B-spline matlab

bspline

R2020a

Plot B-spline and its polynomial pieces

collapse all in page

Syntax

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

Description

example

`bspline(t)` plots the B-spline with knot sequence `t`, as well as the polynomial pieces of which it is composed. For more information about spline fitting, see About Splines in Curve Fitting Toolbox.

Plot curva b-spline

Tracciamo una curva B-spline assegnando una sequenza di nodi t.

Assegniamo la seguente sequenza di nodi

t = [0 1.5 2.3 4 5];

Plot curva b-spline

Tracciamo una curva B-spline assegnando una sequenza di nodi t.

Assegniamo la seguente sequenza di nodi

```
t = [0 1.5 2.3 4 5];
```

Richiamiamo la function matlab bspline con input
Il vettore dei nodi t, utilizzando la pp-form

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

Stampa della ppform

```
pp =
struct with fields:      form: 'pp'
breaks: [0 1.5000 2.3000 4 5]
coefs: [4×4 double]
pieces: 4
order: 4
```

Plot curva b-spline

Tracciamo una curva B-spline assegnando una sequenza di nodi t.

Assegniamo la seguente sequenza di nodi

t = [0 1.5 2.3 4 5];

Richiamiamo la function matlab bspline con input
Il vettore dei nodi t, utilizzando la pp-form

pp =bspline(t)

Stampa della ppform

```
pp =
struct with fields:      form: 'pp'
breaks: [0 1.5000 2.3000 4 5]
coefs: [4×4 double]
pieces: 4
order: 4
```

Vettore contenente elementi che rappresentano l'inizio e
La fine di ciascuno degli intervalli su cui sono definiti i polinomi

fnplt(pp)   (oppute bspline(t))

Plot curva b-spline

Tracciamo una curva B-spline assegnando una sequenza di nodi t.

Assegniamo la seguente sequenza di nodi

```
t = [0 1.5 2.3 4 5];
```

Richiamiamo la function matlab bspline con input
Il vettore dei nodi t, utilizzando la pp-form

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

Stampa della ppform

```
pp =
struct with fields:      form: 'pp'
breaks: [0 1.5000 2.3000 4 5]
coefs: [4×4 double]
pieces: 4
order: 4
```

Vettore contenente elementi che rappresentano l'inizio e
La fine di ciascuno degli intervalli su cui sono definiti i polinomi

Coefficienti dei polinomi «locali»

Plot curva b-spline

Tracciamo una curva B-spline assegnando una sequenza di nodi t.

Assegniamo la seguente sequenza di nodi

t = [0 1.5 2.3 4 5];

Richiamiamo la function matlab bspline con input
Il vettore dei nodi t, utilizzando la pp-form

pp =bspline(t)

Stampa della ppform

```
pp =
struct with fields:     form: 'pp'
breaks: [0 1.5000 2.3000 4 5]
coefs: [4×4 double]
pieces: 4
order: 4
```

Vettore contenente elementi che rappresentano l'inizio e
La fine di ciascuno degli intervalli su cui sono definiti i polinomi

Coefficienti dei polinomi «locali»

4 polinomi «locali»

Plot curva b-spline

Tracciamo una curva B-spline assegnando una sequenza di nodi t.

Assegniamo la seguente sequenza di nodi

t = [0 1.5 2.3 4 5];

Richiamiamo la function matlab bspline con input
Il vettore dei nodi t, utilizzando la pp-form

pp =bspline(t)

Stampa della ppform

```
pp =
struct with fields:      form: 'pp'
breaks: [0 1.5000 2.3000 4 5]
coefs: [4×4 double]
pieces: 4
order: 4
```

Vettore contenente elementi che rappresentano l'inizio e
La fine di ciascuno degli intervalli su cui sono definiti i polinomi

Coefficienti dei polinomi «locali»

4 polinomi «locali»

Ordine della curva bspline

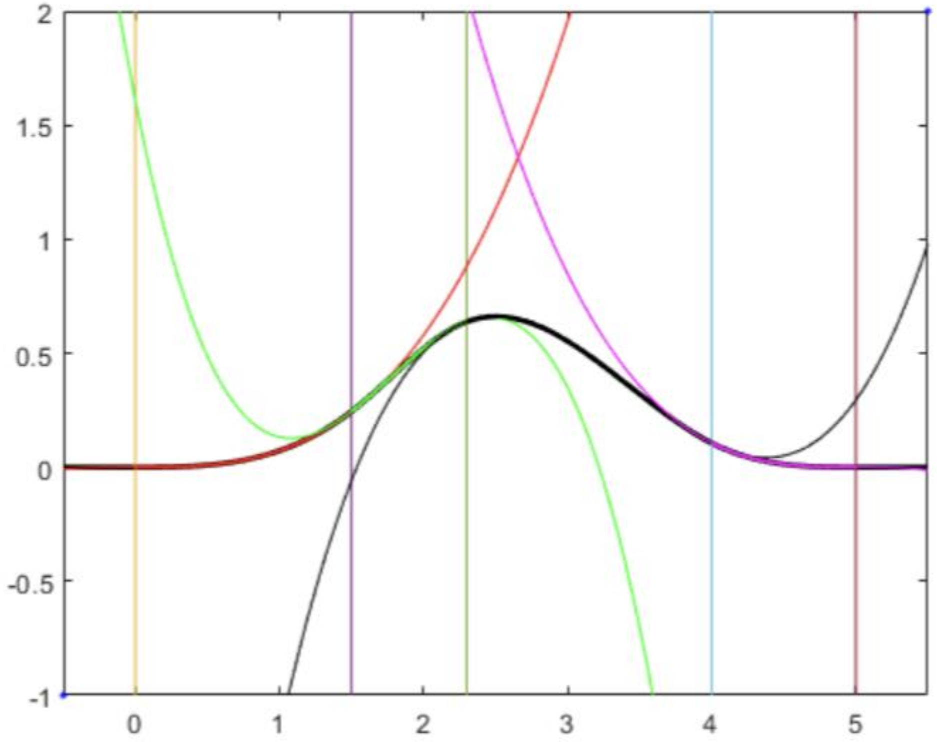
Plot curva b-spline

Per il grafico della curva b-spline possiamo digitare direttamente

bspline(t)

Oppure se utilizziamo la pp-form

fnplt(pp)



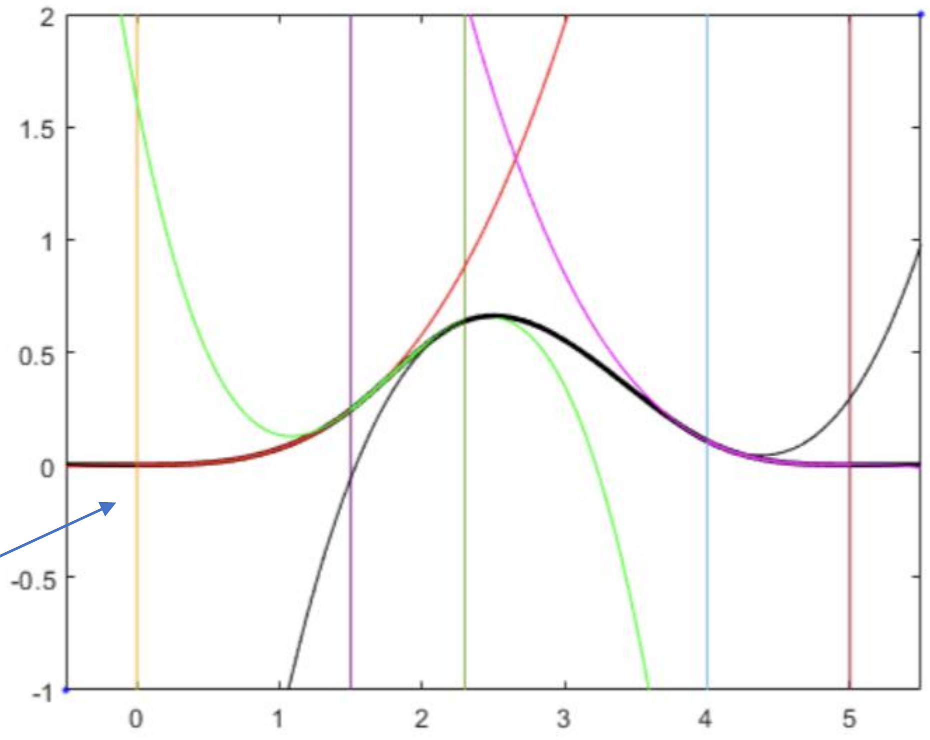
Plot curva b-spline

Per il grafico della curva b-spline possiamo digitare direttamente

bspline(t)

Oppure se utilizziamo la pp-form

fnplt(pp)



curva evidenziata in nero è la
Curva B-spline

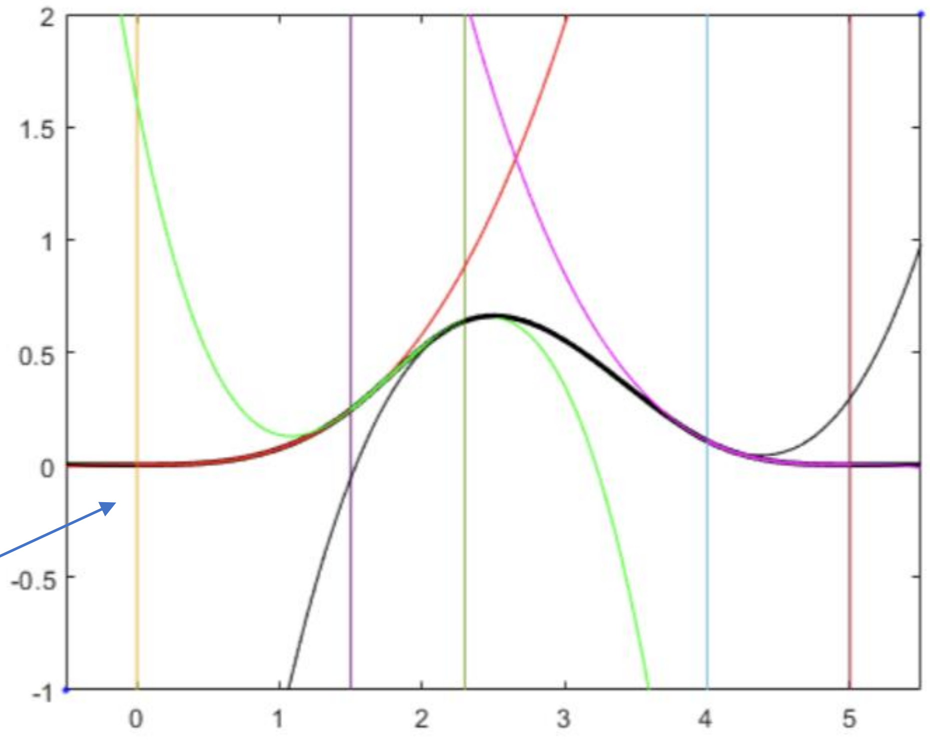
Plot curva b-spline

Per il grafico della curva b-spline possiamo digitare direttamente

bspline(t)

Oppure se utilizziamo la pp-form

fnplt(pp)



curva evidenziata in nero è la
Curva B-spline

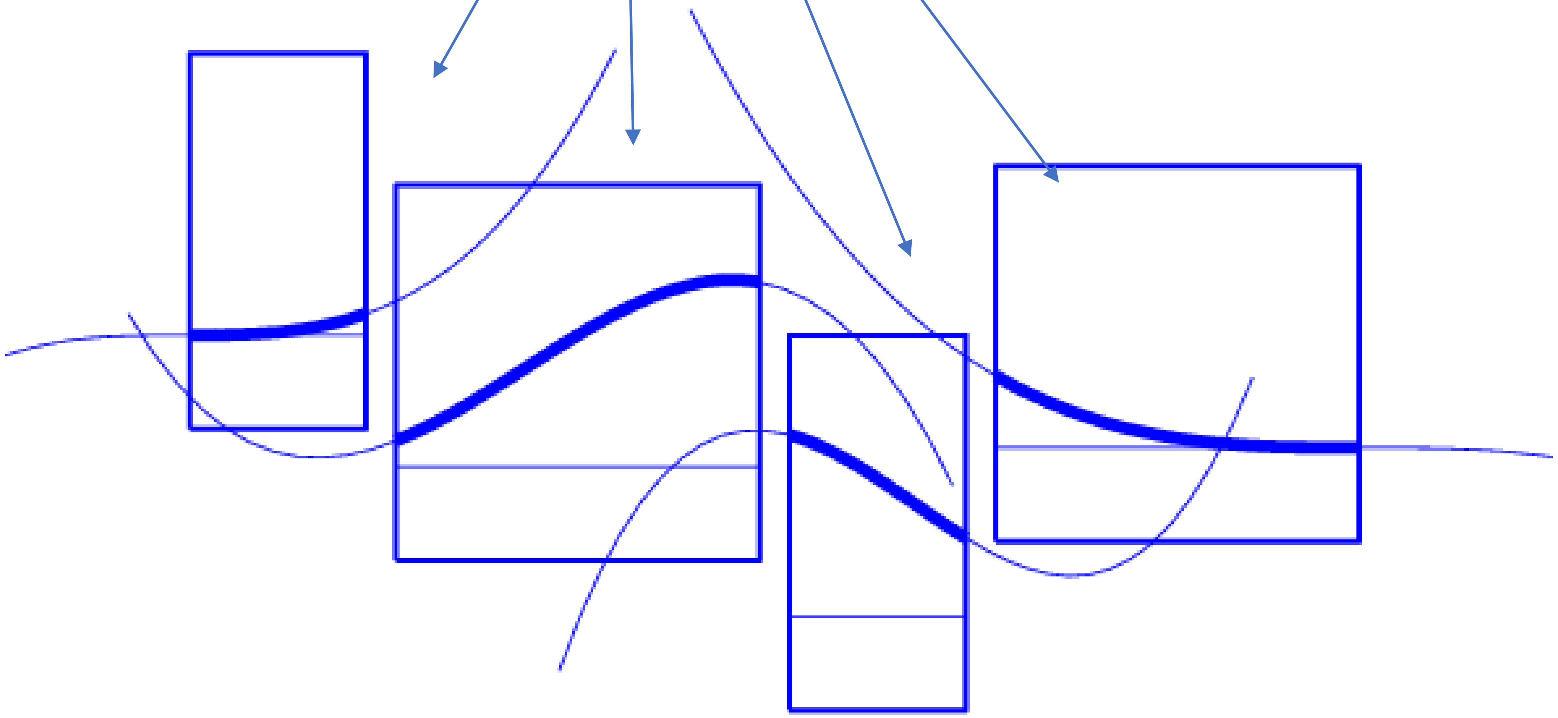
Evidenziamo le funzioni bsline

Plot curva b-spline

Vettore breaks

t = [0 1.5 2.3 4 5];

4 funzioni b-spline

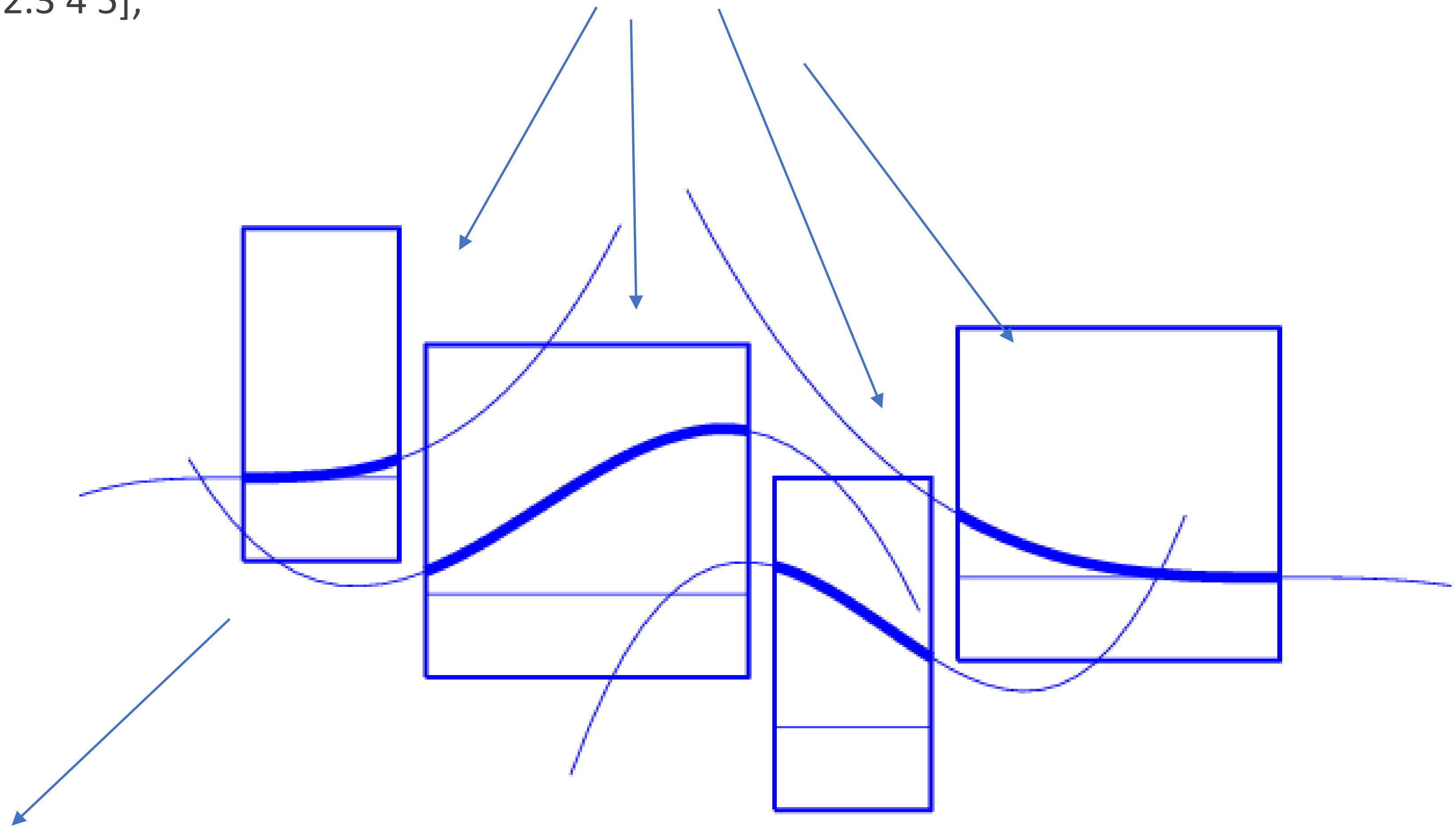


Plot curva b-spline

Vettore breaks

t = [0 1.5 2.3 4 5];

4 funzioni b-spline

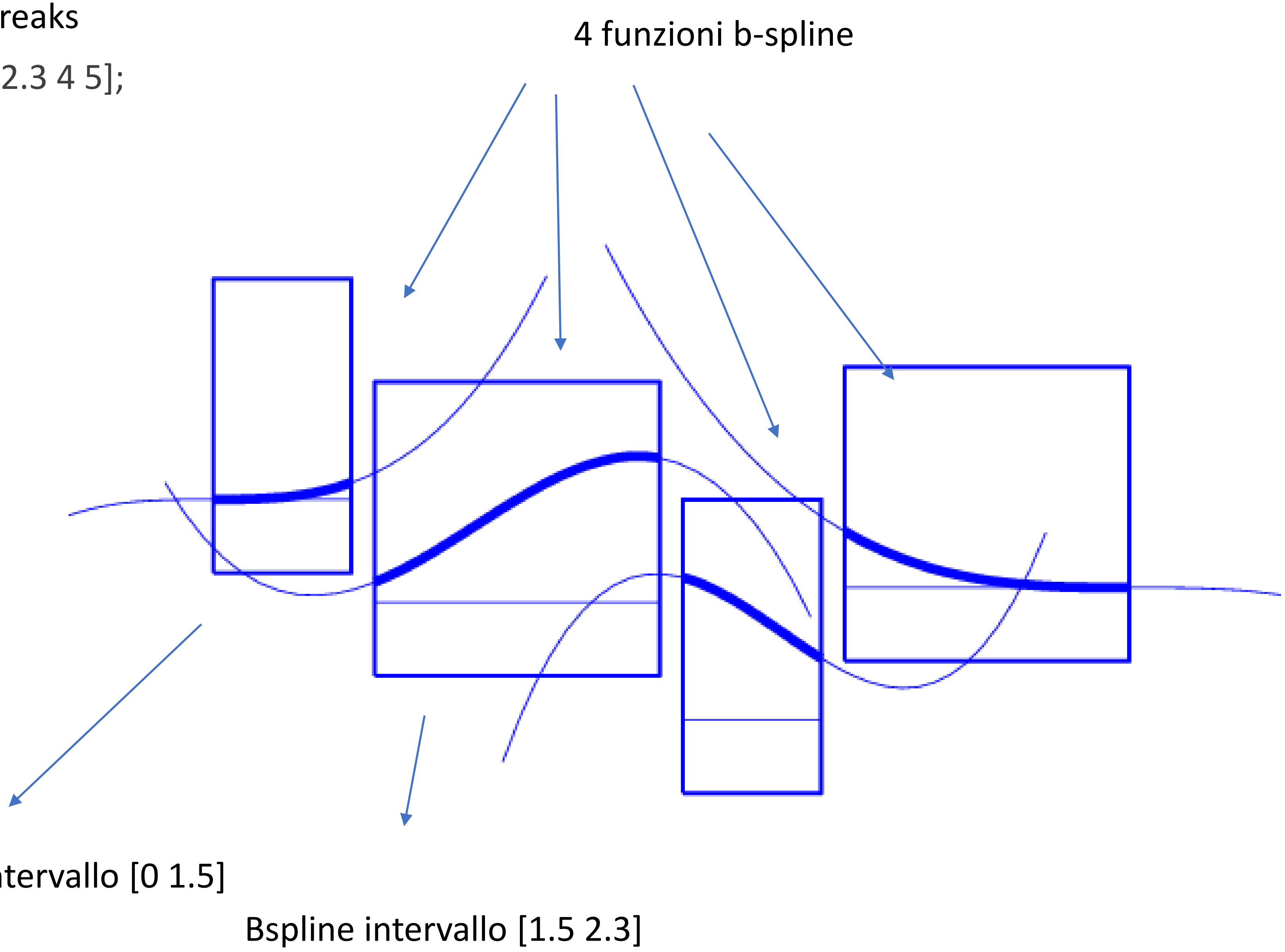


Bspline intervallo [0 1.5]

Plot curva b-spline

Vettore breaks

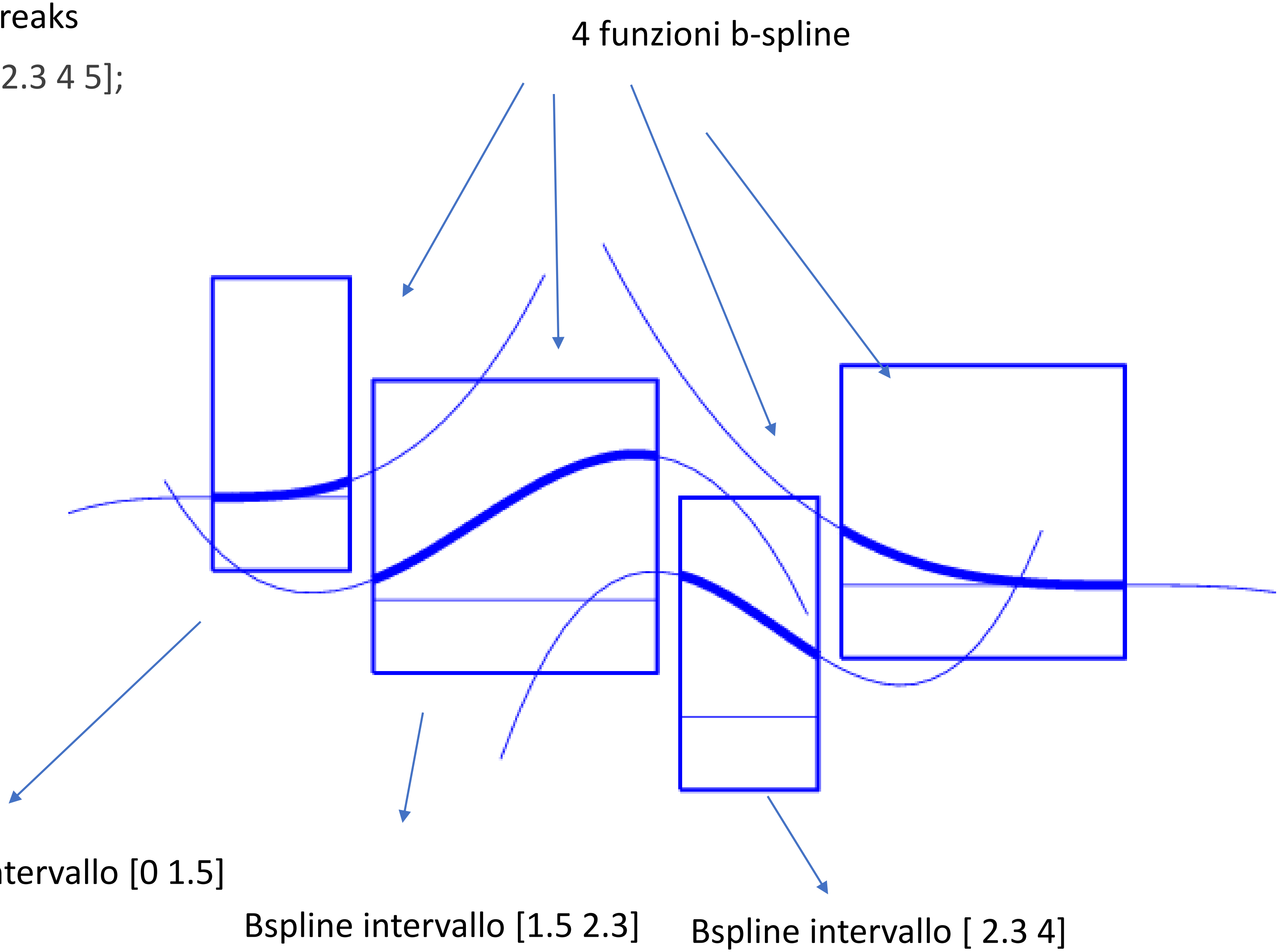
t = [0 1.5 2.3 4 5];



Plot curva b-spline

Vettore breaks

t = [0 1.5 2.3 4 5];

