P_i;**
- **Strong convex-hull: la curva $C(t)$ è contenuta nel dominio convesso generato dai suoi punti di controllo (se $w_i \geq 0 \ \forall i$);**

Invarianza alle trasformazioni affini

Una proprietà molto importante che possiedono le curve NURBS è l'invarianza alle trasformazioni affini. Dato un generico punto di controllo P, indichiamo una trasformazione affine per tale punto come: $A[P] = L[P] + T$.

Se indichiamo una generica curva NURBS con $C(t)$, allora:

$$C(t) = \sum_i P_i * R_{i,h}(t)$$

$$A[C(t)] = L[C(t)] + T = \sum_i \{L[P_i] * R_{i,h}(t)\} + T$$

Invarianza alle trasformazioni affini

Se invece consideriamo la curva NURBS costruita con le trasformate affini dei punti di controllo:

$$C'(t) = \sum_i A[P_i] * R_{i,h}(t) = \sum_i \{L[P_i] * R_{i,h}(t)\} + T \sum_i R_{i,h}(t)$$

$$\sum_i R_{i,h}(t) = 1$$

$$C'(t) = \sum_i \{L[P_i] * R_{i,h}(t)\} + T = A[C(t)]$$

Invarianza alle trasformazioni affini

Dunque:

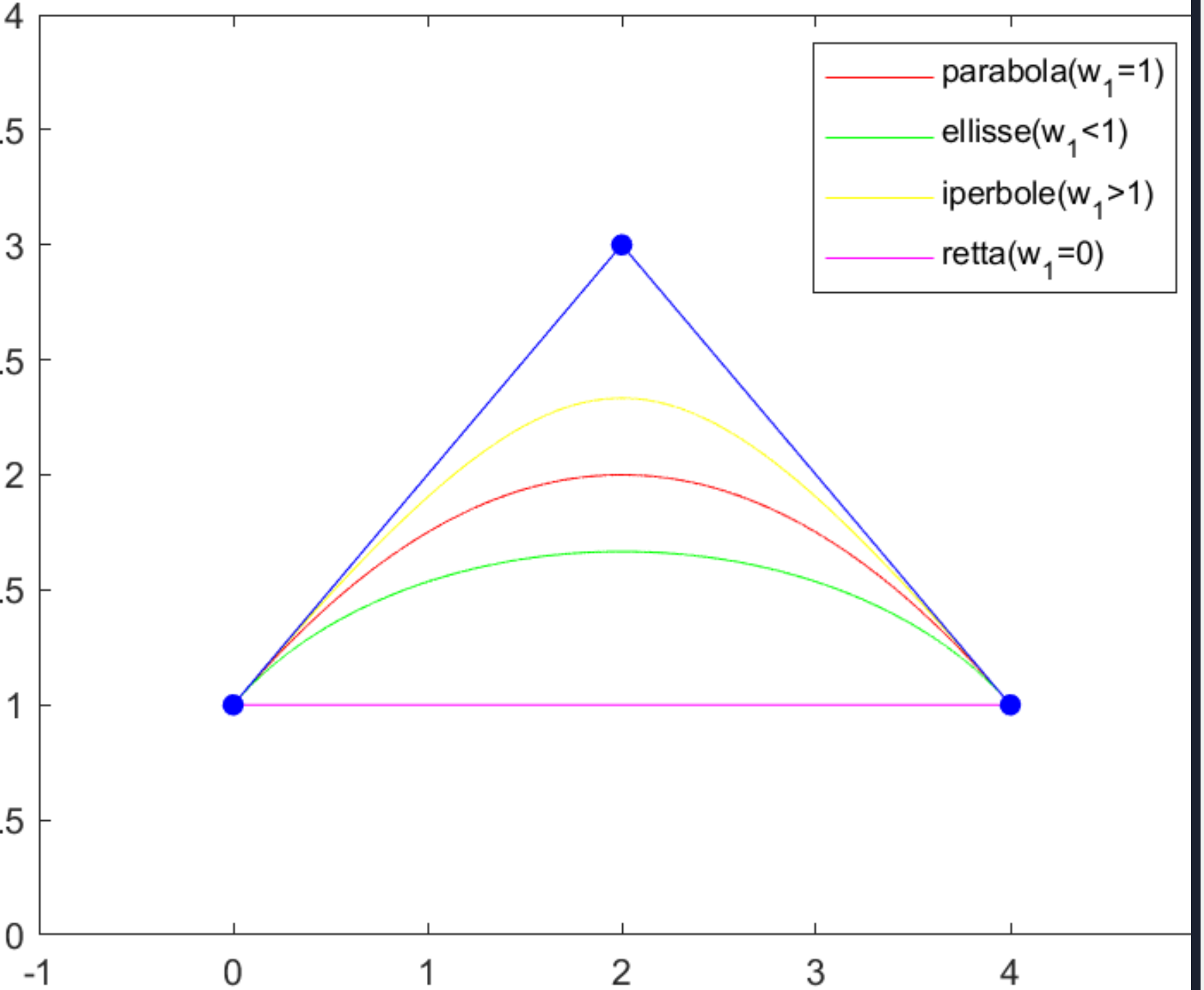
$$A[C(t)] = \sum_i A[P_i] * R_{i,h}(t)$$

