
Laurea Magistrale in Data Science, Calcolo Scientifico & Intelligenza Artificiale

Numerical approximation for
data modeling

Università degli Studi di Firenze
a.a. 2024/2025

Carlotta Giannelli

Curve parametriche

Preliminari

Curve di Bézier

Curve B-spline

- 1 Considerare la rappresentazione trigonometrica e razionale di una (semi)-circonferenza centrata nell'origine e raggio R ,

$$X(t) = R \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \quad \theta \in [-\frac{\pi}{2}, +\frac{\pi}{2}], \quad X(t) = R \begin{pmatrix} \frac{1-t^2}{1+t^2} \\ \frac{2t}{1+t^2} \end{pmatrix} \quad t \in [-1, 1]$$

e vedere la distribuzione dei punti sulla circonferenza al variare di intervalli parametrici uniformi nei due casi.

- 2 Scrivere la function $\kappa = \text{curvature}(dX, d2X)$ che, date la derivata prima e seconda di una curva, calcoli la curvatura (con segno nel caso piano):

$$\kappa(t) := \frac{|\dot{X}(t) \wedge \ddot{X}(t)|}{|\dot{X}(t)|^3}$$

al variare di t nell'intervallo parametrico.

- 3 Verificare che nel caso della circonferenza la curvatura è costante (e uguale a $1/R$) per ogni valore del parametro di definizione.

Curve parametriche nello spazio

Preliminari

Curve di
Bézier

Curve
B-spline

- 1 Scrivere la function $\text{tau} = \text{torsion}(\text{dX}, \text{d2X}, \text{d3X})$ che, date la derivata prima, seconda e terza di una curva nello spazio, calcoli la torsione:

$$\tau(t) = \frac{\det(\dot{X}(t) \ddot{X}(t) \ddot{\ddot{X}}(t))}{|\dot{X}(t) \wedge \ddot{X}(t)|^2}$$

Considerare l'elica circolare retta

$$X(\theta) = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \\ a\theta \end{pmatrix}, \quad \theta \in [0, T]$$

e verificare che in questo caso curvatura e torsione sono costanti e uguali a

$$\kappa(\theta) = \frac{1}{1+a^2}, \quad \tau(\theta) = \frac{a}{1+a^2}.$$

Polinomi di Bernstein e curve di Bézier

- 1 Scrivere la function `B = bernstein(i,n,t)` che calcola l' i -esimo polinomio di Bernstein di grado n :

$$B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}, \quad \binom{n}{i} = \frac{n!}{i!(n-i)!}, \quad i = 0, \dots, n$$

Fare il plot dei polinomi di base al variare di n fra 1 e 10 e verificare la proprietà di partizione dell'unità.

- 2 Utilizzando la funzione `ginput` per definire una serie di punti di controllo,
 - visualizzate la corrispondente curva di Bézier piana insieme al grafico della sua curvatura (dopo aver calcolato i punti sulla curva e sulle sue derivate prima e seconda rispetto ad una tabulazione fissata);
 - verificate il valore della curvatura negli estremi dell'intervallo parametrico.
- 3 Verificare cosa succede scambiando due punti di controllo o modificandone uno, o invertendo l'ordine di tutti i punti di controllo.
- 4 Verificare la proprietà di linear precision.
- 5 Definire una curva di Bézier nello spazio e visualizzarla insieme al grafico della sua curvatura e torsione (dopo aver calcolato i punti sulla curva e sulle sue derivate prima, seconda e terza rispetto ad una tabulazione fissata).