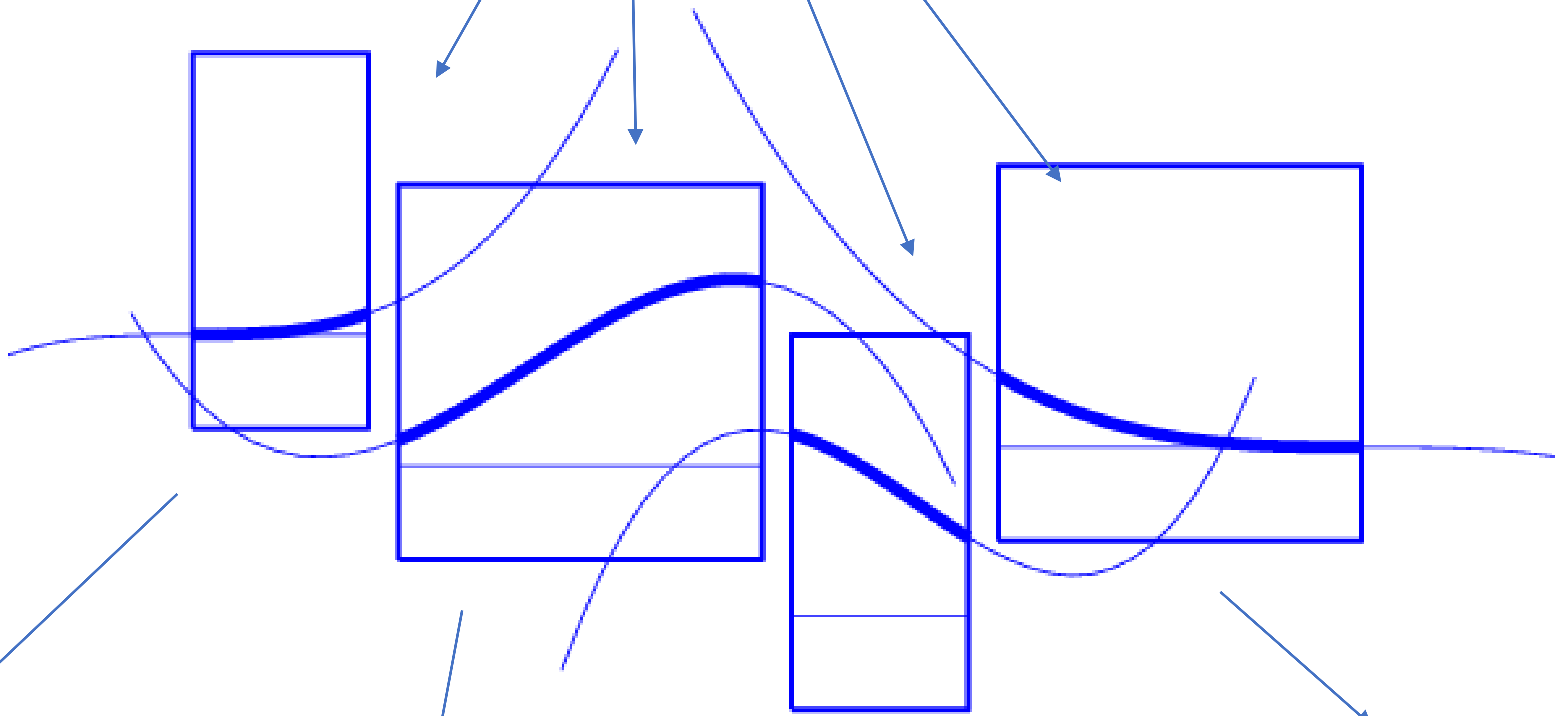


Plot curva b-spline

Vettore breaks

t = [0 1.5 2.3 4 5];

4 funzioni b-spline



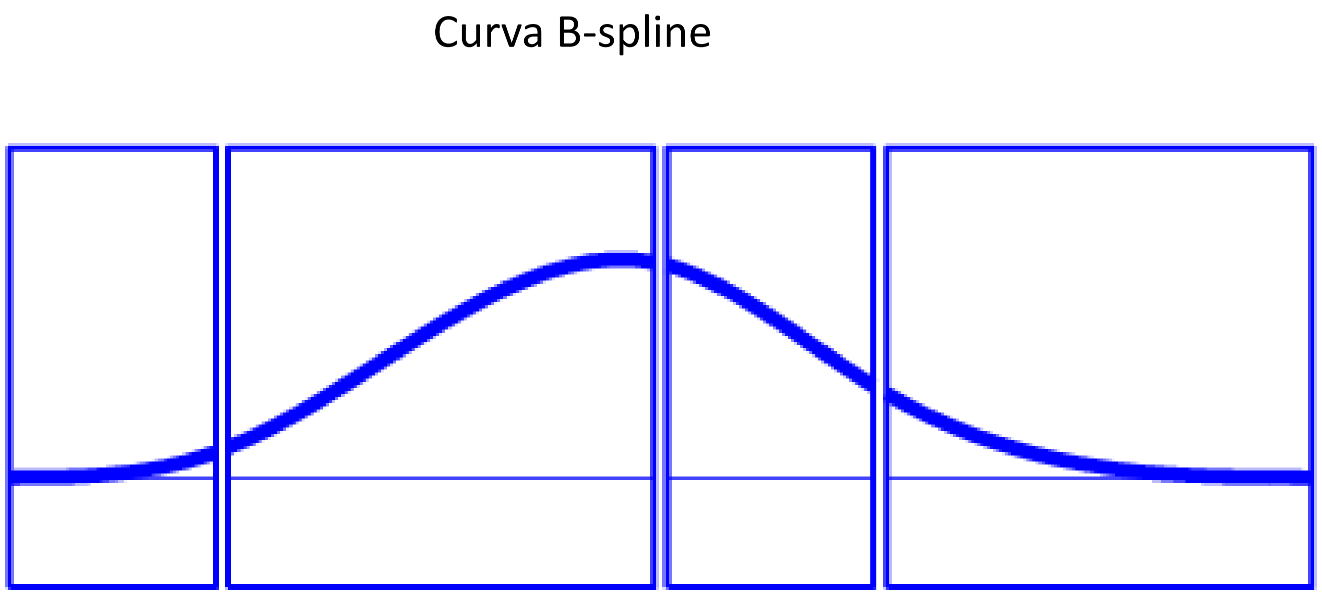
Bspline intervallo [0 1.5]

Bspline intervallo [1.5 2.3]

Bspline intervallo [2.3 4]

Bspline intervallo [4 5]

Plot curva b-spline



Matlab function bsplinepolytraj per generare una traiettoria polinomiale utilizzando le B-spline cubica

Resources ▾

bsplinepolytraj

R2020a

Generate polynomial trajectories using B-splines

collapse all in page

Syntax

```
[q,qd,qdd,pp] = bsplinepolytraj(controlPoints,tInterval,tSamples)
```

Description

example

`[q,qd,qdd,pp] = bsplinepolytraj(controlPoints,tInterval,tSamples)` generates a piecewise cubic B-spline trajectory that falls in the control polygon defined by `controlPoints`. The trajectory is uniformly sampled between the start and end times given in `tInterval`. The function returns the positions, velocities, and accelerations at the input time samples, `tSamples`.

Generare una b-spline cubica utilizzando la routine bsplinepolytraj

Scelgo i seguenti punti di controllo
>> cpts = [1 4 4 3 -2 0; 0 1 2 4 3 1];

Scelgo l'intervallo temporale inserendo solamente l'istante iniziale e finale

tpts = [0 5];

Campioni temporali per la traiettoria, specificati come vettore;

>> tvec = 0:0.01:5;

Richiamo la routine matlab per tracciare il plot della curva bsline
[q, ~,~,~,pp] = bsplinepolytraj(cpts,tpts,tvec);

Output nella pp-form

```
pp =   struct with fields:
  form: 'pp'
  breaks: [-1 0 1.6667 3.3333 5 6]
  coefs: [10×4 double]
  pieces: 5
   order: 4
```

Plot curva b-spline

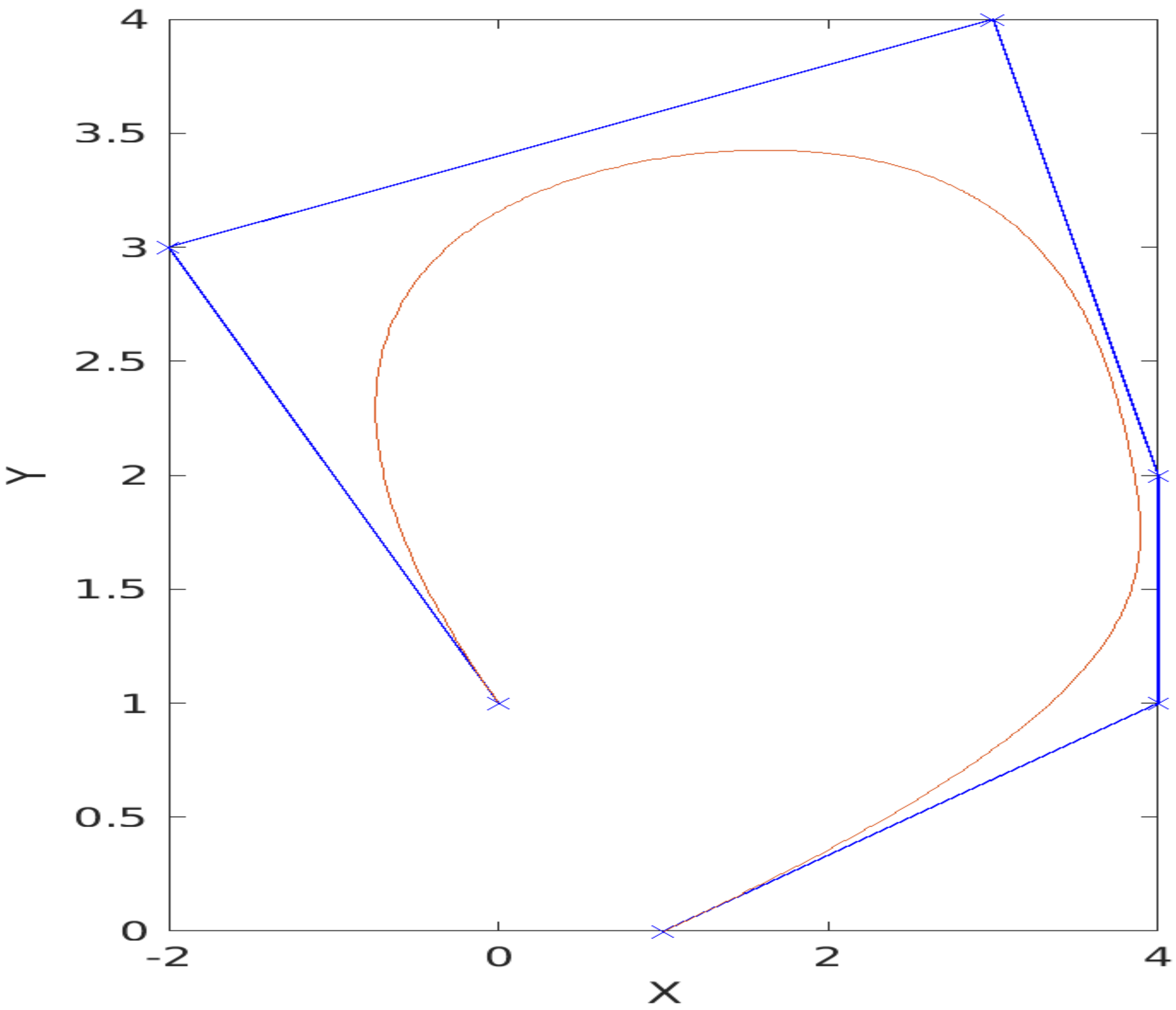
```
>> figure
```
Grafico dei punti di controllo

```
plot(cpts(1,:),cpts(2,:),'xb-')
```
Sovrapposizione dei grafici

```
hold all
```

Plot della curva bspline

```
fnplt(pp)

xlabel('X')
ylabel('Y')
```



Plot curva b-spline

Come ottenere una curva b-spline chiusa?

Il punto iniziale P0 e P5 devono coincidere.

coef=

$$
\begin{array}{cccccc}
1 & 4 & 4 & 3 & -2 & 1 \\
0 & 1 & 2 & 4 & 3 & 0 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
P0 & P1 & P2 & P3 & P4 & P5
\end{array}
$$

plot(cpts(1,:),cpts(2,:),'xb-')
Sovrapposizione dei grafici
hold all

Plot della curva bspline
fnplt(pp)

xlabel('X')
ylabel('Y')

Plot curva b-spline

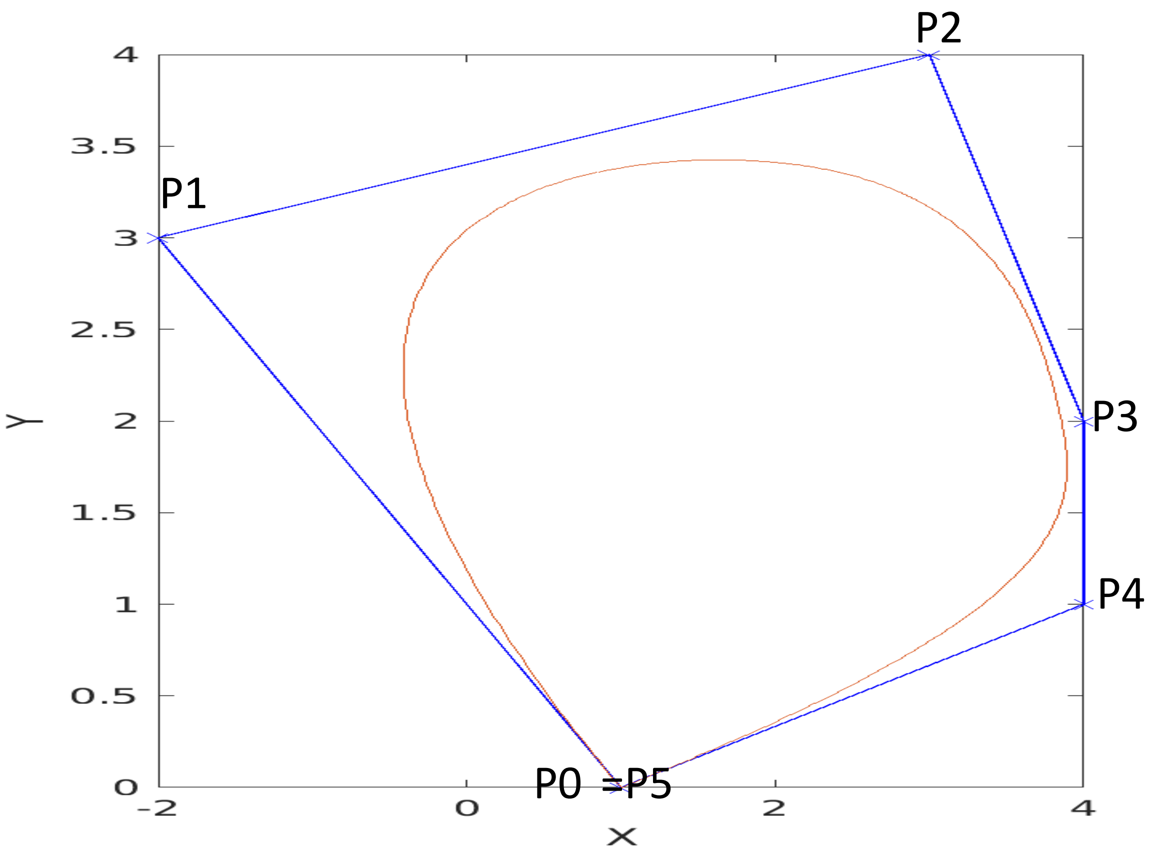
Come ottenere una curva b-spline chiusa?

Il punto iniziale P0 e P5 devono coincidere.

Plot curva b-spline
plot(cpts(1,:),cpts(2,:),'xb-')
Sovrapposizione dei grafici
hold all

Plot della curva bspline
fnplt(pp)

xlabel('X')
ylabel('Y')



Plot curva b-spline

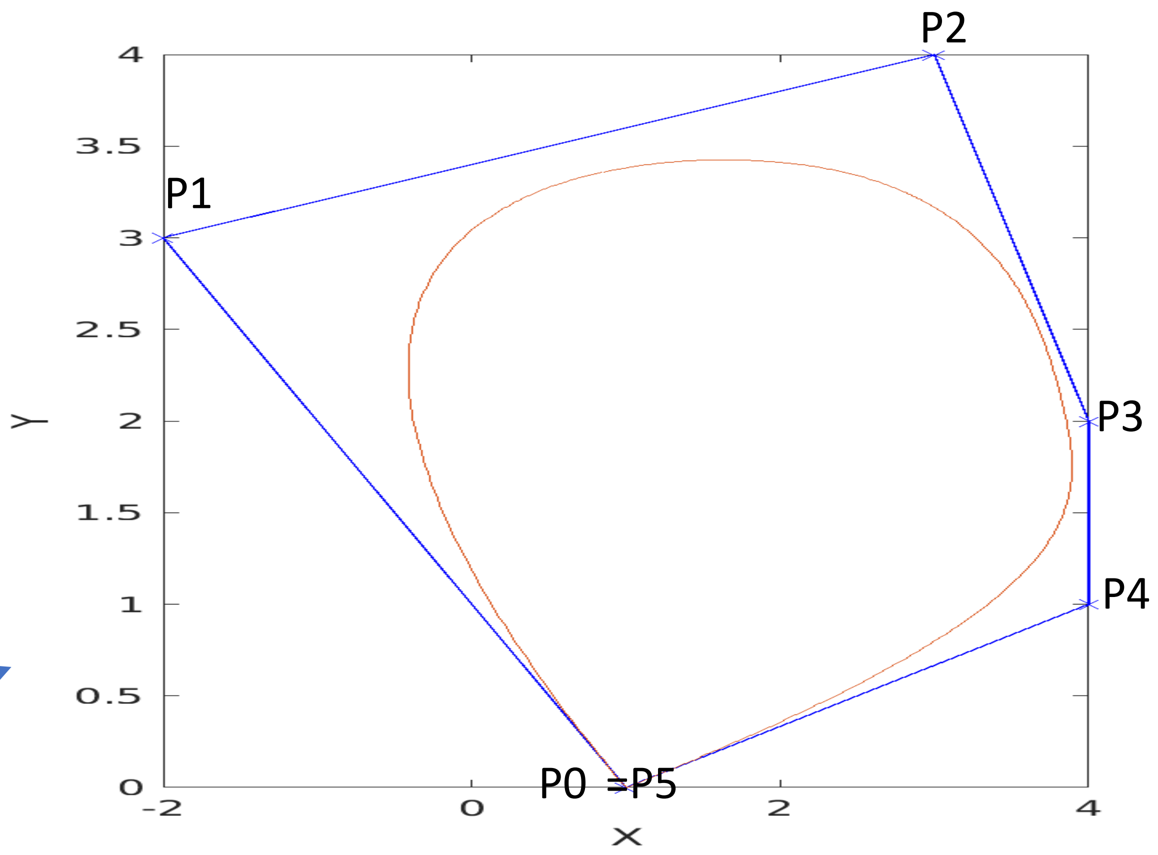
Come ottenere una curva b-spline chiusa?

Il punto iniziale P0 e P5 devono coincidere.

Plot curva b-spline
plot(cpts(1,:),cpts(2,:),'xb-’)
Sovrapposizione dei grafici
hold all

Plot della curva bspline
fnplt(pp)

xlabel('X’)
ylabel('Y')



Curva chiusa

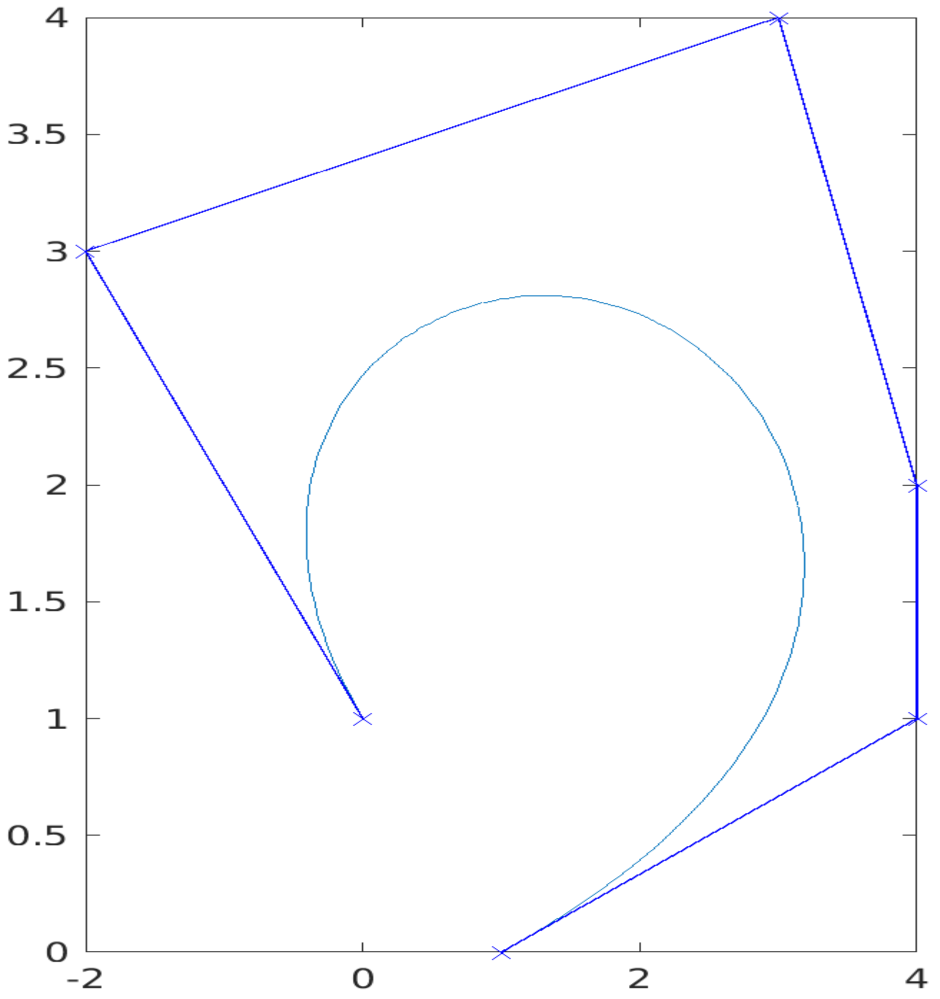
Confronto curva bspline e curva di Beizer

Consideriamo i seguenti punti di controllo

```
>> cpts = [1 4 4 3 -2 1; 0 1 2 4 3 0];
```

```
cpts =

   1   4   4   3  -2   1
   0   1   2   4   3   0
```

P0 P1 P2 P3 P4 P5

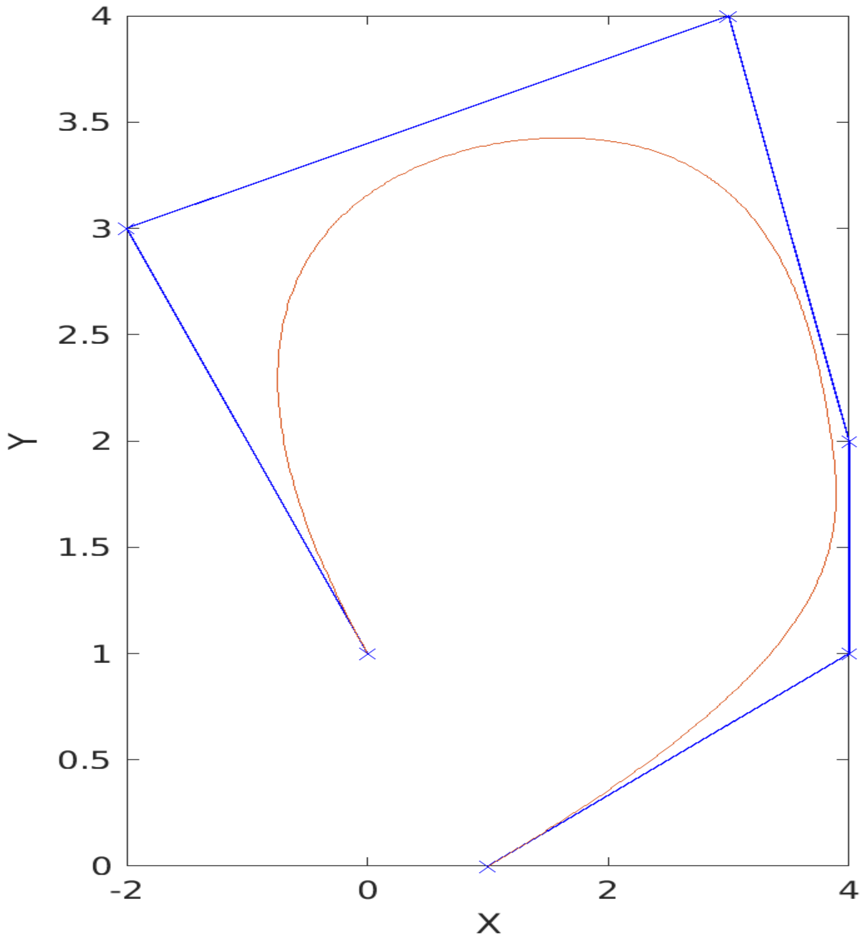


Generiamo la curva di Beizer a partire dai polinomi di bernstein

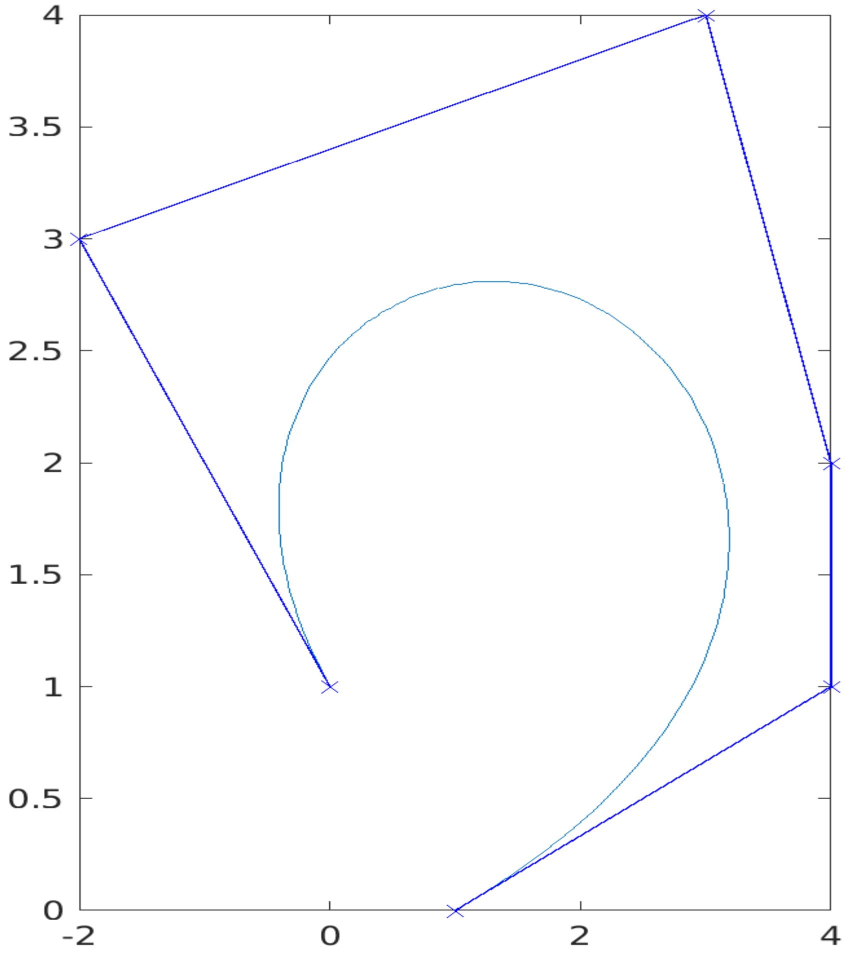
```
>> syms t
B = bernsteinMatrix(5, t);
>> bezierCurve = simplify(B*cpts')
```

