Appunti dal corso di Metodi Numerici per la Grafica

Carli Samuele
E-mail: carlisamuele@csspace.net
www.csspace.net

17 ottobre 2010

Indice

Capitolo 1

Nozioni di base

Definizioni e richiami

Notazione

- v è un vettore
- v_i è l'i-esma componente del vettore v
- α, α_i è un numero (array di)
- P è un punto (ovvero un vettore d-dimensionale)
- X è una curva

dove non diversamente specificato.

Rappresentazione in Scilab

I vettori sono convenzionalmente intesi come vettori colonna[†].

Bulk operations

Gli oggetti grafici sono usualmente rappresentati come liste di punti; in generale una curva sarà rappresentata con una matrice $d \times n$[‡].
Per migliorare l'efficienza le operazioni tra vettori sono implementate in modo da eseguire la stessa operazione in blocco su liste di vettori di lunghezza arbitraria (ovvero punto a punto lungo una curva).

[†] da dichiarare esplicitamente tali in Scilab
[‡] In questo contesto d=2 o 3

Operazioni tra vettori

Prodotto scalare

Definizione 1.1 (Prodotto scalare)**.**

$$\bullet, \cdot^{\dagger} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

Proprietà

- $vw = |v||w| \cos \alpha = v^T w$
- $v, w \neq 0 \implies vw = 0 \Leftrightarrow v \perp w$
- $vw = wv$
- $\lambda vw = (\lambda v)w = v(\lambda w)$
- $v(u + w) = vu + vw$

Su una base canonica: $vw = \sum v_i w_i = v^T w$
Dati $v(t), w(t)$:

$$\frac{d}{dt} v(t)w(t) = \frac{dv(t)}{dt} w(t) + \frac{dw(t)}{dt} v(t)$$

Prodotto vettoriale

Definizione 1.2 (Prodotto vettoriale)**.**

$$\times, \wedge : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$$

con $d = 2, 3$

Proprietà

- $v \wedge w = u$
- $u = w \wedge v = 0 \Leftrightarrow v = 0$ o $w = 0$ o $w = v = 0$
- $u = w \wedge v = 0$ se w parallelo v
- u è perpendicolare al piano individuato da u, w con direzione convenzionale
- $|u| = |v||w||\sin \alpha|$
- $\lambda v \wedge w = (\lambda v) \wedge w = v \wedge (\lambda w)$

[†]Per brevità talvolta si omette completamente l'operatore: vw

Dati $v(t), w(t)$:

$$\frac{d}{dt}(v(t) \wedge w(t)) = \frac{dv(t)}{dt} \wedge w(t) + \frac{dw(t)}{dt} \wedge v(t)$$

Casi particolari:

$$\begin{pmatrix} a \\ b \end{pmatrix} \wedge \begin{pmatrix} c \\ d \end{pmatrix} = (ad - cb)$$

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \wedge \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} v_2 w_3 - v_3 w_2 \\ v_3 w_1 - v_1 w_3 \\ v_1 w_2 - v_2 w_1 \end{pmatrix}$$

Cross product 2D/3D su array di vettori

```
function cp = crossprod(x,y)
  [rx,cx] = size(x);
  [ry,cy] = size(y);
  if or([rx~=ry,cx~=cy]) then printf("Cannot croosprod different vectors!"); cp=0; end;
  if rx==2 then
    cp = diag( x' * [y(2,:);-y(1,:)] )'
  end;
  if rx==3 then
    cp = []
    for n=1:cx
      cp = [cp,[x(2,n)*y(3,n)-x(3,n)*y(2,n); x(3,n)*y(1,n)-x(1,n)*y(3,n); x(1,n)*y(2,n)-x
          (2,n)*y(1,n)]]
    end;
   end;
endfunction
```

Combinazioni di punti

Una combinazione di punti:

$$P = \sum_i \alpha_i P_i, \quad \alpha_i \in \mathbb{R},\ P_i \in \mathbb{E}^d$$

si dice *baricentrica* se $\sum_i \alpha_i = 1$, *convessa* se anche $\alpha_i > 0 \, \forall \, i$
Se la combinazione è baricentrica, può essere scritta come la somma di uno qualsiasi degli addendi e un vettore:

$$\sum_i \alpha_i P_i = P_k + \sum_i \alpha_i P_i - \sum_i \alpha_i P_k = P_k + \sum_i \alpha_i (P_i - P_k)$$

Una combinazione baricentrica convessa è utile per rappresentare parametrizzazioni di oggetti:

Esempio 1.1 (Punto medio di un segmento)**.**
$s = \overline{b_0 b_1}; \quad b_0, b_1 \in \mathbb{R}^2;$
$M = \sum_{i=1,2} \alpha_i b_i = b_0 + \alpha_0(b_0 - b_0) + \alpha_1(b_1 - b_0) = b_0 + \alpha_1(b_1 - b_0); \quad \alpha_1 = \alpha_2 = \frac{1}{2}$

Esempio 1.2 (Punti di un segmento)**.** Generalizzando l'esempio precedente:
$s = \overline{b_0 b_1}; \quad b_0, b_1 \in \mathbb{R}^2; \quad t \in [0, 1];$
$P(t) = t b_0 + (1 - t) b_1 = b_0 + (1 - t)(b_1 - b_0)$

Spazi di lavoro

Definizione 1.3 (Spazio di lavoro)**.** Si definisce *spazio affine* di ordine d, a partire da un punto $O \in \mathbb{R}^d$, l'insieme:

$$\mathbb{E}^d = \left\{ P = O + v \,|\, v \in \mathbb{R}^d \right\}$$

Definizione 1.4 (Convex Hull)**.**

$$CH_{b_0 \ldots b_n} = \left\{ P \in \mathbb{E}^d \,|\, P = \sum_i \alpha_i b_i \right\}$$

in cui gli α_i formano una combinazione baricentrica convessa.
CH è il più piccolo insieme convesso che contiene i punti $b_0 ... b_n$

Trasformazioni affini

Definizione 1.5 (Trasformazione affine)**.**

$$\Phi(x) : \mathbb{E}^d \to \mathbb{E}^d$$

lineare rispetto ad x.

Generalmente si usano funzioni del tipo

$$\Phi(x) = Ax + v \tag{1.1}$$

in cui $A \in \mathcal{M}^{d \times d}$ e $v \in \mathbb{R}^d$ che sono lineari e permettono di definire facilmente le trasformazioni più comuni in grafica.

Movimenti rigidi

Traslazione La traslazione si ottiene con una funzione del tipo (1.1) in cui $A \equiv I$, per cui $\Phi(x) = x + v$. Alternativamente è possibile modificare la matrice A:

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + a \\ y + b \\ 1 \end{pmatrix}$$

Rotazione La rotazione intorno all'origine (del sistema di riferimento ortonormale corrente) si ottiene ancora con una funzione del tipo (1.1) in cui $v \equiv 0$ (vettore nullo, non c'è traslazione) ed in cui la matrice, nel caso tridimensionale, assume la forma[†]:

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

[†]In questo esempio ruota intorno all'asse z; per cambiare asse è sufficiente modificare coerentemente la matrice. Ad esempio per ruotare intorno all'asse x: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$

Scalatura La scalatura si ottiene imponendo un fattore di scala ad ogni dimensione della matrice identità:

$$\begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{pmatrix}$$

Proprietà delle trasformazioni affini della famiglia (1.1)

Le trasformazioni affini sono lineari rispetto alle combinazioni baricentriche. Infatti:

$$\Phi(P) = \Phi\left(\sum_i \alpha_i P_i\right) = A\left(\sum_i \alpha_i P_i\right) + v\overbrace{\sum_i \alpha_i}^{=1} = \sum_i \alpha_i (AP_i + v) = \sum_i \alpha_i \Phi(P_i)$$

Inoltre, se A è ortogonale[†], una trasformazione della famiglia (1.1) lascia invariati angoli e lunghezze.
Infatti, dati x, y vettori, per le lunghezze:

$$\|\Phi(x) - \Phi(y)\|^2 = \|Ax + v - Ay - v\|^2 = \|A(x-y)\|^2 =_{def} (A(x-y))^T(A(x-y)) = \|x-y\|^2$$

Per gli angoli, se x, y sono tali che $x^T y = |x||y|\cos\theta$ ed, in quanto vettori, possono essere letti come $x = a - b; y = c - d$ (con $a, b, c, d \in \mathbb{R}^d$):

$$\Phi(x)^T\Phi(y) = (\Phi(a) - \Phi(b))^T(\Phi(c) - \Phi(d)) = (Ax)^T Ay = x^T y$$

Rappresentazioni parametriche

Definizione 1.6 (Curva)**.** Si definisce curva l'immagine di un'applicazione $X : I \to \mathbb{E}^d$ continua e localmente iniettiva.

Definizione 1.7 (Rappresentazione Parametrica)**.** L'applicazione $X : I \to \mathbb{E}^d$ continua e localmente iniettiva si dice rappresentazione a parametri in I della curva; in generale una stessa curva può ammettere più di una rappresentazione parametrica.[‡]

Definizione 1.8 (Rappresentazione parametrica differenziabile)**.** La rappresentazione $X(t)$ di dice differenziabile se esiste la funzione:

$$\dot{X}(t) = \frac{dX(t)}{dt}$$

Se $\dot{X} \neq 0$ su tutto l'intervallo di definizione I, X si dice *differenziabile regolare*

[†] $A^T A = AA^T = I$

[‡] Ad esempio, un segmento di retta (curva degenere) tra l'origine e il punto (1,1) può essere rappresentata in forma implicita ($ax + by + c = 0$), esplicita ($y = mx + q$) o parametrica su un'intervallo $I(= [0, 1]$ per esempio):
$X(t) = \begin{cases} x = t \\ y = t \end{cases}$ o equivalentemente $X(t) = \begin{cases} x = 1 - t^k \\ y = 1 - t^k \end{cases}$

Riparametrizzazioni

La non unicità della rappresetazione parametrica suggerisce che si possa passare da una rappresentazione all'altra di una stessa curva, utilizzando di volta in volta la parametrizzazione con le caratteristiche più adatte al contesto.

Definizione 1.9 (Funzioni di riparametrizzazione)**.** Sia $\tau : I \to I_2$. Si definisce *riparametrizzazione di X rispetto a τ* una funzione $\tilde{X}$ tale che $\tilde{X}(\tau(t)) = X(t)$.

Si ha che:

$$\frac{d\tilde{X}}{d\tau} = \frac{dX}{dt}\frac{1}{\frac{d\tau}{dt}} = \frac{dX}{dt}\frac{dt}{d\tau}$$

in cui dX/dt è la tangente alla curva e $dt/d\tau$ è uno scalare > 0.
Se $\tau \in \mathcal{C}^1$ e $\dot{\tau} > 0$ su tutto I la riparametrizzazione si dice *ammissible*; in questo caso la curva è percorsa tutta nello stesso verso e la sua tangente non cambia verso né direzione, ma solo il modulo.

Osservazione 1.1 (Tangente alla curva). La tangente alla curva non può essere usata per caratterizzarla, in quanto potenzialmente cambia modulo al cambiare della rappresentazione.
Per caratterizzare una curva occorre definire il *versore tangente*:

$$e_1(t) = \frac{\dot{X}(t)}{|\dot{X}(t)|}, \quad \dot{X}(t) \neq 0$$

Definizione 1.10 (Rappresentazione parametrica regolare)**.** Se $\dot{X}(t) \neq 0 \ \forall t \in I$ la rappresentazione parametrica si dice *regolare*.

Lunghezza dell'arco

Definizione 1.11 (Lunghezza dell'arco)**.** Si definisce *lunghezza dell'arco* la funzione:

$$S(t) = \int_a^t \left|\frac{dX(z)}{dz}\right| dz$$

La lunghezza dell'arco può essere usata come funzione di riparametrizzazione ammissibile, che ha anche la caratteristica di essere lineare rispetto alle lunghezze sulla curva. In particolare $S : [a, b] \to [0, l]$, e $\frac{dS(t)}{dt} = \left|\frac{dX(t)}{d(t)}\right|$ implica che se $X(t)$ è regolare anche $S(t)$ lo è, e vale anche:

$$\frac{dX}{dS} = \frac{dX}{dt}\frac{dt}{dS} = \frac{dX}{dt}\frac{1}{\left|\frac{dX}{dt}\right|} = 1$$

Osservazione 1.2 (Invarianza rispetto alla rappresentazione). La lunghezza dell'arco è invariante rispetto alla rappresentazione parametrica della curva, infatti, cambiando parametrizzazione:

$$S(z) = \int_a^z \left|\frac{dX}{dy}\right| dy = \int_a^k \left|\frac{dX}{dt}\frac{dt}{dy}\right| \frac{dy}{dt} dt = S(k)$$

Frenet Frame

Definizione 1.12 (Fernet Frame)**.** Il *Frenet Frame* è un sistema di coordinate ortonormali definito localmente in un punto di una curva $X(t)$.

È possibile definire un Frenet frame[†] a partire dalle derivate di una qualsiasi rappresentazione parametrica $X(t)$ di una curva:

$$\begin{cases} e_1(t) = \dfrac{\dot{X}(t)}{|\dot{X}(t)|} & \text{versore tangente} \\ e_2(t) = \dfrac{\frac{de_1(t)}{dt}}{|\frac{de_1(t)}{dt}|} = \dfrac{\dot{X}(t) \times (\ddot{X}(t) \times \dot{X}(t))}{|\dot{X}(t)||\ddot{X}(t) \times \dot{X}(t)|} = \dfrac{\dot{X}(t)(\dot{X}(t) \cdot \ddot{X}(t))}{|\dot{X}(t)(\dot{X}(t) \cdot \ddot{X}(t))|} & \text{versore normale} \\ e_3(t) = e_1(t) \times e_2(t) & \text{versore binormale}^{\ddagger} \end{cases}$$

Curvatura e Torsione

Definizione 1.13 (Curvatura)**.** Si definisce *curvatura* la quantità:

$$K(t) = \frac{|\dot{X}(t) \times \ddot{X}(t)|}{|\dot{X}(t)|^3}$$

La curvatura è una grandezza caratteristica della curva indipendente dalla sua rappresentazione parametrica.
L'espressione della curvatura si semplifica notevolmente quando la rappresentazione è lunghezza dell'arco:

$$K(s) = |\ddot{X}(s)|$$

in quanto $\dot{X}(s) \equiv 1$.

Calcolo della curvatura

```
function K = curvature(dx,ddx)
  K=(sum(crossprod(dx,ddx).^2,'r').^0.5)./(sum(dx.^2,'r').^0.5).^3
endfunction
```

Definizione 1.14 (Torsione)**.** Si definisce *torsione* la quantità:

$$\tau(t) = \frac{\det(\left[\dot{X}(t); \ddot{X}(t); \dddot{X}(t)\right])}{\left|\dot{X}(t) \times \ddot{X}(t)\right|^2}$$

[†]L'insieme di tutti i Frenet frame è rappresentato tramite tre vettori ortonormali e_1, e_2, e_3; in realtà ogniuno di essi è una funzione del parametro della rappresentazione della curva (C'è un Frenet frame diverso per ogni punto della curva).

[‡]Identità "BAC-CAB": $A \times (B \times C) = B(A \cdot C) - C(A \cdot B)$.

La torsione si annulla solo quando il prodotto misto è nullo:

$$(\dot{X}(t) \times \ddot{X}(t)) \cdot \dddot{X}(t) = \det(\left[\dot{X}(t); \ddot{X}(t); \dddot{X}(t)\right]) = 0$$

La torsione è non nulla solo nello spazio e non su un piano.

Calcolo della torsione

```
function T = torsione(dx,ddx,dddx)
  [d,n] = size(dx)
  den = (sum(crossprod(dx,ddx).^2,'r'));
  T = [];
  for i=1:n
    T = [T,det([dx(:,i),ddx(:,i),dddx(:,i)])/den(i)];
    end;
endfunction
```

Capitolo 2

Curve parametriche

Polinomi di Bernstein

Definizione 2.1 (Base di Bernstein)**.** I *polinomi di Bersntein* sono una famiglia di polinomi definiti, per ogni $n \in \mathbb{N}^+$, come:

$$\begin{cases} B_0^0(t)^\dagger = 1 \\ B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \in \Pi_n,\ i \leq n \\ B_i^n(t) = 0,\ i > n,\ i < 0 \ ^\ddagger \end{cases}$$

Definizione 2.2 (Polinomi di Bernstein in forma ricorsiva)**.**

$$\begin{aligned} B_0^0 &= 1 \\ B_j^n &= 0, \quad j \neq 0, 1, ..., n \\ B_i^n &= (1-t)B_i^{n-1} + tB_{i-1}^{n-1} \end{aligned}$$

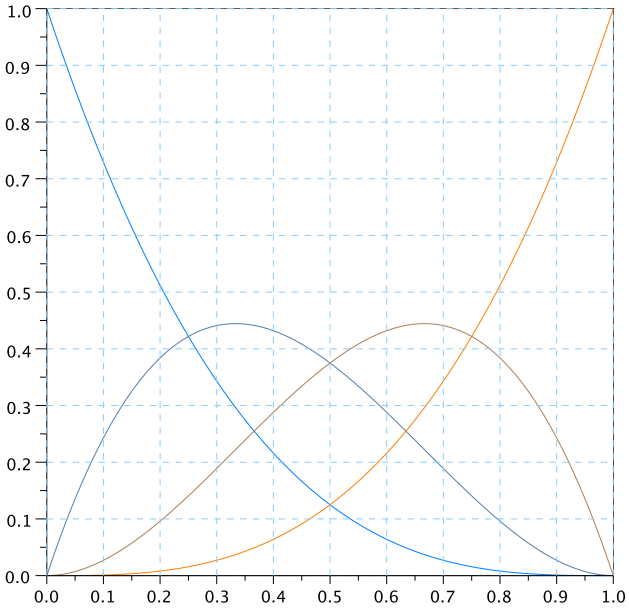
Le due definizioni sono equivalenti, infatti:

Caso ($i = 0$ e $i = n$)*.*

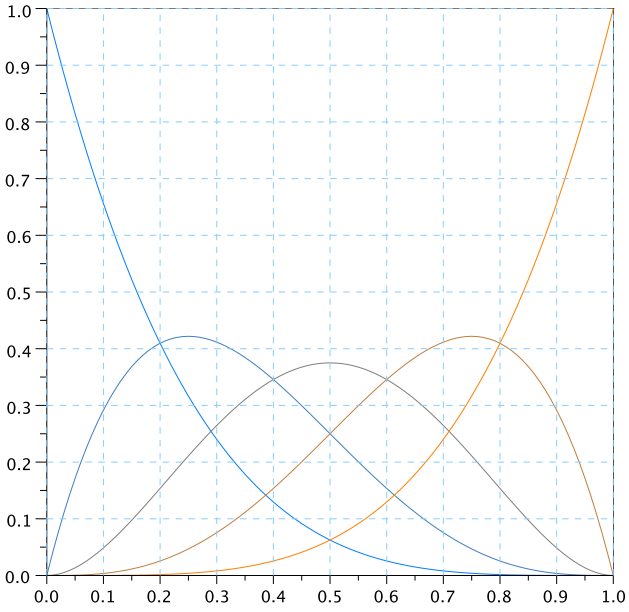
$$B_0^n(t) = (1-t)^n, \quad B_n^n = t^n$$

†Per brevità talvolta si omette di specificare il parametro (t)

‡Successivamente utile ai fini della rappresentazione e del calcolo



(a) Terzo grado



(b) Quarto grado

Figura 2.1: Basi di Bernstein $B_i^n(t)$

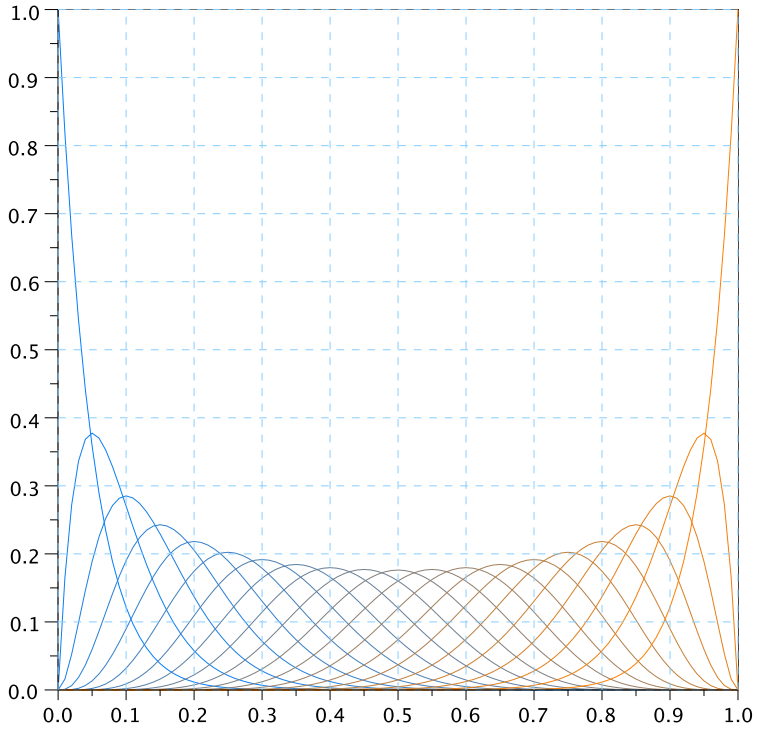


Figura 2.2: Basi di Bernstein di ventesimo grado

Caso $(0 < i < n)$. in quanto $\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$:

$$\begin{aligned} B_i^n(t) &= \binom{n}{i} t^i (1-t)^{n-i} = \left(\binom{n-1}{i} + \binom{n-1}{i-1} \right) t^i (1-t)^{n-i} \\ &= (1-t)\binom{n-1}{i} t^i (1-t)^{n-i-1} + t\binom{n-1}{i-1} t^{i-1} (1-t)^{n-i} \\ &= (1-t) B_i^{n-1} + t B_{i-1}^{n-1} \end{aligned}$$

Definizione 2.3 (Polinomi di Berstein di grado n)**.** Si definisce *Polinomio di Bernstein di grado* n la serie:

$$p(t) = \sum_{k=0}^{n} c_k B_k^n(t)$$

Proprietà

I polinomi di Bernstein sono definiti su tutto $\mathbb{R}$, ma ci si limita a studiarne le proprietà nell'intervallo $[0, 1]$ (sufficiente in questo contesto).

Osservazione 2.1 (Mappabilità della base). I polinomi di Bernstein possono essere definiti, in maniera equivalente, su qualsiasi intervallo $[a, b]$ mappabile a $[0, 1]$:

$$B_i^n(t) = \binom{n}{i} \frac{(t-1)^i (b-t)^{n-i}}{(b-a)^n}, \quad t \in [a, b]$$

Non negatività

Ogni polinomio della base è non negativo in quanto prodotto di componenti non negative ($\binom{n}{i}, t^i, (1-t)^{n-i}$).