B-splines nel MATLAB toolbox

Preliminari

Curve di
Bézier

Curve
B-spline

Documentation » Curve fitting toolbox » Splines

The screenshot shows the MATLAB Documentation website. The browser address bar displays "MATLAB Documentation". The page header includes a search bar and navigation links like "Explore Examples" and "Explore Add-Ons". On the left, a "CONTENTS" sidebar lists various MATLAB products, with "Curve Fitting Toolbox" highlighted. The main content area is titled "Documentation" and features a search bar. Below the header, there's a section for "My Products" with a list of toolboxes. The "Curve Fitting Toolbox" is selected, showing its description: "MATLAB® is the high-level language and interactive environment used by millions of engineers and scientists worldwide. The matrix-based language is a natural way to express computational mathematics." To the right, there's a section for "Getting Started with MATLAB" with links to "Functions in MATLAB", "Release Notes", and "Installation". Below this, there's a "My Products" section with three columns: "MATLAB® Family", "Simulink® Family", and "Hardware Support". The "MATLAB® Family" column lists various toolboxes, including "Curve Fitting Toolbox", "Global Optimization Toolbox", "Neural Network Toolbox", "Optimization Toolbox", "Partial Differential Equation Toolbox", "Statistics and Machine Learning Toolbox", and "Symbolic Math Toolbox". The "Simulink® Family" column lists "Simulink", "Event-Based Modeling", "SimEvents", "Stateflow", "Physical Modeling", "SimRF", "Simscape", "Simscape Electronics", "Simscape Fluids", "Simscape Multibody", "Simscape Power Systems", "Control Systems", and "Simulink Control Design". The "Hardware Support" column has a link to "Hardware Support".

Documentation

Search Help

CONTENTS Close

My Products

MATLAB

Simulink

Bioinformatics Toolbox

Communications System Toolbox

Computer Vision System Toolbox

Control System Toolbox

Curve Fitting Toolbox

Database Toolbox

Datafeed Toolbox

DSP System Toolbox

Econometrics Toolbox

Embedded Coder

Filter Design HDL Coder

Financial Instruments Toolbox

Financial Toolbox

Fixed-Point Designer

Fuzzy Logic Toolbox

Global Optimization Toolbox

Image Acquisition Toolbox

Image Processing Toolbox

Instrument Control Toolbox

LTE System Toolbox

Mapping Toolbox

MATLAB

MATLAB® is the high-level language and interactive environment used by millions of engineers and scientists worldwide. The matrix-based language is a natural way to express computational mathematics.

Explore Examples Explore Add-Ons

Getting Started with MATLAB

Functions in MATLAB

Release Notes

Installation

Edit Preferences

My Products

MATLAB® Family

MATLAB

Parallel Computing

MATLAB Distributed Computing Server

Parallel Computing Toolbox

Math, Statistics, and Optimization

Curve Fitting Toolbox

Global Optimization Toolbox

Neural Network Toolbox

Optimization Toolbox

Partial Differential Equation Toolbox

Statistics and Machine Learning Toolbox

Symbolic Math Toolbox

Control Systems

Simulink® Family

Simulink

Event-Based Modeling

SimEvents

Stateflow

Physical Modeling

SimRF

Simscape

Simscape Electronics

Simscape Fluids

Simscape Multibody

Simscape Power Systems

Control Systems

Simulink Control Design

Hardware Support

For a complete list of hardware solutions, see [Hardware Support](#).

B-splines nel MATLAB toolbox

Preliminari

Curve di
Bézier

Curve
B-spline

Documentation » Curve fitting toolbox » Splines

The screenshot shows the MATLAB documentation interface for the 'Splines' section of the Curve Fitting Toolbox. The browser window has a tab labeled 'Splines'. The page header is blue with the word 'Documentation' and a search bar. A left sidebar contains a 'CONTENTS' menu with a 'Close' button. The sidebar lists the following items: 'Documentation Home', 'Curve Fitting Toolbox' (with a sub-menu icon), 'Getting Started with Curve Fitting Toolbox', 'Linear and Nonlinear Regression', 'Interpolation', 'Smoothing', 'Fit Postprocessing', 'Splines' (highlighted), 'Construction', 'Postprocessing', 'Breaks, Knots, and Sites', 'Examples', 'Functions', 'Apps', 'Release Notes', and 'PDF Documentation'. The main content area is titled 'Splines' and contains the following text: 'Construct splines with or without data; ppform, B-form, tensor-product, rational, and stform thin-plate splines'. Below this, there are three sections: 'Construction' (Create splines including B-form, tensor-product, NURBs and other rational splines), 'Postprocessing' (Evaluate splines, plot, find minimum or zero-crossings, integrate or differentiate), and 'Breaks, Knots, and Sites' (Optimize knots and breaks).