Confronto curva bspline e curva di Beizer

Curva B-spline



Curva di Bezier



Confronto curva bspline e curva di Beizer

Genero **curva b-spline** considerando i seguenti punti di controllo

cpts = [1 4 4 3 -2 1; 0 1 2 4 3 0];

cpts =

$$\begin{array}{cccccc} 1 & 4 & 4 & 3 & -2 & 1 \\ 0 & 1 & 2 & 4 & 3 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ P0 & P1 & P2 & P3 & P4 & P5 \end{array}$$

E considerando i seguenti punti di controllo

cpts’ = [1 4 5 3 -2 1; 0 1 2 4 3 0];

cpts’ =

$$\begin{array}{cccccc} 1 & 4 & 5 & 3 & -2 & 1 \\ 0 & 1 & 2 & 4 & 3 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ P0' & P1' & P2' & P3' & P4' & P5' \end{array}$$

P0≡P0’

P1≡P1’

P2≠P2’

P3≡P3’

P4≡P4’

P5≡P5’

Confronto curva bspline e curva di Beizer

Genero **curva b-spline** considerando i seguenti punti di controllo

cpts = [1 4 4 3 -2 1; 0 1 2 4 3 0];

cpts =

$$\begin{matrix} 1 & 4 & 4 & 3 & -2 & 1 \\ 0 & 1 & 2 & 4 & 3 & 0 \end{matrix}$$

P0 P1 P2 P3 P4 P5

E considerando i seguenti punti di controllo

cpts’ = [1 4 5 3 -2 1; 0 1 2 4 3 0];

cpts’ =

$$\begin{matrix} 1 & 4 & 5 & 3 & -2 & 1 \\ 0 & 1 & 2 & 4 & 3 & 0 \end{matrix}$$

P0’ P1’ P2’ P3’ P4’ P5’

P0≡P0’

P1≡P1’

$\boxed{P2 \neq P2'}$ Cambia solamente un punto

P3≡P3’

P4≡P4’

P5≡P5’

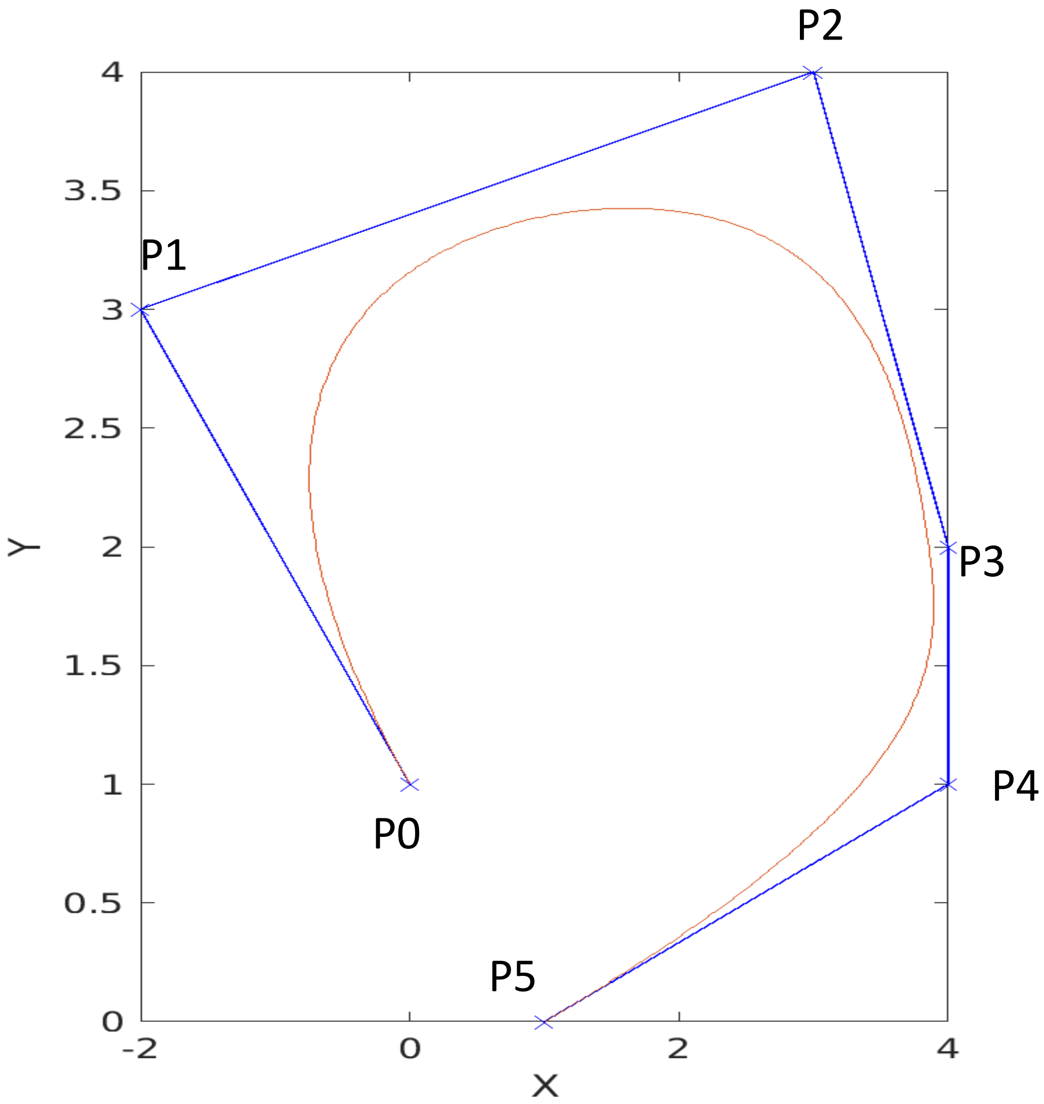
Confronto curva bspline e curva di Beizer

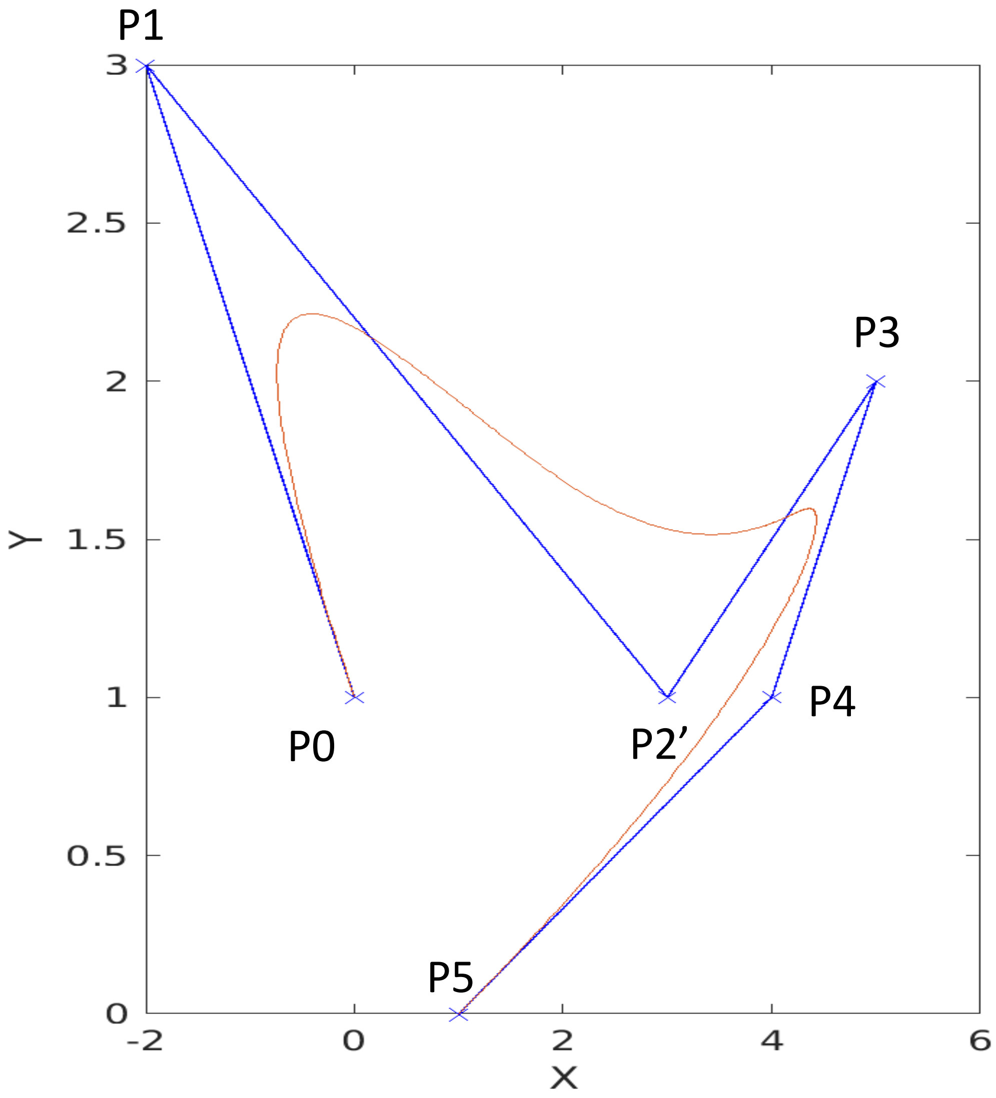
Cambio solamente un punto

Consideriamo i seguenti punti di controllo (1 punto diverso)

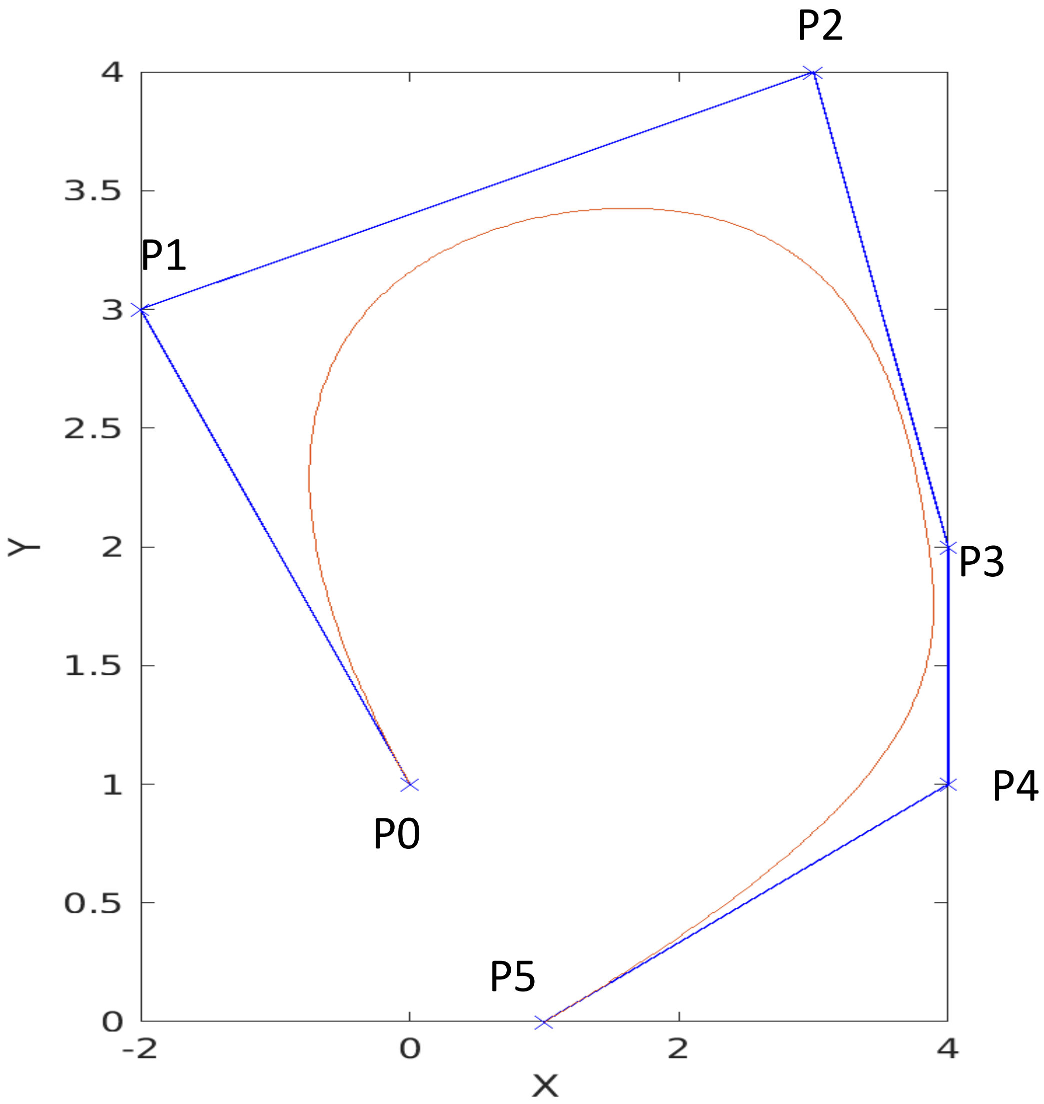
```
>>cpts = [1 4 4 3 -2 1; 0 1 2 4 3 0];
>> cpts1 = [1 4 5 3 -2 0; 0 1 2 1 3 1];
>> [q, ~,~, ~, pp] = bsplinepolytraj(cpts,tpts,tvec);
```

Confronto curva bspline e curva di Beizer

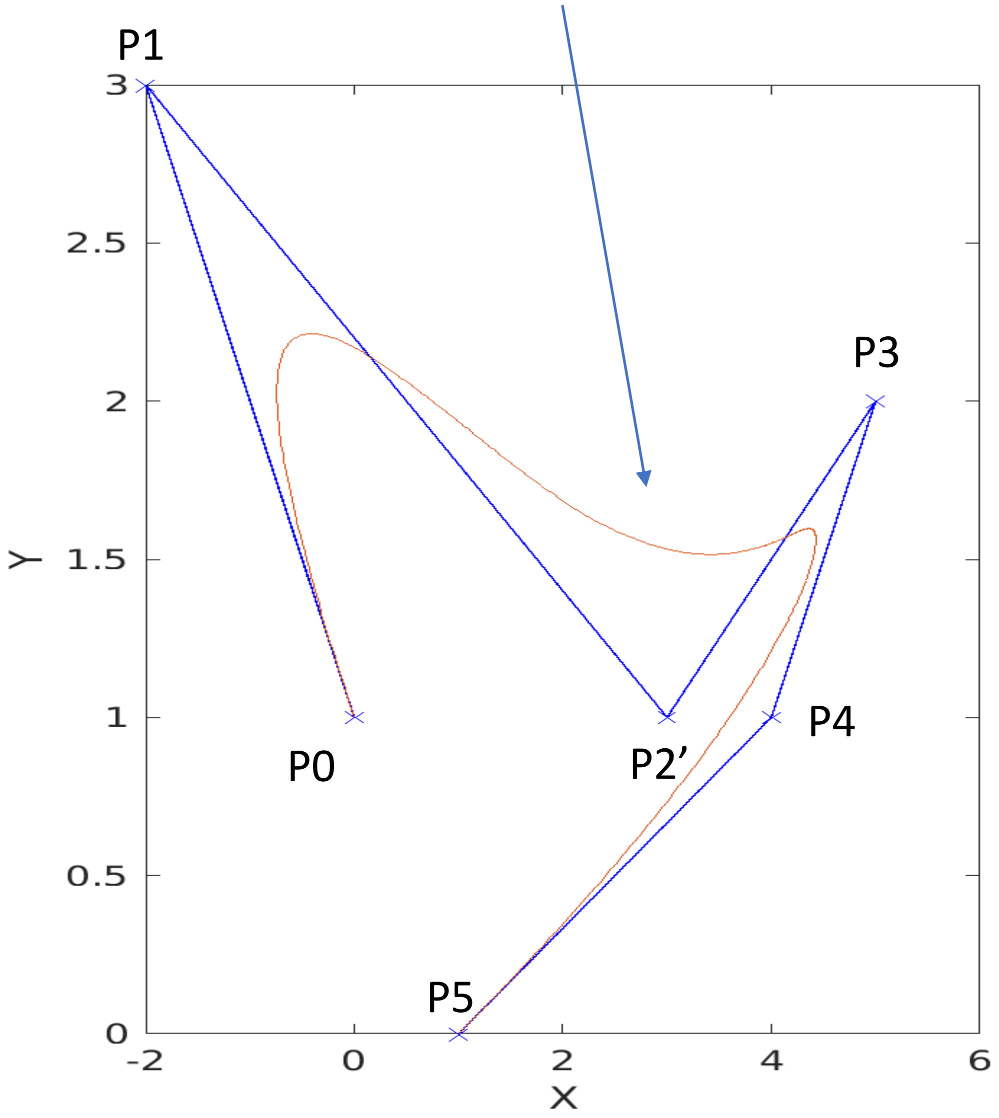




Confronto curva bspline e curva di Beizer



Cambiamento locale



Confronto curva bspline e curva di Beizer

Genero **curva di Beizer** considerando i seguenti punti di controllo

cpts = [1 4 4 3 -2 1; 0 1 2 4 3 0];

cpts =

1	4	4	3	-2	1
0	1	2	4	3	0

P0 P1 P2 P3 P4 P5

E considerando i seguenti punti di controllo

cpts1= [1 4 5 3 -2 1; 0 1 2 4 3 0];

cpts1=

1	4	5	3	-2	1
0	1	2	4	3	0

P0’ P1’ P2’ P3’ P4’ P5’

P0≡P0’

P1≡P1’

P2≠P2’

P3≡P3’

P4≡P4’

P5≡P5’

Confronto curva bspline e curva di Beizer

Genero **curva di Beizer** considerando i seguenti punti di controllo

cpts = [1 4 4 3 -2 1; 0 1 2 4 3 0];

cpts =

$$\begin{array}{cccccc} 1 & 4 & 4 & 3 & -2 & 1 \\ 0 & 1 & 2 & 4 & 3 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ P0 & P1 & P2 & P3 & P4 & P5 \end{array}$$

E considerando i seguenti punti di controllo

cpts1= [1 4 5 3 -2 1; 0 1 2 4 3 0];

cpts1=

$$\begin{array}{cccccc} 1 & 4 & 5 & 3 & -2 & 1 \\ 0 & 1 & 2 & 4 & 3 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ P0' & P1' & P2' & P3' & P4' & P5' \end{array}$$

P0≡P0’

P1≡P1’

$\boxed{\text{P2≠P2’}}$ Cambia solamente un punto

P3≡P3’

P4≡P4’

P5≡P5’

Confronto curva bspline e curva di Beizer

Curve di Bezier relativa ai seguenti
punti di controllo

```
cpts = [1 4 4 3 -2 0; 0 1 2 1 3 1];
```

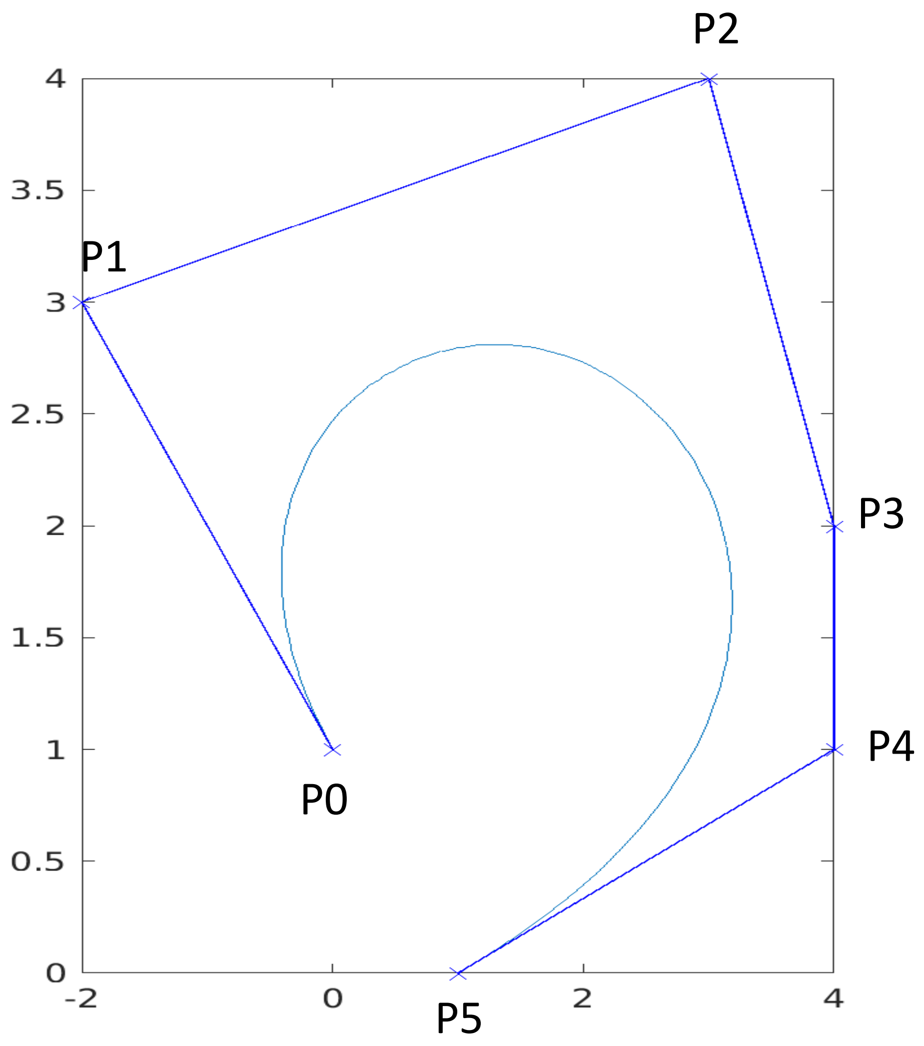
```
>> syms t
B = bernsteinMatrix(5, t);
>> bezierCurve = simplify(B*cpts’)
>> fplot(bezierCurve(1), bezierCurve(2), [0, 1])
```

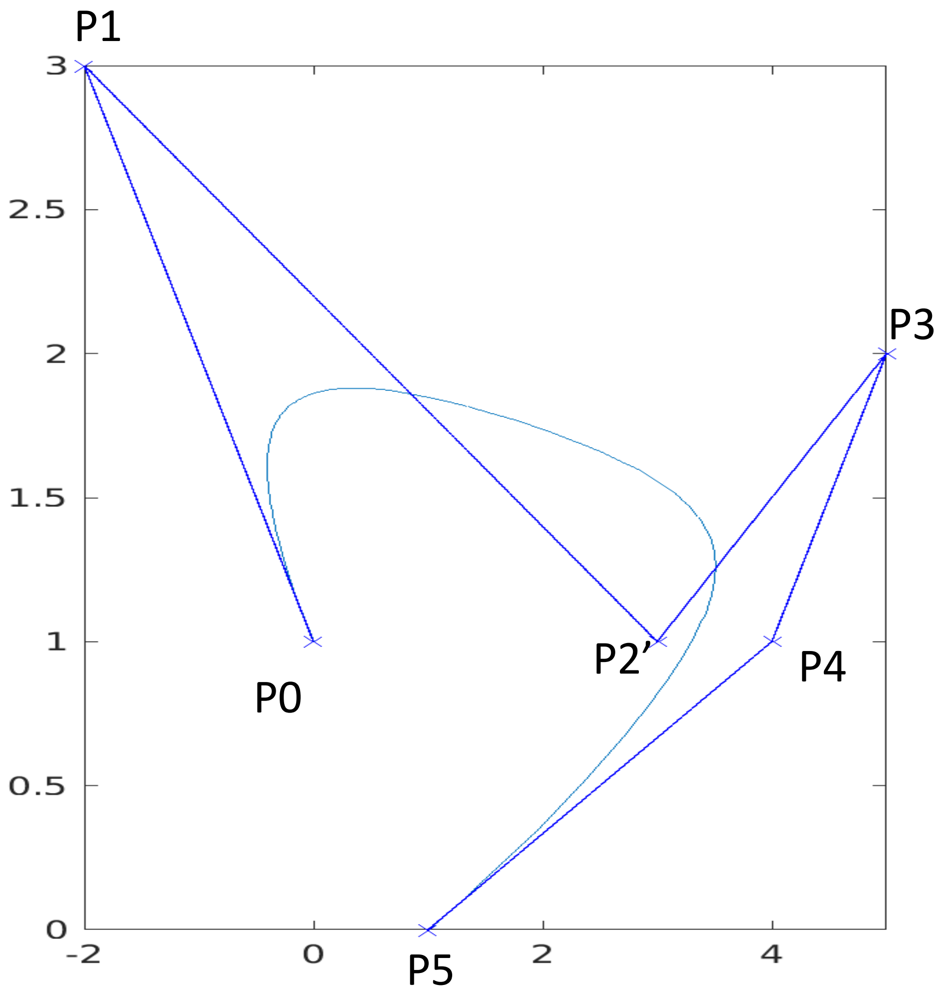
E la curva di Bezier
Relativa ai seguenti punti di controllo

```
cpts1 = [1 4 5 3 -2 1; 0 1 2 4 3 0];
```