cioè l'immagine affine di una curva NURBS si ottiene trasformando i punti di controllo e lasciando i pesi inalterati.

Sezioni coniche

Mostriamo come una qualunque sezione conica può essere rappresentata mediante una curva NURBS di grado 2, con vettore dei nodi $T = (0,0,0,1,1,1)$ e tre punti di controllo P_0, P_1, P_2 , lavorando unicamente sul peso w_1 (avendo posto $w_0 = w_2 = 1$). In particolare, se:

- $w_1 = 1$, **otteniamo una parabola;**
- $w_1 < 1$, **otteniamo un'ellisse;**
- $w_1 > 1$, **otteniamo una iperbole;**
- $w_1 = 0$, **otteniamo una retta.**



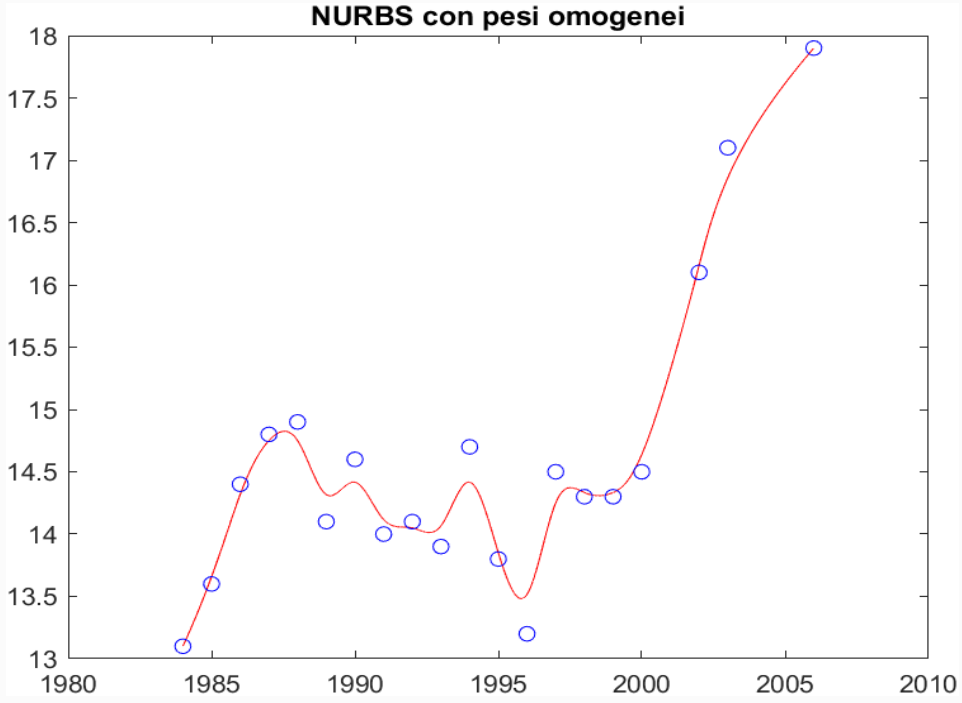
Sezioni coniche

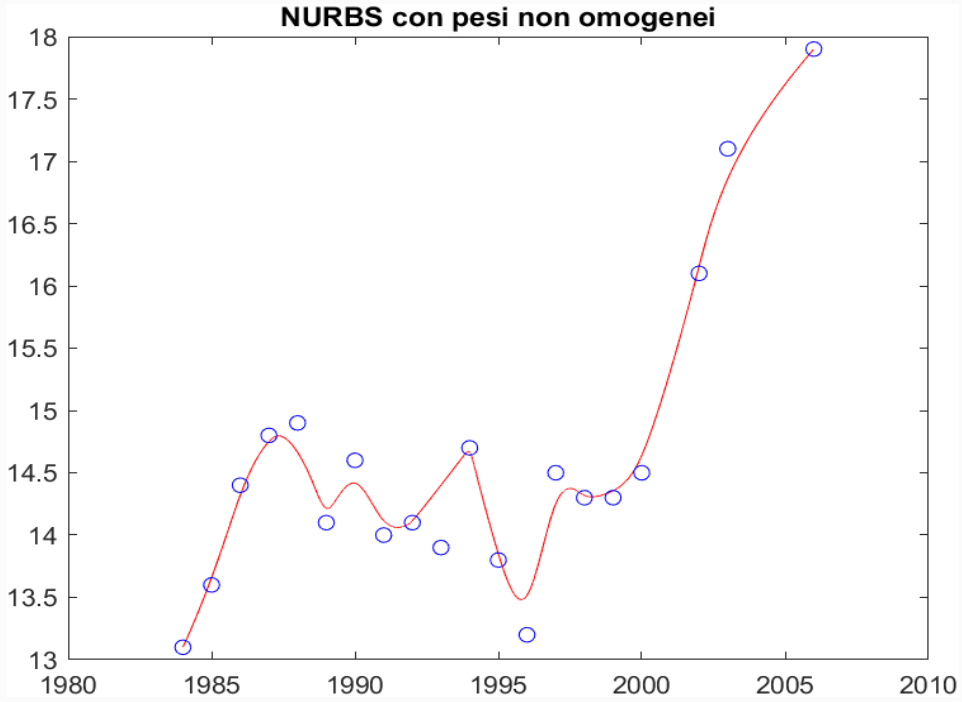
- $w_1 = 1$, parabola;
- $w_1 < 1$, ellisse;
- $w_1 > 1$, iperbole;
- $w_1 = 0$,retta.

Caso studio

```
p = tavg';

n_punti = length(p(1,:));

h = 3;

w1 = ones(1,n_punti);

t = [zeros(1,h+1) 1/(n_punti-h)*(1:n_punti-h-1) ones(1,h+1)];

u = linspace(0,1,1e4);

c = nurbsfun(h+1,t,w1,p,u);

plot(c(1,:),c(2,:),'r');

title('NURBS con pesi omogenei')

hold on

scatter(p(1,:),p(2,:),'bo')

%utilizzando pesi disomogenei

w2 = [0.2*ones(1,5) ones(1,4) 0 5*ones(1,5) ones(1,5)];

c = nurbsfun(h+1,t,w2,p,u);

figure(2)

plot(c(1,:),c(2,:),'r');

title('NURBS con pesi non omogenei')

hold on

scatter(p(1,:),p(2,:),'bo')
```

Caso studio


NURBS con pesi omogenei
18
17.5
17
16.5
16
15.5
15
14.5
14
13.5
13
1980
1985
1990
1995
2000
2005
2010


NURBS con pesi non omogenei
18
17.5
17
16.5
16
15.5
15
14.5
14
13.5
13
1980
1985
1990
1995
2000
2005
2010

Interpolazione e qualità delle immagini

Vedremo come è possibile, attraverso l'interpolazione, migliorare la qualità di immagini sgranate.