Partizionamento dell'unità

$$\sum_{i=0}^{n} B_i^n(t) = 1$$

infatti, $\forall\, t \in \mathbb{R}$:

$$1 = (t + (1-t))^n =_{def} \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \equiv \sum_{i=0}^{n} B_i^n(t)$$

Simmetria

$$B_i^n(t) = B_{n-i}^n(1-t)$$

infatti:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} = \binom{n}{n-i} (1-t)^{n-i} t^i =^{\dagger} B_{n-i}^n(1-t)$$

Indipendenza lineare

I polinomi di Bernstein sono una base per Π_n; infatti, B_i^n può essere scritto come:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} = t^i \binom{n}{i} \sum_{j=0}^{n-i} \binom{n-i}{j} [1^j] - 1^{n-i-j} t^{n-i-j}$$

[†] Per il binomio di Newton vale la proprietà $\binom{n}{i} = \binom{n}{n-i}$

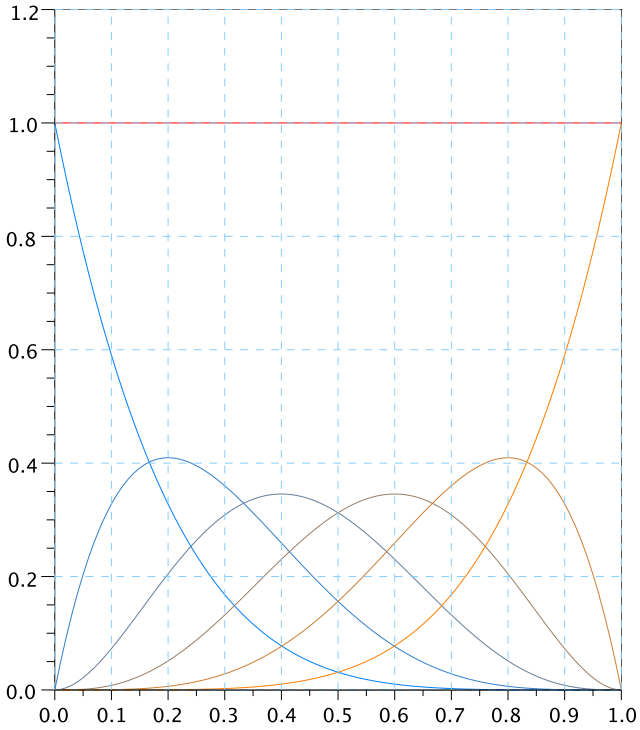


Figura 2.3: Partizionamento dell'unità, la somma dei polinomi è evidenziata in rosso

da cui:

$$\begin{pmatrix} B_0^n \\ \vdots \\ B_n^n \end{pmatrix} = \begin{pmatrix} 1 & \dots & \dots & \dots & 1 \\ 0 & \ddots & & & \vdots \\ \vdots & & \binom{n}{i} & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & & & & 1 \end{pmatrix} \begin{pmatrix} t^0 \\ \vdots \\ t^n \end{pmatrix}$$

Se la matrice del cambio di base è non singolare i polinomi di Bernstein sono linearmente indipendenti; la matrice è non singolare in quanto triangolare superiore con tutti elementi non nulli sulla diagonale (determinante non nullo).

Precisione lineare

$$\sum_{i=0}^{n} \frac{i}{n} B_i^n(t) = t$$

Infatti, per induzione su n:

$$B_1^1(t) = t$$

e supponendo che valga per $n-1$:

$$\begin{aligned}
\sum_{i=0}^{n} \tfrac{i}{n} B_i^n(t) &= \sum_{i=0}^{n} \frac{i}{n}\left[(1-t)B_i^{n-1} + tB_{i-1}^{n-1}\right] \\
&= \frac{n-1}{n}(1-t)\sum_{i=0}^{n-1}\frac{i}{n-1}B_i^{n-1}(t) + t\frac{n-1}{n}\sum_{i=1}^{n}\frac{i}{n-1}B_{i-1}^{n-1}(t) \\
&= \frac{n-1}{n}(1-t)t + \frac{n-1}{n}t\sum_{j=0}^{n-1}\frac{j+1}{n-1}B_j^{n-1}(t) = \\
&= \frac{n-1}{n}(1-t)t + \frac{n-1}{n}t\left[\sum_{j=0}^{n-1}\frac{j}{n-1}B_j^{n-1}(t) + \frac{1}{n-1}\sum_{j=0}^{n-1}\frac{j+1}{n-1}B_j^{n-1}(t)\right] \\
&= \frac{n-1}{n}(1-t)t + \frac{n-1}{n}t\left[t + \frac{1}{n-1}\right] = \\
&= \frac{n-1}{n}(1-t)t + \frac{n-1}{n}t^2 + \frac{t}{n} = t
\end{aligned}$$

Variation diminishing

Un polinomio di Bernstein $p(t) = \sum_{k=0}^{n} c_k B_k^n(t)$ a coefficienti $c_0, .., c_n$, dato V il numero di variazioni di segno della sequenza di costanti e N numero di radici reali di $p(t)$ contate con la loro molteplicità, gode della proprietà:

$$N \leq V, \quad \text{e} \quad N = V - 2k, \quad t \in [0,1].$$

Infatti,

$$p(t) = \sum_{k=0}^{n} c_k \binom{n}{k} t^k (1-t)^{n-k}$$

parametrizzato su $u \in [0, \infty]$, ovvero considerando $t = \frac{u}{1+u}$ e $(1-t) = \frac{1}{1+u}$, diventa:

$$p(t(u)) = \sum_{k=0}^{n} c_k \binom{n}{k} (\tfrac{u}{1+u})^k (\tfrac{1}{1+u})^{n-k} = \sum_{k=0}^{n} c_k \binom{n}{k} \frac{u^k}{(1+u)^n} = \frac{1}{(1+u)^n}\sum_{k=0}^{n} c_k \binom{n}{k} u^k$$

che è un polinomio sulla base canonica per il quale vale la regola di Cartesio[†].

[†]Il numero di radici positive (contato con molteplicità) è dato dal numero di cambi di segno (variazioni) fra due coefficienti consecutivi.

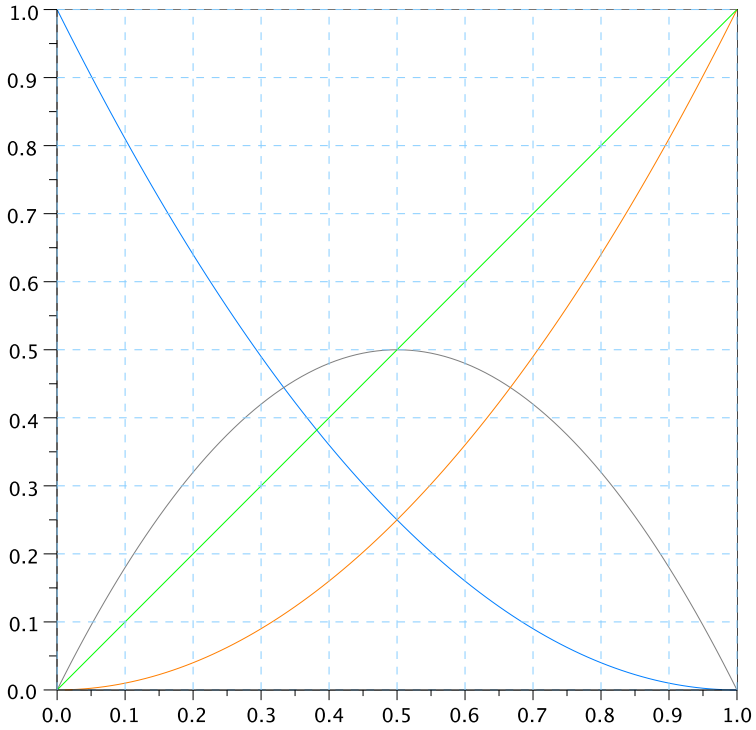


Figura 2.4: Precisione lineare: in verde $\sum_{i=0}^{n} \frac{i}{n} B_i^n(t)$.

Valori agli estremi

$$B_i^n(0) = \begin{cases} 1, & i = 0 \\ 0, & i \neq 0 \end{cases}$$

$$B_i^n(1) = \begin{cases} 1, & i = n \\ 0, & i \neq n \end{cases}$$

Derivata

La derivata dei polinomi di Bernstein può essere espressa in termini della base di Bernstein stessa:

$$\begin{aligned}\frac{dB_i^n}{dt} &= \frac{d}{dt}\binom{n}{i}t^i(1-t)^{n-i} = \\ &= \frac{in!}{i!(n-i)!}t^{i-1}(1-t)^{n-1} - \frac{(n-i)n!}{i!(n-i)!}t^i(1-t)^{n-1-i} \\ &= n\frac{(n-1)!}{(i-1)!(n-i)!}t^{i-1}(1-t)^{n-i} - n\frac{(n-i)!}{i!(n-i-1)!}t^i(1-t)^{n-i-1} \\ &= n\left(\binom{n-1}{i-1}t^{i-1}(1-t)^{n-1} - \binom{n-1}{i}t^i(1-t)^{n-i-1}\right) \\ &= n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right)\end{aligned}$$

La stessa derivata però può anche essere rappresentata in modo da evidenziare il comportamento della funzione:

$$\begin{aligned}\frac{dB_i^n}{dt} &= \frac{d}{dt}\binom{n}{i}t^i(1-t)^{n-i} \\ &= \binom{n}{i}t^{i-1}(1-t)^{n-i-1}[i(1-t) - t(n-i)] \\ &= \binom{n}{i}t^{i-1}(1-t)^{n-i-1}[i-nt]\end{aligned}$$

da cui si nota che i punti non banali in cui la derivata si annulla sono facilmente determinabili: $\hat{t}_i = \frac{i}{n}$, con $i = 0, \ldots, n$

Curve di Bézier

Definizione 2.4 (Curva di Bézier)**.** Dati $b_0, ..., b_n \in \mathbb{E}^d$ insieme di punti di controllo, si definisce *curva di Bézier* il polinomio:

$$X(t) = \sum_{i=0}^{n} b_i B_i^n(t), \quad t \in [0,1]$$

Definizione 2.5 (Poligono di controllo)**.** Si definisce *poligono di controllo* il poligono che si ottiene unendo ordinatamente i punti di controllo.

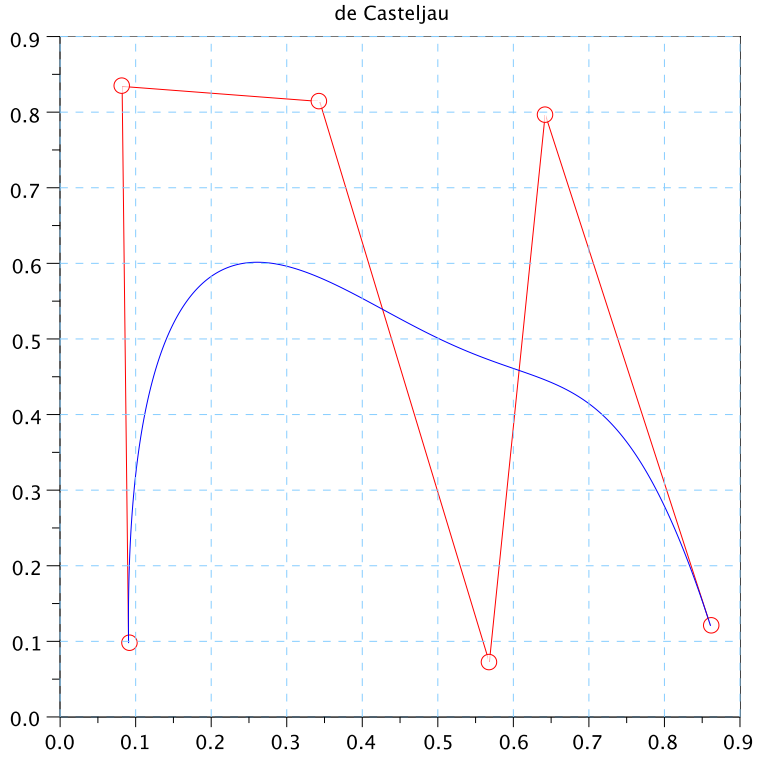


Figura 2.5: Esempio di curve di Bézier; in rosso il poligono di controllo

Proprietà

Convex Hull

La curva giace completamente nella convex hull dei punti di controllo, in quanto ogni punto della curva è una combinazione baricentrica di questi[†].

Interpolazione agli estremi

Segue direttamente dai valori agli estremi dei polinomi di Bernstein che $X(0) \equiv b_0$ e $X(1) \equiv b_n$.

Invarianza alle trasformazioni affini

Data $\Phi : \mathbb{E}^d \to \mathbb{E}^d$ di tipo $\Phi(p) = Ap + v$, si ha:

$$\Phi(X(t)) = AX(t) + v = \sum_{i=0}^{n} Ab_i B_i^n(t) + \sum_{i=0}^{n} vB_i^n(t) = \sum_{i=0}^{n}(Ab_i + v)B_i^n(t) = \sum_{i=0}^{n} \Phi(b_i)B_i^n(t)$$

[†]I polinomi di Bernstein partizionano l'unità

Simmetria

Le curve di Bézier sono simmetriche rispetto all'ordine dei punti di controllo, infatti, dati $c_i = b_{n-i}$:

$$\tilde{X}(t) = \sum_{i=0}^{n} c_i B_i^n(t) = \sum_{i=0}^{n} c_i B_{n-i}^n(1-t) = \sum_{i=0}^{n} b_i B_i^n(1-t) = X(1-t)$$

Precisione lineare

Se i punti di controllo sono allineati ed equidistanti la convex hull è il segmento $\overline{b_0 b_n}$ e la curva giace su una retta:

$$X(t) = \sum_{i=0}^{n}(b_0 + \tfrac{i}{n}(b_n - b_0))B_i^n(t) = b_0 \sum_{i=0}^{n} B_i^n(t) + (b_n - b_0)\sum_{i=0}^{n} \tfrac{i}{n} B_i^n(t) = {}^{\dagger}\, b_0 + t(b_n - b_0)$$

Controllo pseudo-locale

Modificando uno dei punti di controllo $b_0, ..., b_n$:

$$c_i = \begin{cases} b_i, & i \neq j \\ b_i + v & i = j \end{cases}$$

si ottiene:

$$\tilde{X}(t) = \sum_{i=0}^{n} c_i B_i^n(t) = \sum_{i=0}^{n} b_i B_i^n(t) + v B_j^n(t)$$

ovvero tutti i punti della curva si spostano nella direzione di v con un'incidenza che dipende da t e dalla forma del $j-$esimo polinomio di Bernstein. In generale, in quanto i polinomi hanno una forma a campana, i punti della curva sono interessati in maniera proporzionale alla distanza dal punto di controllo modificato.

Variation diminishing

La proprietà di variation diminishing aiuta a capire l'andamento della curva; si può infatti sapere il numero massimo di intersezione che la curva può avere con una retta (N) a partire da quello del poligono di controllo (N_p). In particolare, $N \leq N_p$ e $N = N_p - 2k$.
Sia $r = \alpha x + \beta y + \gamma$, le intersezioni si possono ricavare, separando le dimensioni, con l'equazione:

$$\alpha \sum_{i=0}^{n} b_{ix} B_i^n(t) + \beta \sum_{i=0}^{n} b_{iy} B_i^n(t) + \gamma \sum_{i=0}^{n} B_i^n(t) = 0$$

da cui:

$$\sum_{i=0}^{n} \underbrace{(\alpha b_{ix} + \beta b_{iy} + \gamma)}_{c_i} B_i^n(t) = 0$$

e la proprietà segue direttamente da quella di variation diminishing dei polinomi di Bernstein.

[†] Precisione lineare dei polinomi di Bernstein

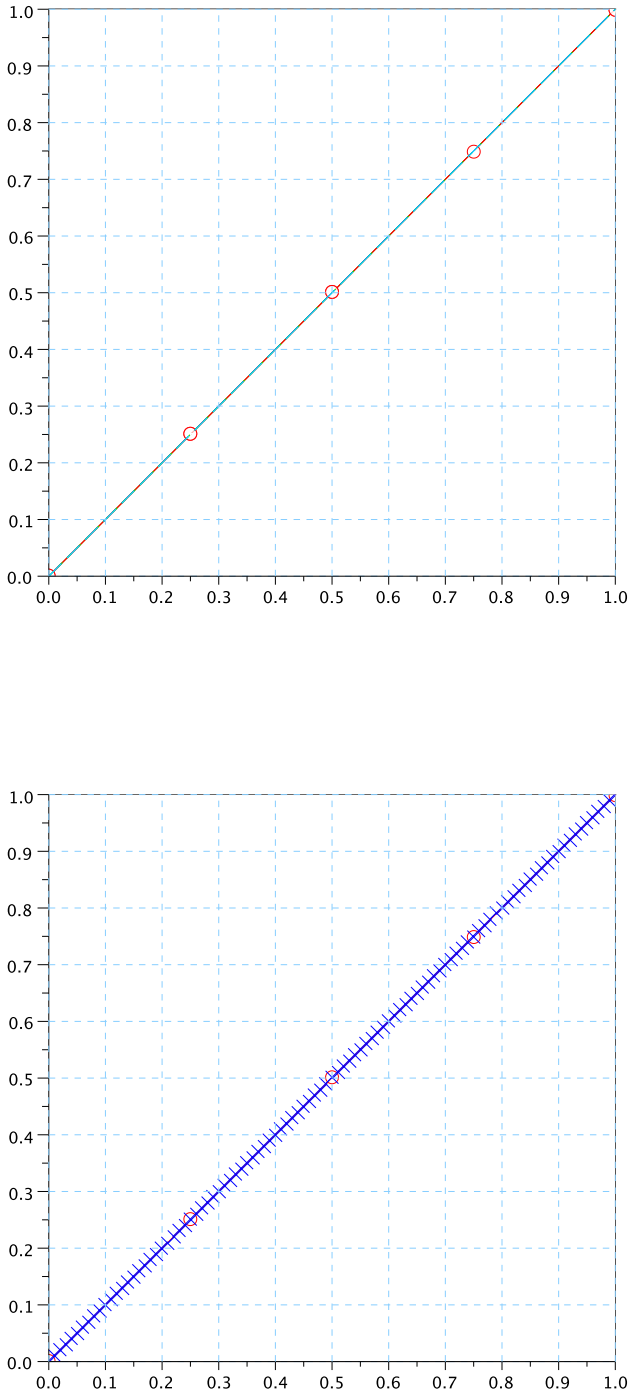


Figura 2.6: Precisione lineare: poligono di controllo e curva che giace su di esso; si nota che i punti della curva sono distribuiti uniformemente rispetto al parametro t.

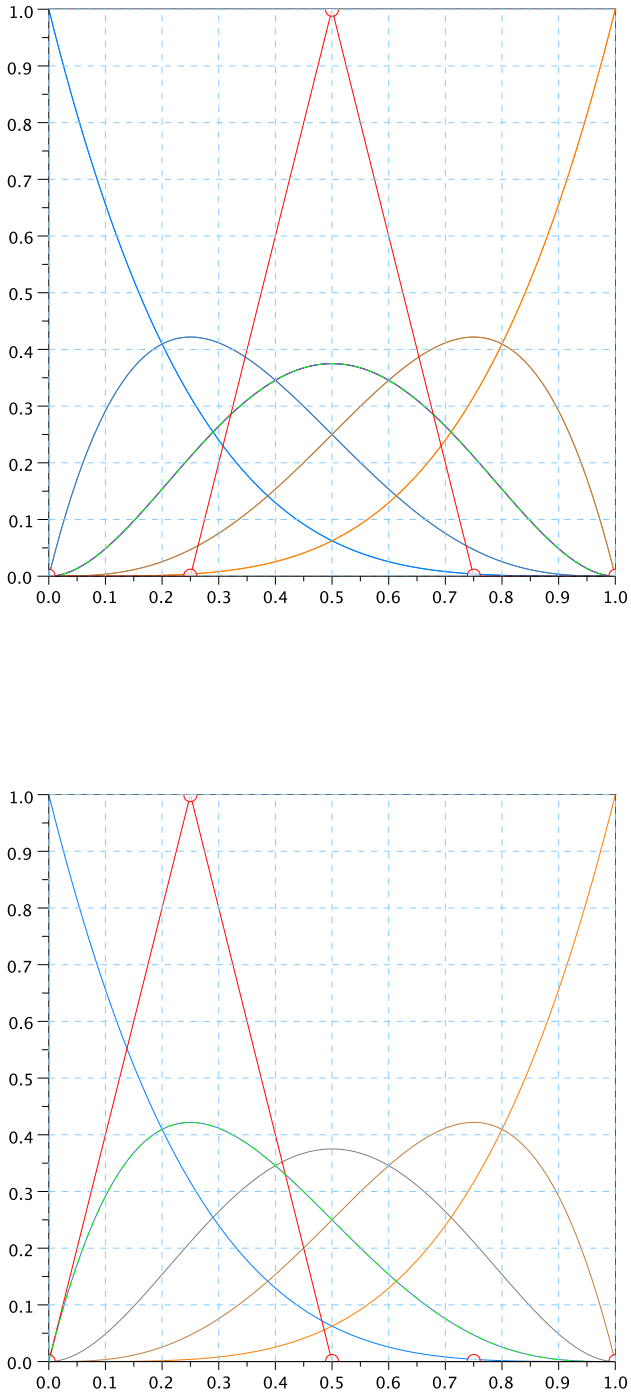


Figura 2.7: Controllo pseudolocale: la curva (verde) può essere fatta coincidere con uno qualsiasi dei polinomi della base

Derivata (o curva Hodograph)

$$\begin{aligned}\dot{X}(t) &= \sum_{i=0}^{n} b_i B_i^n(t) = n \sum_{i=0}^{n} b_i \left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t) \right) \\ &= n \left[(b_1 - b_0) B_0^{n-1}(t) + ... + (b_n - b_{n-1}) B_{n-1}^{n-1} \right] \\ &= n \sum_{i=0}^{n-1} (b_{i+1} - b_i) B_i^{n-1}(t) = n \sum_{i=0}^{n-1} \Delta b_i B_i^{n-1}(t)\end{aligned}$$

dove $\Delta b_i =_{def} (b_{i+1} - b_i)$.
La derivata di una curva di Bézier è quindi ancora una curva dello stesso tipo ma di grado minore e con punti di controllo differenti.

Calcolo della hodograph

```
function bh=Bezier_hodograph(b)
[d,n] = size(b)
for i=1:d
        bh(i,:) = diff(b(i,:));
end
bh=bh*n;
endfunction
```

Derivate successive

Dalla forma della derivata seconda:

$$\ddot{X}(t) = n(n-1) \sum_{i=0}^{n-2} \Delta^2 b_i B_i^{n-2}(t)$$

si può generalizzare[†]:

$$X^{(r)}(t) = n^{\underline{r}} \sum_{i=0}^{n-r} \Delta^r b_i B_i^{n-r}(t)$$

Tangenza agli estremi

In quanto:

$$\dot{X}(0) = n\Delta b_0, \quad \dot{X}(1) = n\Delta b_{n-1}$$

la curva di Bézier è sempre tangente ai segmenti che uniscono i primi e gli ultimi due punti del poligono di controllo.