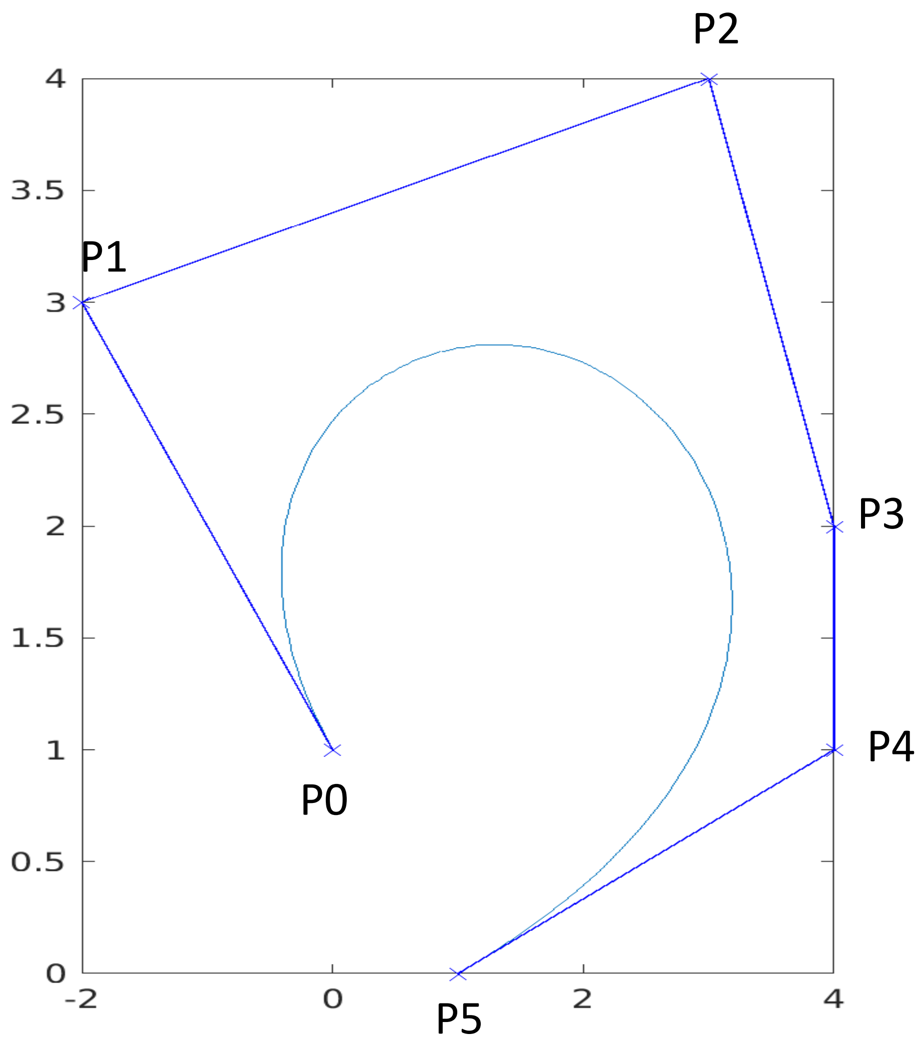
```
>> bezierCurve = simplify(B*cpts’)
```

Confronto curva bspline e curva di Beizer

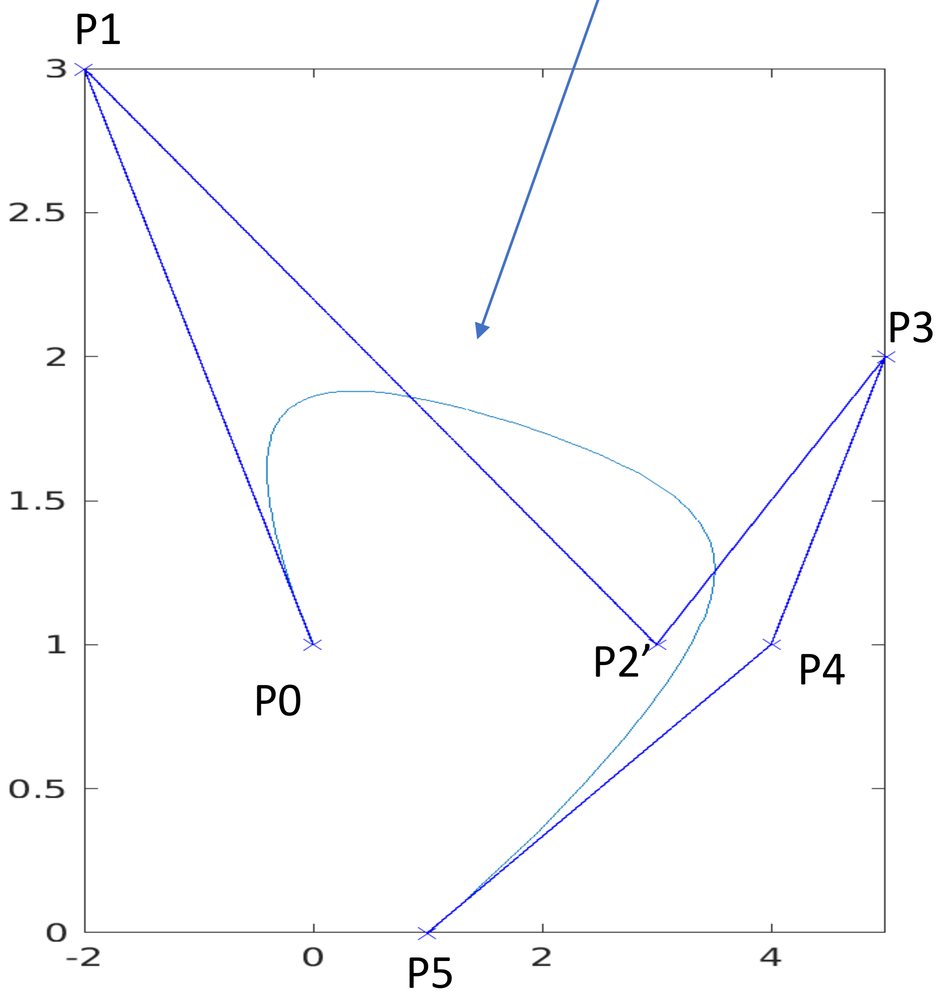




Confronto curva bspline e curva di Beizer



Cambiamento globale



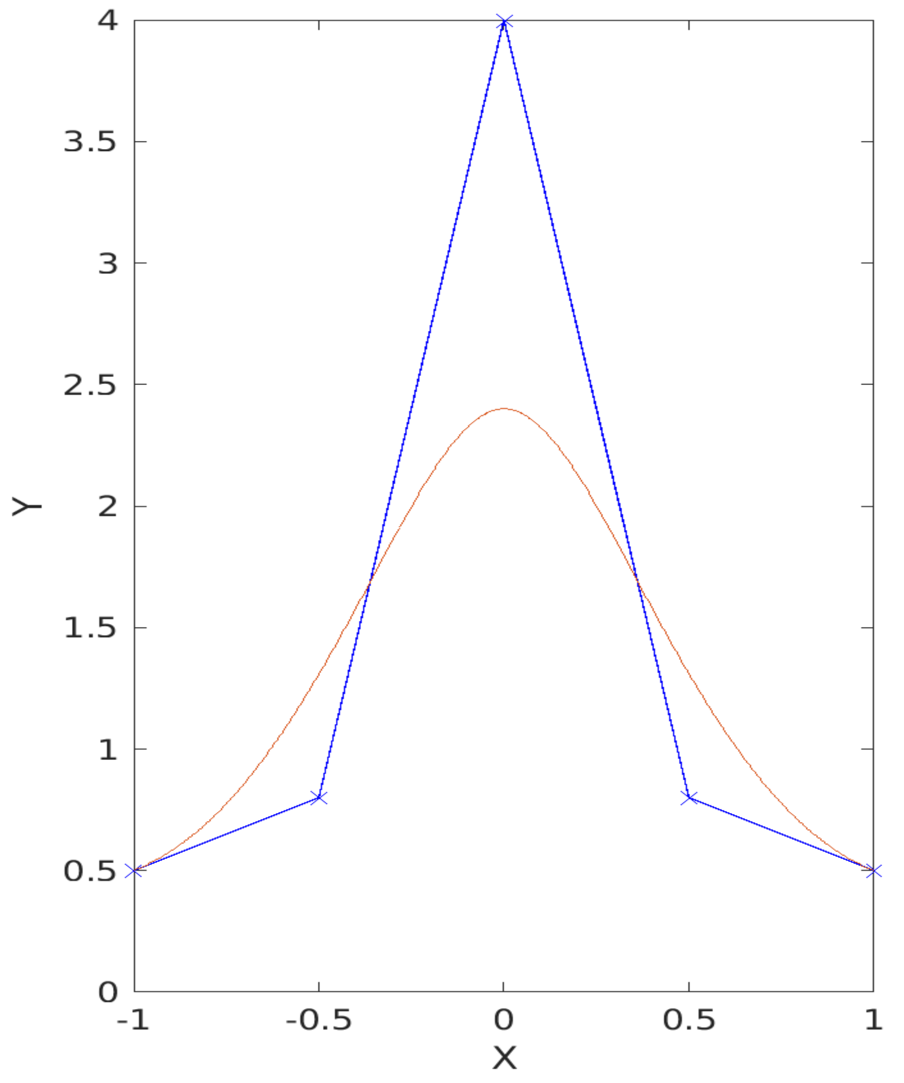
Curva B-spline al variare della molteplicità di un punto di controllo

Genero una curva B-spline a partire dai seguenti punti di controllo

```
>> cpts=[ -1 -0.5 0 0.5 1; 0.5 0.8 4 0.8 0.5];
cpts =
```

P0	P1	P2	P3	P4
-1.0000	-0.5000	0	0.5000	1.0000
0.5000	0.8000	4.0000	0.8000	0.5000

In particolare P2 ha molteplicità 1



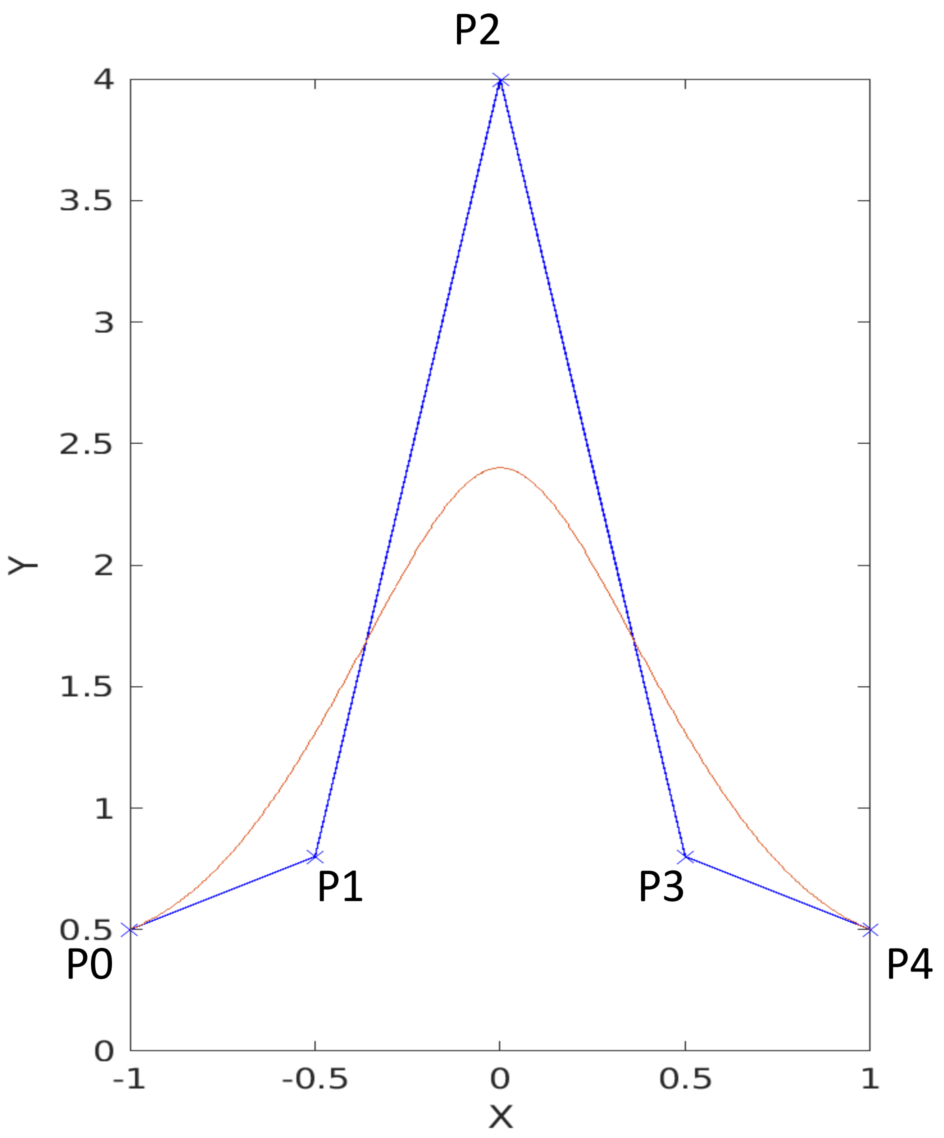
Curva B-spline al variare della molteplicità di un punto di controllo

Genero una curva B-spline a partire dai seguenti punti di controllo

>> cpts=[-1 -0.5 0 0.5 1; 0.5 0.8 4 0.8 0.5];
cpts =

P0	P1	P2	P3	P4
-1.0000	-0.5000	0	0.5000	1.0000
0.5000	0.8000	4.0000	0.8000	0.5000

In particolare P2 ha molteplicità 1



Curva B-spline al variare della molteplicità di un punto di controllo

Genero una curva B-spline a partire dai seguenti punti di controllo

```
>> cpts=[ -1 -0.5 0 0 0 0.5 1; 0.5 0.8 4 4 4 0.8 0.5];

cpts =
```

-1.0000	-0.5000	0	0	0	0.5000	1.0000
0.5000	0.8000	4.0000	4.0000	4.0000	0.8000	0.5000
P0	P1	P2 ≡	P3 ≡	P4	P5	P6

P2 ha molteplicità 3

Curva B-spline al variare della molteplicità di un punto di controllo

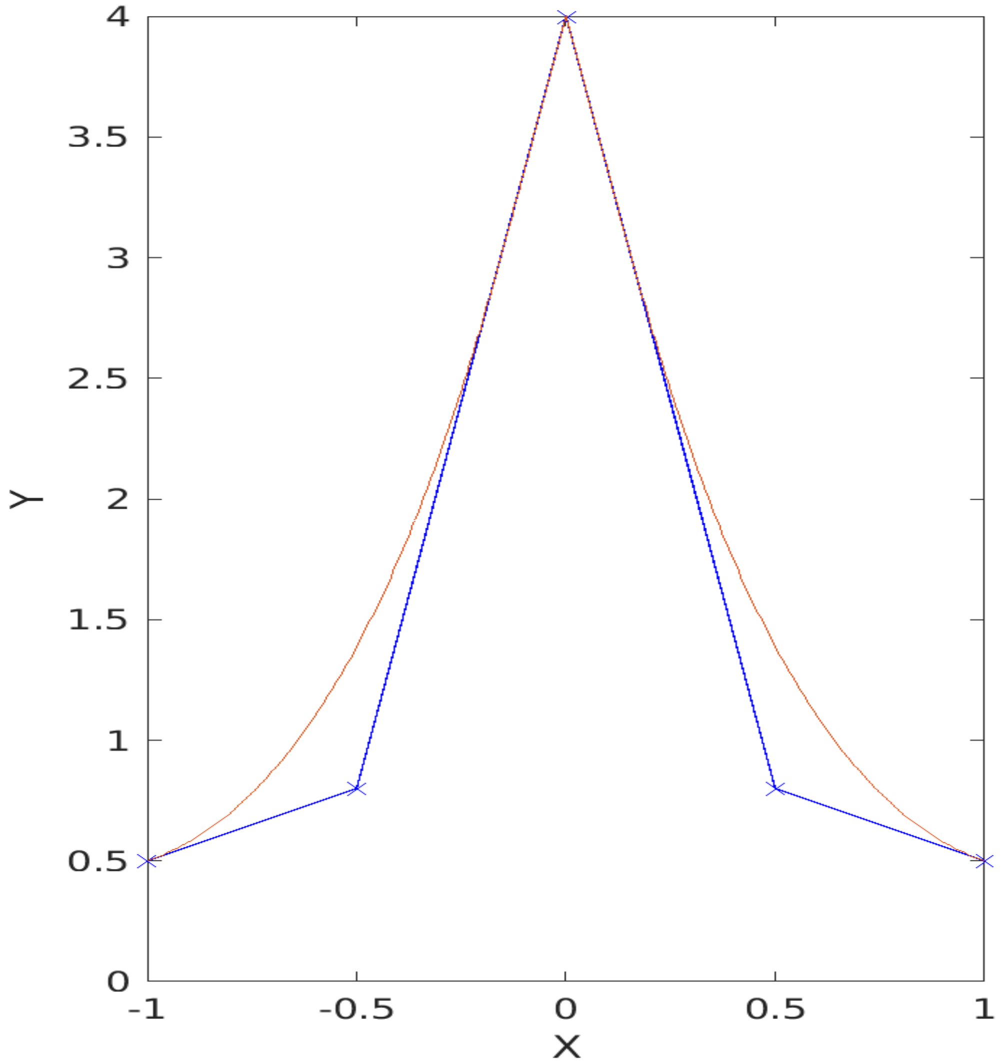
Genero una curva B-spline a partire dai seguenti punti di controllo

<u>>> cpts=[-1 -0.5 0 0 0 0.5 1; 0.5 0.8 4 4 4 0.8 0.5];</u>

<u>cpts =</u>

-1.0000	-0.5000	0	0	0	0.5000	1.0000
0.5000	0.8000	4.0000	4.0000	4.0000	0.8000	0.5000
P0	P1	P2 ≡	P3 ≡	P4	P5	P6

P2 ha molteplicità 3



Curva B-spline al variare della molteplicità di un punto di controllo

Interpola (0,4)

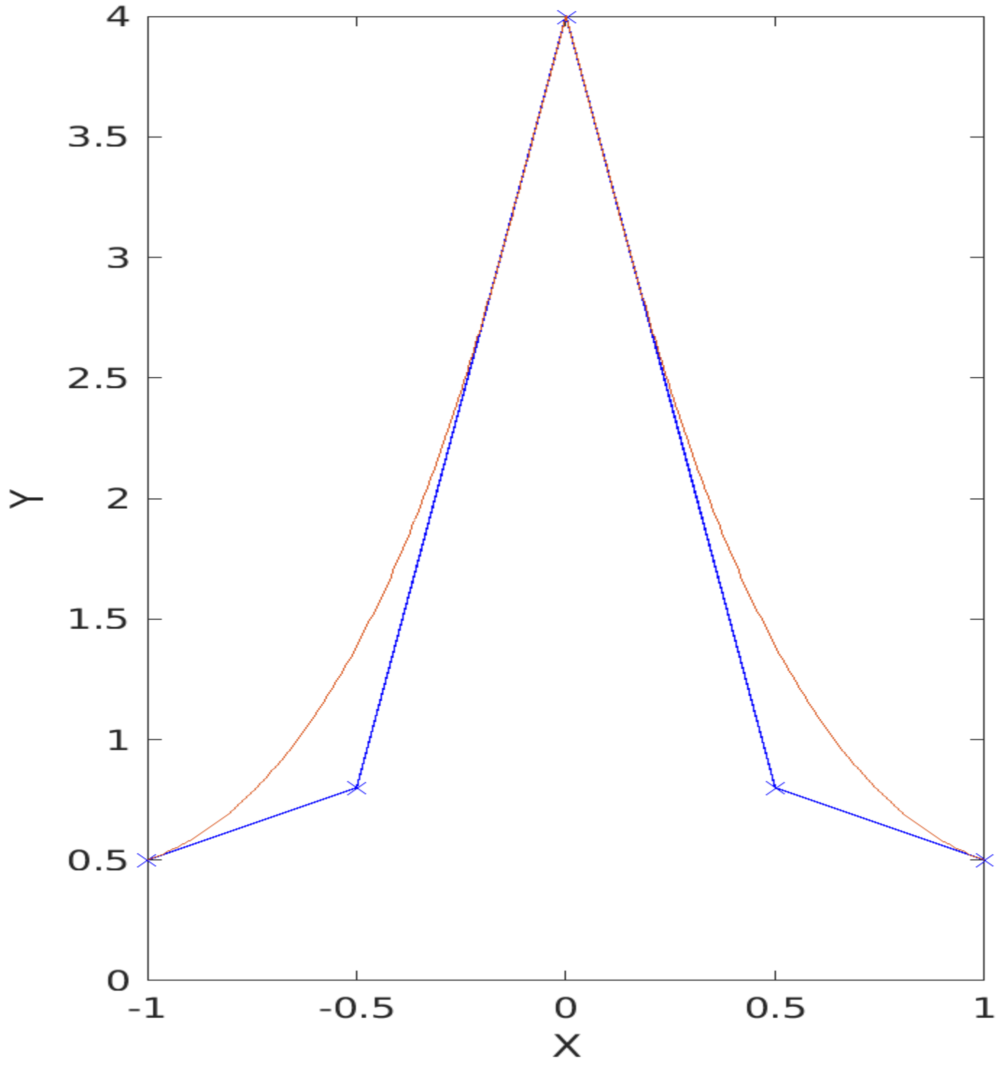
Genero una curva B-spline a partire dai seguenti punti di controllo

>> cpts=[-1 -0.5 0 0 0 0.5 1; 0.5 0.8 4 4 4 0.8 0.5];

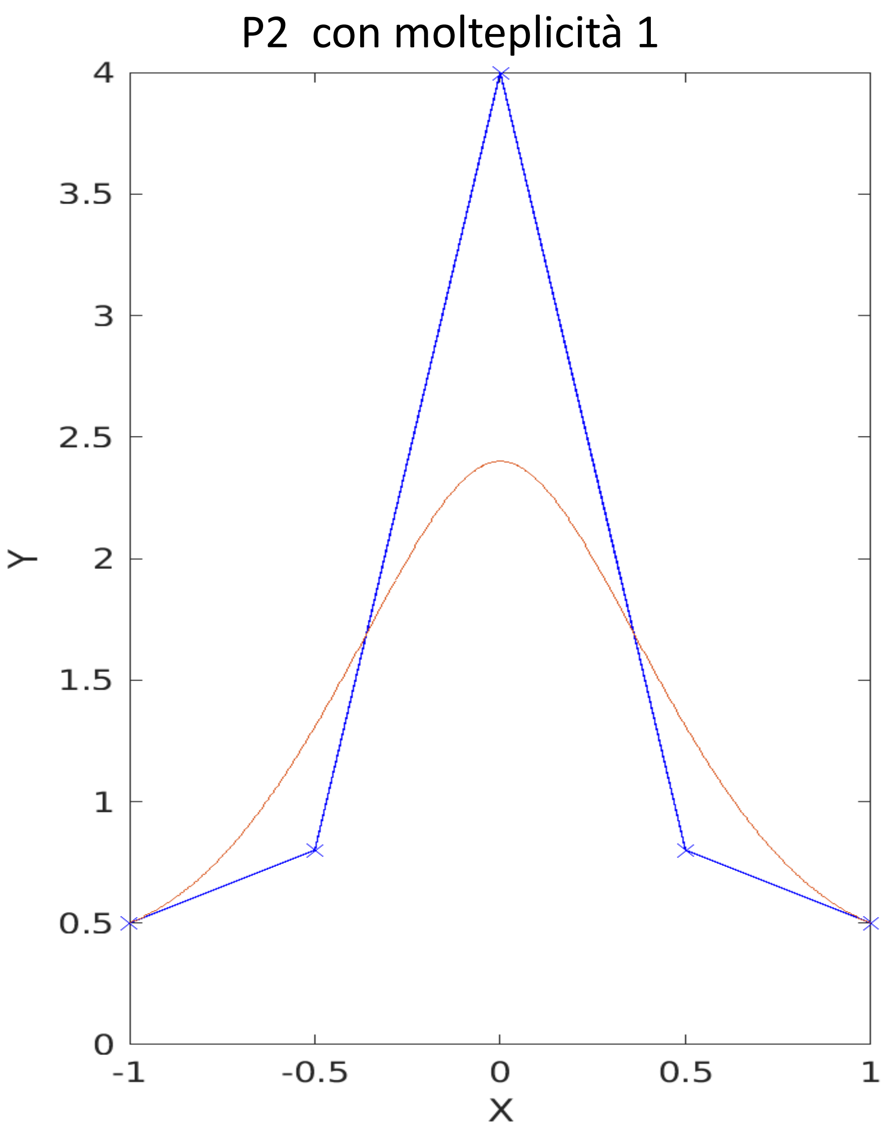
cpts =

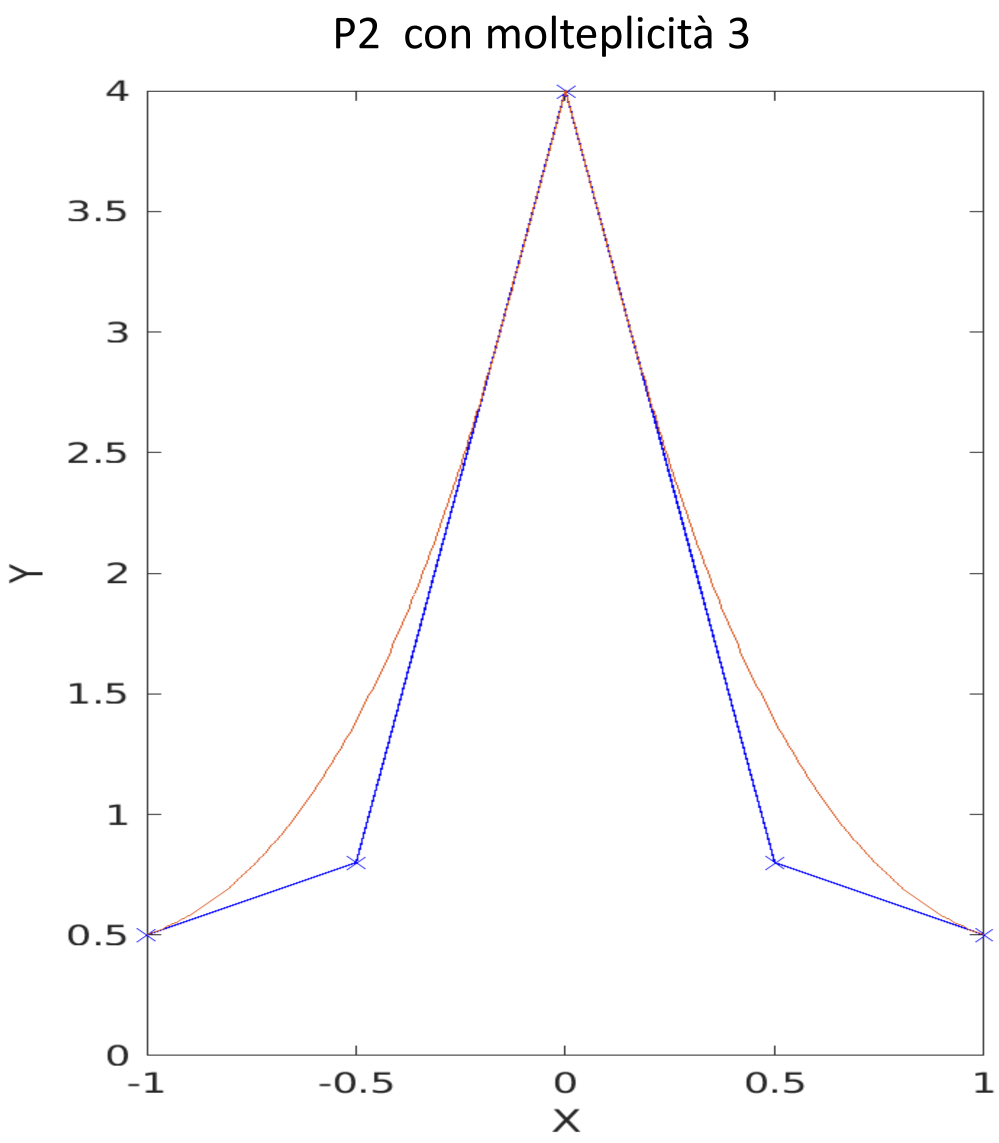
-1.0000	-0.5000	0	0	0	0.5000	1.0000
0.5000	0.8000	4.0000	4.0000	4.0000	0.8000	0.5000
P0	P1	P2 ≡	P3 ≡	P4	P5	P6

P2 ha molteplicità 3



Curva B-spline al variare della molteplicità di un punto di controllo





Matlab, function fit permette di approssimare un set di dati specificando il metodo e il tipo di funzione approssimante.