Documentation

Search Help

CONTENTS Close

Documentation Home

Curve Fitting Toolbox

Getting Started with Curve Fitting Toolbox

Linear and Nonlinear Regression

Interpolation

Smoothing

Fit Postprocessing

Splines

Construction

Postprocessing

Breaks, Knots, and Sites

Examples

Functions

Apps

Release Notes

PDF Documentation

Splines

Construct splines with or without data; ppform, B-form, tensor-product, rational, and stform thin-plate splines

Construction

Create splines including B-form, tensor-product, NURBs and other rational splines

Postprocessing

Evaluate splines, plot, find minimum or zero-crossings, integrate or differentiate

Breaks, Knots, and Sites

Optimize knots and breaks

Construction

Preliminari

Curve di
Bézier

Curve
B-spline

Documentation » Curve fitting toolbox » Splines » Construction

- **Functions**
 - `bspline` \Rightarrow provare `bspligui`

Construction

Preliminari

Curve di
Bézier

Curve
B-spline

Documentation » Curve fitting toolbox » Splines » Construction

- **Functions**

- bspline \Rightarrow provare bspligui
- B-spline di grado 1 e 2 su nodi uniformi

1) Confrontare i comandi

bspline([0 1 2]) e bspline([0 1 2 3])

con

$$N_{0,2}(t) = \begin{cases} t & \text{se } t \in [0, 1) \\ 2 - t & \text{se } t \in [1, 2] \end{cases} \quad N_{0,3}(t) = \begin{cases} t^2 & \text{se } t \in [0, 1) \\ \frac{t(2-t)+(3-t)(t-1)}{2} & \text{se } t \in [1, 2) \\ \frac{(3-t)^2}{2} & \text{se } t \in [2, 3] \end{cases}$$

Construction

Preliminari

Curve di
Bézier

Curve
B-spline

Documentation » Curve fitting toolbox » Splines » Construction

- **Functions**

- bspline \Rightarrow provare bspligui
- B-spline di grado 1 e 2 su nodi uniformi

1) Confrontare i comandi

bspline([0 1 2]) e bspline([0 1 2 3])

con

$$N_{0,2}(t) = \begin{cases} t & \text{se } t \in [0, 1) \\ 2 - t & \text{se } t \in [1, 2] \end{cases} \quad N_{0,3}(t) = \begin{cases} t^2 & \text{se } t \in [0, 1) \\ \frac{t(2-t)+(3-t)(t-1)}{2} & \text{se } t \in [1, 2) \\ \frac{(3-t)^2}{2} & \text{se } t \in [2, 3] \end{cases}$$

2) Definire una B-spline su un vettore dei nodi locale generico

Documentation » Curve fitting toolbox » Splines » Construction

- **Functions**

- `bspline` \Rightarrow provare `bspligui`
- B-spline di grado 1 e 2 su nodi uniformi

1) Confrontare i comandi

`bspline([0 1 2])` e `bspline([0 1 2 3])`

con

$$N_{0,2}(t) = \begin{cases} t & \text{se } t \in [0, 1) \\ 2 - t & \text{se } t \in [1, 2] \end{cases} \quad N_{0,3}(t) = \begin{cases} t^2 & \text{se } t \in [0, 1) \\ \frac{t(2-t)+(3-t)(t-1)}{2} & \text{se } t \in [1, 2) \\ \frac{(3-t)^2}{2} & \text{se } t \in [2, 3] \end{cases}$$

2) Definire una B-spline su un vettore dei nodi locale generico

3) Utilizzando il comando `spco1`, scrivere una function che dati il vettore dei nodi e il grado da considerare, disegni il grafico della base delle B-spline.