Interpolazione e qualità delle immagini

```
%Caricamento immagine ridimensionata
img = imresize(imread('cane.jpg'), [64 NaN]);
[r,c,v] = size(img);
```

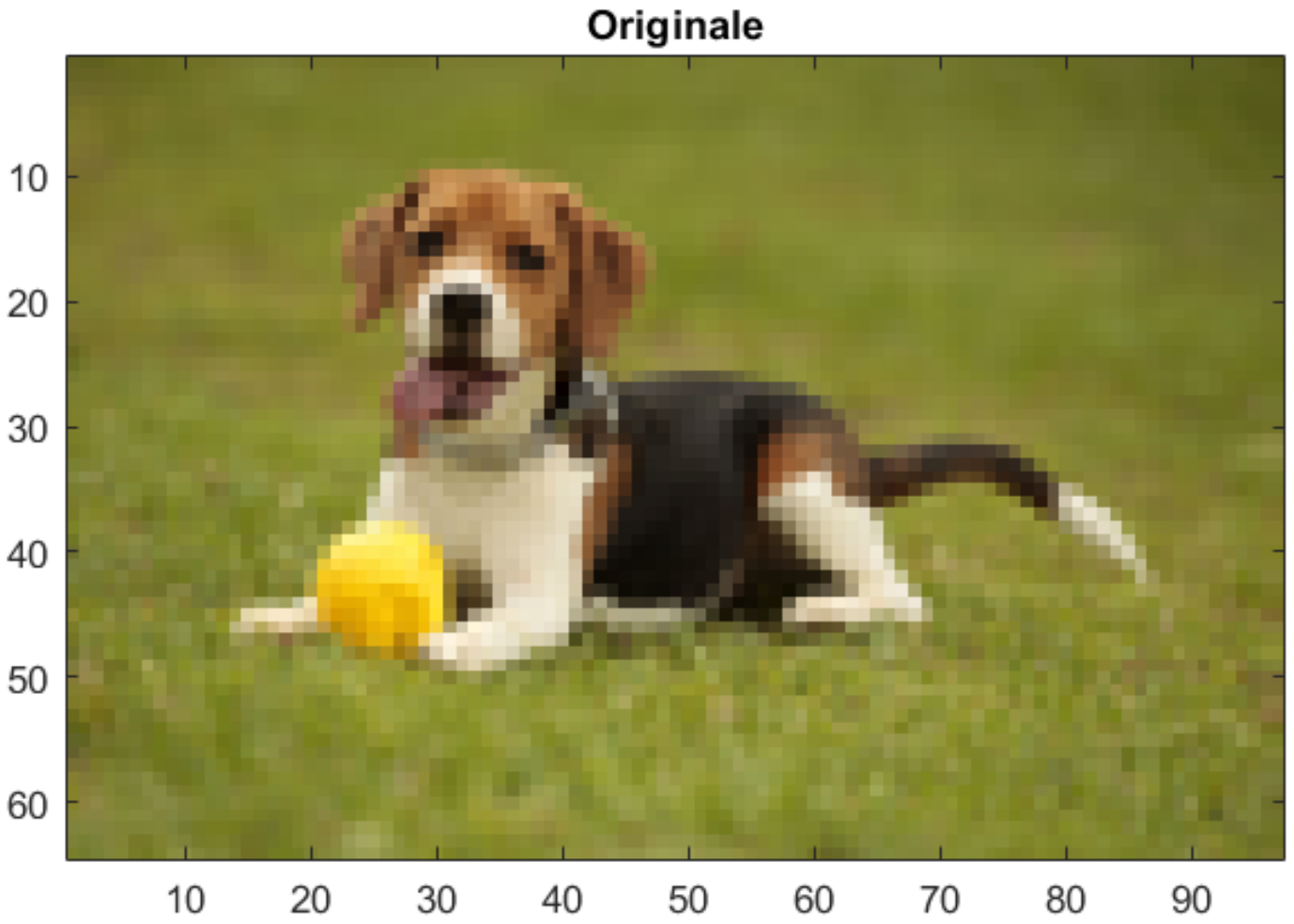
Preallochiamo le nuove immagini e generiamo una griglia di pixel più fitta. Saranno preallocate due immagini, poiché mostreremo un miglioramento dell'immagine sia attraverso l'interpolazione lineare sia attraverso l'interpolazione Spline:

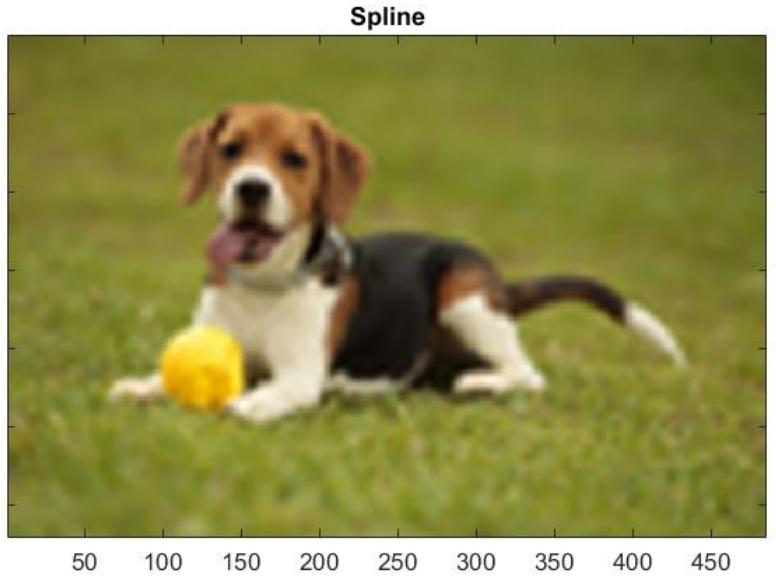
```
%Preallocazione delle nuove immagini
NEW_I = zeros(5*r,5*c,v,'uint8');
NEW_I_L = zeros(5*r,5*c,v,'uint8');

%Determino la griglia dell'immagine   di  partenza
x = 1:c;
y = 1:r;

%Determino la griglia dell'immagine  che voglio  ottenere
new_x = linspace(1,c,5*c)';
new_y = linspace(1,r,5*r);
```

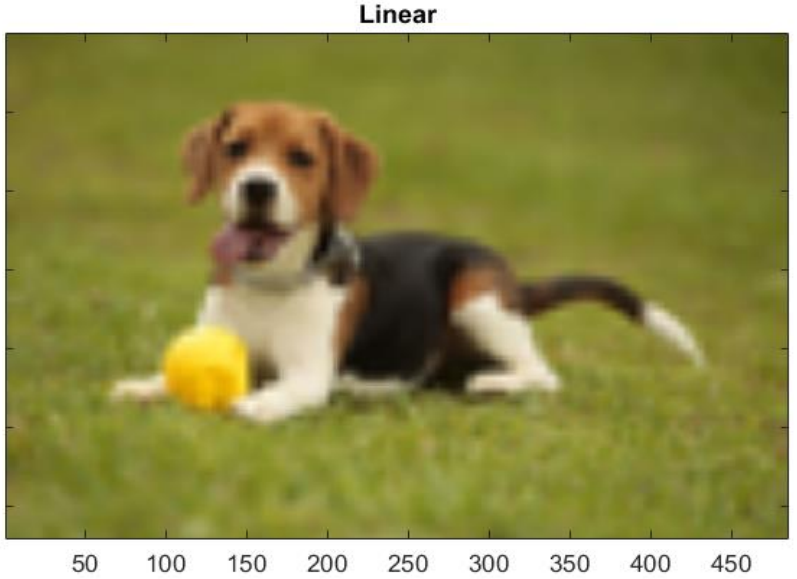
Interpolazione e qualità delle immagini

Effettuiamo l'interpolazione sulla griglia originale, attraverso la funzione *interp2:*

```
%Generazione    tramite interpolazione  delle due   nuove   immagini
for i=1:3

NEW_I(:,:,i)=uint8(interp2(x,y,double(img(:,:,i)),new_x,new_y,'Spline'));

NEW_I_L(:,:,i)=uint8(interp2(x,y,double(img(:,:,i)),new_x,new_y,'linear'));
end
```


Originale
10
20
30
40
50
60
10
20
30
40
50
60
70
80
90


Spline
50
100
150
200
250
300
350
400
450


Linear
50
100
150
200
250
300
350
400
450