Resources ▾

fit

R2020

Fit curve or surface to data

collapse all in pa

Syntax

```
fitobject = fit(x,y,fitType)
fitobject = fit([x,y],z,fitType)
fitobject = fit(x,y,fitType,fitOptions)
fitobject = fit(x,y,fitType,Name,Value)
```

Resources ▾

Description

`fitobject = fit(x,y,fitType)` creates the fit to the data in x and y with the model specified by `fitType`. exam

`fitobject = fit([x,y],z,fitType)` creates a surface fit to the data in vectors x, y, and z. exam

`fitobject = fit(x,y,fitType,fitOptions)` creates a fit to the data using the algorithm options specified by the `fitOptions` object. exam

Approssimazione matlab

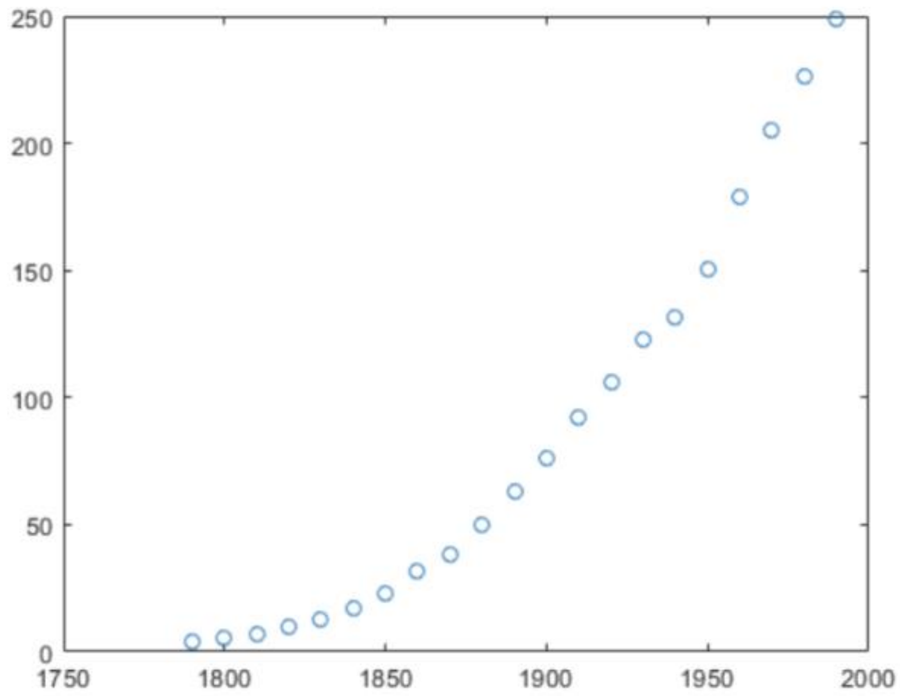
Carichiamo i seguenti dati di Matlab:

```
load census;
```

E generiamo il plot dei dati caricati

```
plot(cdate,pop,'o')
```

Distribuzione dati



Approssimazione matlab

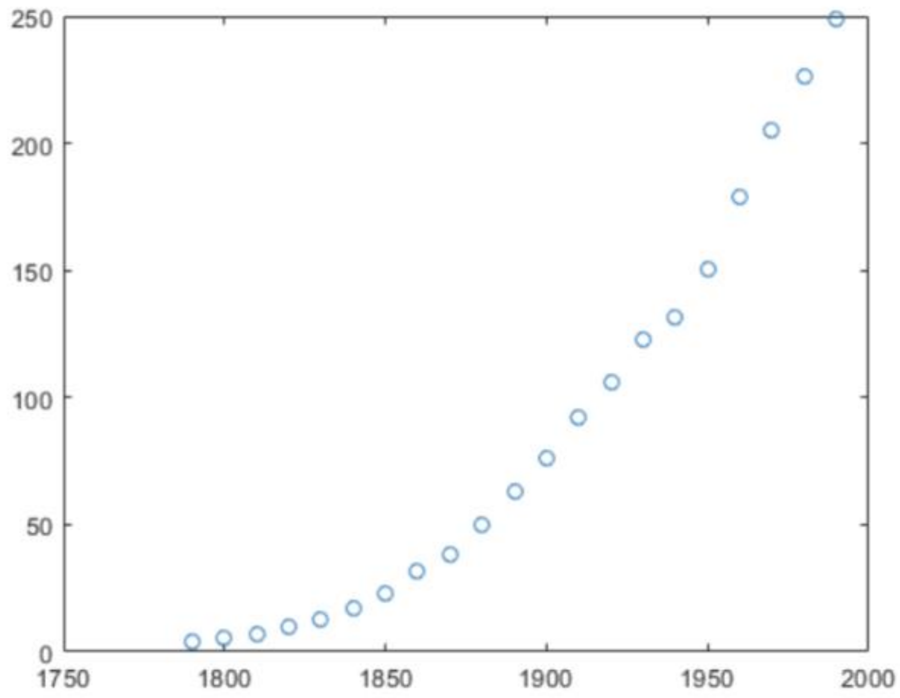
Carichiamo i seguenti dati di Matlab:

```
load census;
```

E generiamo il plot dei dati caricati

```
plot(cdate,pop,'o')
```

Distribuzione dati




```
f =      Linear model Poly2:
    f(x) = p1*x^2 + p2*x + p3
     Coefficients :
  p1 =    0.006541  (0.006124, 0.006958)
  p2 =      -23.51  (-25.09, -21.93)
  p3 =   2.113e+04  (1.964e+04, 2.262e+04)
```

```
f=fit(cdate,pop,'poly2')
   plot(f,cdate,pop)
```

Approssimazione matlab

Dati da caricare

```
load census;
```

Tramite la function fit generiamo una funzione approssimate i Dati utilizzando un polinomio di grado 2

```
f=fit(cdate,pop,'poly2')
```

Visualizzazione della funzione approssimate (e i suoi coeff.)

```
f =      Linear model Poly2:
   f(x) = p1*x^2 + p2*x + p3
    Coefficients :
  p1 =    0.006541  (0.006124, 0.006958)
  p2 =      -23.51  (-25.09, -21.93)
  p3 =   2.113e+04  (1.964e+04, 2.262e+04)
```

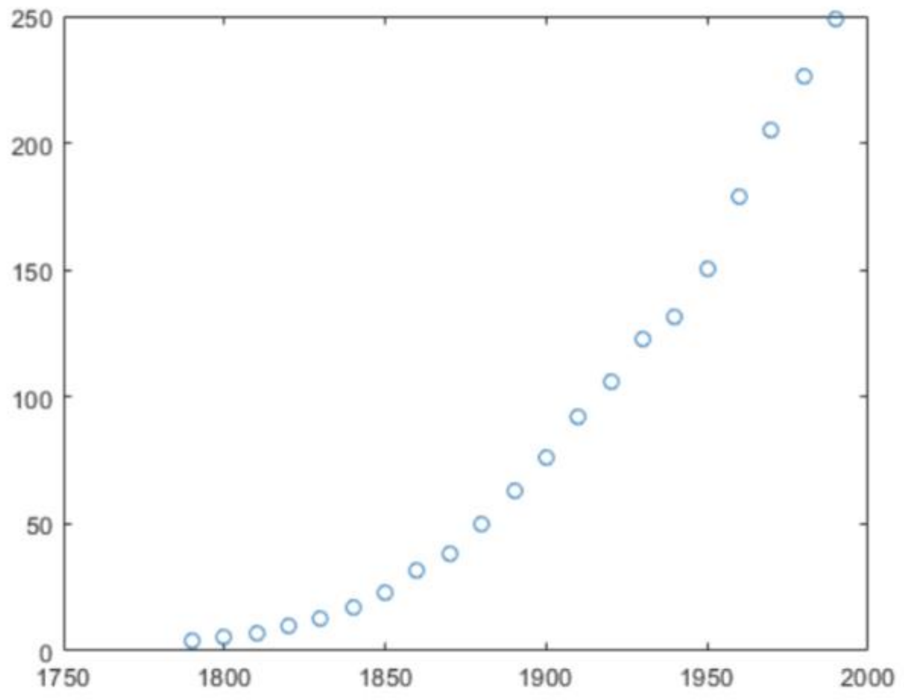


Grafico della funzione

```
plot(f,cdate,pop)
```

Approssimazione matlab

Dati da caricare

load census;

Tramite la function fit generiamo una funzione approssimate i Dati utilizzando un polinomio di grado 2

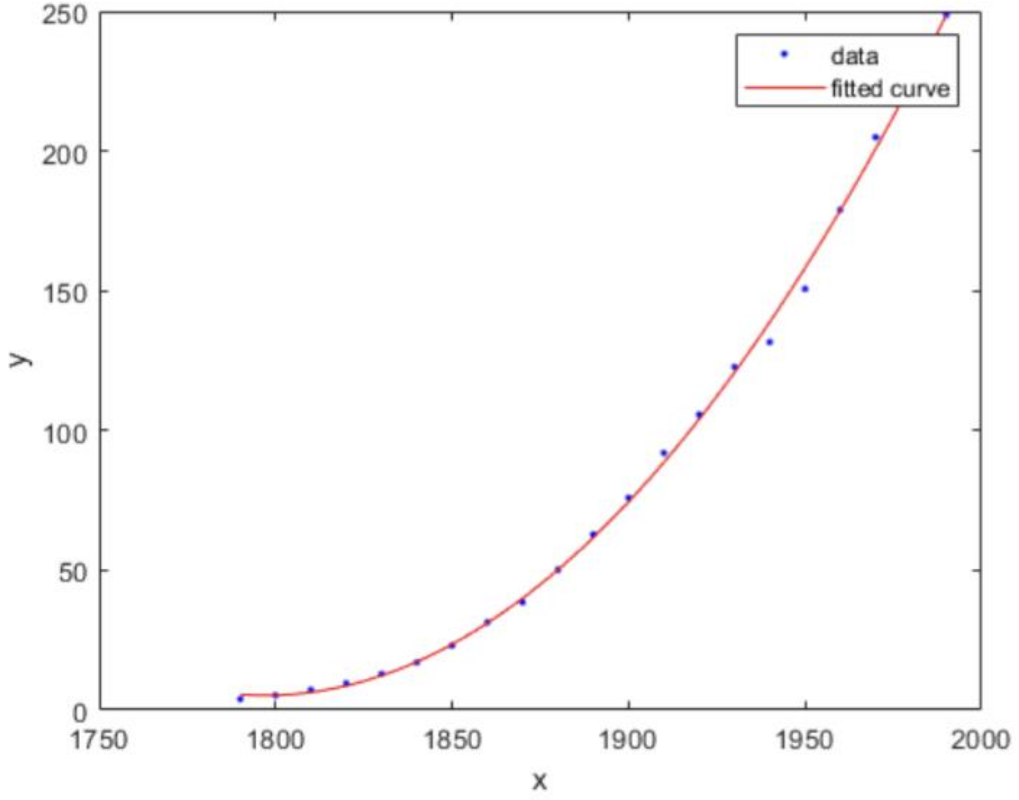
f=fit(cdate,pop,'poly2')

Visualizzazione della funzione approssimate (e i suoi coeff.)

```
f =     Linear model Poly2:
   f(x) = p1*x^2 + p2*x + p3
    Coefficients :
  p1 =    0.006541  (0.006124, 0.006958)
  p2 =      -23.51  (-25.09, -21.93)
  p3 =   2.113e+04  (1.964e+04, 2.262e+04)
```

Grafico della funzione

plot(f,cdate,pop)



Approssimazione matlab nello spazio

Carichiamo i seguenti dati di Matlab
load franke

Tramite la function fit generiamo una funzione approssimate i
Dati utilizzando un polinomio di grado 2 in x e di grado 3 in y

sf = fit([x, y],z,'poly23')

Visualizzazione della funzione approssimate (e i suoi coeff.)

```
sf(x,y) = p00 + p10*x + p01*y + p20*x^2 + p11*x*y + p02*y^2 +
p21*x^2*y + p12*x*y^2 + p03*y^3
    Coefficients
p00 =       1.118  (0.9149, 1.321)
p10 =  -0.0002941  (-0.000502, -8.623e-05)
p01 =       1.533  (0.7032, 2.364)
p20 =  -1.966e-08  (-7.084e-08, 3.152e-08)
p11 =   0.0003427  (-0.0001009, 0.0007863)
p02 =      -6.951  (-8.421, -5.481)
p21 =   9.563e-08  (6.276e-09, 1.85e-07)
p12 =  -0.0004401  (-0.0007082, -0.0001721)
p03 =       4.999  (4.082, 5.917)
```

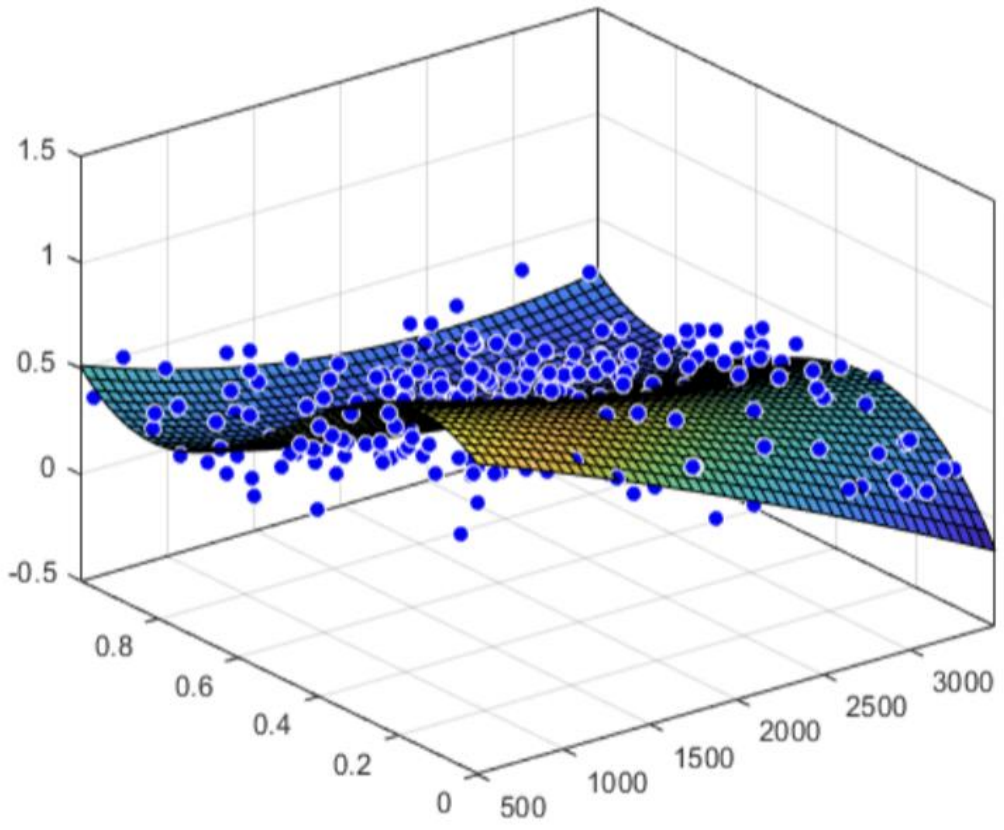
Approssimazione matlab nello spazio

Carichiamo i seguenti dati di Matlab
load franke

Tramite la function fit generiamo una funzione approssimate i
Dati utilizzando un polinomio di grado 2 in x e di grado 3 in y

Grafico della funzione e dei punti

plot(sf,[x,y],z)



Approssimazione matlab scegliendo il modello approssimante

Carichiamo i seguenti dati di Matlab

```
load census
```

Scegliamo le seguenti opzione per la funzione approssimate

```
fo = fitoptions('Method','NonlinearLeastSquares',...
'Lower',[0,0],'Upper',[Inf,max(cdate)],'StartPoint',[1 1]);
ft = fittype('a*(x-b)^n','problem','n','options',fo);
```

Richiamo la funzione fit per generare la funzione approssimate

```
[curve2,gof2] = fit(cdate,pop,ft,'problem',2)
```

Visualizzazione della funzione approssimate (e i suoi coeff.)

```
curve2 =
     General model:
     curve2(x) = a*(x-b)^n
     Coefficients
       a =    0.006092  (0.005743, 0.006441)
       b =        1789  (1784, 1793)
     Problem parameters:
       n =           2
```

Approssimazione matlab scegliendo il modello approssimante

Carichiamo i seguenti dati di Matlab

```
load census
```

Scegliamo le seguenti opzione per la funzione approssimate

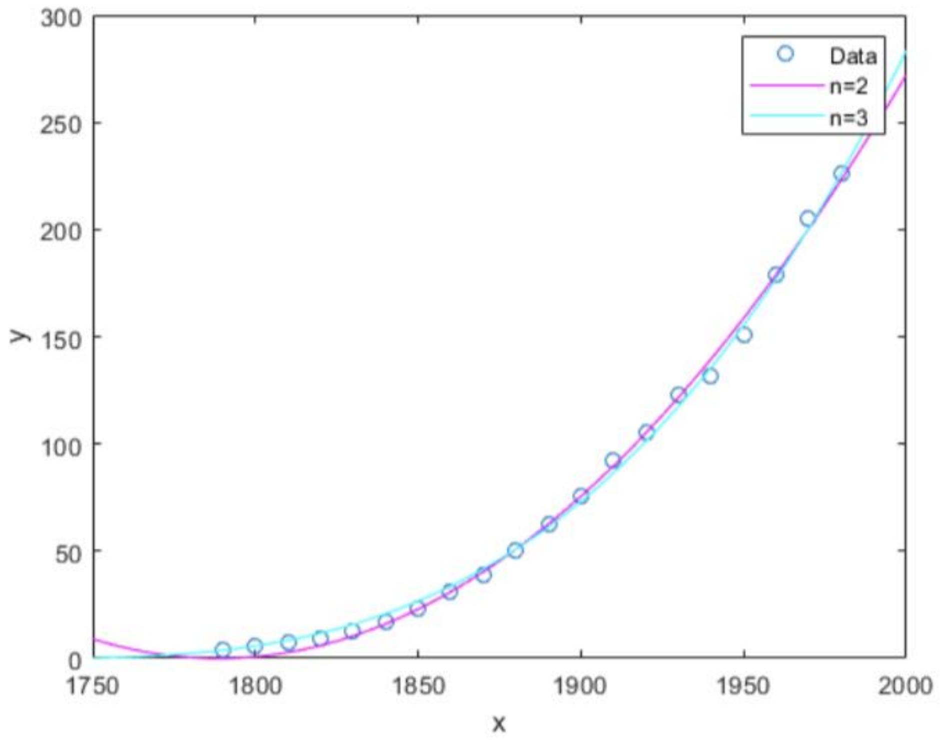
```
fo = fitoptions('Method','NonlinearLeastSquares',...
'Lower',[0,0],'Upper',[Inf,max(cdate)],'StartPoint',[1 1]);
ft = fittype('a*(x-b)^n','problem','n','options',fo);
```

Richiamo la funzione fit per generare la funzione approssimate

```
[curve2,gof2] = fit(cdate,pop,ft,'problem',2)
```

Visualizzazione della funzione approssimate (e i suoi coeff.)

```
curve2 =
     General model:
     curve2(x) = a*(x-b)^n
     Coefficients
       a =    0.006092  (0.005743, 0.006441)
       b =        1789  (1784, 1793)
     Problem parameters:
       n =           2
```

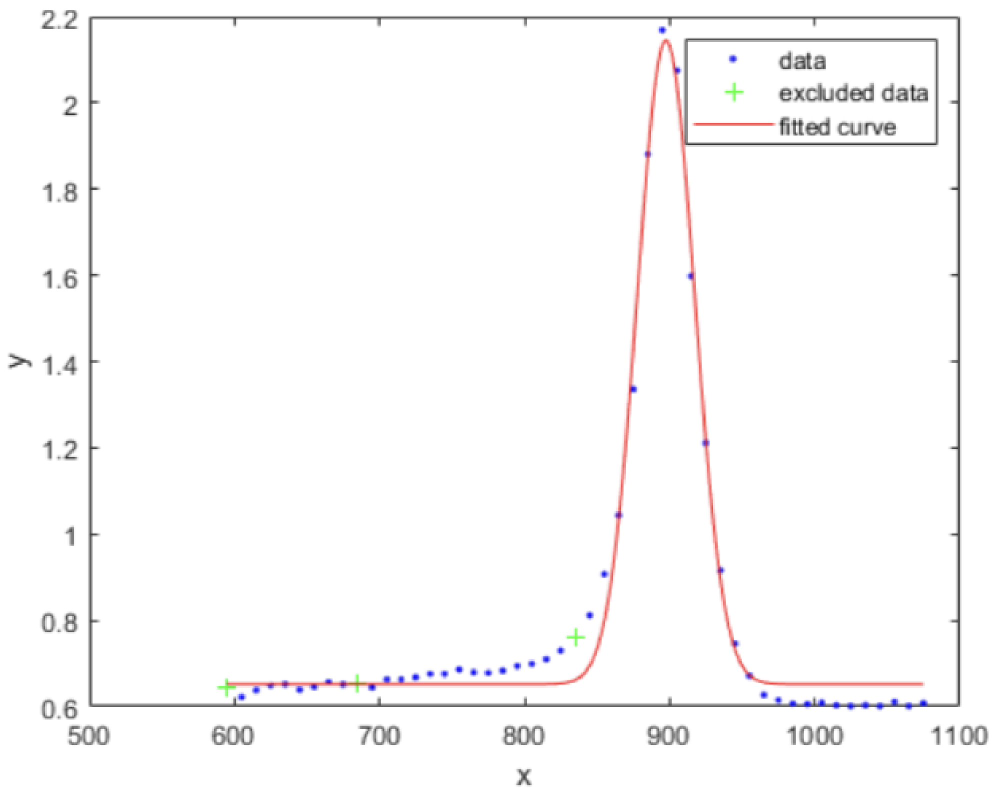


Approssimazione matlab escludendo alcuni punti

Carichiamo i seguenti dati

[x, y] = titanium;

restituisce misure di una determinata proprietà del titanio in funzione della temperatura



Funzione Gaussiana

gaussEqn = 'a*exp(-((x-b)/c)^2)+d'

Punti esclusi dall'approssimazione

exclude1 = [1 10 25];

startPoints = [1.5 900 10 0.6]

f1 = fit(x',y',gaussEqn,'Start', startPoints, 'Exclude', exclude1);

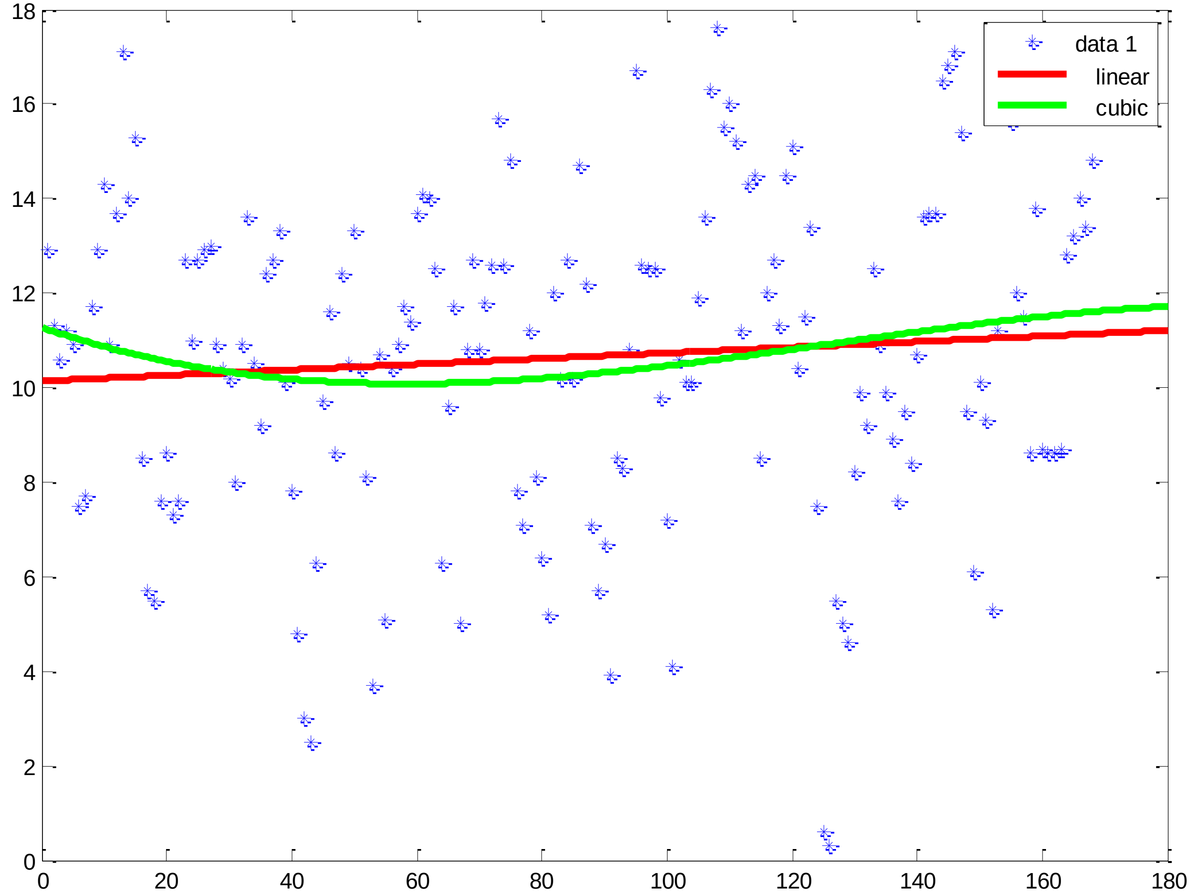
plot(f1,x,y,exclude1)

Approssimazione matlab escludendo alcuni punti

Carichiamo I seguenti dati

>> load enso

valori misurati ogni mese della differenza di pressione atmosferica tra l'est dell'Islanda e Darwin in Australia (influenza il vento nell'emisfero australe)