B-spline di grado d

Preliminari

Curve di
Bézier

Curve
B-spline

```
function getBsplineBasis(t,d)
    x = [t(1):.1:t(end)];
    c = spcol(t,d+1,x);
    [l,m] = size(c);
    figure (1); hold on;
    for tt=t, plot([tt tt],[0 1],'-'), end
    plot(x,c,'linew',2); axis image; box;
    figure(2); hold on;
    c = c+ones(l,1)*[0:m-1];
    for tt=t,plot([tt tt],[0 m],'-'), end
    plot(x,c,'linew',2), axis image; box;
end
```

B-spline

Preliminari

Curve di
Bézier

Curve
B-spline

4) Utilizzando i comandi `spmak` e `fnplt` visualizzare i polinomi di Bernstein di grado 1, 2, 3 ($k + 1$ zeri e uni nel vettore dei nodi)

B-spline

Preliminari

Curve di
Bézier

Curve
B-spline

4) Utilizzando i comandi `spmak` e `fnplt` visualizzare i polinomi di Bernstein di grado 1, 2, 3 ($k + 1$ zeri e uni nel vettore dei nodi)

5) Scrivere una function che visualizza i polinomi di Bernstein di grado d utilizzando `spcol`.

B-spline

Preliminari

Curve di
Bézier

Curve
B-spline

4) Utilizzando i comandi `spmak` e `fnplt` visualizzare i polinomi di Bernstein di grado 1, 2, 3 ($k + 1$ zeri e uni nel vettore dei nodi)

5) Scrivere una function che visualizza i polinomi di Bernstein di grado d utilizzando `spcol`.

```
function getBerp(d)
    knots = [zeros(1,d+1) ones(1,d+1)];
    x = [0:.01:1];
    c = spcol(knots,d+1,x);
    plot(x,c,'linew',2)
    title(['Polinomi di Bernstein di grado ', num2str(d)]);
    axis equal; axis([0 1 0 1]);
end
```

6) Verifica della proprietà di partizione dell'unità

B-spline

Preliminari

Curve di
Bézier

Curve
B-spline

4) Utilizzando i comandi `spmak` e `fnplt` visualizzare i polinomi di Bernstein di grado 1, 2, 3 ($k + 1$ zeri e uni nel vettore dei nodi)

5) Scrivere una function che visualizza i polinomi di Bernstein di grado d utilizzando `spcol`.

```
function getBerp(d)
    knots = [zeros(1,d+1) ones(1,d+1)];
    x = [0:.01:1];
    c = spcol(knots,d+1,x);
    plot(x,c,'linew',2)
    title(['Polinomi di Bernstein di grado ', num2str(d)]);
    axis equal; axis([0 1 0 1]);
end
```

6) Verifica della proprietà di partizione dell'unità

7) Interpolazione funzionale

- Considerare il vettore dei breakpoints $[0, 0.25, 0.5, 0.75, 1]$.
- Utilizzare il comando `augknt` per definire il vettore esteso dei nodi t (con nodi ausiliari ripetuti) al variare di $k = 2, 3, 4$.
- Utilizzare `aveknt` per definire il vettore $x = [x_0, \dots, x_n]$ delle ascisse di interpolazione (ascisse di Greville)

$$x_i = \frac{t_{i+1} + \dots + t_{k-1}}{k-1} = \frac{\sum_{j=1}^{k-1} t_{i+j}}{k-1}, \quad i = 0, \dots, n,$$

definite come valor medio di $k-1$ nodi consecutivi ($n+1 = \text{length}(t) - k$).

- Associare un valore y_i ad ogni ascissa di interpolazione (es. con la funzione `rand(length(x), 1)`) per definire il vettore di valori da interpolare $y = [y_0, \dots, y_n]^T$.
- Calcolare la funzione B-spline $s(t)$ interpolante i valori y_i nelle ascisse x_i :

$$s(t) = \sum_{j=0}^n d_j N_{j,k}(t) \quad \text{t.c.} \quad s(x_i) = \sum_{j=0}^n d_j N_{j,k}(x_i) = y_i, \quad i = 0, \dots, n,$$

risolvendo il sistema $Cd = y$, dove C è la matrice di collocazione delle B-spline (`spcol`), per calcolare il vettore $d = [d_0, \dots, d_n]^T$.

- Confrontare il risultato con la spline calcolata da `spapi`.