[†]Falling factorial: $x^{\underline{k}} = x(x-1)...(x-k+1)$, ovvero i primi k termini del fattoriale. Nota: $x^{\underline{k}} = \frac{x!}{(x-k)!}$

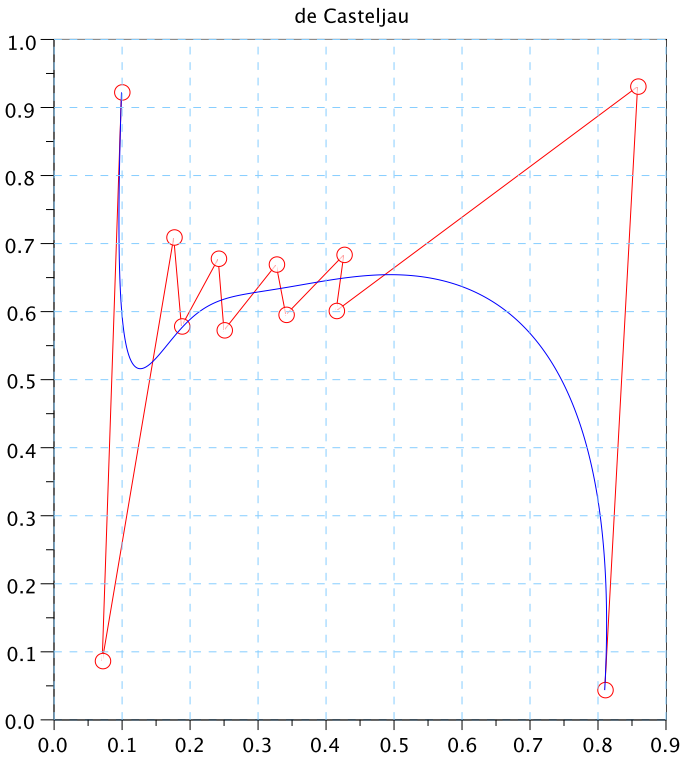


Figura 2.8: Tangenza agli estremi

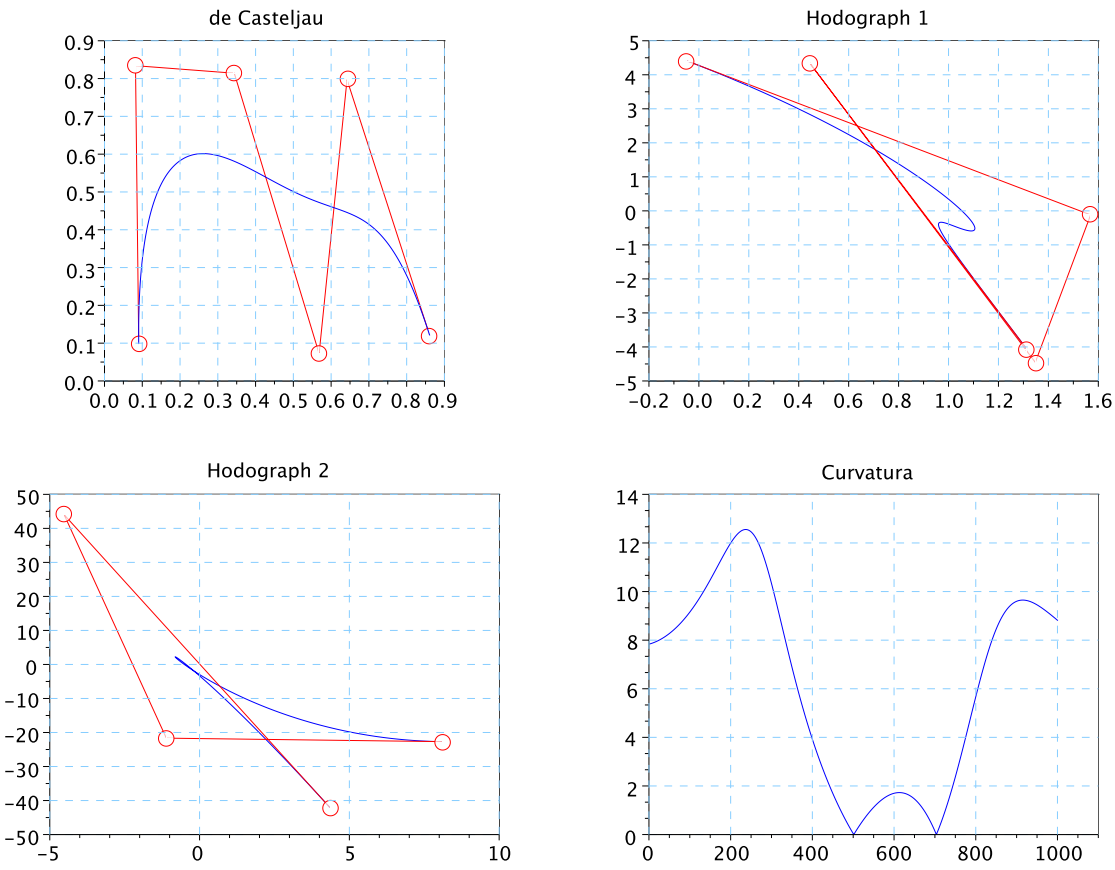


Figura 2.9: Esempio di hodograph prima e seconda e curvatura

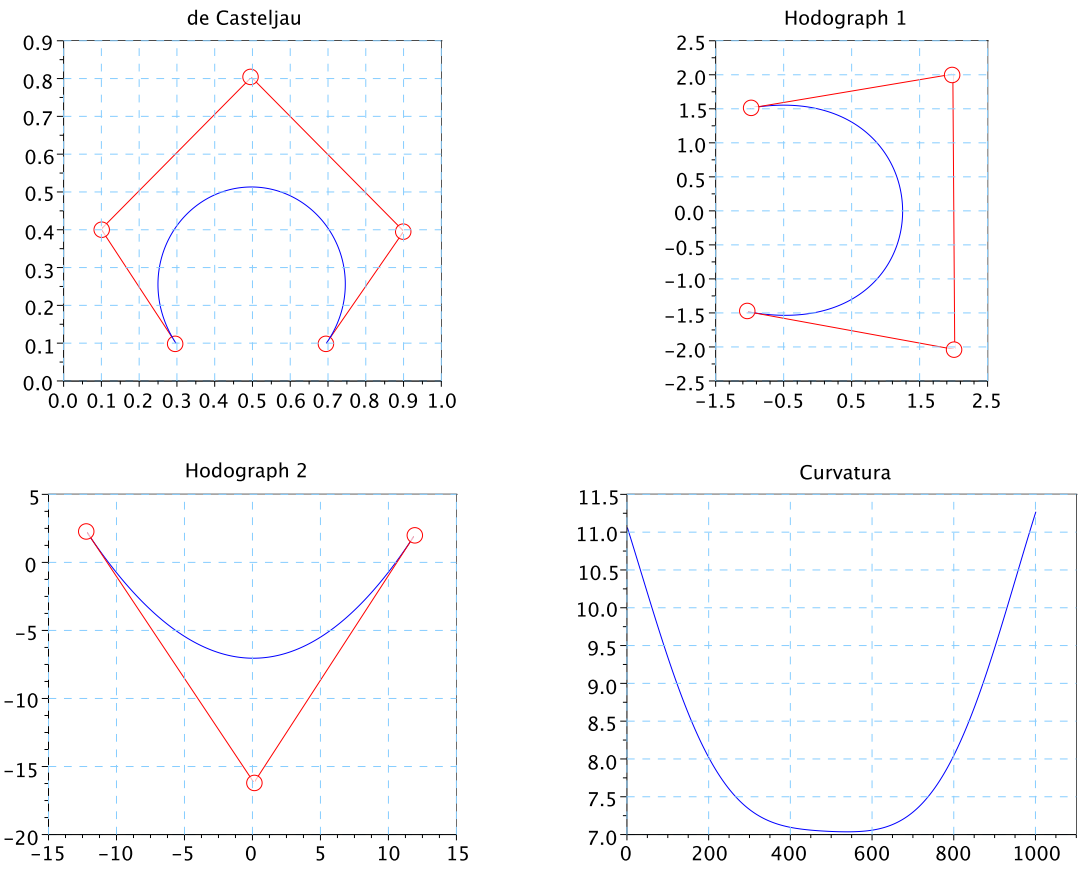


Figura 2.10: Altro esempio di hodograph prima e seconda e curvatura

Algoritmi per curve di Bézier

Disegno della curva: de Casteljau

I punti della curva possono essere calcolati ricorsivamente definendo:

$$\begin{cases} b_i^0 = b_i, & i = 0, ..., n \\ b_i^r = (1-t)b_i^{r-1} + tb_{i+1}^{r-1}, & i = 0, ..., n-r;\ r = 1, ..., n \end{cases}$$

Si dimostra per induzione che questa formula ricorsiva, in r passi, calcola i punti della curva. Per $r = 1$:

$$b_i^1 = (1-t)b_i + tb_{i+1} = \sum_{j=0}^{1} b_{i+j} B_j^1(t)$$

Supponendo che valga per $r-1$, si ha:

$$\begin{aligned} b_i^r(t) &= (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t) \\ &= (1-t)\sum_{j=0}^{r-1} b_{i+j} B_j^{r-1}(t) + t\sum_{j=0}^{r-1} b_{i+j+1} B_j^{r-1}(t) \\ &= (1-t)\sum_{k=i}^{i+r-1} b_k B_{k-i}^{r-1}(t) + t\sum_{k=i+1}^{i+r} b_k B_{k-i-1}^{r-1}(t) \\ &= \sum_{k=i}^{i+r} \left[(1-t)b_k B_{k-i}^{r-1}(t) + tb_k B_{k-i-1}^{r-1}(t)\right] \\ &= \sum_{k=i}^{i+r} b_k B_{k-i}^r(t) = \sum_{j=0}^{r} b_{j+i} B_j^r(t) \end{aligned}$$

ovvero variando t si possono ottenere tutti i punti della curva.

Algoritmo di de Casteljau

```
function Curve = Bezier_casteljau(b,t) //b poligono, t punti di valutazione
lt = length(t)
[d,n] = size(b)
f = t
e = 1-t
b = b'

for i=1:d
    C(:,:,i) = b(1:n-1,i)*e+b(2:n,i)*f
end
for r=2:n-1
    for row=1:n-r
        for i=1:d
            C(row,:,i) = C(row,:,i).*e+C(row+1,:,i).*f
        end
    end
end
for i=1:d
    Curve(i,:)=C(1,:,i)
end
endfunction
```

Degree elevation

Una curva di Bézeier è rappresentabile con un'altra di grado più elevato, ovvero esiste soluzione all'equazione:

$$X(t) = \sum_{i=0}^{n} b_i B_i^n(t) = \sum_{i=0}^{n+1} c_i B_i^{n+1}(t)$$

Per la proprietà di interpolazione agli estremi si deve avere $c_0 = b_0$ e $c_{n+1} = b_n$.
Moltiplicando a sinistra per $(t + (1-t))$ e usando la proprietà di partizionamento dell'unità dei polinomi di Bernstein, si ha:

$$(t + (1-t)) \sum_{i=0}^{n} b_i B_i^n(t) = \sum_{i=0}^{n} b_i \binom{n}{i} \left[t^i (1-t)^{n-i+1} + t^{i+1} (1-t)^{n-i} \right] = \sum_{i=0}^{n+1} c_i B_i^{n+1}(t)$$

da cui, considerando $b_{-1} = b_{n+1} = 0$:

$$\begin{aligned}
&\sum_{i=0}^{n} b_i \binom{n}{i} \left[t^{i+1} (1-t)^{n-i} + t^i (1-t)^{n-i+1} \right] \\
&= \sum_{i=0}^{n} b_i \binom{n}{i} t^{i+1} (1-t)^{n-i} + \sum_{i=0}^{n} b_i \binom{n}{i} t^i (1-t)^{n-i+1} \\
&= \sum_{i=0}^{n+1} b_{i-1} \binom{n}{i-1} t^i (1-t)^{n-i+1} + \sum_{i=0}^{n+1} b_i \binom{n}{i} t^i (1-t)^{n-i+1} \\
&= \sum_{i=0}^{n+1} \frac{\left[\binom{n}{i-1} b_{i-1} + \binom{n}{i} b_i \right]}{\binom{n+1}{i}} B_i^{n+1}(t)
\end{aligned}$$

Si sono trovati i punti di controllo c_i:

$$c_i = \frac{\left[\binom{n}{i-1} b_{i-1} + \binom{n}{i} b_i \right]}{\binom{n+1}{i}} = \frac{i}{n+1} b_{i-1} + \left(1 - \frac{i}{n+1} \right) b_i$$

Osservazione 2.2. I punti di controllo c_i sono ottenuti come combinazione lineare dei due vertici adiacenti del poligono di controllo di partenza, quindi il nuovo poligono è inscritto nel vecchio.

Degree elevation

```
function B = Bezier_degelev(b)
[d,n] = size(b)
B(:,1) = b(:,1)
B(:,n+1) = b(:,n)
for i=2:n
      k = (i-1)/(n);
      B(:,i) = k*b(:,i-1) + (1-k)*b(:,i);
end
endfunction
```

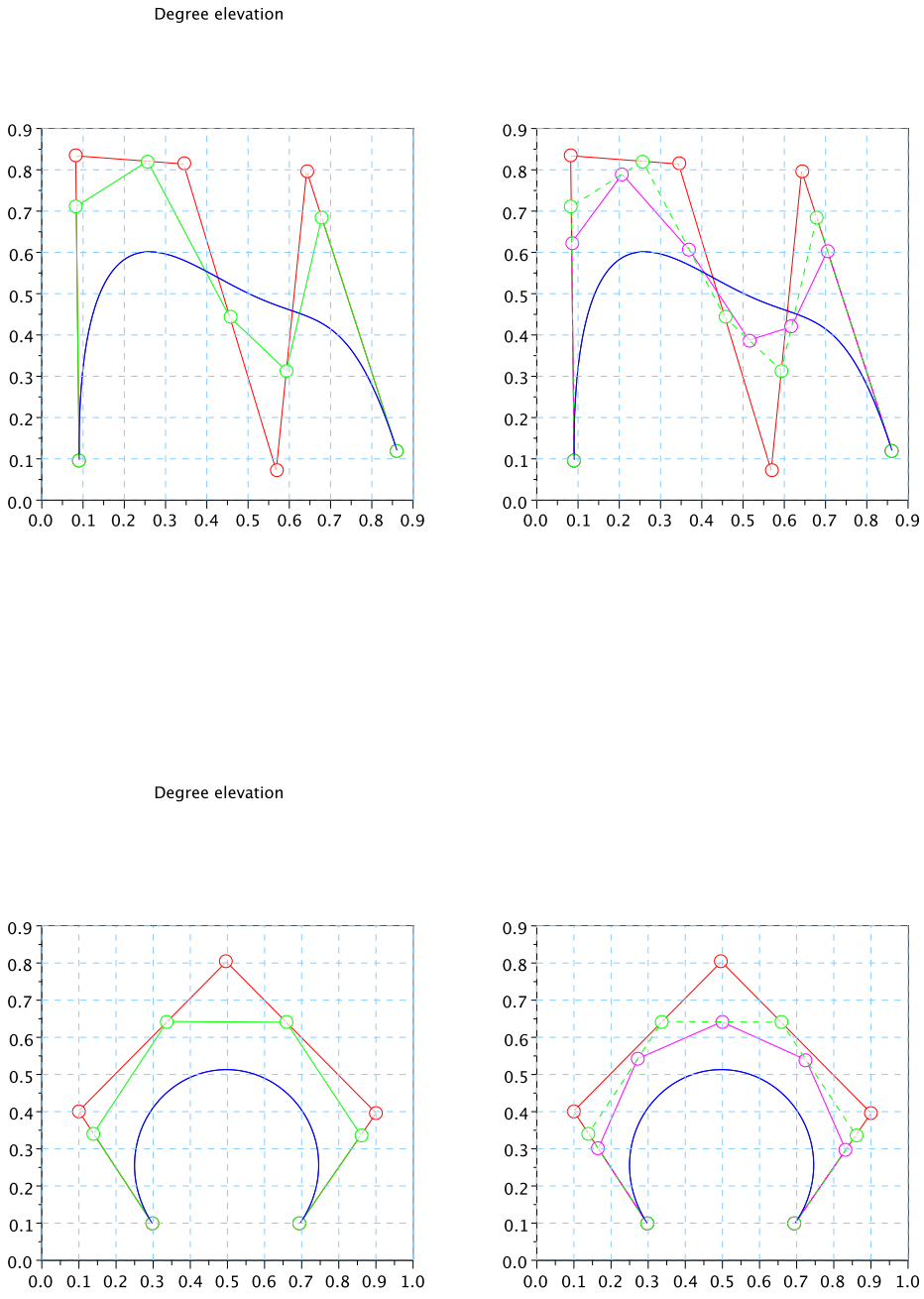


Figura 2.11: Esempi di elevazione del grado

Osservazione 2.3 (Repeated degree elevation). Per disegnare una curva si può pensare di innalzare il grado del poligono di controllo finnchè questo non sia sufficientemente vicino alla curva; i coefficienti dell'r-esima iterazione sono:

$$c_i^{(r)} = \sum_{j=0}^{n} b_i \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}}$$

Fissato t, se $r \to \infty$ il poligono tende effettivamente alla curva, ma il costo computazionale è molto più elevato dell'algoritmo di de Casteljau in quanto servono molte elevazioni per raggiungere una precisione paragonabile.

Suddivisione

Una curva può essere suddivisa in due curve dello stesso grado giunte in un punto.
Dati $\hat{t} \in]0,1[$ e:

$$X(t) = \sum_{i=0}^{n} b_i B_i^n(t)$$

si cercano:

$$\begin{cases} C_L(\tau) = \sum_{k=0}^{n} c_k B_k^n(\tau) & = X(t),\ t \in [0,\hat{t}],\ \tau = \dfrac{t}{\hat{t}} \in [0,1] \\ C_R(u) = \sum_{k=0}^{n} d_k B_k^n(\tau) & = X(t),\ t \in [\hat{t},1],\ u = \dfrac{t-\hat{t}}{1-t} \in [0,1] \end{cases}$$

Definizione 2.6 (Operatori identità e shift)**.**
Si definiscono gli operatori I, E tali che $Ib_i = b_i$, $Eb_i = b_{i+1}$.

Osservazione 2.4 (Rappresentazione alternativa delle curve di Bézier). I punti della curva di Bézier (algoritmo di de Casteljau) si possono rappresentare a partire dagli operatori shift e identità: $b_i^r = \left[(1-t)I + tE\right]^{r-1} b_i$.

I coefficienti c_k e d_k si ottengono dall'algoritmo di de Casteljau come:

$$c_k = b_0^k(\hat{t}), \quad d_k = b_k^{n-k}(\hat{t})$$

infatti[†]:

$$
\begin{aligned}
C_L(\tau) &= \sum_{k=0}^{n} b_0^k(\hat{t}) \binom{n}{k} \tau^k (1-\tau)^{n-k} \\
&= \sum_{k=0}^{n} \left[(1-\hat{t})I + \hat{t}E\right]^k b_0 \binom{n}{k} \tau^k (1-\tau)^{n-k} \\
&= \sum_{k=0}^{n} \binom{n}{k} \tau^k \left[(1-\hat{t})I + \hat{t}E\right]^k (1-\tau)^{n-k} I^{n-k} b_0 \\
&= \sum_{k=0}^{n} \binom{n}{k} \left[\tau 1 - \tau\hat{t}I + \tau\hat{t}E\right]^k \left[(1-\tau)I\right]^{n-k} b_0 \\
&= \left[\tau I - \tau\hat{t}I + \tau\hat{t}E + I - \tau I\right]^n b_0 \\
&= \left[(1-\hat{t})I + \hat{t}E\right]^n b_0
\end{aligned}
$$

Subdivision

```
function [b1,b2] = Bezier_subdiv(b,t)
[d,n] = size(b)
f = t
e = 1-t
b1(:,1) = b(:,1)
b2(:,n) = b(:,n)
b = b'
for i=1:d
    C(:,:,i) = b(1:n-1,i)*e+b(2:n,i)*f
end
for i=1:d
    b1(i,2) = C(1,1,i)
    b2(i,n-1) = C(n-1,1,i)
end
for r=2:n-1
    for row=1:n-r
        for i=1:d
            C(row,:,i) = C(row,:,i).*e+C(row+1,:,i).*f
        end
    end
for i=1:d
    b1(i,r+1) = C(1,1,i)
    b2(i,n-r) = C(n-r,1,i)
end
end
endfunction
```

[†]Il caso C_R si dimostra in maniera analoga

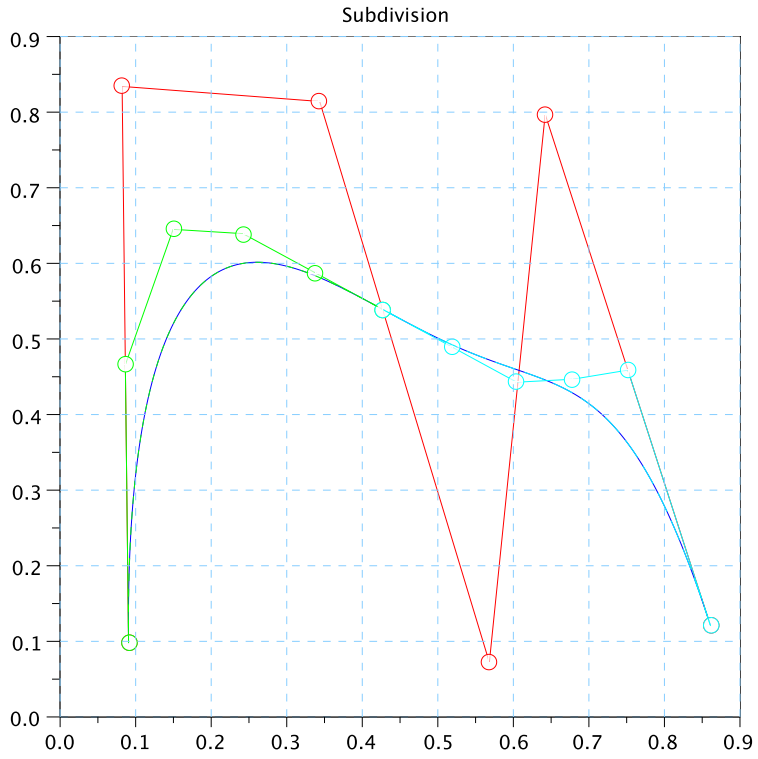


Figura 2.12: Esempio di subdivision

Curve di Bézier razionali

Le curve di Bézieri razionali sono un'estensione alle curve di Bézier che mira a migliorarne la controllabilità.