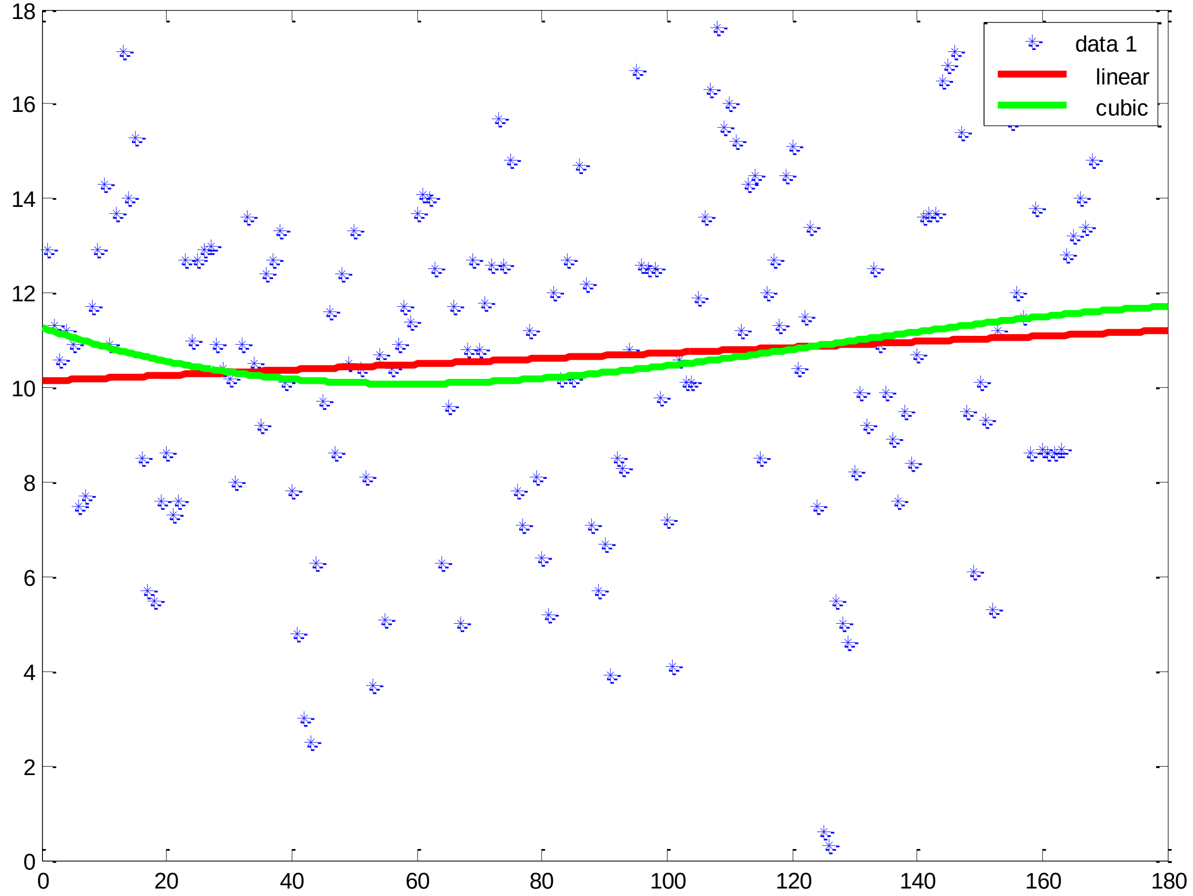


Approssimazione matlab escludendo alcuni punti

Carichiamo I seguenti dati

>> load enso

valori misurati ogni mese della differenza di pressione atmosferica tra l’est dell’Islanda e Darwin in Australia (influenza il vento nell’emisfero australe)

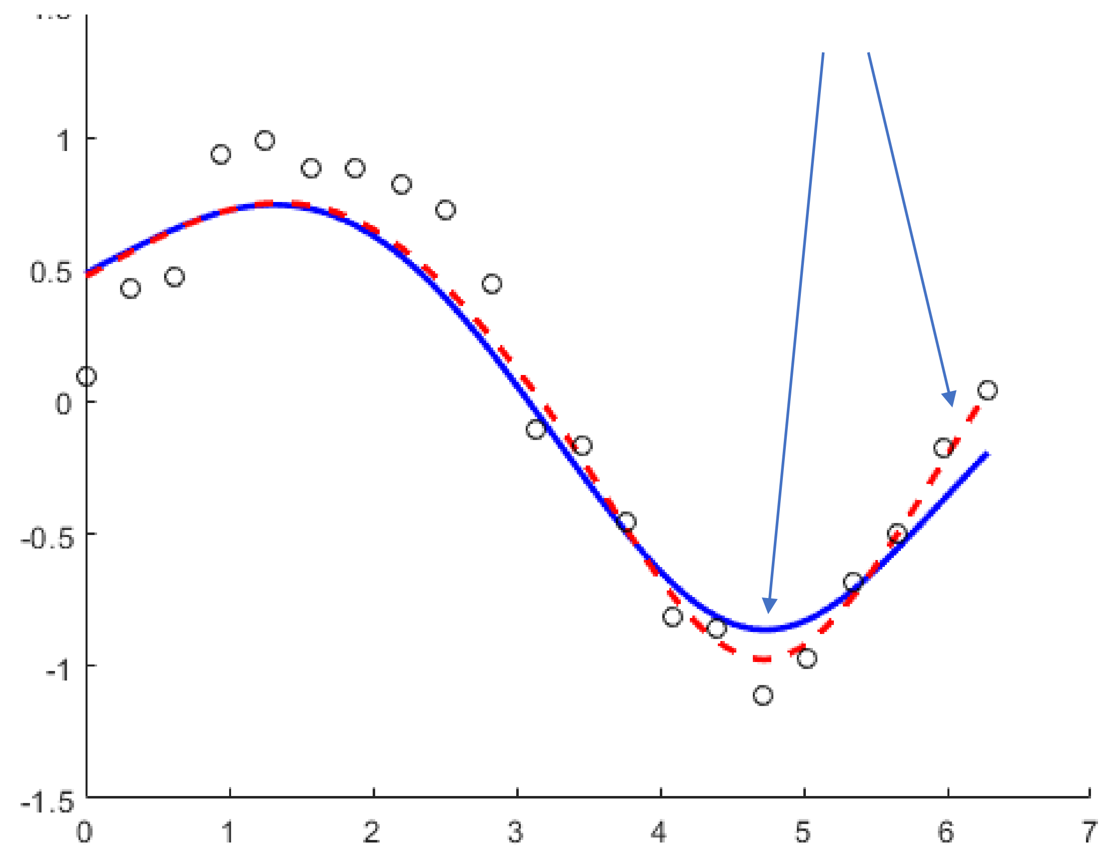


Migliorare l’approssimazione

Se si è a conoscenza dell’errore sui dati e quindi dei pesi sui dai.

Approssimazione matlab

2 approssimazioni differenti, dipendono dai pesi dei dati



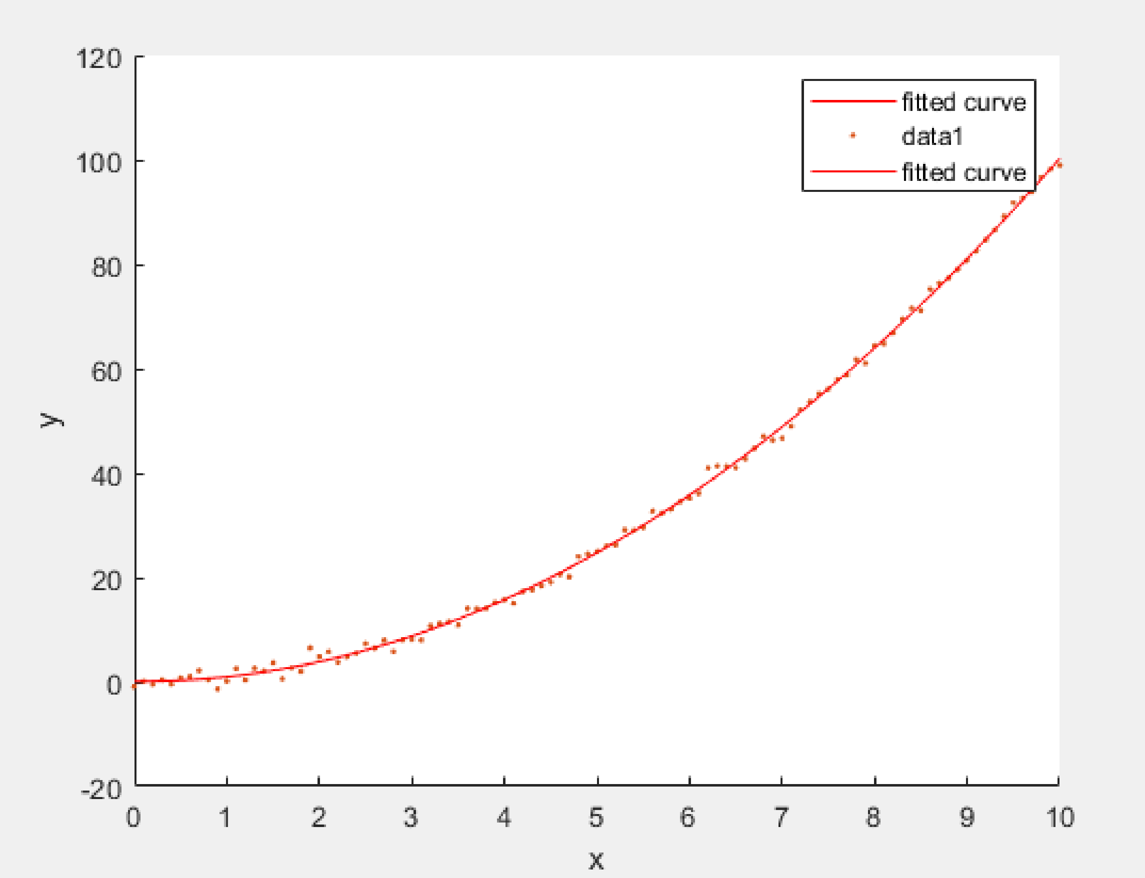
Approssimazione matlab tenendo conto dei pesi

Dati nell’intervallo [0 10]
x = 0:.1:10;

Valore della parabola in corrispondenza dei punti x
Con l’aggiunta di un errore random
y = x.*x + randn(size(x));

Pesi associati ai dati
w = linspace(.5, .7,length(x));

%plot dati
plot(x,y,’.’)
%fit considerando i pesi
ft = fittype('poly2’)
cf = fit(x’,y’,ft,'Weight’,w’)
% Plot
hold on
plot(cf,'fit')



Approssimazione matlab tenendo conto dei pesi

Cambiamo i pesi rispetto all'esercizio precedente

Dati nell'intervallo [0 10]

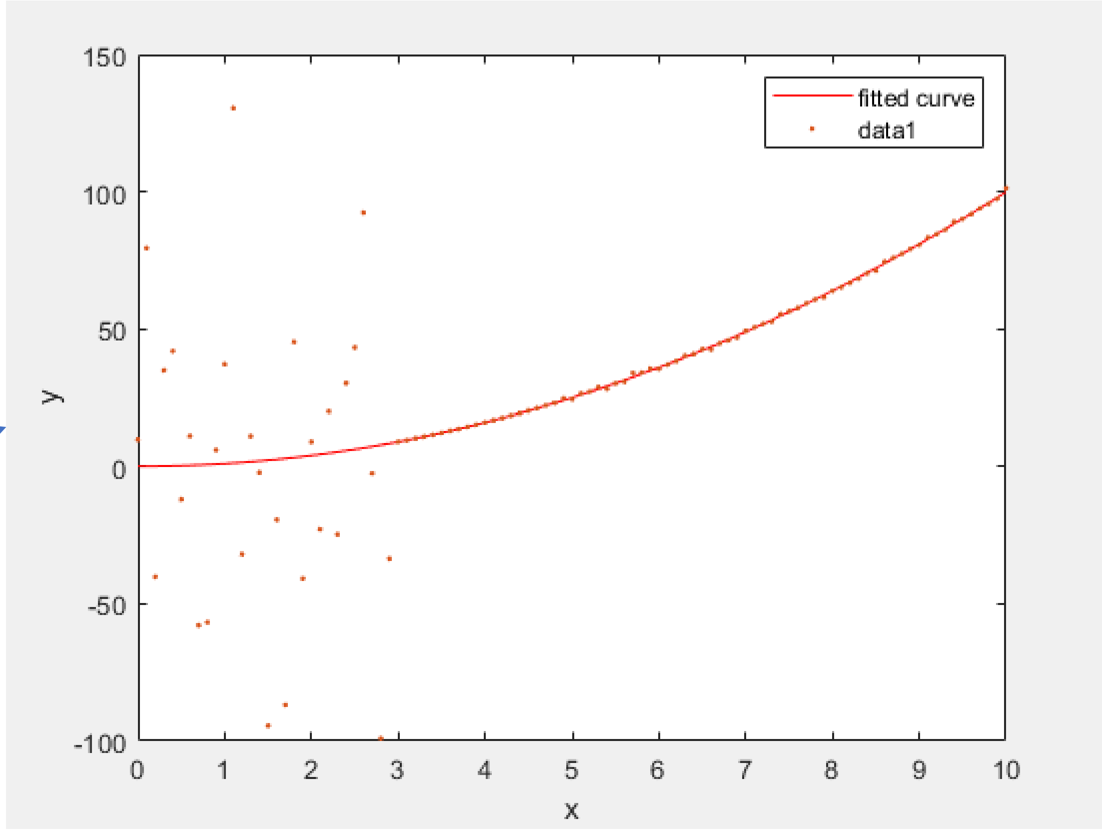
x = 0:.1:10;

Valore della parabola in corrisponenza dei punti x
Con l'aggiunta di un errore random

```
y = x.*x + [50*randn(30,1)' zeros(19,1)' 0.5*randn(52,1)'];
```

Pesi associati ai dati

```
w = [zeros(30,1)' 5*ones(19,1)' 0.5*ones(52,1)'];
%plot dati
plot(x,y,'.')
ft = fittype('poly2')
%fit considerando i pesi
cf = fit(x',y',ft,'Weight',w')
% Plot
hold on
plot(cf,'fit')
```



Approssimazione matlab tenendo conto dei pesi

Cambiamo i pesi rispetto all'esercizio precedente

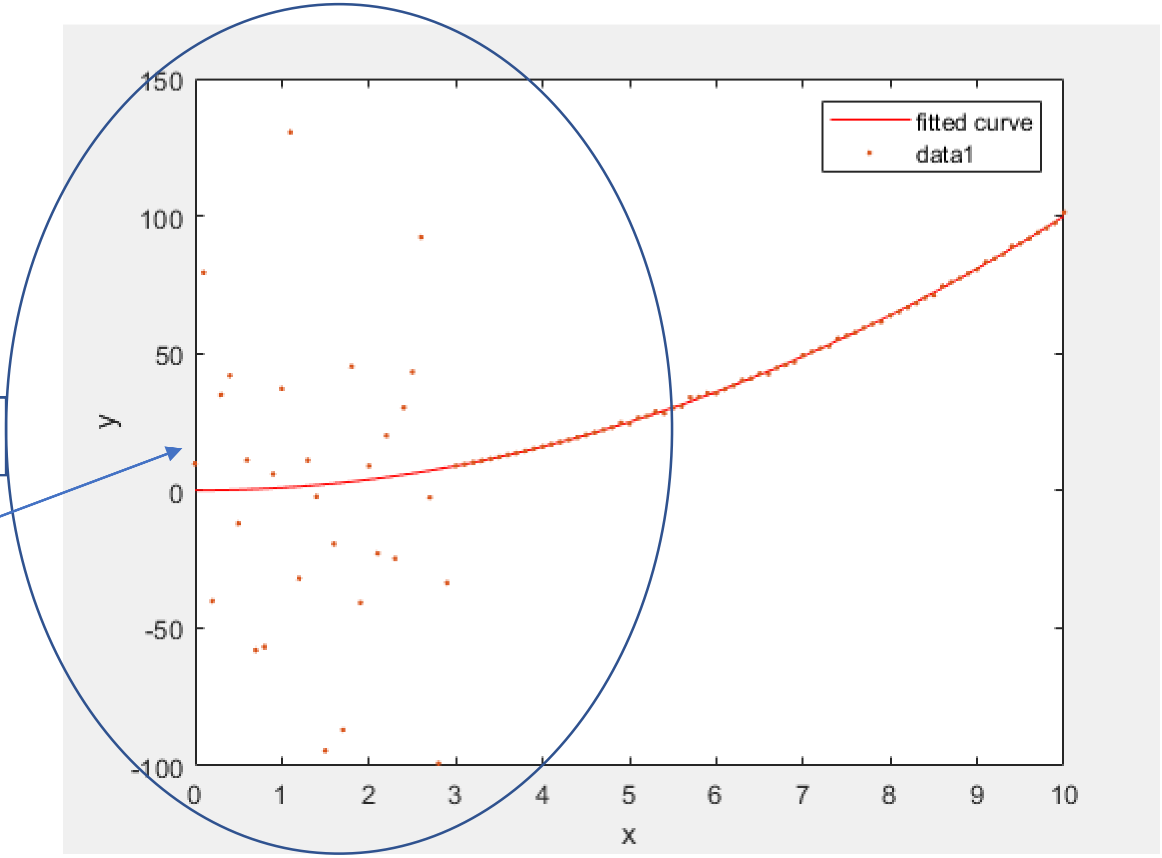
Dati nell'intervallo [0 10]

x = 0:.1:10;

Valore della parabola in corrisponenza dei punti x
Con l'aggiunta di un errore random

y = x.*x + [50*randn(30,1)' zeros(19,1)' 0.5*randn(52,1)'];

Pesi associati ai dati
w = [zeros(30,1)' 5*ones(19,1)' 0.5*ones(52,1)'];
%plot dati
plot(x,y,'.')
ft = fittype('poly2')
%fit considerando i pesi
cf = fit(x',y',ft,'Weight',w')
% Plot
hold on
plot(cf,'fit')



Approssimazione matlab tenendo conto dei pesi

Cambiamo i pesi rispetto all'esercizio precedente

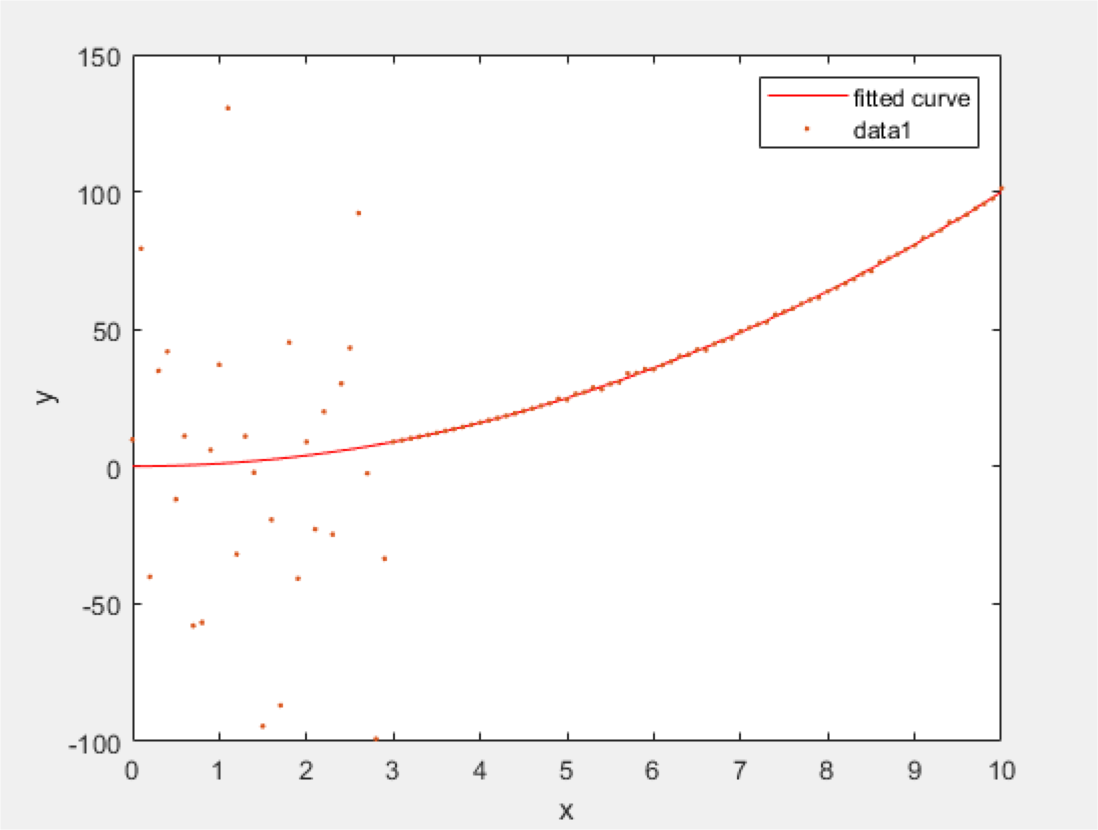
Dati nell'intervallo [0 10]

x = 0:.1:10;

Valore della parabola in corrisponenza dei punti x
Con l'aggiunta di un errore random

y = x.*x + [50*randn(30,1)' zeros(19,1)' 0.5*randn(52,1)'];

Pesi associati ai dati
w = [zeros(30,1)' 5*ones(19,1)' 0.5*ones(52,1)'];
%plot dati
plot(x,y,'.')
ft = fittype('poly2')
%fit considerando i pesi
cf = fit(x',y',ft,'Weight',w')
% Plot
hold on
plot(cf,'fit')



Meno peso ai punti iniziale, la funzione approssimante continua ad avere l'andamento di una parabola

Approssimazione matlab tenendo conto dei pesi

Cambiamo i pesi rispetto all’esercizio precedente

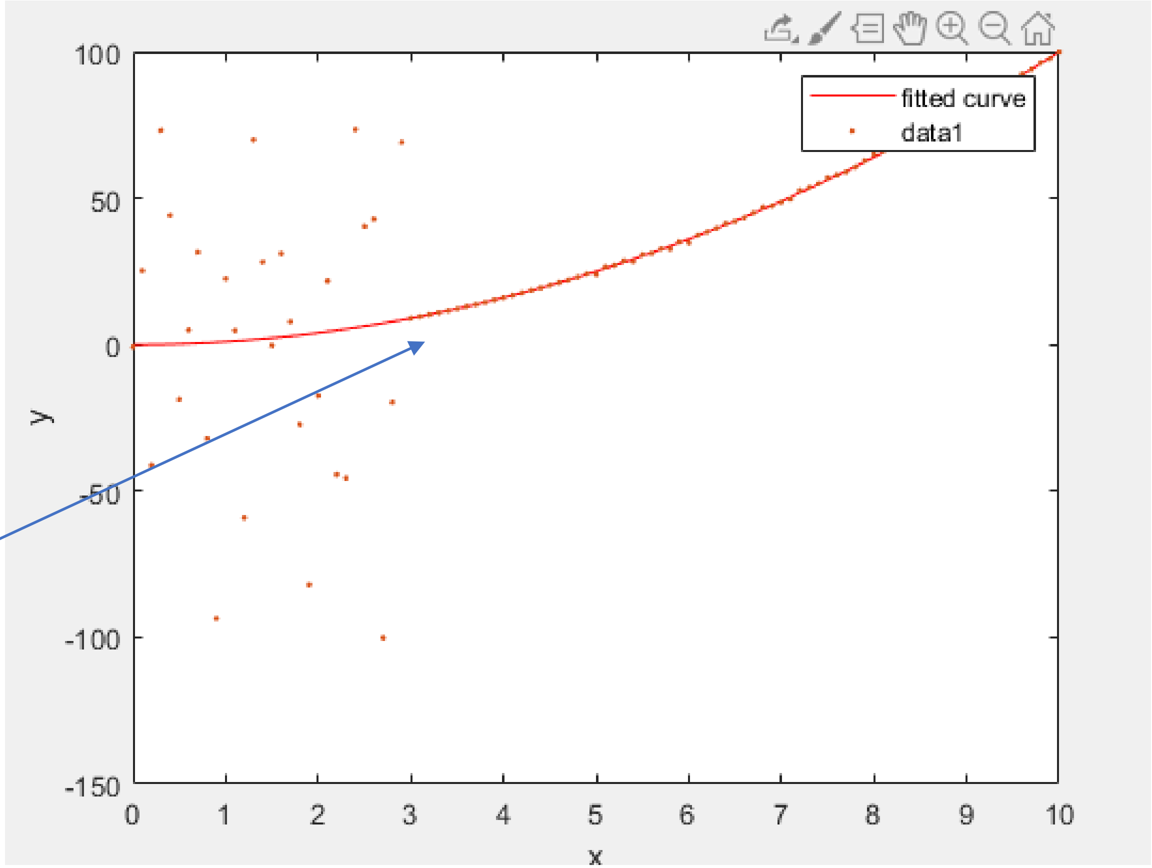
Dati nell’intervallo [0 10]

x = 0:.1:10;

Valore della parabola in corrisponenza dei punti x
Con l’aggiunta di un errore random

y = x.*x + [50*randn(30,1)' zeros(19,1)' 0.5*randn(52,1)’];

Pesi associati ai dati
w = [zeros(30,1)' 5*ones(19,1)' 0.5*ones(52,1)’];
%plot dati
plot(x,y,'.’)
ft = fittype('poly2’)
%fit considerando i pesi
cf = fit(x’,y’,ft,'Weight’,w’)
% Plot
hold on
plot(cf,'fit')



Cambiamo i pesi

Cambiamo ancora i pesi rispetto all'esercizio precedente

Dati nell'intervallo [0 10]