7) Interpolazione funzionale

- Considerare il vettore dei breakpoints $[0, 0.25, 0.5, 0.75, 1]$.
- Utilizzare il comando `augknt` per definire il vettore esteso dei nodi t (con nodi ausiliari ripetuti) al variare di $k = 2, 3, 4$.
- Utilizzare `aveknt` per definire il vettore $x = [x_0, \dots, x_n]$ delle ascisse di interpolazione (ascisse di Greville)

$$x_i = \frac{t_{i+1} + \dots + t_{k-1}}{k-1} = \frac{\sum_{j=1}^{k-1} t_{i+j}}{k-1}, \quad i = 0, \dots, n,$$

definite come valor medio di $k-1$ nodi consecutivi ($n+1 = \text{length}(t) - k$).

- Associare un valore y_i ad ogni ascissa di interpolazione (es. con la funzione `rand(length(x), 1)`) per definire il vettore di valori da interpolare $y = [y_0, \dots, y_n]^T$.
- Calcolare la funzione B-spline $s(t)$ interpolante i valori y_i nelle ascisse x_i :

$$s(t) = \sum_{j=0}^n d_j N_{j,k}(t) \quad \text{t.c.} \quad s(x_i) = \sum_{j=0}^n d_j N_{j,k}(x_i) = y_i, \quad i = 0, \dots, n,$$

risolvendo il sistema $Cd = y$, dove C è la matrice di collocazione delle B-spline (`spcol`), per calcolare il vettore $d = [d_0, \dots, d_n]^T$.

- Confrontare il risultato con la spline calcolata da `spapi`.

8) Curve B-spline

Spline Function Naming Conventions

Preliminari

Curve di
Bézier

Curve
B-spline

Most of the spline commands in this toolbox have names that follow one of the following patterns:

- `cs...` commands construct cubic splines (in ppform)
- `sp...` commands construct splines in B-form
- `fn...` commands operate on spline functions
- `..2...` commands convert something
- `..api` commands construct an approximation by interpolation
- `..aps` commands construct an approximation by smoothing
- `..ap2` commands construct a least-squares approximation
- `...knt` commands construct (part of) a particular knot sequence
- `...dem` commands are examples.

Parametrizzazioni

Preliminari

Curve di
Bézier

Curve
B-spline

Siano

- $\mathbf{t} := \{t_0, \dots, t_{n+k}\}$: vettore esteso dei nodi associato a $[a, b] = [0, 1]$;
- $\{\mathbf{p}_i\}_{i=0, \dots, n}$: punti di interpolazione.

Parametrizzazioni

Preliminari

Curve di
Bézier

Curve
B-spline

Siano

- $\mathbf{t} := \{t_0, \dots, t_{n+k}\}$: vettore esteso dei nodi associato a $[a, b] = [0, 1]$;
- $\{\mathbf{p}_i\}_{i=0, \dots, n}$: punti di interpolazione.

I valori del parametro $\{x_i\}_{i=0, \dots, n}$ in cui interpolare i punti \mathbf{p}_i possono essere scelti tramite:

- ascisse di Greville:

$$x_i = \frac{t_{i+1} + \dots + t_{k-1}}{k-1} = \frac{\sum_{j=1}^{k-1} t_{i+j}}{k-1}, \quad i = 0, \dots, n,$$

- parametrizzazione uniforme:

$$x_i = i/n$$

- parametrizzazione chordlength: posto $\Delta \mathbf{p}_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|$, $i = 0, \dots, n-1$,

$$x_0 = 0, \quad x_{i+1} = x_i + \frac{\Delta \mathbf{p}_i}{\Delta \mathbf{p}}, \quad \text{con} \quad \Delta \mathbf{p} = \sum_{i=0}^{n-1} \Delta \mathbf{p}_i$$

- parametrizzazione centripeta: posto $\Delta \mathbf{p}_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|^{1/2}$,
 $i = 0, \dots, n-1$,

$$x_0 = 0, \quad x_{i+1} = x_i + \frac{\Delta \mathbf{p}_i}{\Delta \mathbf{p}}, \quad \text{con} \quad \Delta \mathbf{p} = \sum_{i=0}^{n-1} \Delta \mathbf{p}_i.$$