Coordinate omogenee

Un insieme è rappresentabile mediante coordinate omogenee se è possibile associare un vettore numerico $(x_1, ..., x_2)$ ad ogni elemento dell'insieme, in modo tale che due vettori distinti siano associati a due oggetti diversi solo se linermente indipendenti.

Definizione 2.7 (Spazio proiettivo)**.** Lo spazio proiettivo $P(K)$ associato ad uno spazio vettoriale K è definito come l'insieme dei suoi sottospazi vettoriali di dimensione uno (rette).

Se ogni vettore di uno spazio vettoriale di dimensione finita è rappresentabile tramite le sue coordinate $(x_1, ..., x_n)$, ogni retta è descrivile come lo span lineare di un vettore non nullo: $\lambda(x_1, ..., x_n)$.

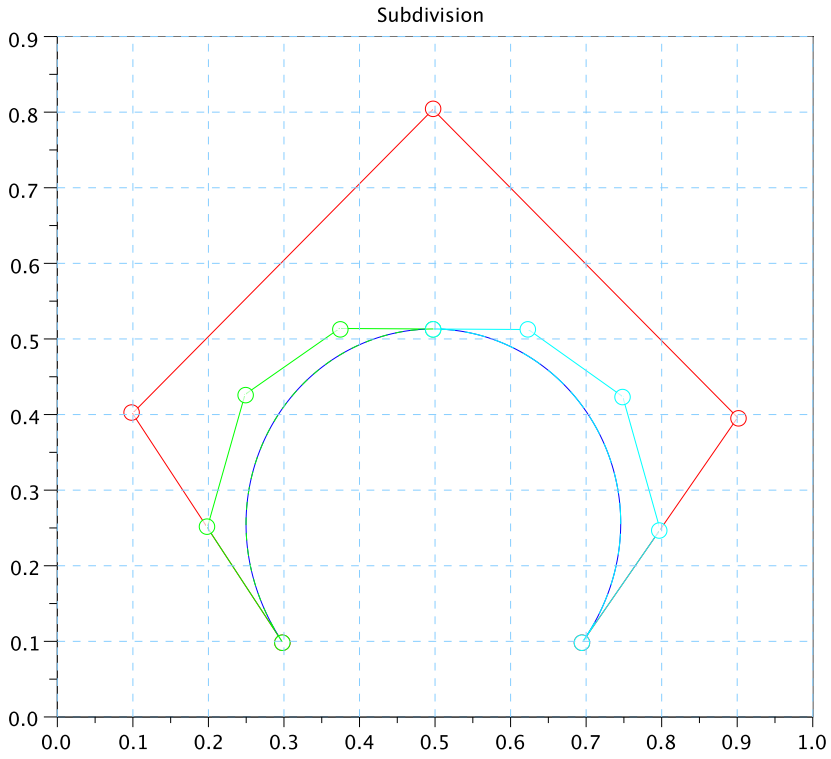


Figura 2.13: Altro esempio di subdivision

Interpretazione geometrica in $d = 3$ Si immagini un piano $\mathbb{E}^2$ immerso in uno spazio $\mathbb{E}^3$ ed allineato col sistema di riferimento, ad esempio perpendicolare all'asse z. I punti di $\mathbb{E}^3$ sono rappresentabili tramite l'insieme delle triple (x, y, z) oppure, a partire dai punti di $\mathbb{E}^2$, come unione dei *punti al finito* ed i *punti all'infinito*:

$$\left\{[x, y, 1] | (x, y) \in \mathbb{E}^2\right\} \cup \left\{[x, y, 0] | (x, y) \in P(\mathbb{E}^2)\right\}$$

Analogamente è possible rappresentare i punti del piano a partire dai punti dello spazio attravero un'operazione di proiezione: $(x, y, z) \to (\frac{x}{z}, \frac{y}{z}, 1)$.

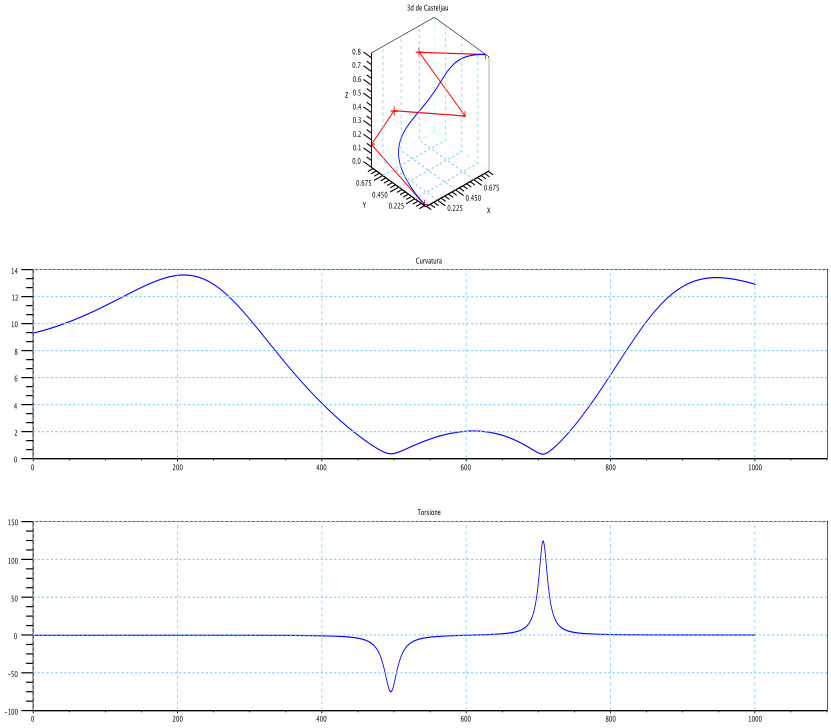


Figura 2.14: Esempio di curva tridimensionale con curvatura e torsione

Curve di Bézier razionali Una curva di Bézier razionale in uno spazio $\mathbb{E}^d$ è la proiezione di una curva di Bézier classica definita in uno spazio $\mathbb{E}^{d+1}$. In particolare si definisce[†]:

$$X(t) = \begin{pmatrix} y_0(t) \\ \vdots \\ y_{d+1}(t) \end{pmatrix} = \sum_{i=0}^{n} \mathbf{B}_i B_i^n(t), \quad \mathbf{B}_i = \beta_i \begin{pmatrix} 1 \\ b_i \end{pmatrix}, \quad b_i \in \mathbb{E}^d$$

in cui i $\mathbf{B}_i$ sono vettori in $\mathbb{E}^{d+1}$. La prima componente della curva $(d+1)$-dimensionale è pertanto:

$$y_0(t) = \sum_{i=0}^{n} \beta_i B_i^n(t)$$

ed applicando direttamente il concetto di proiezione si ottiene la curva proiettiva in $\mathbb{E}^d$:

$$X(t) = \begin{pmatrix} y_0(t) \\ \vdots \\ y_d(t) \end{pmatrix} = \frac{\sum_{i=0}^{n} \beta_i b_i B_i^n(t)}{\sum_{i=0}^{n} \beta_i B_i^n(t)}, \quad \beta_i > 0$$

[†]Per comodità di notazione si intende lo spazio proiettivo perpendicolare alla prima componente del vettore, ovvero $(x, y, z, ..., c_n) \to (1, \frac{x}{c_n}, \frac{y}{c_n}, ...)$

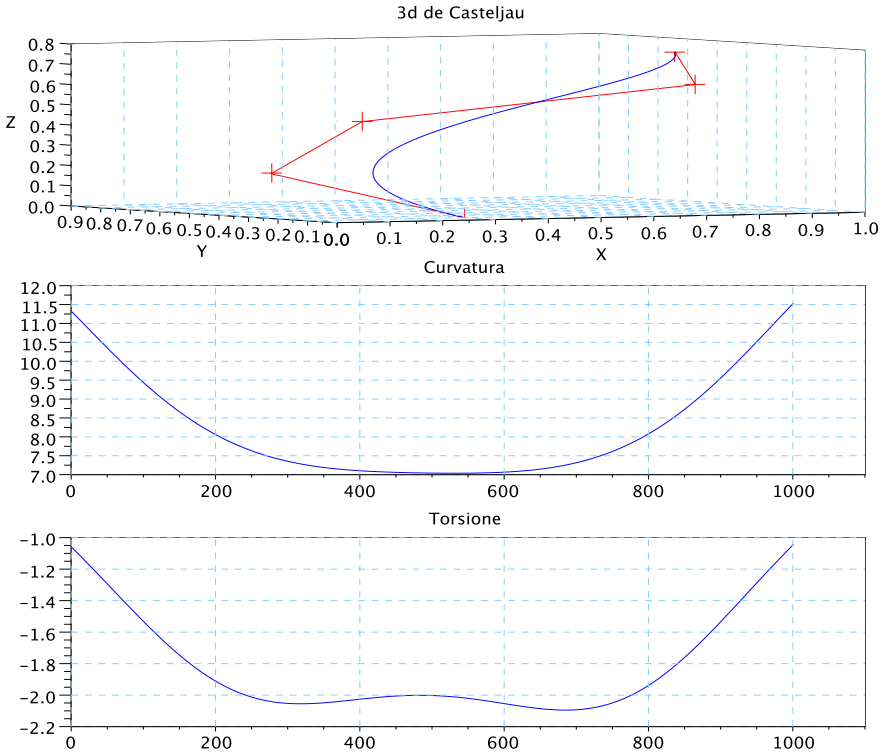


Figura 2.15: Altro sempio di curva tridimensionale con curvatura e torsione

Osservazione 2.5. Se $\beta_0 = \beta_1 = ... = \beta_n$, $X(t) \equiv \sum_{i=0}^{n} b_i B_i^n(t)$ ovvero rimane una Bézier definita dai punti di controllo b_i.

Algoritmo di Casteljau per curve di Bézeir razionali

```
function Curve = RATBezier_casteljau(b,beta,t)
[d,n] = size(b)
n2 = length(beta)
B=[beta]
for i=1:d //Aumenta di una dimensione
      B = [B;beta.*b(i,:)]
end
C = Bezier_casteljau(B,t)
for i=1:d
Curve(i,:) = C(i+1,:)./C(1,:) //Proietta
end
endfunction
```

Proprietà

Simmetria

Sia $s = 1 - t$. La proprietà segue direttamente da quella dei polinomi di Bernstein:

$$X(s) = \frac{\sum_{i=0}^{n} \beta_{n-i} b_{n-i} B_i^n(s)}{\sum_{i=0}^{n} \beta_{n-i} B_i^n(s)} = \frac{\sum_{i=0}^{n} \beta_{n-i} b_{n-i} B_{n-i}^n(t)}{\sum_{i=0}^{n} \beta_{n-i} B_{n-i}^n(t)} = \frac{\sum_{j=0}^{n} \beta_j b_j B_j^n(t)}{\sum_{j=0}^{n} \beta_j B_j^n(t)}$$

Convex Hull

Definendo la curva come:

$$X(t) = \sum_{i=0}^{n} b_i v_i(t), \quad v_i(t) = \frac{\beta_i B_i^n(t)}{w(t)}, \quad w(t) = \sum_{i=0}^{n} \beta_i B_i^n(t)$$

si ha che:

$$\sum_{i=0}^{n} v_i(t) = \frac{1}{w(t)} \underbrace{\sum_{i=0}^{n} \beta_i B_i^n(t)}_{w(t)} = 1$$

ovvero la curva è una combinazione convessa dei punti di controllo ed appartiene alla convex hull di questi.

Invarianza per trasformazioni affini

Deriva direttamente dal fatto che la curva è una combinazione baricentrica di punti; la dimostrazione è analoga a quella per le curve di bézier classiche.

Linear precision

In generale non vale, infatti:

$$X(t) = \sum_{i=0}^{n} (b_0 + \tfrac{i}{n} v) v_i(t) = b_0 + \sum_{i=0}^{n} (\tfrac{i}{n} v_i(t)) v = b_0 + \frac{1}{w(t)} \sum_{i=0}^{n} \tfrac{i}{n} \beta_i B_i^n(t) v$$

da cui:

$$X(t) = b_0 + \frac{t}{w(t)} v$$

Se i β_i sono omogenei però la proprietà si mandiene valida.

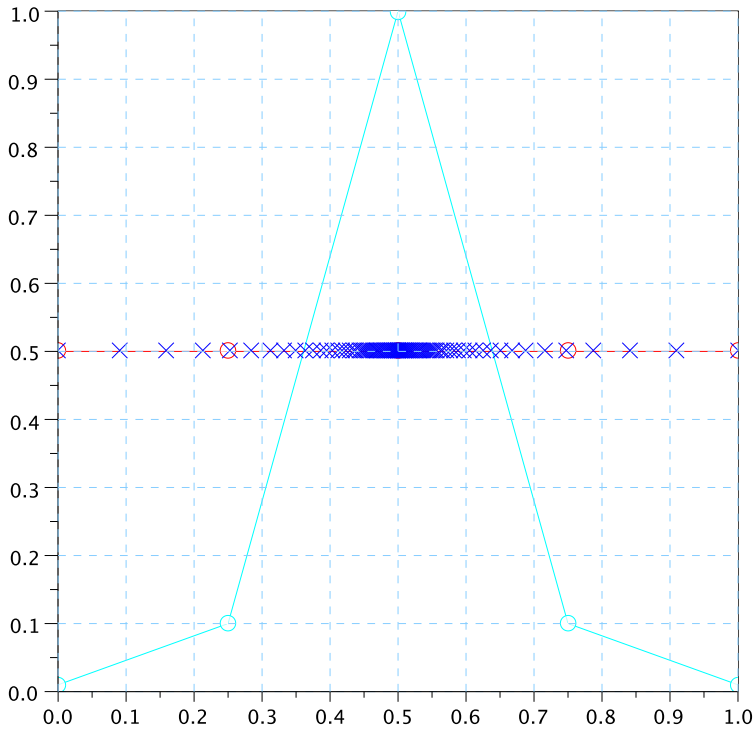


Figura 2.16: Linear precision: la distribuzione dei pesi (ciano) varia la densità dei punti della curva (blu) rispetto al parametro t.

Valori agli estremi

La curva interpola i punti di controllo agli estremi:

$$X(0) = \frac{\beta_0 b_0}{\beta_0} = b_0$$
$$X(1) = \frac{\beta_n b_n}{\beta_n} = b_n$$

Variation diminishing

I punti della curva $X(t) = \sum_{i=0}^{n} b_i v_i(t)$, con $v_i(t) = \frac{\beta_i B_i^n(t)}{w(t)}, w(t) = \sum_{i=0}^{n} \beta_i B_i^n(t)$, che giacciono sulla retta $ax + by + c = 0$ sono:

$$a\sum_{i=0}^{n} b_{ix} v_i(t) + b\sum_{i=0}^{n} b_{iy} v_i(t) + c\sum_{i=0}^{n} v_i(t) = 0$$
$$\sum_{i=0}^{n} (ab_{ix} + bb_{iy} + c) v_i(t) = 0$$
$$\sum_{i=0}^{n} \frac{\beta_i (ab_{ix} + bb_{iy} + c)}{w(t)} B_i^n(t) = 0$$

quindi, ponendo $d_i = \frac{\beta_i (ab_{ix} + bb_{iy} + c)}{w(t)}$ si rientra nel caso della curve di Bézier classica, per cui la proprietà è dimostrata.

Controllo pseudolocale

Modificando uno dei pesi:

$$\hat{\beta}_j = \beta_j + \Delta\beta_j$$

si ottiene la curva:

$$\hat{X}(t) = \frac{\sum_{i=0}^{n} \beta_i b_i B_i^n(t) + \Delta\beta_j b_j B_j^n(t)}{\underbrace{\sum_{i=0}^{n} \beta_i B_i^n(t) + \Delta\beta_j B_j^n(t)}_{l(t)}}$$

Confrontando la curva modificata con quella originale si ha:

$$\begin{aligned}(\hat{X}(t) - X(t))l(t) &= \sum_{i=0}^{n} \beta_i b_i B_i^n(t) + \Delta\beta_j b_j B_j^n(t) - \sum_{i=0}^{n} \beta_i b_i B_i^n(t) - X(t)\Delta\beta_j B_j^n(t) \\ &= \Delta\beta_i B_i^n(t)[b_j - X(t)]\end{aligned}$$

La differenza tra le curve vale quindi:

$$(\hat{X}(t) - X(t)) = \frac{\Delta\beta_i B_i^n(t)}{\sum_{i=0}^{n} \beta_i B_i^n(t) + \Delta\beta_j B_j^n(t)}[b_j - X(t)]$$

Modificando un peso si ha quindi un effetto su ogni punto della curva, direttamente proporzionale alla distanza $b_j - X(t)$: la curva tende ad avvicinarsi al punto b_j se $\Delta\beta_j$ è positivo, allontanarsi altrimenti.

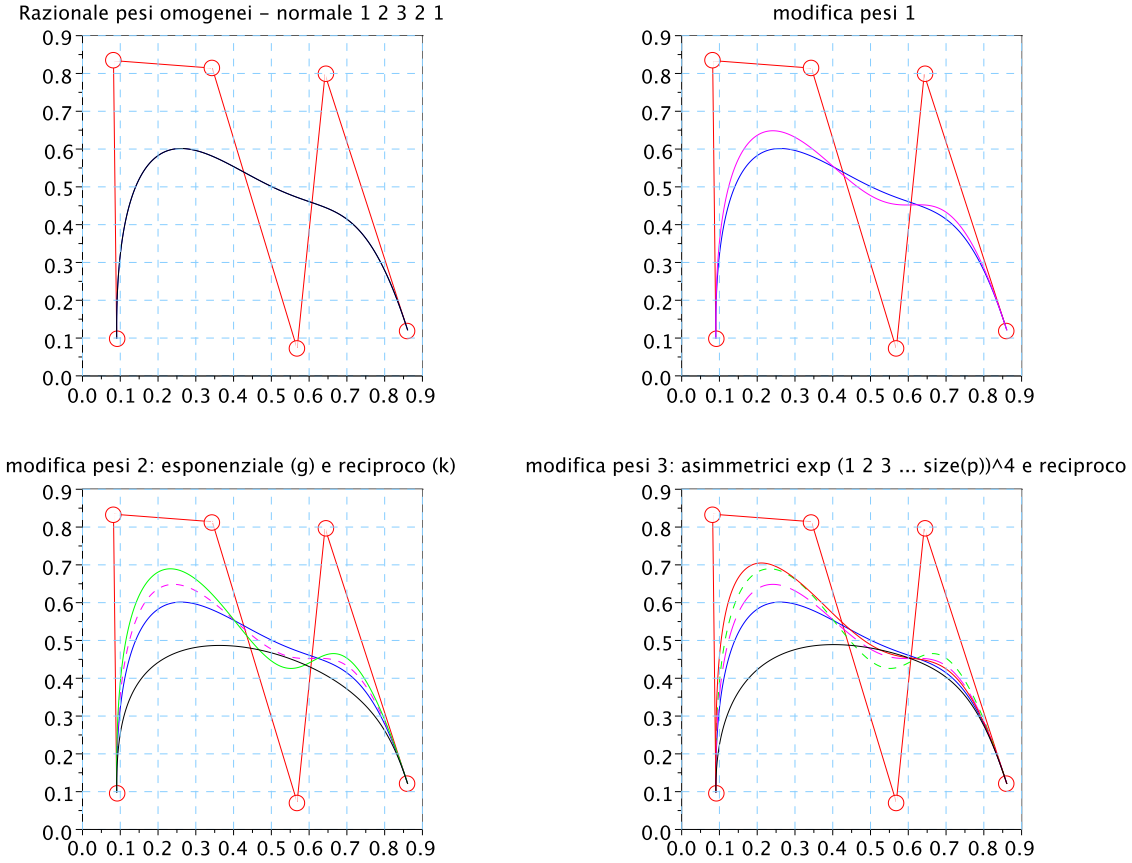


Figura 2.17: Esempio di differenti distribuzioni dei pesi

Derivata

Sia:

$$X(t) = \frac{c(t)}{w(t)} = \frac{\sum_{i=0}^{n} \beta_i b_i B_i^n(t)}{\sum_{i=0}^{n} \beta_i B_i^n(t)}$$

Si può calolare la derivata di $X(t)$:

$$c' = (Xw)' = \dot{X}w + Xw' \Rightarrow \dot{X} = \frac{c' - Xw'}{w}$$

Le derivate di $c(t)$ e $w(t)$ sono note:

$$c^{(r)} = n^{\underline{r}} \sum_{i=0}^{n-r} \Delta^r \beta_i b_i B_i^{n-r}$$
$$w^{(r)} = n^{\underline{r}} \sum_{i=0}^{n-r} \Delta^r \beta_i B_i^{n-r}$$

quindi, sostituendo in $\dot{X}(t)$ si ottiene:

$$\dot{X}(t) = \frac{n\left[\sum_{i=0}^{n-r} \Delta\beta_i b_i B_i^{n-r}(t) - \left(\sum_{i=0}^{n-r} \Delta\beta_i B_i^{n-r}(t)\right) X(t)\right]}{\sum_{i=0}^{n} \beta_i B_i^n(t)}$$

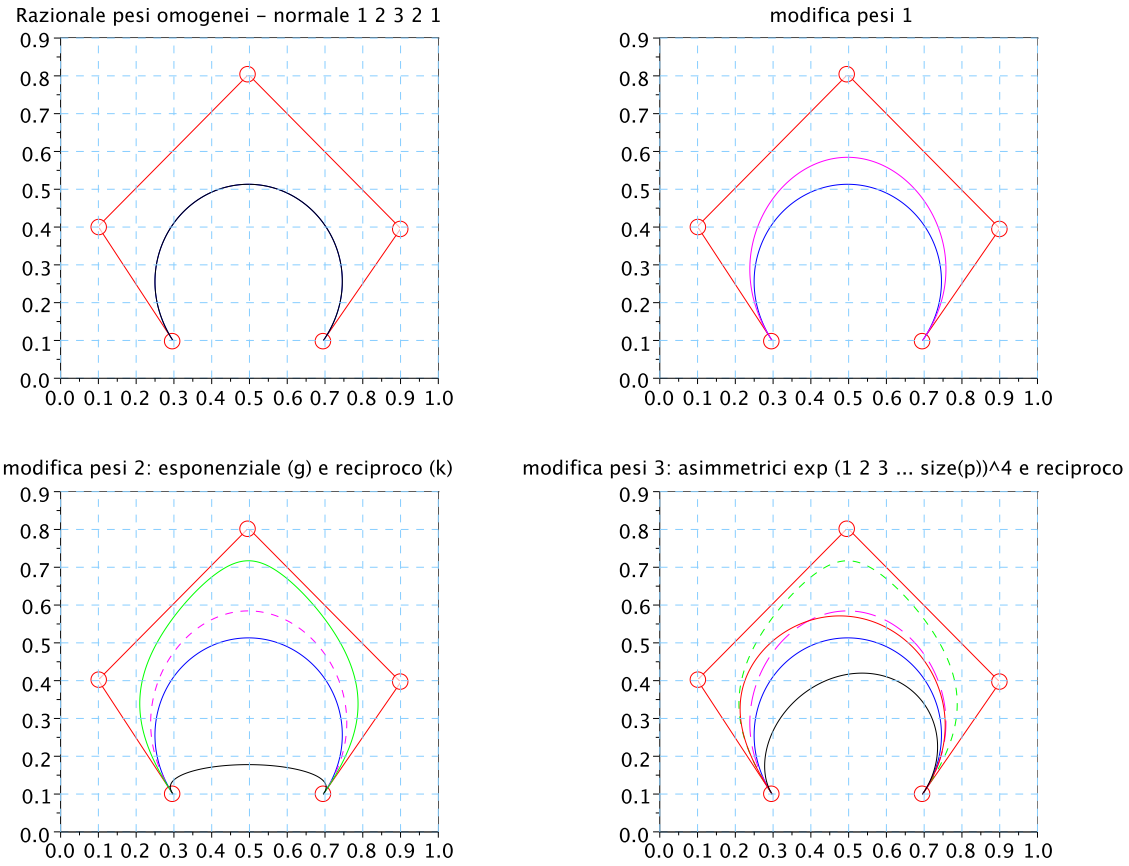


Figura 2.18: Un altro esempio di differenti distribuzioni dei pesi

Tangenza agli estremi

La tangenza agli estremi è mantenuta, infatti:

$$\dot{X}(0) = n\frac{\beta_1}{\beta_0}\Delta b_0$$
$$\dot{X}(1) = n\frac{\beta_{n-1}}{\beta_n}\Delta b_{n-1}$$

Se si volesse ottenere una curva chiusa si deve quindi imporre:

$$(\beta_{n-1} + \beta_1)b_0 = \beta_{n-1}b_{n-1} + \beta_1 b_1$$

Curve di Bézier razionali in forma standard

Definizione 2.8 (Curva di Bézier razionale in forma standard)**.** Una curva di Bézier razionale si dice in forma standard se i pesi associati agli estremi del poligono di controllo sono unitari.