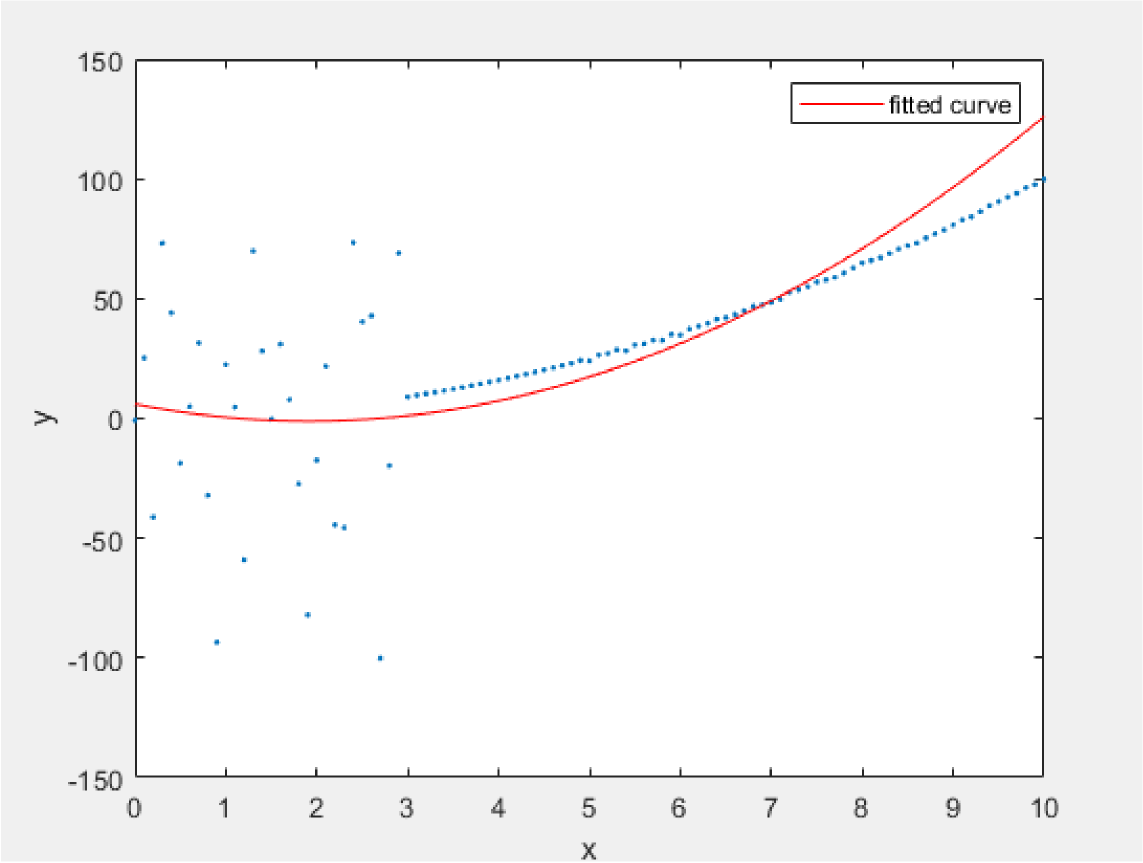
```
x = 0:.1:10;
```

Valore della parabola in corrisponenza dei punti x
Con l'aggiunta di un errore random

```
y = x.*x + [50*randn(30,1)' zeros(19,1)' 0.5*randn(52,1)'];
```

Pesi associati ai dati

```
w = [50*ones(30,1)' 5*ones(19,1)' 0.5*ones(52,1)'];
%plot dati
plot(x,y,'.')
ft = fittype('poly2')
%fit considerando i pesi
cf = fit(x',y',ft,'Weight',w')
% Plot
hold on
plot(cf,'fit')
```



Cambiamo i pesi

Cambiamo ancora i pesi rispetto all'esercizio precedente

Dati nell'intervallo [0 10]

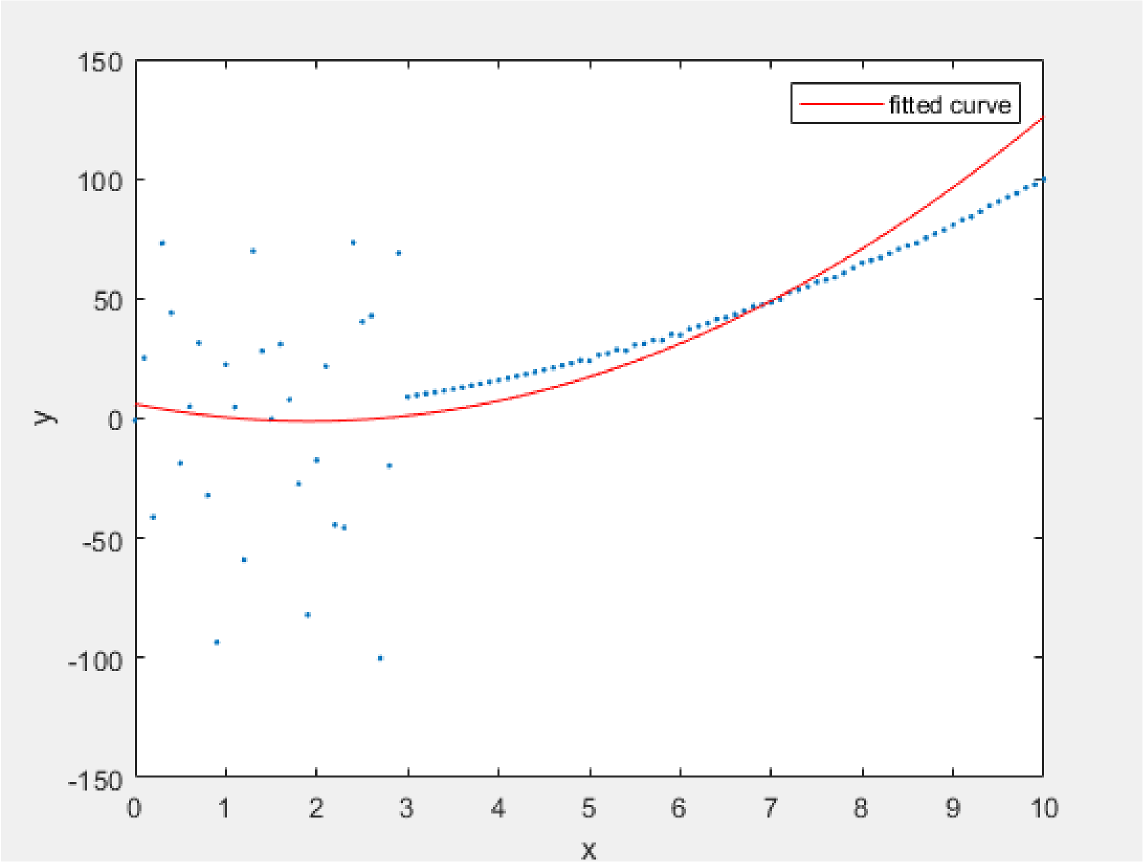
```
x = 0:.1:10;
```

Valore della parabola in corrisponenza dei punti x
Con l'aggiunta di un errore random

```
y = x.*x + [50*randn(30,1)' zeros(19,1)' 0.5*randn(52,1)'];
```

Pesi associati ai dati

```
w = [50*ones(30,1)' 5*ones(19,1)' 0.5*ones(52,1)'];
%plot dati
plot(x,y,'.')
ft = fittype('poly2')
%fit considerando i pesi
cf = fit(x',y',ft,'Weight',w')
% Plot
hold on
plot(cf,'fit')
```



Peso maggiore ai dati iniziali l'andamento della
Funzione approssimante cambia.

Confronto tra funzioni approssimanti

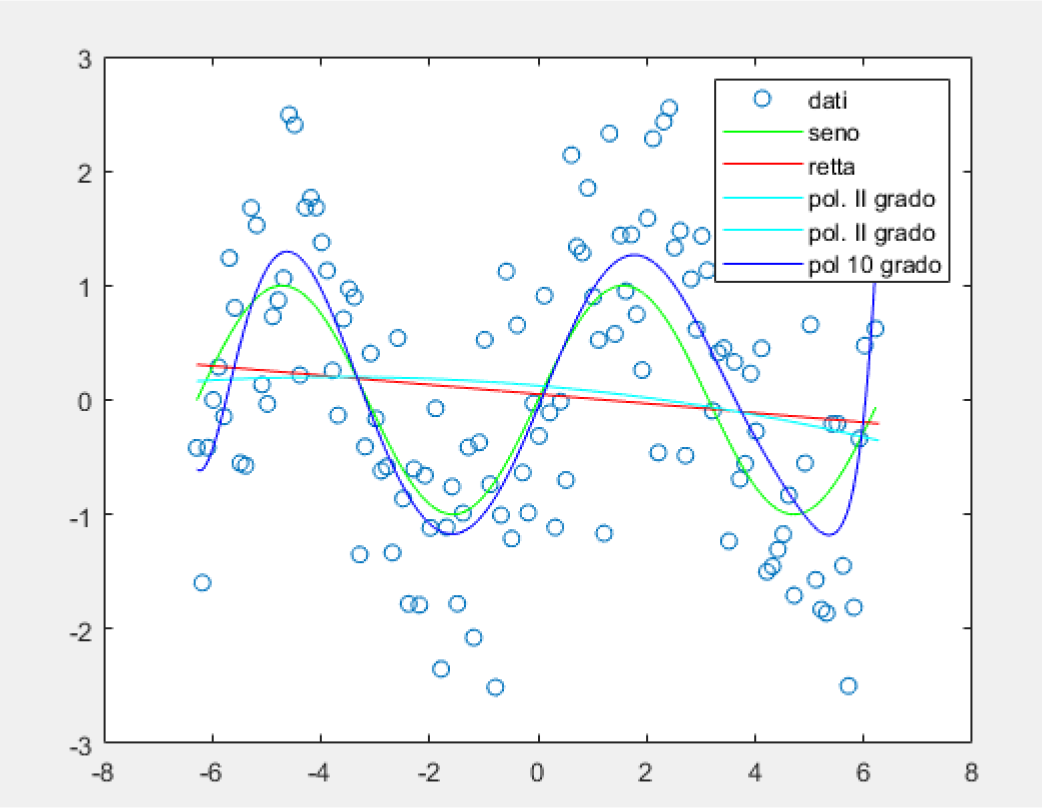
```
% confronto retta e pol.II grado
x=-2*pi:0.1:2*pi;
y=sin(x)+randn(1,length(x));
plot(x,y,'o')
hold on
ys=sin(x);
plot(x,ys,'g')
t=-2*pi:0.005:2*pi;
c=polyfit(x,y,1);
v=polyval(c,t);
plot(t,v,'r')
c=polyfit(x,y,2);
v=polyval(c,t);
plot(t,v,'c')
hold on
c=polyfit(x,y,10);
v=polyval(c,t);
plot(t,v,'c')
```

Aggiungiamo un errore

retta

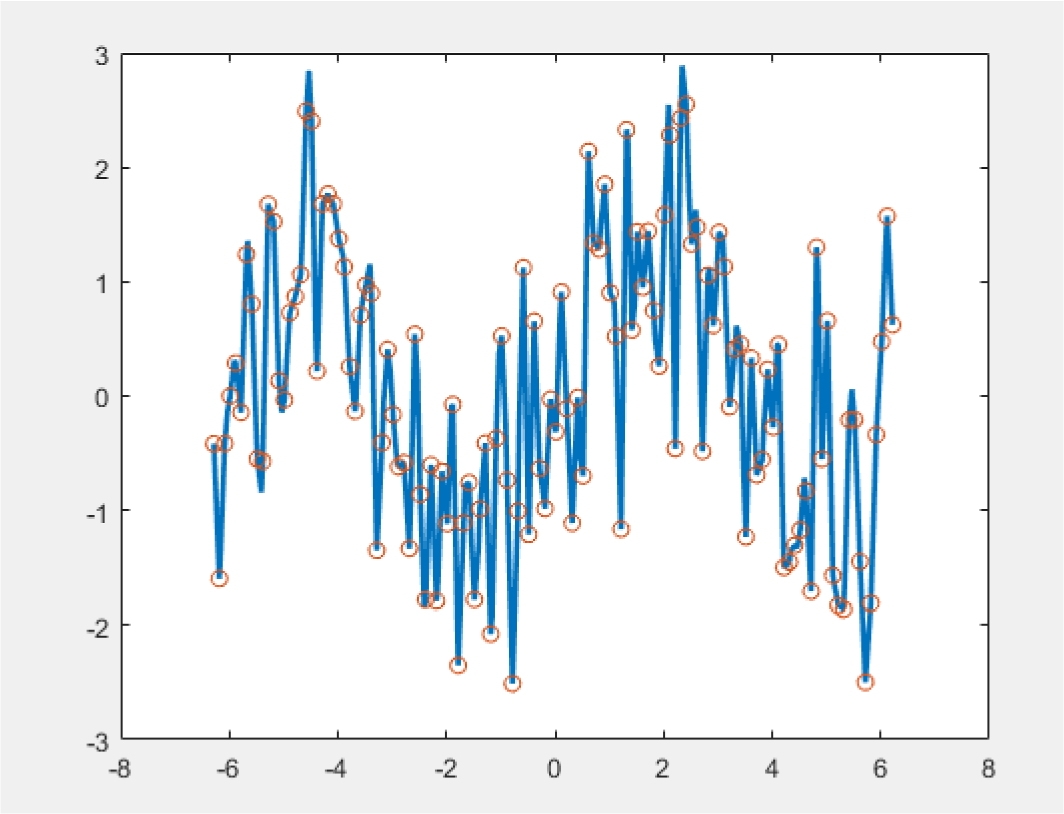
Polinomio di II grado

Polinomio di grado 10

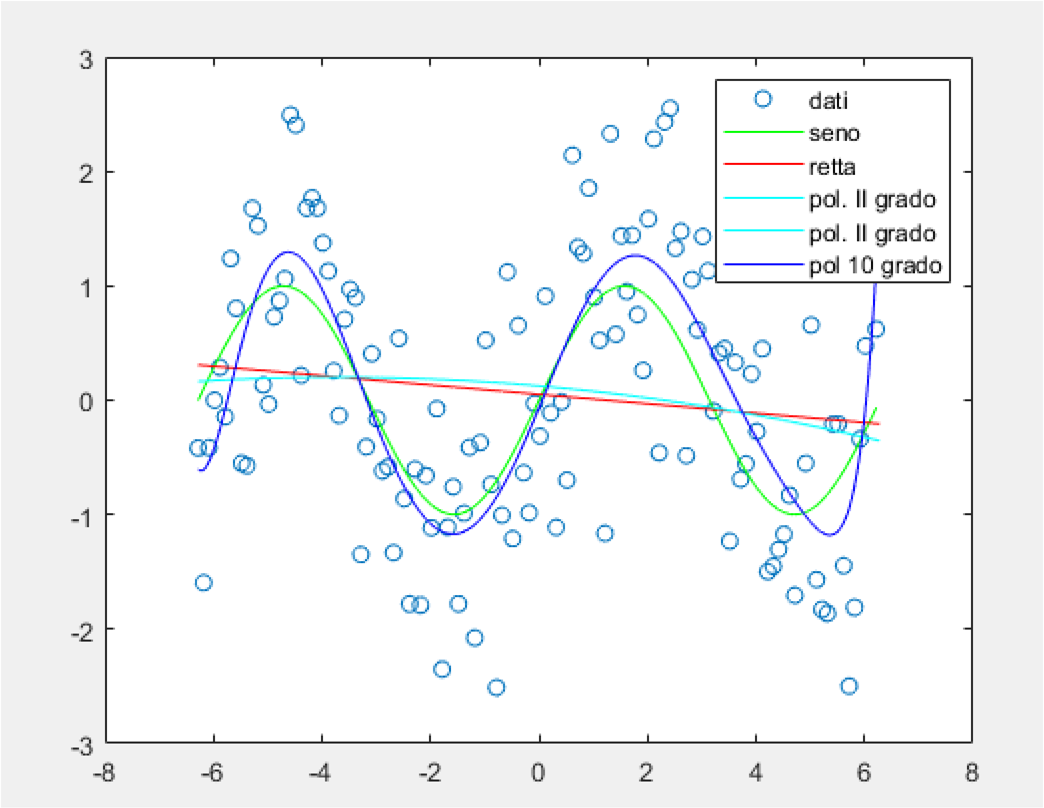


Confronto interpolazione-approssimazione

Interpolazione



Approssimazione



Problema dei minimi quadrati

retta

f(x)=a0+a1x

minimizzare

$$\sum_{i=1}^{m}\left[y_i-(a_0+a_1x_i)\right]^2$$

Per calcolare i coeff. Della retta dei minimi quadrati (a0,a1)

Con matlab tramite la funzione polyfit e polyval per la valutazione del polinomio in corrispondenza di un vettore input x

polyfit

Polynomial curve fitting

collapse

Syntax

```
p = polyfit(x,y,n)
[p,S] = polyfit(x,y,n)
[p,S,mu] = polyfit(x,y,n)
```

Description

`p = polyfit(x,y,n)` returns the coefficients for a polynomial `p(x)` of degree `n` that is a best fit (in a least-squares sense) for the data in `y`. The coefficients in `p` are in descending powers, and the length of `p` is `n+1`

$$p(x) = p_1 x^n + p_2 x^{n-1} + \dots + p_n x + p_{n+1}.$$

MATLAB

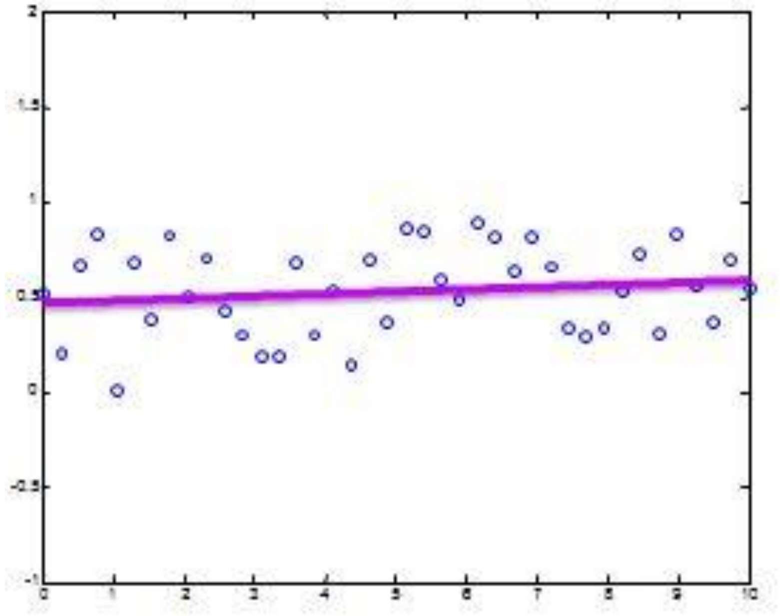
coef=polyfit(x,y,g)

- ✓x= vettore di ascisse(distinte) lungo n
- ✓y= vettore di valori noti lungo n
- ✓g=grado del polinomio di minimi quadrati(<n-1)
- ✓coef=coefficienti del polinomio

f=polyval(coef,t)

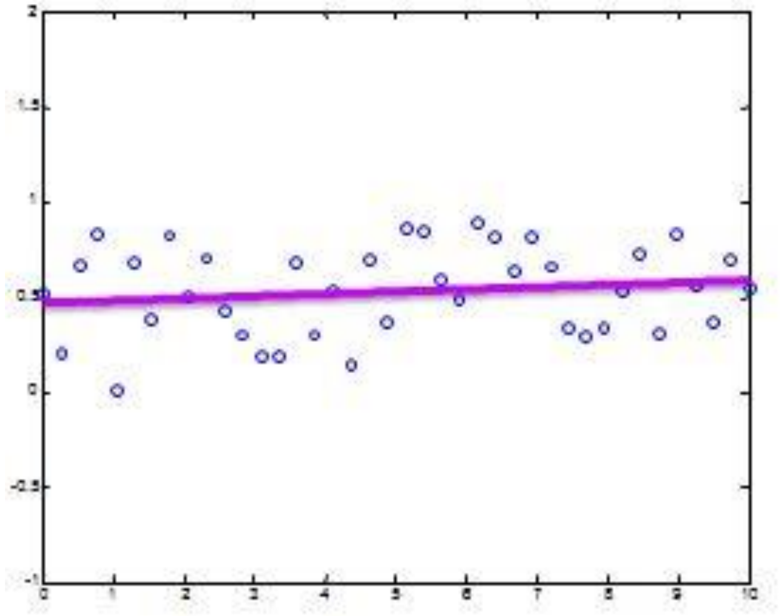
- ✓coef=coefficienti del polinomio
- ✓t=vettore di punti in cui valutare il polinomio
- ✓f= vettore di valori del polinomio nei punti t

```
>> x=linspace(0,10,40);
>> y=rand(1,40);
>> t=0:0.1:10;
>> p=polyfit(x,y,1);
>> s=polyval(p,t);
>> plot(x,y,'o',t,tt)
>>axis([0 10 -1 2])
```



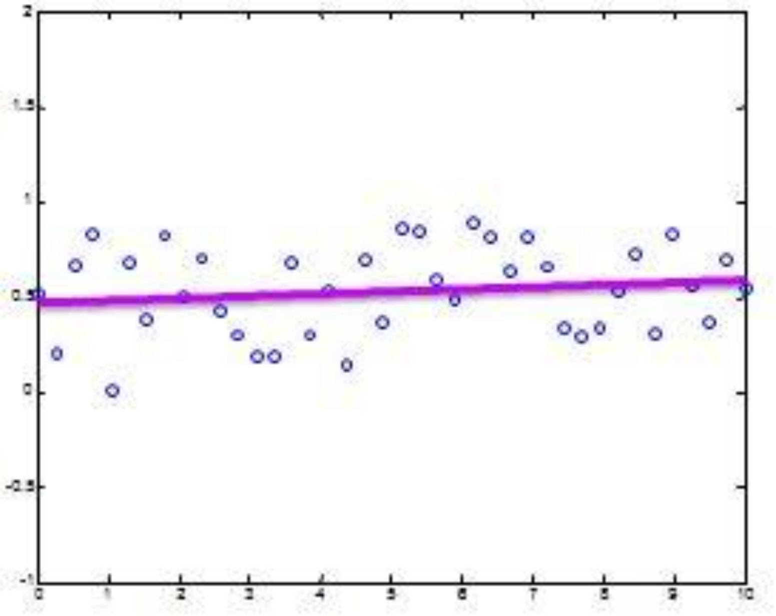
Retta minimi quadrati

```
>> x=linspace(0,10,40);
>> y=rand(1,40);
>> t=0:0.1:10;
>> p=polyfit(x,y,1);
>> s=polyval(p,t);
>> plot(x,y,'o',t,tt)
>>axis([0 10 -1 2])
```

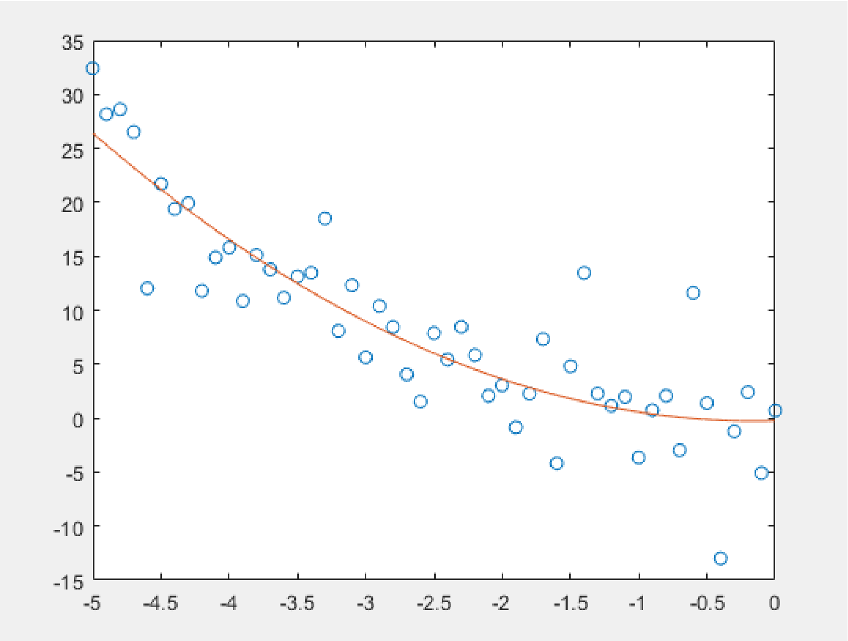


```
>> x=linspace(0,10,40);
>> y=rand(1,40);
>> t=0:0.1:10;
>> p=polyfit(x,y,1);
>> s=polyval(p,t);
>> plot(x,y,'o',t,tt)
>>axis([0 10 -1 2])
```

Retta minimi quadrati

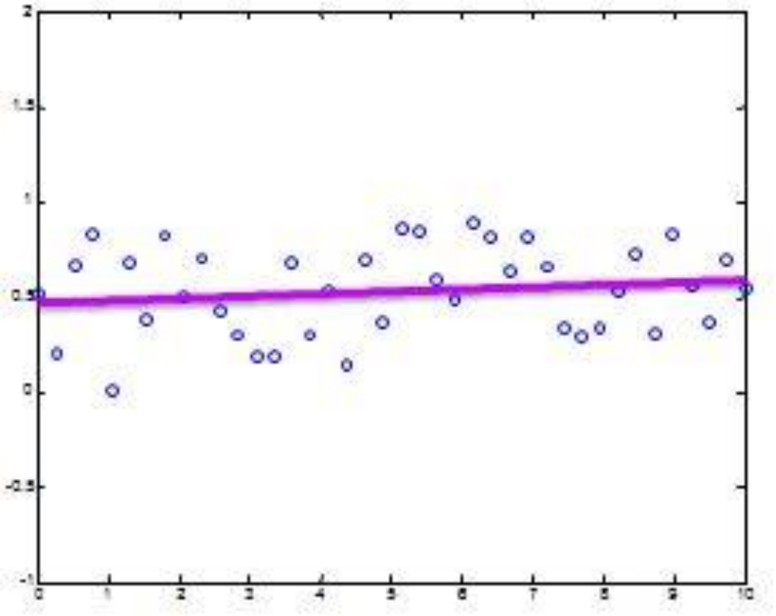


```
x=-5:0.1:0;
y=x.^2;
p=5*randn(1,length(x));
yp=y+p;
c=polyfit(x,yp,2);
t=-5:0.001:0;
v=polyval(c,t);
plot(x,yp,'o',t,v)
```



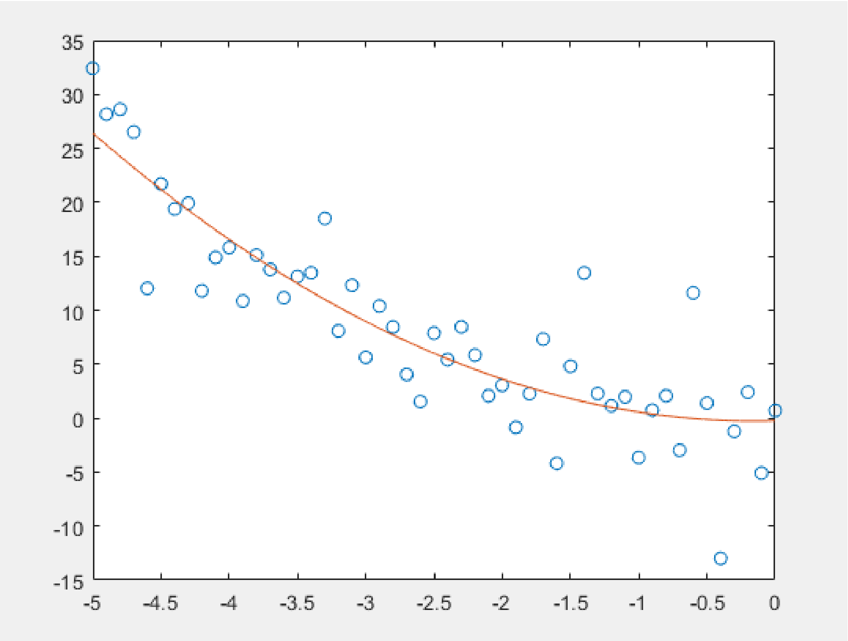
```
>> x=linspace(0,10,40);
>> y=rand(1,40);
>> t=0:0.1:10;
>> p=polyfit(x,y,1);
>> s=polyval(p,t);
>> plot(x,y,'o',t,tt)
>>axis([0 10 -1 2])
```

Retta minimi quadrati



```
x=-5:0.1:0;
y=x.^2;
p=5*randn(1,length(x));
yp=y+p;
c=polyfit(x,yp,2);
t=-5:0.001:0;
v=polyval(c,t);
plot(x,yp,'o',t,v)
```

Parabola dei minimi quadrati



Problema dei minimi quadrati

retta

f(x)=a0+a1x

minimizzare

$$\sum_{i=1}^{m}\left[y_i-(a_0+a_1x_i)\right]^2$$

Problema dei minimi quadrati

retta

f(x)=a0+a1x

minimizzare

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

Formulazione generale del problema dei minimi quadrati

$$\underset{x}{\textbf{argmin}} \; \| A x - b \|_2^2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij}x_j - b_i\right)^2$$

Problema dei minimi quadrati

retta

f(x)=a0+a1x

minimizzare

$$\sum_{i=1}^{m}\left[y_i-(a_0+a_1x_i)\right]^2$$

Formulazione generale del problema dei minimi quadrati

$$\underset{x}{\textbf{argmin}} \quad \| Ax-b\|_2^2=\sum_{i=1}^{m}\left(\sum_{j=1}^{n}a_{ij}x_j-b_i\right)^2 \longrightarrow$$

$A \in R^{mxn}$

Con m>n o m<n

Problema dei minimi quadrati

retta

f(x)=a0+a1x

minimizzare

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

Formulazione generale del problema dei minimi quadrati

$$\underset{x}{\textbf{argmin}} \; \| A\,x - b \|_2^2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij} x_j - b_i\right)^2$$

$A \in R^{mxn}$

Con m>n o m<n

Equivale a risolvere il seguente sistema

$$\left(A^t A\right) x = A^t b$$

Sistema equazioni normali

Problema dei minimi quadrati

retta

f(x)=a0+a1x

minimizzare

$$\sum_{i=1}^{m}\left[y_i-(a_0+a_1x_i)\right]^2$$

Formulazione generale del problema dei minimi quadrati

$$\underset{x}{\textbf{argmin}} \; \| A x - b \|_2^2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij}x_j - b_i\right)^2$$

$A \in R^{mxn}$

Con m>n o m<n

Equivale a risolvere il seguente sistema

Matrice dei coeff. quadrata (nxn)

$$\left(A^t A\right)x = A^t b$$

Sistema equazioni normali

Matlab mette a disposizione la routine lscov per calcolare la soluzione di problemi dei minimi quadrati

lscov

R2020a

Least-squares solution in presence of known covariance

Syntax

```
x = lscov(A,B)
x = lscov(A,B,w)
x = lscov(A,B,V)
x = lscov(A,B,V,alg)
[x,stdx] = lscov(...)
[x,stdx,mse] = lscov(...)
[x,stdx,mse,S] = lscov(...)
```

Description

`x = lscov(A,B)` returns the ordinary least squares solution to the linear system of equations `A*x = B`, i.e., `x` is the n-by-1 vector that minimizes the sum of squared errors `(B - A*x)'*(B - A*x)`, where `A` is m-by-n, and `B` is m-by-1. `B` can also be an m-by-k matrix, and `lscov` returns one solution for each column of `B`. When `rank(A) < n`, `lscov` sets the maximum possible number of elements of `x` to zero to obtain a "basic solution".