Per trasformare una curva razionale in una in forma standard, si utilizza la riparametrizzazione:

$$t = \frac{(1+\alpha)u}{1+\alpha u} \equiv \phi(u), \quad \alpha \in \mathbb{R}, u \in [0,1]$$

La derivata della funziona di riparametrizzazione $\phi(u)$ vale:

$$\phi'(u) = \frac{1}{(1+\alpha u)^2}\left[(1+\alpha)(1+\alpha) - \alpha(1+\alpha)u\right] = \frac{1+\alpha}{(1+\alpha u)^2}$$

quindi la funzione è monotona per $\alpha > -1$, e la riparametrizzazione è ammissibile in quel range. Ha senso quindi applicarla:

$$X(t(u)) = \frac{\sum_{i=0}^{n}\beta_i b_i\binom{n}{i}\frac{(1+\alpha)^i u^i}{(1+\alpha u)^i}\frac{(1-u)^{n-i}}{(1+\alpha u)^{n-i}}}{\sum_{i=0}^{n}\beta_i\binom{n}{i}\frac{(1+\alpha)^i u^i(1-\alpha u)^{n-i}}{(1+\alpha u)^n}} = \frac{\sum_{i=0}^{n}\beta_i b_i \overbrace{\binom{n}{i}u^i(1-u)^{n-i}(1+\alpha)^i}^{B_i^n}}{\sum_{i=0}^{n}\beta_i\binom{n}{i}u^i(1-u)^{n-i}(1+\alpha)^i} =$$

$$= \frac{\sum_{i=0}^{n}\beta_i b_i(1+\alpha)^i B_i^n}{\sum_{i=0}^{n}\underbrace{\beta_i(1+\alpha)^i}_{\tilde{\beta}_i}B_i^n} = \frac{\sum_{i=0}^{n}\tilde{\beta}_i b_i B_i^n}{\sum_{i=0}^{n}\tilde{\beta}_i B_i^n}$$

e richiedere che siano verificate:

$$\begin{cases} \tilde{\beta}_n \equiv \beta_n(1+\alpha)^n = 1, & \alpha = \frac{1}{\sqrt[n]{\beta_n}} - 1 \\ \\ \tilde{\beta}_0 \equiv \beta_0(1+\alpha)^0 = 1, & \text{se } \beta_0 = 1 \end{cases} \tag{2.1}$$

La forma standard si può ottenere solo nel caso in cui la condizione $\beta_0 = 1$ sia già verificata inizialmente.

Algorito per il calcolo dei beta normalizzati

```
function beta=RATBezier_scalebeta(beta)
n = length(beta);
beta=beta/beta(1)
alpha=(1/beta(n))^(1/n-1);
for i=1:n
      beta(i) = (alpha^(i-1))*beta(i);
end
endfunction
```

Sezioni coniche

Le sezioni coniche sono rappresentabili utilizzando curve di Bézier di secondo grado.

Coordinate baricentriche bidimensionali

Un punto p che giace nella convex hull di altri tre $(b_0, b_1, b_2)^\dagger$ che delimitano un triangolo è identificabile univocamente con un sistema di coordinate baricentriche (u_0, u_1, u_2) a riferimenti b_0, b_1, b_2:

$$\begin{cases} p & = u_0 b_0 + u_1 b_1 + u_2 b_2 \\ 1 & = u_0 + u_1 + u_2 \end{cases}$$

Il sistema di coordinate sopra descritto è univocamente determinato, si dimostra che in:

$$\underbrace{\begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix}}_{A} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix} = \underbrace{\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}}_{p}$$

la matrice A è nonsingolare e quindi il sistema ammette soluzione unica. Il determinante di A è legato all'area del triangolo $b_0 b_1 b_2$; fissando una base per esso si ha:

$$area(A) = \tfrac{1}{2}|\Delta| = \tfrac{1}{2}|(b_1 - b_0) \times (b_2 - b_0)|$$

Applicando il metodo ci Cramer:

$$\det A_0(x, y) = \det \begin{pmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix};$$

$$\det A_1(x, y) = \det \begin{pmatrix} x_0 & x & x_2 \\ y_0 & y & y_2 \\ 1 & 1 & 1 \end{pmatrix};$$

[†] ogniuno ha componenti x_i, y_i

$$\det A_2(x,y) = \det \begin{pmatrix} x_0 & x_1 & x \\ y_0 & y_1 & y \\ 1 & 1 & 1 \end{pmatrix};$$

e quindi:

$$u_i = \frac{\det A_i}{\det A} \equiv \frac{\Delta_i(x,y)}{\Delta}$$

dove $\Delta_i = \det A_i(x,y)$ e $\Delta = \det A$. Il determinante di A è non nullo quindi le u_i sono tutte definite. Il sistema perde però l'unicità nel caso in cui tutti e tre i punti siano allineati.

Rappresentazione di curve coniche

In generale una curva conica è rappresentabile come:

$$ax^2 + 2bxy + cy^2 + dx + ey + f = 0$$

o in forma matriciale, dato il punto $p = \begin{pmatrix} x \\ y \end{pmatrix}$:

$$p^T \begin{pmatrix} a & b \\ b & c \end{pmatrix} p + (d,e)p + f = 0$$

Il determinante della matrice determina il carattere della conica:

$$det \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{cases} > 0 & \text{ellisse} \\ = 0 & \text{parabola} \\ < 0 & \text{iperbole} \end{cases}$$

Si ricordano alcune definizioni:

- **Ellisse**: luogo dei punti del piano per cui la somma delle distanze da due punti fissati (detti fuochi) è costante.

  $\sqrt{(x+c)^2+y^2} + \sqrt{(x-c)^2+y^2} = R,$ fuochi allineati sull'asse x e centro nell'origine

- **Parabola**: luogo dei punti equidistanti da una retta ed un punto disgiunti.

  $\left(y + \frac{1}{4a}\right) = \sqrt{x^2 + \left(y\frac{1}{4a}\right)^2},$ fuoco in $(0, \frac{1}{4a})$, retta $y = -\frac{1}{4a}$

- **Iperbole**: luogo dei punti per i quali la differenza delle distanze da due punti assegnati è costante.

  $\sqrt{(x+c)^2+y^2} - \sqrt{(x-c)^2+y^2} = R,$ fuochi allineati sull'asse y e centro nell'origine

Archi di coniche con curve di Bézier

Siviluppando la formula di una curva di Bézier razionale in forma standard di secondo grado e etichettandone in modo opportuno i termini:

$$X(t) = \frac{b_0 \overbrace{B_0^1(t)}^{u_0(t)} + b_1 \overbrace{\beta_1 B_1^2(t)}^{u_1(t)} + b_2 \overbrace{B_2^2(t)}^{u_2(t)}}{\underbrace{B_0^2(t) + \beta_1 B_1^2(t) + B_2^2(t)}_{w(t)}}, \quad v_0(t) = \frac{u_0(t)}{w(t)}, v_1(t) = \frac{u_1(t)}{w(t)}, v_2(t) = \frac{u_2(t)}{w(t)}$$

i punti della curva possono essere espressi in maniera semplice come:

$$X(t) = b_0 v_0(t) + b_1 v_1(t) + b_2 v_2(t)$$

Dalle proprietà dei polinomi di Bernstein segue direttamente che $\sum v_i(t) = 1$ ovvero ogni punto della curva è una combinazione baricentrica dei tre punti di controllo, leggibile anche come coordinate baricentriche rispetto agli stessi.
Calcolando il quadrato della componente v_1 si ottiene:

$$v_1^2(t) = 4\beta_1 \frac{t^2(1-t)^2}{w^2(t)} = 4\beta_1 v_0(t) v_2(t) \Rightarrow v_1^2(t) - 4\beta_1 v_0(t) v_2(t) = 0$$

sostituendo ai v_i la relativa notazione risultante dal teorema di Cramer vista nella sezione precedente ($v_i = \Delta_i(x,y)/\Delta$), si ha che punti della curva rispettano:

$$\Delta_1^2(x,y) - 4\beta_1 \Delta_0(x,y)\Delta_2(x,y) = 0$$

che un'equazione di secondo grado, ovvero una curva conica che cambia comportamento in base al valore di β_1:

$$\beta_1 \begin{cases} < 1 & \text{ellisse} \\ = 1 & \text{parabola} \\ > 1 & \text{iperbole} \end{cases}$$

Algoritmo per il calcolo di curve coniche

```
function C = RATBezier_coniche(b,s,ttab)
// beta = s/(s-1)
C = RATBezier_casteljau(b,[1,s,1],ttab)
endfunction
```

Calcolo delle coniche mostrate in fig (2.19)

```
pc = [-1,0,1;1,-1,1];
ellipse1 = RATBezier_coniche(pc,0.5,xcol);
ellipse2 = RATBezier_coniche(pc,-0.5,xcol);
parabola = RATBezier_coniche(pc,1,xcol);
iperbole = RATBezier_coniche(pc,3,xcol);
pc2 = [.5,0,1;1,0,.5];
circ1 = RATBezier_coniche(pc2,0.5,xcol);
circ2 = RATBezier_coniche(pc2,-0.5,xcol);
}
```

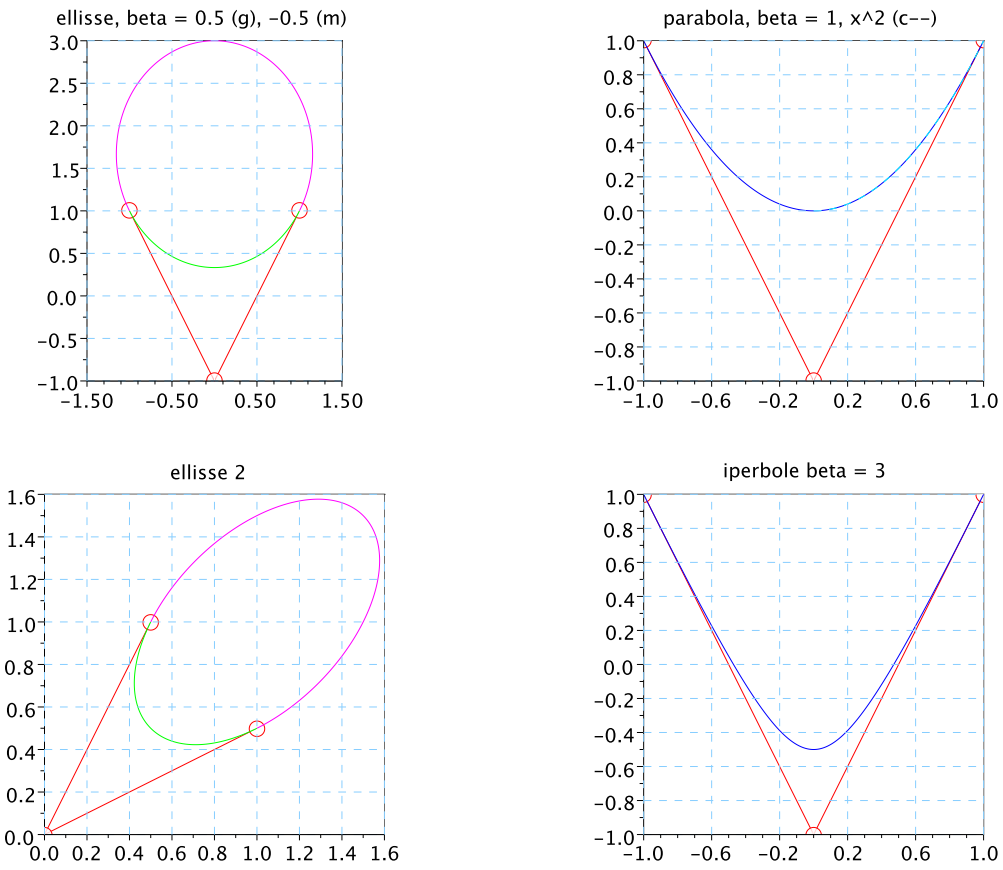


Figura 2.19: Rappresentazione di curve coniche

Capitolo 3

Curve SPLINE

Bézier-spline

Uno dei possibili modi per migliorare la controllabilità e località di una curva di Bézier è quello di considerare sottointervalli della curva come curve separate, preoccupandosi di gestire i raccordi. Dato un intervallo di definizione della curva $[a, b]$ ed una sua partizione in m sottointervalli $\tau : a \equiv u_1 < u_2 < ... < u_m < u_{m+1} \equiv b$ disgiunti ed adiacenti, si definisce la curva polinomiale a tratti:

$$X(u) : [a, b] \to \mathbb{E}^d, \quad X(u) = \bigcup X_i(t), t \in [u_i, u_{u+1})$$

in cui:

$$X_i(u) = \sum_{k=0}^{n} b_{k,i} B_k^n \left(\frac{u - u_i}{h_i} \right), \quad h_i = u_{i+1} - u_i$$

e $b_{k,i}$ è il k-esimo punto di controllo della curva i-esima. I punti della curva $X(u_i)$ sono detti nodi (knot) o punti di giunzione. La condizione di continuità si ottiene imponendo:

$$X_i(u_{i+1}) = X_{i+1}(u_{i+1}) \Rightarrow b_{n,i-1} = b_{0,i}$$

grazie alla proprietà di end-point interpolation; ad esempio date due curve disconnesse si può aggiungere un punto a tutte e due che stia spazialmente tra l'ultimo della prima e il primo della seconda.

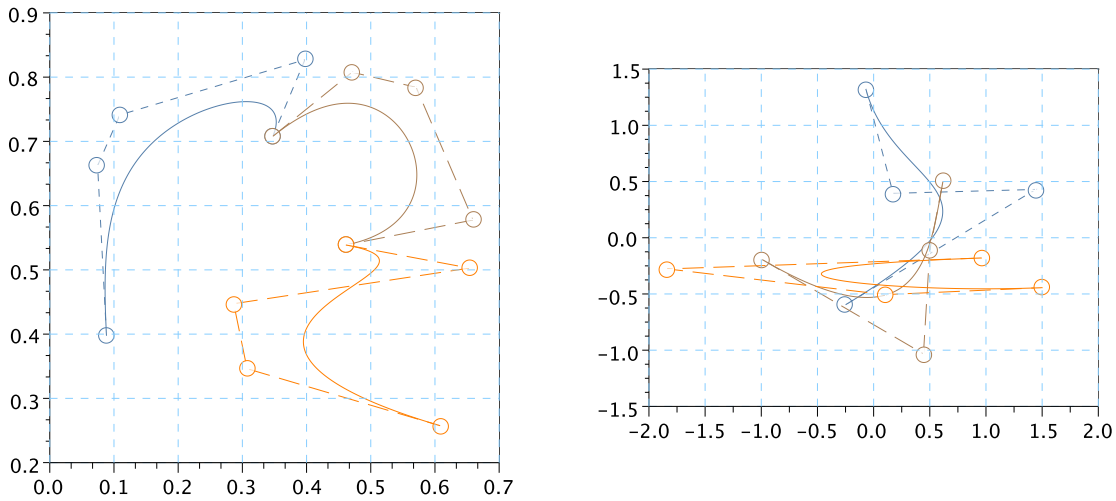


Figura 3.1: Beziér-spline con continuità C0: anche visivamente la curva è continua ma ha dei punti in cui non è derivabile. A destra la hodograph, visibilmente disconnessa.

Continuità dei raccordi

Per aumentare la regolarità della curva si può voler imporre la continuità delle derivate; ad esempio per ottenere una curva in $\mathcal{C}^1$ si impone:

$$\begin{cases} X_i(u_{i+1}) = X_{i+1}(u_{i+1}) \\ \\ \dfrac{d}{du}X_i(u_{i+1}) = \dfrac{d}{du}X_{i+1}(u_{i+1}) \end{cases}$$

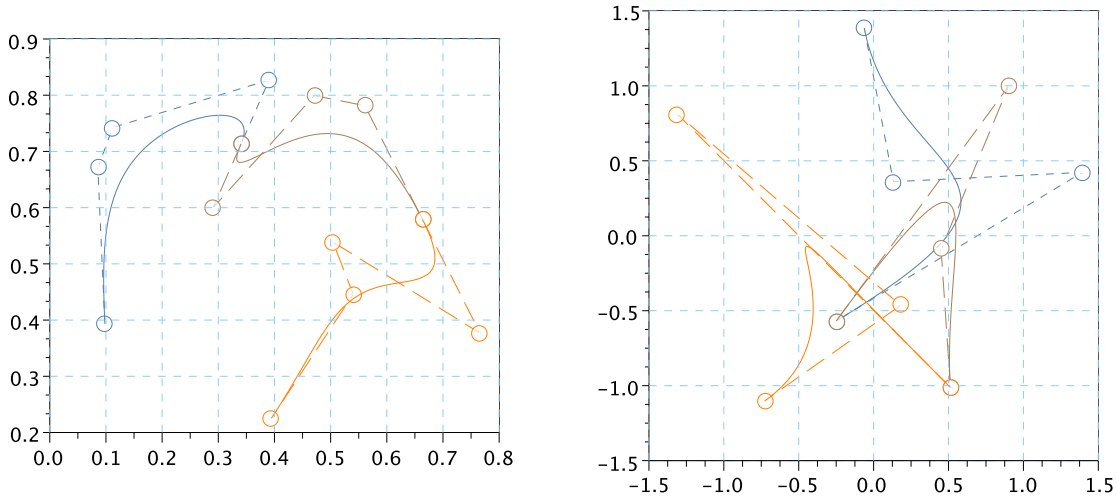


Figura 3.2: Beziér-spline con continuità C1: la curva appare continua. A destra la hodograph, di continuità C0.

Si ha che:

$$\frac{d}{du}X_i(u_{i+1}) = \frac{d}{du}X_{i+1}(u_{i+1})$$
$$\frac{d}{du}X_i(1)\frac{d}{du}\frac{u-u_i}{h_i} = \frac{d}{du}X_{i+1}(0)\frac{u-u_{i+1}}{h_{i+1}}$$
$$\frac{\Delta b_{n-1,i}}{h_i} = \frac{\Delta b_{0,i+1}}{h_{i+1}}$$

ovvero:

$$b_{1,i+1} = b_{0,i+1} + \frac{h_{i+1}}{h_i}(b_{n,i} - b_{n-1,i}).$$

Analogamente per avere continuità $\mathcal{C}^2$ si impone anche:

$$\frac{d^2}{du}X_i(u_{i+1}) = \frac{d^2}{du}X_{i+1}(u_{i+1})$$

da cui:

$$\frac{\Delta^2 b_{n-2,i}}{h_i^2} = \frac{\Delta^2 b_{0,i+1}}{h_{i+1}^2}$$

ovvero:

$$\Delta b_{1,i+1} = \Delta b_{0,i+1} + \frac{h_{i+1}}{h_i}(\Delta b_{n,i} - \Delta b_{n-1,i}).$$

Quanto appena descritto è un metodo per ottenere una curva di Bézier definita a tratti parametrizzata in modo tale che rispetto al parametro u la velocità di percorrenza della curva sia proporzionale alla dimensione dei diversi intervalli assegnati ad ogni curva. In pratica è comodo considerare l'intervallo $[a, b]$ suddiviso equamente tra i vari tratti di curva (curva polinomiale a tratti uniformi); in questo caso la funzione di riparametrizzazione è la stessa per tutte (a meno dell'intervallo) e non c'è bisogno di riparametrizzarle, perchè si può dare per scontato che ogni curva sarà valutata in $[0, 1]$ quando u è nell'intervallo corrispondente. Si ha che:
Le derivate delle due curve sono facili da derivare:

$$\frac{d}{dt}X_i(t) = \frac{d}{du}X_{i+1}(u)$$
$$n\sum_{k=0}^{n-1}\Delta b_{k,i}B_k^n(t) = m\sum_{k=0}^{m-1}\Delta b_{k,i+1}B_k^n(u)$$

Sappiamo di voler allineare l'ultimo punto della prima curva con il primo della seconda, quindi:

$$n\sum_{k=0}^{n-1}\Delta b_{k,i}B_k^{n-1}(1) = m\sum_{k=0}^{m-1}\Delta b_{k,i+1}B_k^{n-1}(0)$$
$$n\Delta b_{n-1,i} = m\Delta b_{0,i+1}$$

in quanto in 0 e 1 non si annullano soltanto rispettivamente il primo e l'ultimo polinomio di Bersntein. Da qui, tenendo conto che per avere $\mathcal{C}^0$ si impone $b_{n.i} = b_{0,i+1}$ segue direttamente la relazione tra i quattro vertici:

$$b_{1,i+1} = (m+n)b_{n,i} - mb_{n-1,i}$$

Analogamente per avere $\mathcal{C}^2$

$$n(n-1)\sum_{k=0}^{n-2}\Delta^2 b_{k,i}B_k^{n-2}(1) = m(m-1)\sum_{k=0}^{m-2}\Delta^2 b_{k,i+1}B_k^{n-2}(0)$$
$$n(n-1)\Delta^2 b_{n-2,i} = m(m-1)\Delta^2 b_{0,i+1}$$

da cui, imponendo le condizioni per $\mathcal{C}^0$ e $\mathcal{C}^1$:

$$b_{2,i+1} = 2b_{1,i+1} - b_{n,i} + \frac{m(m-1)}{n(n-1)}\left(b_{n,i} - 2b_{n-1,i} + b_{n-2,i}\right)$$

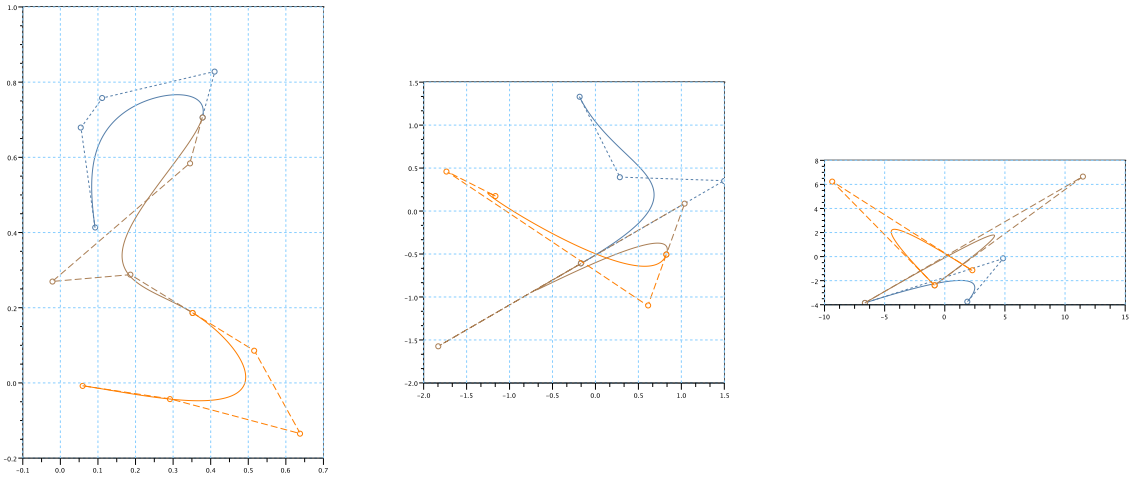


Figura 3.3: Beziér-spline con continuità C2: visivamente la curva è continua e regolare. A destra la prima e la seconda hodograph: la prima è C1 e la seconda C0.

Continuità geometrica

Due curve si raccordano con continuità geometrica $\mathcal{G}^s$ nel punto p se esiste una riparametrizzazione ammissibile per cui le curve hanno continuità $\mathcal{C}^s$ in p.
Per avere continuità $\mathcal{G}^1$, date due curve $X(u), Y(t)$ e la funzione $u(t) : [t_0, t_1] \to [u_0, u_1]$:

$$\frac{d}{dt}X(t_1) = \frac{d}{dt}Y(t_1)$$

$$\omega_1 \frac{d}{du}X(u_1) = \frac{d}{dt}Y(t_1), \quad \omega_1 = \frac{d}{dt}u(t_1) > 0$$

ovvero, formalmente due curve si raccordano $\mathcal{G}^1$ se:

$$\exists\ \omega_1 > 0 \text{ tale che } \frac{d}{dt}Y(t_1) = \omega_1 \frac{d}{du}X(u_1)$$

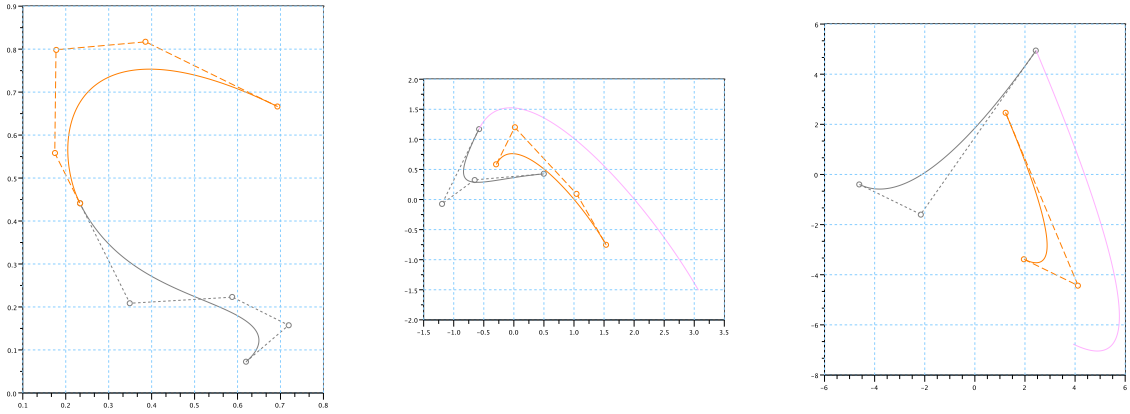


Figura 3.4: Beziér-spline con continuità G2: visivamente la curva è continua e regolare. A destra la prima e la seconda hodograph: la prima è C1 e la seconda C0 dopo che sia stata applicata la scalatura relativa al segmento di spline.

Il caso $\mathcal{G}^2$ è analogo:

$$\frac{d^2}{dt}X(t_1) = \frac{d^2}{d^2t}Y(t_1)$$

$$\frac{d}{du}\left(\frac{d}{du}X(u_1)\frac{d}{dt}u(t_1)\right) = \frac{d^2}{d^2t}Y(t_1)$$

$$\frac{d}{du}X(u_1)\frac{d^2}{d^2t}u(t_1) + \frac{d^2}{d^2u}X(u_1)\left(\frac{d}{dt}u(t_1)\right)^2 = \frac{d^2}{d^2t}Y(t_1)$$

e come prima le due curve hanno raccordo $\mathcal{G}^2$ se:

$$\exists\ \omega_1 > 0, \omega_2 > 0,\ \text{tali che}\ \frac{d^2}{d^2t}Y(t_1) = \omega_2\frac{d^2}{d^2u}X(u_1) + \omega_1\frac{d}{du}X(u_1)$$

Come prima, nel caso pratico considerando equiparametrizzate le curve si ottiene, per le continuità $\mathcal{G}^1$ e $\mathcal{G}^2$:

$$n\Delta b_{n-1,i} = \omega_1 m\Delta b_{0,i+1} \rightarrow b_{1,i+1} = \frac{(n+m\omega_1)b_{n,i} - nb_{n-1,i}}{m\omega_1}$$

$$n(n-1)\Delta^2 b_{n-2,i} = m(m-1)\Delta^2 b_{0,i+1} \rightarrow b_{2,i+1} =$$

$$\frac{(n(n-1) - m(m-1)\omega_2)b_{n,i} - 2n(n-1)b_{n-1,i} + n(n-1)b_{n-2,i} + 2\omega_2 m(m-1)b_{1,i+1}}{m(m-1)\omega_2}$$

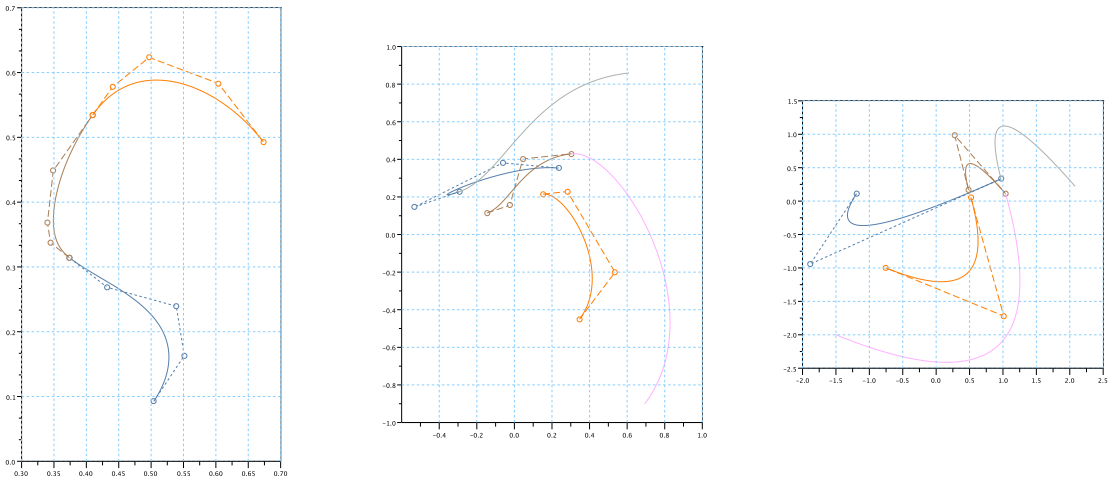


Figura 3.5: Beziér-spline con continuità G2: visivamente la curva è continua e regolare. A destra la prima e la seconda hodograph: la prima è C1 e la seconda C0 dopo che sia stata applicata la scalatura relativa al segmento di spline.

Implementazioni

Vedi appendice A.1.

B-SPLINE

Curve SPLINE monovariate

In generale, una curva SPLINE monovariata è una funzione $S : [a, b] \to \mathbb{R}^d$ polinomiale a tratti. In particolare, definita una partizione τ dell'intervallo $[a, b]$:

$$[a, b] = \bigcup I_i, \quad I_i = [\tau_i, \tau_{i+1}), \quad \tau : a \equiv \tau_1 < \tau_2 < ... < \tau_n < \tau_{n+1} \equiv b$$

la funzione S è rappresentabile come:

$$S = \bigcup P_i, \quad P_i : [\tau_i, \tau_{i+1}] \to \mathbb{R}^d, \quad P_i \in \Pi_m$$

Si indica con $S_{m,\tau}$ una generica funzione SPLINE sulla partizione τ e composta di polinomi tutti di grado m. Considerando l'intervallo $[a, b]$ suddiviso in n sottointervalli, si è definito uno spazio di dimensione:

$$dim(S_{m,\tau}) = n \times (m + 1)$$

$I_i \cap I_j = \emptyset$ con $i \neq j$, $\quad I_i \cup I_{i+1} \equiv [t_i, t_{i+2})$

in quanto ogni polinomio di grado m è identificato da $m+1$ coefficienti. In pratica, volendo rappresentare una curva, si vuole che S sia continua (ovvero $\lim_{t\to t_{i+1}} P_i(t) = P_{i+1}(t_{i+1})$) e che anche le derivate si raccordino ($S \in \mathcal{C}^{m-1}$)†; questo lascia uno spazio di dimensione:

$$dim(S_{m,\tau}) = n \times (m+1) - (n-1) \times m = n + m$$

infatti ci sono $n-1$ punti di raccordo per ogniuno dei quali si impogono m condizioni di di raccordo (una di continuità e le derivate).

Base delle potenze troncate

Si definiscono:

$$(x-\tau_i)_+^m = \begin{cases} (x-\tau_i)_+^m, & x < \tau_i \\ 0, & x \leq \tau_i \end{cases}$$

Definizione 3.1 (Base delle potenze troncate)**.** Sulla partizione $\tau : a \equiv \tau_1 < ... < \tau_l \equiv b$ si definisce *base delle potenze troncate* l'insieme di polinomi:

$$\{1, x^0, x^1, ..., x^m, (x-\tau_1)_+^m, ..., (x-\tau_{l-1})_+^m\}$$

Teorema 3.1. *L'insieme delle potenze troncate è una baste per lo spazio delle spline $S_{m,\tau}$.*

Dimostrazione. Si deve dimostrare che i polinomi della base sono $l+m$ linearmente indipendenti, ovvero che:

$$\sum_{i=0}^{m} a_i x^i + \sum_{i=1}^{l-1} b_i (x-\tau_i)_+^m = 0 \Leftrightarrow a_i = b_i = 0$$

Se $x \in I_1 \equiv [\tau_1, \tau_2]$ si ha:

$$\sum_{i=0}^{m} a_i x^i + \sum_{i=1}^{l-1} b_i \underbrace{(x-\tau_i)_+^m}_{=0} = 0$$

ovvero la tesi è dimostrata in quanto la base è quella canonica. Per $x \in I_2$ si ha:

$$\sum_{i=0}^{m} a_i x^i + b_1 (x-\tau_1)_+^m + \sum_{i=2}^{l-1} b_i \underbrace{(x-\tau_i)_+^m}_{=0} = 0$$

ma la proprità vale per gli a_i, il che implica che debba valere anche b_1. Procedendo allo stesso modo si dimostra la tesi per tutti i b_i □

Osservazione 3.1 (Derivata). $(x-\tau)_+^m \in \mathcal{C}^m$ e $\dfrac{d^m}{dx}(x-\tau)_+^m = m!$ se $x > \tau$.

†Non si può chiedere la derivabilità $\mathcal{C}^m$ perchè questo ridurebbe lo spazio a $k+1$ dimensioni, ovvero si perderebbe il controllo sulla posizione dei punti di raccordo

Base delle B-SPLINE

La base delle potenze troncate presenta problemi di malcondizionamento; questi possono essere risolti definendo, sullo stesso concetto, una base a supporto compatto.
Si definisce una partizione di $[a, b]$[†] più rilassata della precedente:

$$\nu : a \equiv \nu_0 \leq \nu_1 \leq ... \leq \nu_l \equiv b, \quad l \text{ intervalli}$$

e, data la base:

$$N_{i,1}(x) = \begin{cases} 1, & \nu_i \leq x \leq \nu_{i+1} \\ 0, & \text{altrimenti} \end{cases}$$

si definisce ricorsivamente:

$$N_{i,k}(x) = \frac{x - \nu_i}{\nu_{i+k-1} - \nu_i} N_{i,k-1}(x) + \frac{\nu_{i+k} - x}{\nu_{i+k} - \nu_{i+1}} N_{i+1,k-1}(x)$$

Nota (Abuso di notazione). Per comodità, se uno dei denominatori di un termine in $N_{i,k}(x)$ si annulla, questo viene considerato nullo.

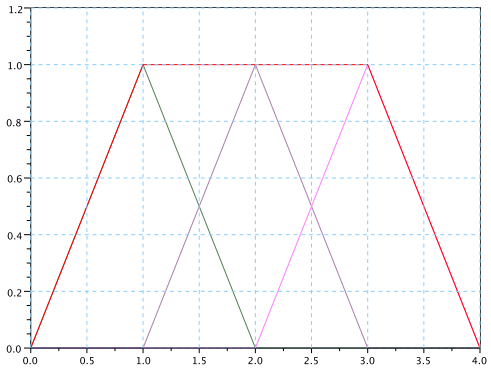
Esempio 3.1 (Caso $k = 2$)**.**

$$N_{i,2}(x) = \begin{cases} \dfrac{x - \nu_i}{\nu_{i+1} - \nu_i}, & x \in [\nu_i, \nu_{i+1}) \\ \dfrac{\nu_{i+2} - x}{\nu_{i+2} - \nu_{i+1}}, & x \in [\nu_{i+1}, \nu_{i+2}) \\ 0 & \text{altrimenti} \end{cases}$$

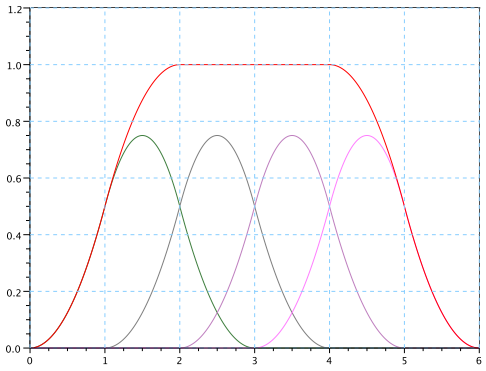
Esempio 3.2 (Caso $k = 3$)**.**

$$N_{i,3}(x) = \begin{cases} \dfrac{x - \nu_i}{\nu_{i+2} - \nu_i} N_{i,2}(x), & x \in [\nu_i, \nu_{i+1}) \\ \dfrac{x - \nu_i}{\nu_{i+2} - \nu_i} N_{i,2}(x) + \dfrac{\nu_{i+3} - x}{\nu_{i+3} - \nu_{i+2}} N_{i+1,2}(x), & x \in [\nu_{i+1}, \nu_{i+2}) \\ \dfrac{\nu_{i+3} - x}{\nu_{i+3} - \nu_{i+1}} N_{i+1,2}, & x \in [t_{i+2}, t_{i+3}) \\ 0 & \text{altrimenti} \end{cases}$$

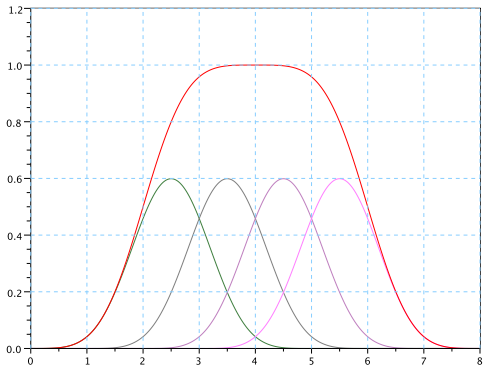
[†]vettore dei nodi



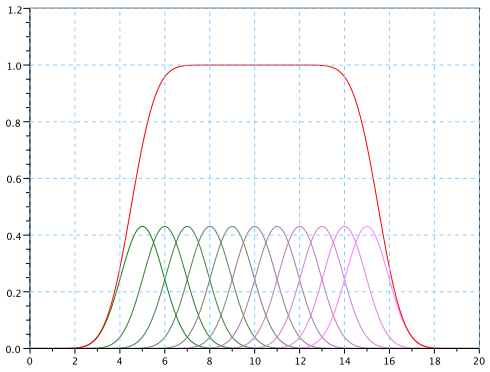
(a) k=2



(b) k=3



(c) k=5



(d) k=10

Figura 3.6: Base delle BSPLINE, in rosso è evidenziata la somma delle componenti.

Proprietà

Supporto compatto

$N_{i,k}(x)$ è a supporto compatto, infatti $N_{i,k} = 0$ se $t \notin [t_i, t_{i+k}]$.

Non negatività

$N_{i,k}(x) \geq 0 \, \forall t \in [t_i, t_{i+k}]$, in quanto somma di soli termini positivi in quell'intervallo.

Partizione dell'unità

$$\sum_{i=0}^{n} N_{i,k}(t) = 1 \, \forall t \in [\nu_{k-1}, \nu_{n+1}]$$

Per induzione, la tesi è vera banalmente per $N_{i,1}(t)$ e $t \in [t_0, t_{n+1}]$; supponendola vera per $N_{i,k-1}(t)$, $t \in [\nu_{k-2}, \nu_{n+1}]$ per $N_{i,k}(t)$ e $t \in [\nu_{k-1}, \nu_{n+1}]$ vale:

$$\begin{aligned}
\sum_{i=0}^{n} N_{i,k}(t) &= \sum_{i=0}^{n} \left[\frac{t - \nu_i}{\nu_{i+k-1} - \nu_i} N_{i,k-1}(t) + \frac{\nu_{i+k} - t}{\nu_{i+k} - \nu_{i+1}} N_{i+1,k-1}(t) \right] \\
&= \frac{t - \nu_0}{\nu_{k-1-\nu_0}} \underbrace{N_{0,k-1}(t)}_{\neq 0 \Leftrightarrow t \in [\nu_0, \nu_{k-1}]} + \sum_{i=1}^{n} \left[\frac{t - \nu_i}{\nu_{i+k-1} - \nu_i} + \frac{\nu_{i+k-1} - t}{\nu_{i+k-1} - \nu_i} \right] N_{i,k-1}(t) \\
&\quad + \frac{\nu_{n+k} - t}{\nu_{n+k} - \nu_{n+1}} \underbrace{N_{n+1,k-1}(t)}_{\neq 0 \Leftrightarrow t \in [\nu_{n+1}, \nu_{n+k-1}]} \\
&= 0 + \sum_{i=1}^{n} N_{i,k-1}(t) + 0 + \underbrace{N_{0,k-1}(t)}_{=0, \, t \in [\nu_{k-1}, \nu_{n+1}]} \\
&= \sum_{i=0}^{n} N_{i,k-1}(t)
\end{aligned}$$

e in quanto $[\nu_{k-1}, \nu_{n+1}] \subset [\nu_{k-2}, \nu_{n+1}]$, la proprietà è dimostrata.

Derivata

La derivata della base delle B-SPLINE è definita come segue:

$$\frac{dN_{i,k}(x)}{dx} = (k-1) \left[\frac{N_{i,k-1}(x)}{\nu_{i+k-1} - \nu_i} - \frac{N_{i+1,k-1}(x)}{\nu_{i+k} - \nu_{i+1}} \right]$$

Indipendenza lineare

$$\sum_{i=0}^{n} a_i N_{i,1}(x) = 0 \Leftrightarrow a_i = 0, \ \forall i$$

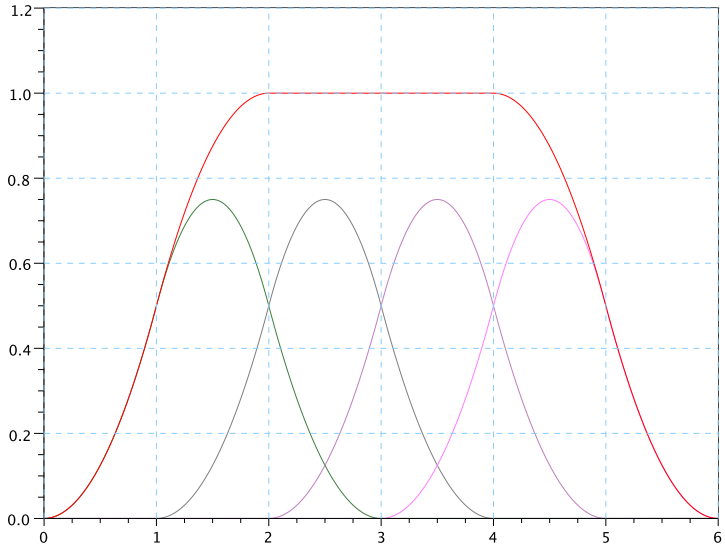


Figura 3.7: Partizione dell'unità: la somma delle basi è unitaria per $t \in [\nu_{k-1}, \nu_{n+1}]$. (in questo caso $k = 3, T = \{0, 1, 2, 3, 4, 5, 6\}, n = 3$)

si verifica banalmente, $\forall t \in [a, b]$. Analogamente, se $k = 2$:

$$\sum_{i=0}^{n} a_i N_{i,2}(x) = 0 \Leftrightarrow a_i = 0, \ \forall i$$

infatti nei nodi $(x = \nu_j)$ solo $N_{k-1,2}(\nu_j) \neq 0$ e a_j deve essere diverso da zero, per ogni j. Per dimostrare la proprietà per $k > 2$ occorre procedere per induzione, utilizzando i precedenti come casi base. Tenendo conto che vale:

$$\sum_{i=0}^{n} a_i N_{i,k}(x) = 0 \Rightarrow (k-1) \sum_{i=0}^{n} a_i \left[\frac{N_{i,k-1}(x)}{\nu_{i+k-1} - \nu_i} - \frac{N_{i+1,k-1}(x)}{\nu_{i+k} - \nu_{i+1}} \right] = 0$$

espandendo e risommando i termini della sommatoria nella derivata si ottiene:

$$(k-1) \sum_{i=1}^{n} \left[\frac{a_i - a_0}{\nu_{i+k-1} - \nu_i} \right] N_{i,k-1}(x) = 0$$

da cui si deve avere $a_i = a_{i+1}$ ovvero tutti i coefficienti coincidenti. Riportando questo risultato alla sommatoria di partenza:

$$a \sum_{i=0}^{n} N_{i,k}(t) = 0 \Leftrightarrow a = 0$$

in quanto gli $N_{i,k}(x)$ sono una combinazione baricentrica.