Risoluzione, tramite la routine matlab lscov, del problema dei minimi quadrati

```
>> A=rand(50,20);
>> b=rand(50,1);
>> V=0.5*eye(50,50);
>> [x]=lscov(A,b,V,'chol')
x =
    0.2868
    0.0607
    0.3014
    0.0739
   -0.1090
   -0.1021
   -0.2157
   -0.2669
    0.0944
    0.0727
    0.3381
    0.1480
   -0.3128
    0.3573
    0.1371
   -0.1898
    0.3461
    0.2118
   -0.1088
   -0.0290
```

```
>> size(x)

ans =

    20     1
```

Risoluzione, tramite la routine matlab lscov, del problema dei minimi quadrati

```
>> A=rand(50,20);
>> b=rand(50,1);
>> V=0.5*eye(50,50);
>> [x]=lscov(A,b,V,'chol')
x =
   0.2868
   0.0607
   0.3014
   0.0739
  -0.1090
  -0.1021
  -0.2157
  -0.2669
   0.0944
   0.0727
   0.3381
   0.1480
  -0.3128
   0.3573
   0.1371
  -0.1898
   0.3461
   0.2118
  -0.1088
  -0.0290
```

Dati

Matrice di covarianza (oppure vettore dei posi)

Metodo da utilizzare

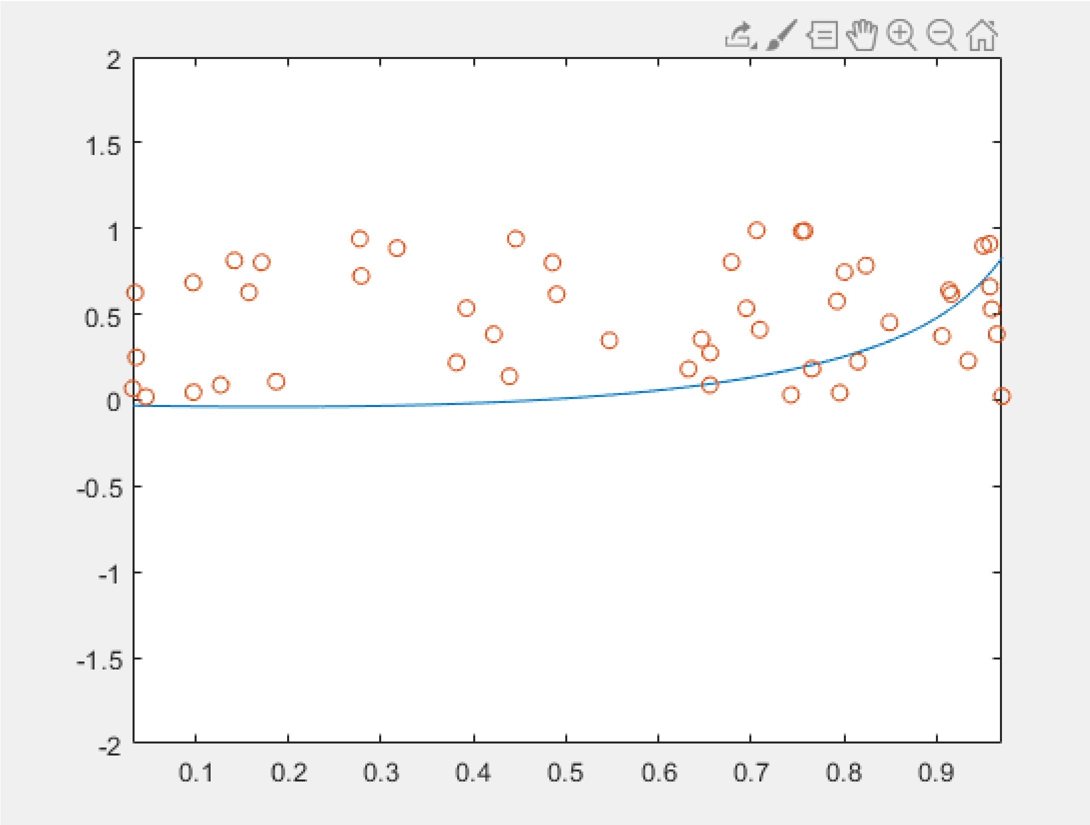
```
>> size(x)

ans =

    20     1
```

Dimensione della soluzione

Risoluzione, tramite la routine matlab lscov, del problema dei minimi quadrati

```
>> A=rand(50,20);
>> b=rand(50,1);
>> V=0.5*eye(50,50);
>> [x]=lscov(A,b,V,'chol')
x =
   0.2868
   0.0607
   0.3014
   0.0739
  -0.1090
  -0.1021
  -0.2157
  -0.2669
   0.0944
   0.0727
   0.3381
   0.1480
  -0.3128
   0.3573
   0.1371
  -0.1898
   0.3461
   0.2118
  -0.1088
  -0.0290
```

```
>> size(x)

ans =

    20     1
```



X, ad esempio, potrebbero essere
I coeff. Di un polinomio approssimante,
Come nel caso della retta dei minimi quadrati
X=[a 0 a1]’.

Consideriamo la seguente matrice A e il vettore b

$$A = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \in \mathbb{R}^{(m_0+m_1)\times n}, \quad b = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \in \mathbb{R}^{m_0+m_1},$$

Problema dei minimi quadrati vincolato $\hat{x} = argmin_{x\in\mathbb{R}^n} J(x)$

Consideriamo la seguente matrice A e il vettore b

$$A = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \in \mathbb{R}^{(m_0+m_1)\times n}, \quad b = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \in \mathbb{R}^{m_0+m_1},$$

Problema dei minimi quadrati vincolato $\hat{x} = argmin_{x \in \mathbb{R}^n} J(x)$

Aggiunta di ulteriore dati

dove

$$J(x) = ||Ax - b||_R^2 = ||H_0 x - y_0||_{R_0}^2 + \boxed{||H_1 x - y_1||_{R_1}^2,}$$

Aggiungo ulteriori dati

```
Nx=20;
k=50;
A=rand(k/2,Nx);
b=rand(k/2,1);
R=eye(k/2,k/2);
[x1]=lscov(A,b,R,'chol');
```

```
>> size(x1)

ans =

    20     1
```

Devo risolvere un altro sistema

```
A=[A;H];
b=[b;y];

Q=0.5*eye(Nx,Nx);
[x]=lscov(A,b,0.5*eye(size(A,1),size(A,1))
'chol');
```

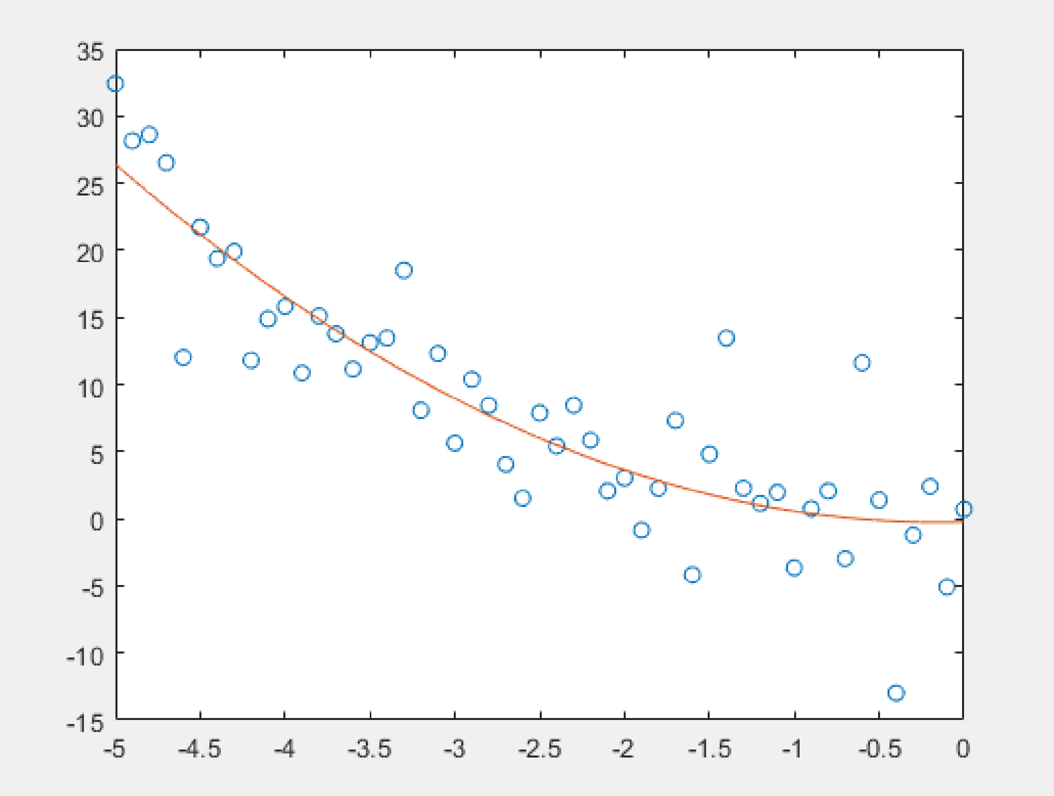
```
>> size(x)

ans =

    20     1
```

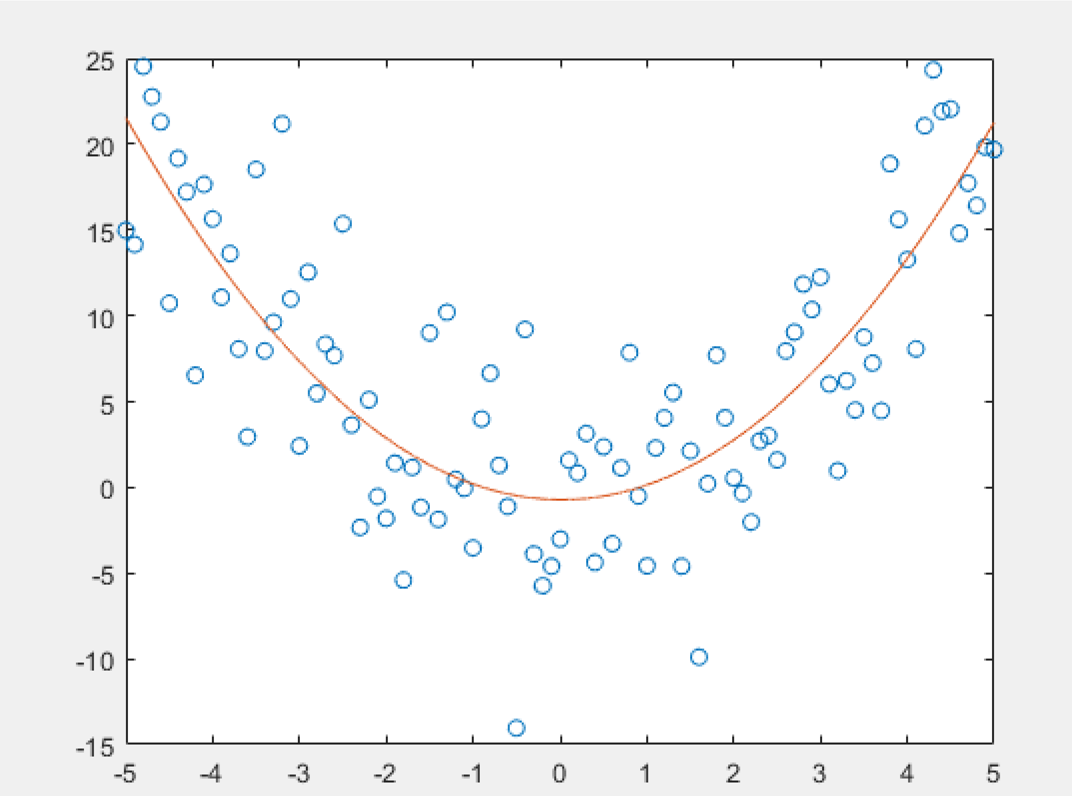
oppure

```
x=-5:0.1:0;
y=x.^2;
p=5*randn(1,length(x));
yp=y+p;
c=polyfit(x,yp,2);
t=-5:0.001:0;
v=polyval(c,t);
plot(x,yp,'o',t,v)
```

Aggiungiamo nuovi dati
Dobbiamo ricalcolare il polinomio

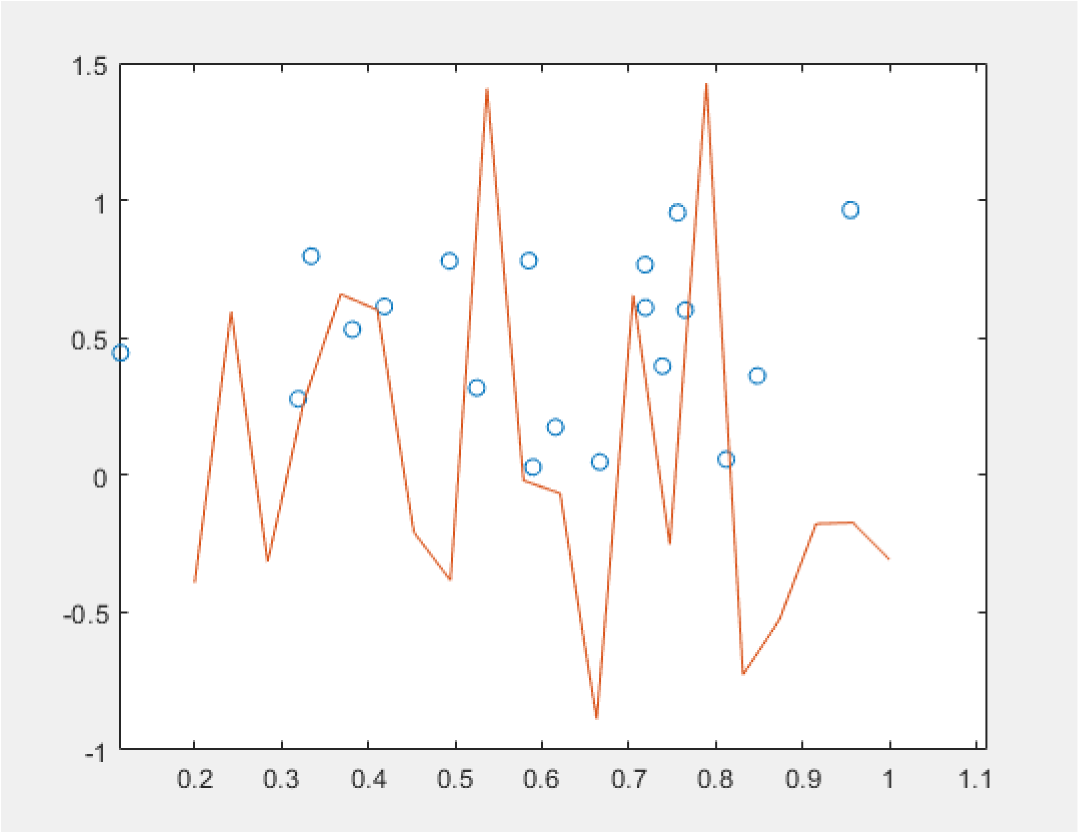
```
x=-5:0.1:5;
y=x.^2;
p=5*randn(1,length(x));
yp=y+p;
c=polyfit(x,yp,2);
t=-5:0.001:5;
v=polyval(c,t);
plot(x,yp,'o',t,v)
```





Kalman Filter

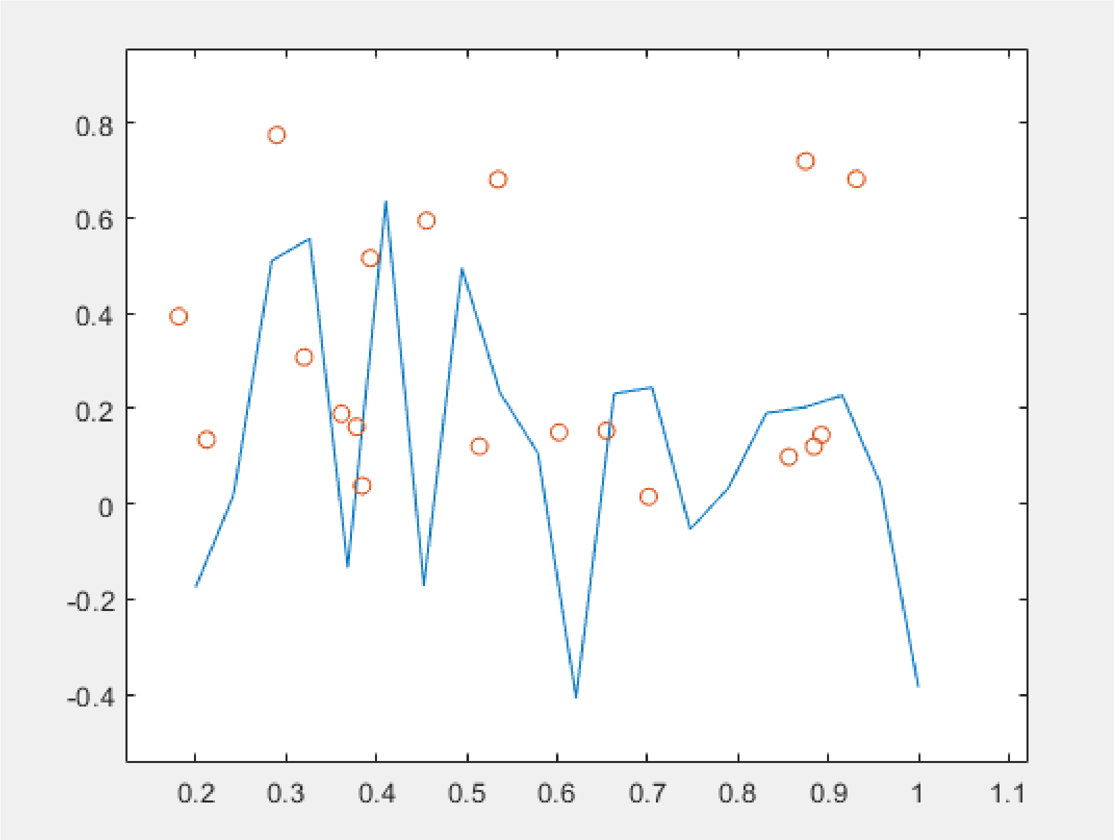
Utilizzando matrici e vettori random: al primo passo



Kalman Filter

Utilizzando matrici e vettori random: secondo passo,

Soluzione calcolata a partire dal quella al passo precedente



Passaggio dal problema dei minimi quadrati al problema della Data Assimilation (DA)

Problema dei minimi quadrati vincolato $\longleftrightarrow$ **Problema Data assimilation (DA)**

Per $k = 0, 1, \ldots, r$:

x^b previous forecast

$$\begin{aligned}\hat{x}_{k+1} &= argmin_{x_{k+1} \in \mathbb{R}^n} J_{k+1}(x_{k+1}) \\ &= argmin_{x_{k+1} \in \mathbb{R}^n} \left\{ ||x_{k+1} - \boxed{M_{k,k+1} \hat{x}_k} ||^2_{Q_k} + ||y_{k+1} - H_{k+1} x_{k+1}||^2_{R_{k+1}} \right\}.\end{aligned}$$

dove $\hat{x}_{k+1}$ stima al tempo tk

Discretizzazione mappa delle osservazioni

Discretizzazione modello

Problema DA

Integriamo al modello le osservazioni

Modello (Shallow water equations)

PDEs

osservazioni

$$\begin{cases} \frac{\partial h}{\partial t} + \frac{\partial vh}{\partial x} = 0 \\ \frac{\partial vh}{\partial t} + \frac{\partial (v^2 h + \frac{1}{2} g h^2)}{\partial x} = 0 \end{cases}$$

t,x variabili indipendenti

h,v variabili dipendenti

$$y(t) = \mathcal{H}_t\,[x(t)],$$

mappa delle osservazioni

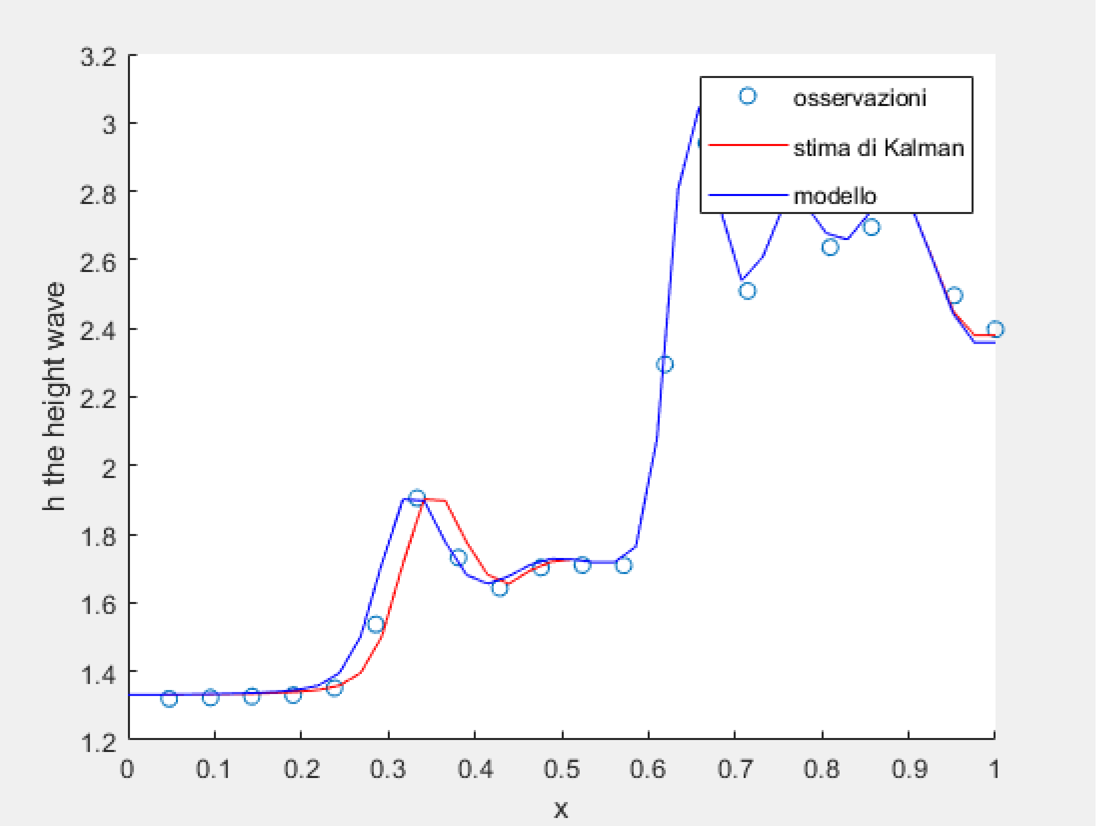
(non necessariamente lineari)

discretizzo

Solutore(Filtro di Kalman)

Problema DA

Fissato un istante t



Problema DA evolutivo

Modello(equazione di avvezione) ⟷ osservazioni

$$\frac{\partial a}{\partial t} + u\frac{\partial a}{\partial x} = 0, \qquad 0 \le x \le L, \quad t \ge 0.$$

t,x variabili indipendenti
a variabili dipendenti

$$y(t) = \mathcal{H}_t\,[x(t)],$$

mappa delle osservazioni
(non necessariamente lineari)

↓

discretizzo

↓

Filtro di Kalman

Problema DA evolutivo