Generalizzazione della base delle B-SPLINE

La definizione di SPLINE data nella sezione precedente richiede che la curva abbia lo stesso tipo di continuità su tutti i nodi. Questo vincolo nella pratica può essere troppo stringente, e talvolta conviele usare uno spazio delle spline allargato.
Data la partizione:

$$\tau : a \equiv \tau_0 < \tau_1 < ... < \tau_n \equiv b, \quad I_i = [\tau_i, \tau_{i+1})$$

ed un vettore:

$$M = m_1, ..., m_{n-1}, \quad 0 \leq m_i \leq k$$

Questo vettore sta ad indicare il tipo di raccordo che si vuole nei punti $\tau_2, .., \tau_n$, se $m_i = 0$ non si chiede neanche la continuità, se $m_i = s$ si chiede la continuità delle derivate dalla zeresima alla s. Si definisce:

$$S_{m,\tau,M} = \Bigg\{ f : [a,b] \to \mathbb{R}^d \text{ per cui } \exists\ p_1(t), ..., p_n(t) \in \Pi_m,\ f(t)|_{I_i} = p_i(t), \text{ e}$$
$$p_{i-1}^{(s)}(\tau_i) = p_i^{(s)}(\tau_i),\ s = \{0, ..., m_i - 1\} \text{ se } m_i > 0,$$
$$\text{nessuna condizione imposta in } \tau_i \text{ se } m_i = 0 \Bigg\}$$

per cui se $m_i = k$ $S_{m,\tau,M} = S_{m,\tau}$ mentre se $m_i = 0$, $S_{m,\tau,M} = P_{m,\tau}$ ovvero non sussiste neanche la condizione di raccordo semplice. Si ha quindi che $\Pi_m \subset S_{m,\tau} \subseteq m, \tau, M \subseteq P_{m,\tau}$, e lo spazio appena definito ha dimensione[†]:

$$dim(S_{m,\tau,M}) = (m+1) \times n - \sum_{i=1}^{n-1} m_i = n + \underbrace{\sum_{i=1}^{n-1} \frac{nm - nm_i + m_i}{n-1}}_{\mu} = n + \mu$$

Per comodità, usualmente il vettore M non si riferisce direttamente al grado di derivabilità nei nodi, ma indica direttamente la molteplicità del nodo i-esimo nel vettore dei nodi, e prende il nome di vettore delle molteplicità. In una curva di grado k la molteplicità del nodo (m) e il grado di derivabilità della curva (d) sono legate dalla relazione:

$$m = k - d + 1 \tag{3.1}$$

ovvero per ottenere una curva a derivabilità massima (che coincide con le B-SPLINE classiche) la molteplicità dei nodi deve essere minima (1) e viceversa, per ottenere la derivabilità minima (nel caso pratico si richiede sempre la continuità $\mathcal{C}^0$) la molteplicità è massima (k).

Definizione 3.2 (Vettore dei nodi esteso)**.** A partire dall'insieme M e dal vettore dei nodi ν (che ha n intervalli) si definisce il vettore dei nodi esteso[‡]:

$$T : \underbrace{t_0 \leq ... \leq t_{k-2}}_{\text{(a)}} \leq \underbrace{t_{k-1} \leq ... \leq t_{n+\mu+1}}_{\text{(b)}} \leq \underbrace{t_{n+\mu+2} \leq ... \leq t_{n+\mu+k}}_{\text{(c)}}$$

[†]Come in precedenza, n intervalli, m grado
[‡]Come in precedenza, k grado, $m_i \in M$

in cui:

(a) è il *vettore dei nodi ausiliari sinistri*, e $\tau_0 = t_0 \leq ... \leq t_{k-2}$

(c) è il *vettore dei nodi ausiliari destri* $\tau_n = t_{n+\mu+2} \leq ... \leq t_{n+\mu+k}$

(b) è tale che:

$$\tau_0 \equiv t_{k-1} < \tau_1 \equiv t_k = ... = t_{k+m_1} < ... < t_{n+\mu-m_n} = ... = t_{n+\mu} \equiv \tau_{n-1} < t_{k+\mu+1} \equiv \tau_n$$

Come prima, si definisce ricorsivamente la base per le spline:

$$N_{i,1}(t) = \begin{cases} 1 & t \in [t_i, t_{i+1}) \\ \\ 0 & \text{altrimenti o se } t_i = t_{i+1} \end{cases}$$

$$N_{i,k}(t) = \begin{cases} 0 & t_i = ... = t_{i+k} \\ \\ 0 + \dfrac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) & t_i = ... = t_{i+k-1} < t_{i+k} \\ \\ \dfrac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + 0 & t_i < ... < t_{i+k-1} = t_{i+k} \\ \\ \dfrac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \dfrac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) & t_i < ... < t_{i+k} \end{cases}$$

Osservazione 3.2 (Caso particolare: polinomi di Bernstein).
Ponendo $[\tau_0, \tau_1] = [0, 1]$, il vettore dei nodi estesi per una spline di grado k diventa:
$t : \tau_0 = t_0 = ... = t_{k-1} < t_k = ... = t_{2k-1} = \tau_1$.
Dalla definizione di $N_{i,k}$, per $i < k$ segue:

$$N_{1,k}(t) = \frac{t - t_1}{t_k - t_1} \underbrace{N_{1,k-1}(t)}_{=0} + \frac{t_{1+k} - t}{t_{1+k} - t_2} N_{2,k-1}(t)$$

quindi, ricorsivamente:

$$N_{2,k-1}(t) = \frac{t - t_2}{t_{k+1} - t_2} \underbrace{N_{2,k-2}(t)}_{=0} + \frac{t_{2+k} - t}{t_{2+k} - t_3} N_{3,k-2}(t)$$

Si può scrivere la formula generica:

$$N_{i,k}(t) = \frac{t_{1+k} - t}{t_{1+k} - t_2} \frac{t_{2+k} - t}{t_{2+k} - t_3} ... N_{i+k-1,1}(t)$$

ma $t_0 = ... = t_{k-1}$ e $t_k = ... = t_{2k-1}$ da cui:

$$N_{i,k}(t) = \left(\frac{\tau_1 - t}{\tau_1 - \tau_0}\right)^{k-1} N_{i+k-1,1}(t)$$

analogamente, per $i \geq k$:

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} \underbrace{N_{i+1,k-1}(t)}_{=0} = ... = \left(\frac{t - \tau_0}{\tau_1 - \tau_0}\right)^{k-1} N_{2k-1,1}(t)$$

Da qui, sostituendo $\tau_0 = 0$ e $\tau_1 = 1$ si ottiene:

$$N_{1,k}(t) = (1-t)^{k-1} \equiv B_1^{k-1}(t), \quad N_{k,k}(t) = t^{k-1} \equiv B_{k-1}^{k-1}, \quad t \in [\tau_0, \tau_1]$$

In generale si può dimostrare che:

$$N_{i+j,k}(t) = B_k^{k-1}(s), \quad t \in [t_{i+j}, t_{i+j+1}],\ s \in [\tau_0, \tau_1]$$

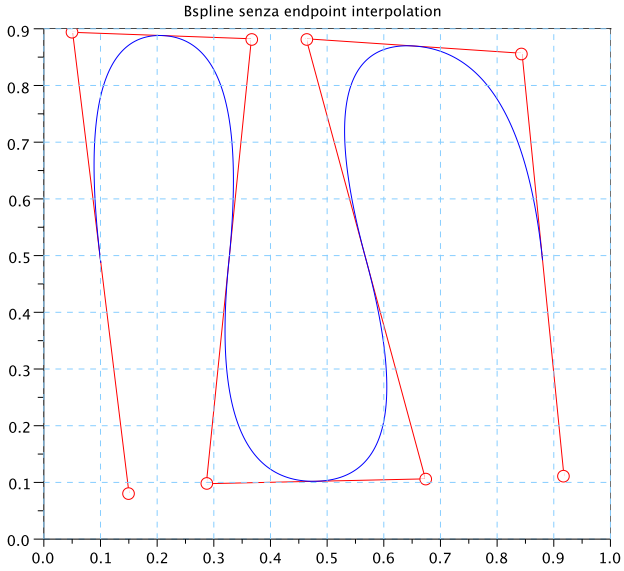
Algoritmo per il calcolo della base delle BSPLINE

```
function base = Bspline_base(i,k,T,t)


if k==1 then
   if t<T(i+1) & T(i)<=t then base=1; else base=0; end;
else
      if  T(i)==T(i+k) then
         base=0;
      else
         if (T(i+k-1)~=T(i)) then
            lp = ((t-T(i))/(T(i+k-1)-T(i)))* Bspline_base(i,k-1,T,t)
         else
            lp = 0
         end

         if (T(i+k)~=T(i+1)) then
            rp = ((T(i+k)-t)/(T(i+k)-T(i+1)))* Bspline_base(i+1,k-1,T,t)
         else
            rp = 0
         end

         base=lp+rp;
      end
end
endfunction
```

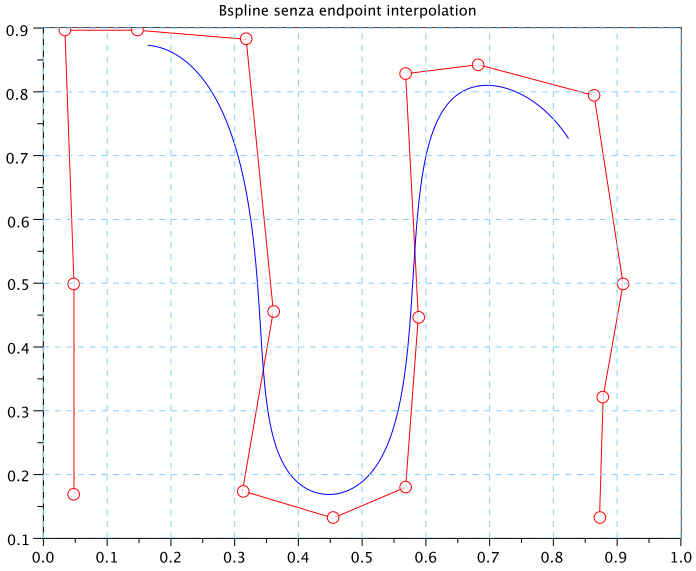
Derivata

La definizione di derivata è l'estensione diretta di quella della base B data in precedenza:

$$\frac{dN_{i,k}(t)}{dt} = (k-1)\left[\frac{N_{i,k-1}(t)}{t_{i+k-1} - t_i} - \frac{N_{i+1,k-1}(t)}{t_{i+k-t_{i+1}}}\right]$$



(a) k=3



(b) k=8

Figura 3.8: Due esempi di curve BSPLINE, una di 3 e una di 8 grado. Si nota chiaramente la mancanza di interpolazione agli estremi.

Curve B-SPLINE

La proprietà di somma unitaria della B-base estesa in $[t_{k-1}, t_{n+1}]$ permette di utilizzarla in modo analogo ai polinomi di Bernstein per definire una curva:

$$X(t) = \sum_{i=0}^{n} d_i N_{i,k}(t), \quad t \in [t_{k-1}, t_{n+1}]$$

in cui d_i sono punti in $\mathbb{E}^d$.

Algoritmo per il disegno delle BSPLINE: ricorsivo

```
function Curve = Bspline_curve(k,T,b,t)
lt = length(t)
[d,n] = size(b)
lT = length(T)
Curve=[]
for p=1:lt-1
      c=[]
      count = 1
      for i=1:lT-k
      c=c+b(:,count).*Bspline_base(i,k,T,t(p))
      count = count+1
      end
      Curve=[Curve,c]
end
endfunction
```

Proprietà

Invarianza alle trasformazioni affini

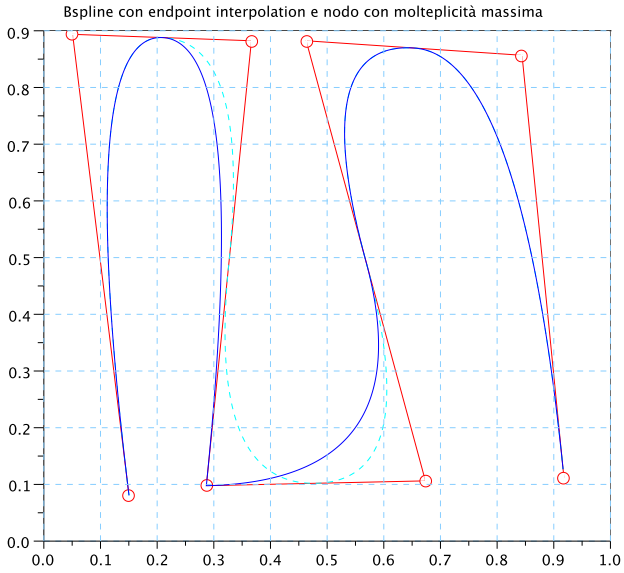
$$\Phi(X(t)) = AX(t) + v = \sum_{i=0}^{n} \Phi(d_i) N_{i,k}(t)$$

Controllo locale Se $t \in [t_r, t_{r+1}]$:

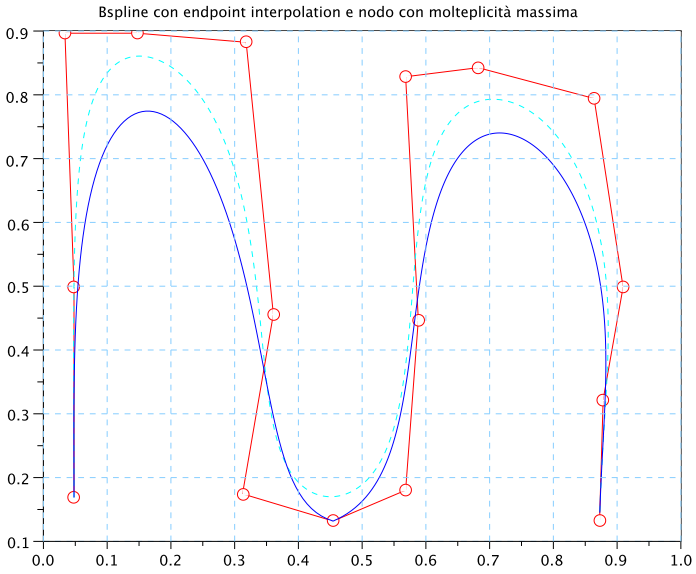
$$X(t) = \sum_{i=r-k+1}^{r} d_i N_{i,k}(t)$$

in quanto tutte le altre $N_{i,k}(t)$ sono nulle, ne consegue che la modifica di d_i ha effetti sulla curva solo nel corrispondente intervallo t_i, t_{i+k}.

Strong ConvexHull Come nel caso dei polinomi di Bernstein, la somma $\sum_{i=0}^{n} d_i N_{i,k}(t), \quad t \in [t_{k-1}, t_{n+1}]$ è una combinazione baricentrica dei d_i, quindi la curva giace nella convex hull di questi; in più la proprietà di località garantisce che la curva, per $t \in [t_r, t_{r+1}]$ sta anche nella convex hull dei punti $[t_i, t_{i+k}]$, ovvero si ha la convex hull forte.



(a) k=3



(b) k=8

Figura 3.9: Molteplicità dei nodi: i due esempi precedenti di curve, con un nodo a molteplicità massima (derivabilità minima).

Linear precision (come conseguenza della strong convex hull) Se si hanno $(k-1)$ punti di controllo allineati $d_{j+i} = d_j + \alpha_i v$, $r = 0, .., k-1$ l'equazione della curva diventa:

$$X(t_{j+k-1}) = \sum_{i=j}^{j+k-1} d_i N_{i,k}(t_{j+k-1}) = d_j + v \left(\sum_{i=j}^{j+k-1} \alpha_{i-j} N_{i,k}(t_{j+k-1}) \right)$$

ovvero se almeno $k-1$ punti sono allineati la curva tocca il segmento su cui giacciono i punti di controllo, e se almeno k lo sono la curva si schiaccia sul segmento stesso.

Derivata

$$\begin{aligned} \frac{dX(t)}{dt} &= (k-1) \sum_{i=0}^{n} d_i \left[\frac{N_{i,k-1}(t)}{t_{i+k-1} - t_i} - \frac{N_{i+1,k-1}(t)}{t_{i+k} - t_{i+1}} \right] \\ &= (k-1) \sum_{i=1}^{n} \frac{d_i - d_{i-1}}{t_{i+k-1} - t_i} N_{i,k-1}(t) \end{aligned}$$

scrivendo il coefficiente $\frac{d_i - d_{i-1}}{t_{i+k-1} - t_i}$ come $d_i^{[1]}$ (e ricorsivamente $d_i^{[r]} = \frac{d_i^{[r-1]} - d_{i-1}^{[r-1]}}{t_{i+k-1} - t_i}$) si può generalizzare:

$$\frac{d^r X(t)}{d^r t} = (k-1)...(k-r) \sum_{i=1}^{n} d_i^{[r]} N_{i,k-1}(t), \quad t \in [t_{k-1}, t_{n+1}]$$

End point interpolation In generale la proprietà di end point interpolation non vale per le B-SPLINE, ma la si può forzare.
Dato il vettore ausiliario $t : t_0, ..., t_i, ..., t_{i+k-2}, ..., t_{k+\mu+k}$, se i k-1 nodi $t_i, ..., t_{i+k-2}$ coincidono si ha che $X(t_i) = d_{i-1}$. Quindi, forzando $t_0 = ... = t_{k-1} = \tau_0$ e $t_{k+\mu+2} = ... = t_{k+\mu+k} = \tau_n$, $N_{0,k}(t) = B_0^{k-1} \left(\frac{t - \tau_0}{\tau_1 - \tau_0} \right)$ con $t \in [\tau_0, \tau_1]$ e $N_{n,k}(t) = B_{k-1}^{k-1} \left(\frac{t - \tau_{n-1}}{\tau_n - \tau_{n-1}} \right)$ con $t \in [\tau_{n-1}, \tau_n]$, ovvero la proprietà è verificata.

Derivata agli estrimi: raccordo Per disegnare una curva chiusa a derivate continue si deve imporre:

$$\frac{d^r X(\tau_0)}{d^r t} = \frac{d^r X(\tau_n)}{d^r t}, \quad r = 0, ..., k-2$$

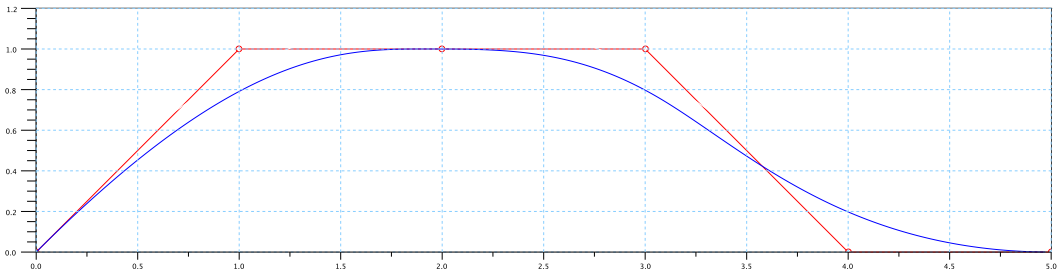
ovvero, per ogni r:

$$(k-1)...(k-r) \sum_{i=1}^{n} d_i^{[r]} N_{i,k-1}(\tau_0) = (k-1)...(k-r) \sum_{i=1}^{n} d_i^{[r]} N_{i,k-1}(\tau_n)$$

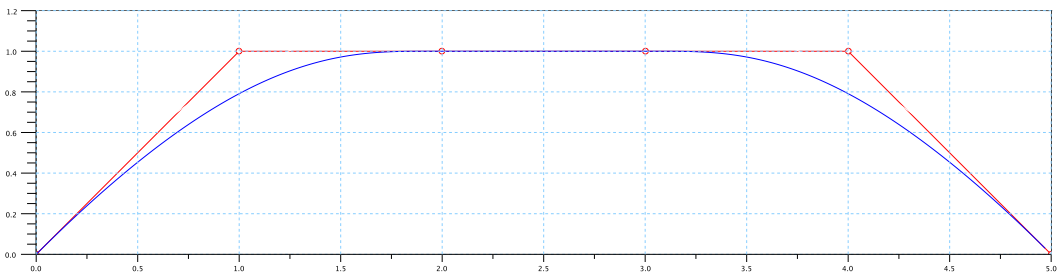
Per le proprietà degli $N_{i,k}$:

$$(k-1)...(k-r) \sum_{i=0}^{k-2} d_i^{[r]} N_{i,k-1}(\tau_0) = (k-1)...(k-r) \sum_{i=n-k+2}^{n} d_i^{[r]} N_{i,k-1}(\tau_n)$$

da cui derivano direttamente le condizioni da imporre, ovvero:

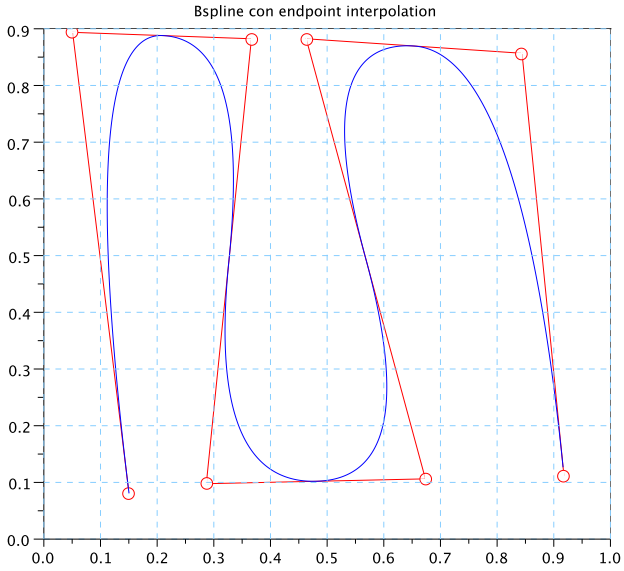


(a) k=4, 3 nodi allineati

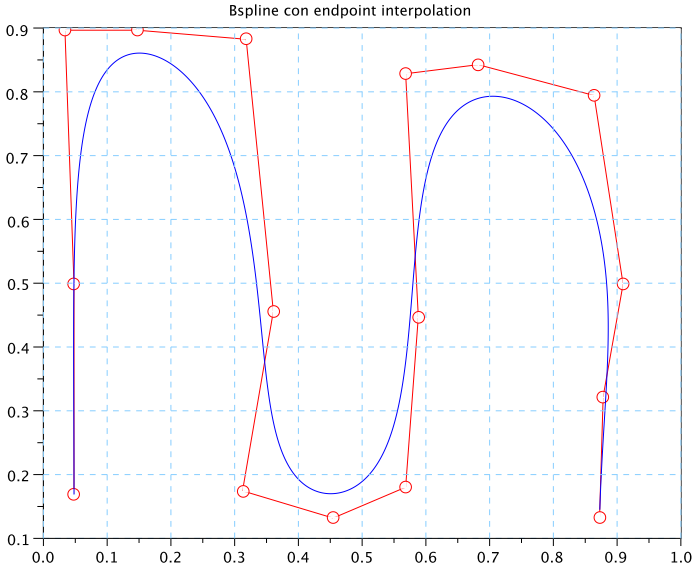


(b) k=4, 4 nodi allineati

Figura 3.10: Linear precision: esempi con $k-1\,(=3)$ e $k\,(=4)$ nodi allineati.



(a) k=3



(b) k=8

Figura 3.11: End point interpolation: i due esempi precedenti di curve a cui è stata forzata l'end point interpolation.

- I primi e gli ultimi punti di controllo devono coincidere: $d_i = d_{n-k+2-i}$, $i = 0, ..., k-2$
- I primi e gli ultimi nodi ausiliari vanno definiti in modo tale da rispettare le distanze:

$$\begin{cases} h_i & = h_{n-k+2+i} \\ h_{n+1+i} & = h_{k-1+i} \end{cases}$$

sempre per $i = 0, ..., k-2$

Algoritmi per curve BSPLINE

Disegno della curva: de Boor

L'algoritmo di de Boor calcola i punti della curva in maniera analoga a quello di de Casteljau per le curve di Bézier.
Considerando $t \in [t_{k-1}, t_{n+1}]$

$$\begin{aligned} X(t) &= \sum_{i=0}^{n} d_i^{(0)} N_{i,k}(t) \\ &= \sum_{i=0}^{n} d_i^{(0)} \left[\frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \right] \\ &= \sum_{i=1}^{n} \underbrace{\left[\underbrace{\frac{t - t_i}{t_{i+k-1} - t_i}}_{\alpha_i^{(1)}(t)} d_i^{(0)} + \underbrace{\frac{t_{i+k-1} - t}{t_{i+k-1} - t_i}}_{1-\alpha_i^{(1)}(t)} d_{i-1}^{(0)} \right]}_{d_i^{(1)}(t)} N_{i,k-1}(t) \end{aligned}$$

a questo punto, definendo $\alpha_i^{(j)}(t) = \dfrac{t - t_i}{t_{i+k-j} - t_i}$ e di conseguenza $d_i^{(j)}(t) = \alpha_i^{(j)}(t) d_i^{(j-1)}(t) + (1 - \alpha_i^{(j)}(t)) d_{i-1}^{(j-1)}(t)$:

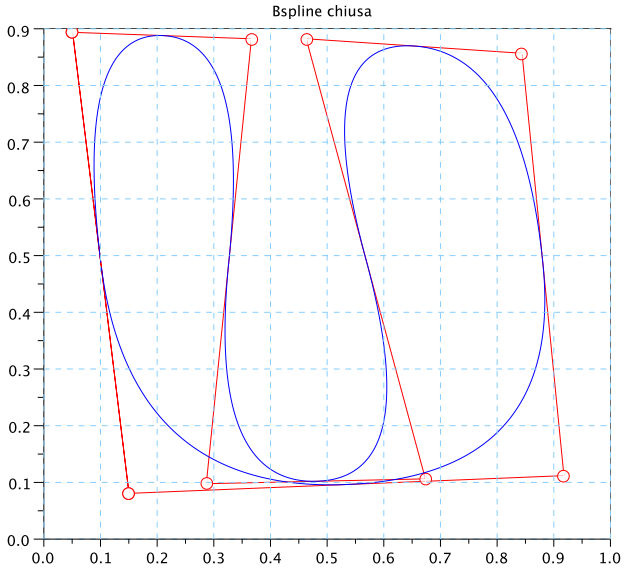
$$\begin{aligned} X(t) &= \sum_{i=j}^{n} d_i^{(j)}(t) N_{i,k-j}(t) \\ &= \sum_{i=k-1}^{n} d_i^{(k-1)}(t) N_{i,1}(t) \end{aligned}$$

ovvero, se $t \in [t_r, t_{r+1}]$, $X(t) = d_r^{(k-1)}(t)$.

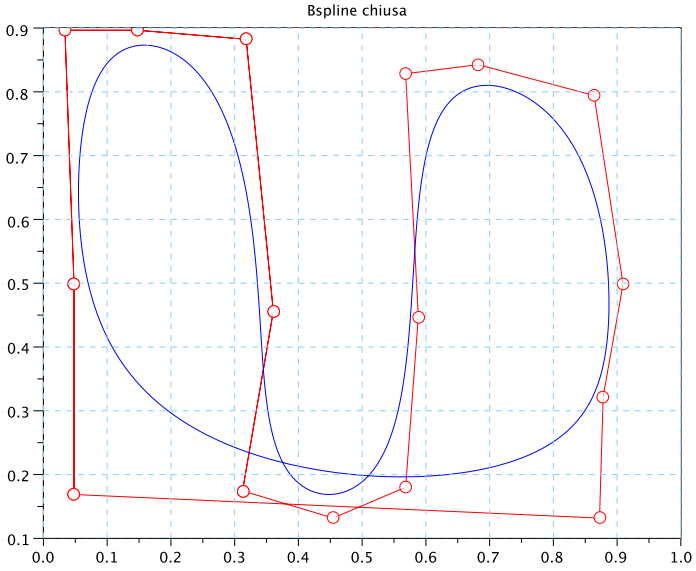
Per $t \in [t_r, t_{r+1}]$ si possono escludere dalla sommatoria iniziale i termini nulli:

$$X(t) = \sum_{i=r-k+1}^{r} d_i^{(k-1)}(t) N_{i,1}(t)$$

ed è possibile ottimizzare l'algoritmo.



(a) k=3



(b) k=8

Figura 3.12: Derivata agli estremi: i due esempi precedenti di curve, con T modificato per avere una curva chiusa.

Algoritmo per il disegno delle BSPLINE: deBoor

```
function Curve = Bspline_deboor(k,T,b,t)
lt = length(t)
[d,n] = size(b)
lT = length(T)


intervals=[]
for i=1:(lT-1)
      if T(i)<T(i+1)
         intervals=[intervals,[T(i);i]]
      end
end
intervals=[intervals,[T(lT);min(find(T==T(lT)))]]

Curve = []
for p=1:lt-1
      r = intervals(2,max(find(intervals(1,:)<=t(p))))
      d=b
      for j=1:k-1
         for i=r:-1:r-k+j+1
            a = (t(p)-T(i))/(T(i+k-j)-T(i))
          d(:,i)=a*d(:,i)+(1-a)* d(:,i-1)
         end
      end
      Curve=[Curve,d(:,r)]
end
endfunction
```

Knot Insertion

Si vuole inserire un nodo $t^* \notin T$ tale che $t^* \in [t_r, t_{r+1}]$ per creare il vettore $T^* = T \cup \{t^*\}$, e definire una curva su T^* che sia identica a quella precedentemente definita su T. In particolare si definisce la curva:

$$X^*(t) = \sum_{i=0}^{n+1} d_i^* N_{i,k}^*(t) \equiv \sum_{i=0}^{n} d_i N_{i,k}(t) = X(t)$$

in cui:

$$T^* : t_i^* = \begin{cases} t_i, & i = 0, ..., r \\ t^*, & i = r+1 \\ t_{i-1}, & i = r+2, ..., n+\mu+k \end{cases}$$

Definendo:

$$\alpha_{L,i} = \begin{cases} \dfrac{t^* - t_i}{t_{i+k-1} - t_i}, & i = r-k+2, ..., r \\ 1, & i = r-k+1 \end{cases}$$

$$\alpha_{R,i} = \begin{cases} \dfrac{t_{i+k-1} - t^*}{t_{i+k-1} - t_i}, & i = r-k+1, ..., r-1 \\ 1, & i = r \end{cases}$$

si posso scrivere i nuovi $N^*_{i,k}$ a partire da quelli vecchi:

$$N^*_{i,k}(t) = \begin{cases} N_{i,k}(t) & i = 0, ..., r-k \\ \alpha_{L,i} N_{i,k}(t) + \alpha_{R,i+1} N_{i+1,k}(t) & i = r-k+1, ..., r \\ N_{i+1,k}(t) & i = r+1, ..., n+1 \end{cases}$$

Espletando parte dell'equazione della nuova curva si ottiene:

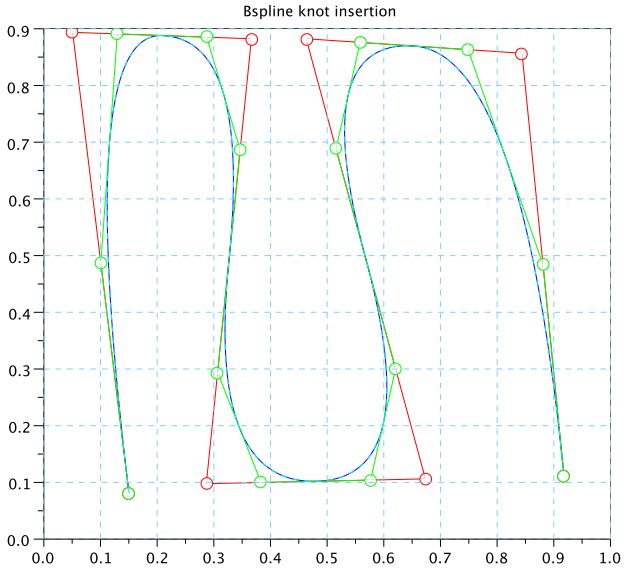
$$\begin{aligned} X(t) &= \sum_{i=0}^{n} d_i N_{i,k}(t) = \sum_{i=0}^{n+1} d^*_i N^*_{i,k}(t) \\ &= \sum_{i=0}^{r-k} d^*_i N_{i,k}(t) + \sum_{i=r-k+1}^{r} d^*_i (\alpha_{L,i} N_{i,k}(t) + \alpha_{R,i+1} N_{i+1,k}(t)) + \sum_{i=r+1}^{n+1} d^*_i N_{i+1,k}(t) \\ &= \sum_{i=0}^{r-k+1} d_i N_{i,k}(t) + \sum_{i=r-k+2}^{r} (d_i \alpha_{L,i} + d_{i-1} \alpha_{R,i}) N_{i,k}(t) + \sum_{i=r+1}^{n+1} d_{i-1} N_{i,k}(t) \end{aligned}$$

da cui si possono ricavare direttamente i d^*_i:

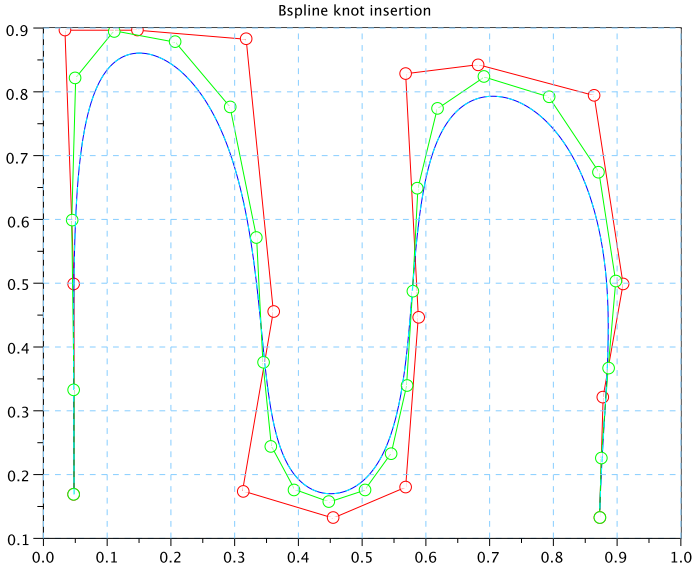
$$d^*_i = \begin{cases} d_i, & i = 0, ..., r-k+1 \\ \alpha_{L,i} d_i + \alpha_{R,i+1} d_{i-1} & i = r-k+2, ..., r \\ d_{i-1} & i = r+1, ..., n+1 \end{cases}$$

Algoritmo di knot insertion

```
function [nb,nT] = Bspline_knotins(k,T,b,p)
[d,n] = size(b);
lT = length(T);


intervals=[]
for i=1:(lT-1)
      if T(i)<T(i+1)
         intervals=[intervals,[T(i);i]]
      end
end
intervals=[intervals,[T(lT);min(find(T==T(lT)))]];

r = intervals(2,max(find(intervals(1,:)<=p)));

nT = [T(1:r),p,T(r+1:lT)];

nb = b(:,1:r-k+1);

for i=r-k+2:r
      a = (p-T(i))/(T(i+k-1)-T(i));
      nb = [nb,a*b(:,i)+(1-a)*b(:,i-1)];
end

nb = [nb,b(:,r:n)];


endfunction
```



(a) k=3



(b) k=8

Figura 3.13: Knot insertion:i due esempi precedenti di curve, con una operazione di knot insertion eseguita per ogni intervallo.

Curve NURBS

Le curve NURBS (Non Uniform Rational B-SPLINEs) sono un'estensione delle B-SPLINEs analoga alla razionalizzazione delle curve di Bézier.
La curva NURBS è definita come:

$$X(t) = \frac{\sum_{i=0}^{n} \beta_i d_i N_{i,k}(t)}{\sum_{i=0}^{n} \beta_i N_{i,k}(t)}, \quad t \in [t_{k-1}, t_{n+1}], \quad T = \{t_0, ..., t_{k-1}, ..., t_{n+1}, ..., t_{n+k}\}$$

Per comodità si indica con $w(t) = \sum_{i=0}^{n} \beta_i N_{i,k}(t)$ e di conseguenza con $R_{i,k}(t) = \frac{\beta_i N_{i,k}(t)}{w(t)}$; così l'espressione della curva diventa:

$$X(t) = \sum_{i=0}^{n} d_i R_{i,k}(t)$$

Algoritmo per il calcolo delle curve NURBS

```
function Curve = Bspline_rational(k,T,b,beta,t)
[d,n] = size(b)
n2 = length(beta)
B=[beta]
for i=1:d //Aumenta di una dimensione
     B = [B;beta.*b(i,:)]
end
C = Bspline_deboor(k,T,B,t)
for i=1:d
Curve(i,:) = C(i+1,:)./C(1,:) //Proietta
end
endfunction
```

Proprietà

Le proprietà delle curve NURBS derivano direttamente da quelle delle B-SPLINEs, se $\beta_i > 0 \; \forall \; i$.

Influenza dei coefficenti

Per studiare l'influenza dei coefficienti si può calcolare cosa succede quando uno di questi cambia. Si definisce:

$$\bar{\beta}_i = \begin{cases} \beta_j + \Delta\beta_j, & i = j \\ \beta_i, & \text{altrimenti} \end{cases}$$

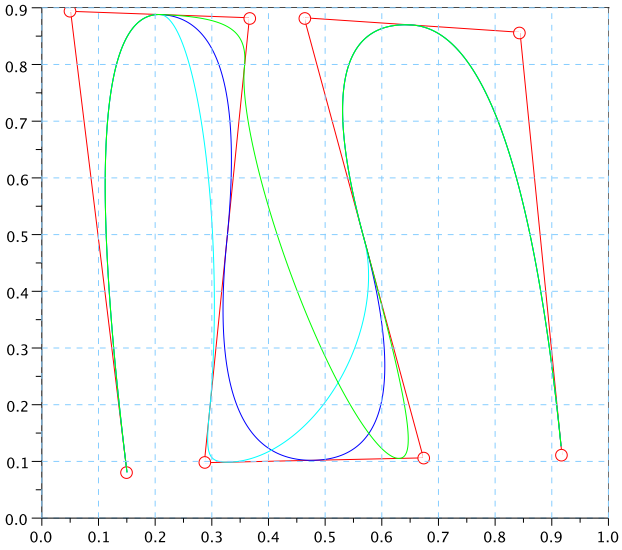
e di conseguenza:

$$\bar{w}(t) = \sum_{i=0}^{n} \bar{\beta}_i N_{i,k}(t) = w(t) + \Delta\beta_j N_{j,k}(t)$$

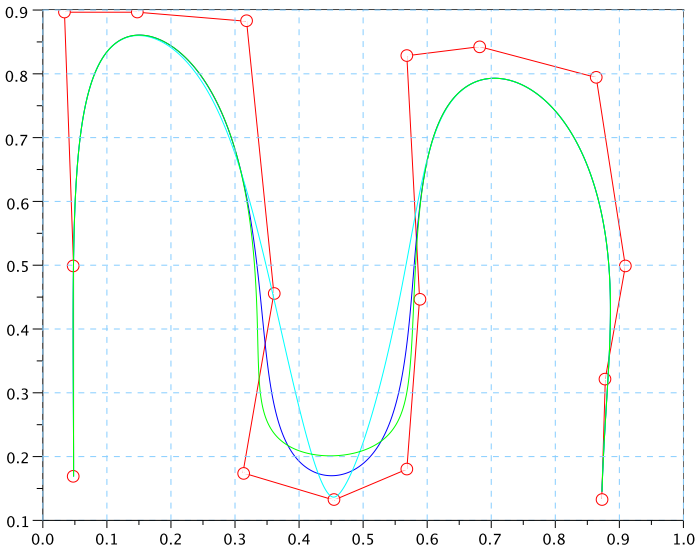
La nuova curva differisce dalla nuova di:

$$\left(\bar{X}(t) - X(t)\right) = \frac{\Delta\beta_j N_{j,k}(t) - X(t)\Delta\beta_j N_{j,k}(t)}{\bar{w}(t)} = \frac{\Delta\beta_j N_{j,k}(t)(d_j - X(t))}{\bar{w}(t)}$$

ovvero l'effetto è limitato dalla proprietà di località di $N_{j,k}(t)$, ha verso $(d_j - X(t))$ (cioè è in direzione del nodo interessato dal cambio di peso) e ha ampiezza $\Delta\beta_j$.



(a) k=3



(b) k=8

Figura 3.14: NURBS: i due esempi precedenti di curve, con il peso relativo ad un punto modificato in direzioni opposte.

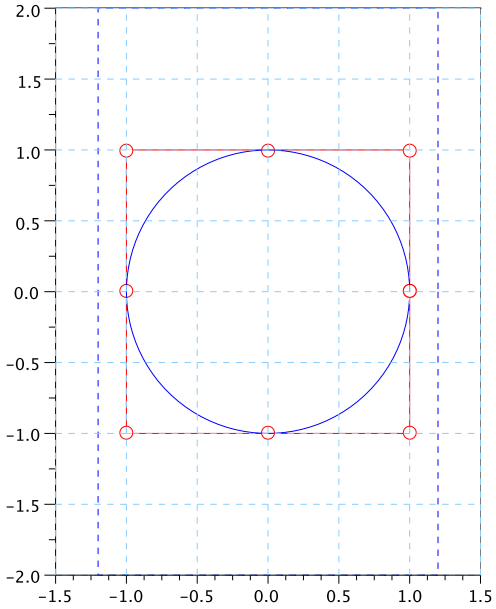


Figura 3.15: Disegno di una circonferenza con una curva NURBS

Capitolo 4

Interpolazione

usando curve parametriche

Dato un'insieme di coppie di valori del tipo:

$$(x_i, f_i), \quad x_0 < ... < x_n, \quad i = 0, ..., n$$

tali che[†] $x_i \in \mathbb{R}^k$ e $f_i \in \mathbb{E}^d$, il problema dell'interpolazione consiste nel trovare una curva $f(t)$ (tipicamente una funzione polinomiale: $f(t) \in \Pi_n(t)$) ed una parametrizzazione $t = g(x_i)$ per cui la funzione interpoli i dati forniti, ovvero:

$$f(g(x_i)) = f_i, \quad i = 0, ..., n$$

e scelte in modo che siano compatibili:

$$\begin{cases} f : [a, b] \to \mathbb{E}^d \\ g : [x_0, x_n] \to [a, b] \end{cases}$$

Parametrizzazioni

Supponendo di vole usare la base di Lagrange[‡] per definire un polinomio interpolante ai dati, si avrebbe un'espressione del tipo:

$$p(x) = \sum_{i=0}^{n} f_i L_{i,n}(x) = \sum_{i=0}^{n} f_i \prod_{\substack{j \neq i \\ j=1}}^{n} \frac{x - x_j}{x_i - x_j} \tag{4.1}$$

[†] In queste note si considerano le funzioni monovariate, ovvero $x_i \in \mathbb{R}$

[‡] $L_{i,n}(x) = \prod_{\substack{j \neq i \\ j=1}}^{n} \frac{x - x_j}{x_i - x_j}$

Da qui, per passare dal polinomio (4.1), strettamente legato agli x_i, a una curva parametrica in t su un intervallo $[a, b]$ arbitrario che sia interpolante gli stessi punti, si definisce:

$$X(t) = \sum_{i=0}^{n} P_i L_{i,n}(t)$$

per la quale i punti di controllo P_i coincidono con i punti da interpolare f_i, e i parametri t_i corrispondenti dipendono dalla parametrizzazione utilizzata.
Alcuni schemi di parametrizzazione sono:

- **Uniforme:** Si sceglie la parametrizzazione in modo che i t_i corrispondenti ai punti di interpolazione siano equispaziati nel dominio $[a, b]$ del parametro, ovvero:

$$t_i = a + \frac{i(b-a)}{n}, \quad i = 0, ..., n$$

Parametrizzazione uniforme

```
function u=l_par_unif(p)
      [d,n] = size(p)
      u=1/(n-1)*(0:n-1)
endfunction
```

- **Chord Length:** Si sceglie la parametrizzazione in modo che i t_i siano distanziati in maniera proporzionale alla distanza dei punti, così da migliorare la corrispondenza tra la posizione dei punti in $[a, b]$ e la loro immagine sulla curva:

$$\begin{cases} t_0 & = a \\ t_{i+1} & = t_i + (b-a)\dfrac{|P_{i+1} - P_i|}{\sum_{j=0}^{n} |P_{j+1-Pj}|} \end{cases}$$

Parametrizzazione Chord Length

```
function u=l_par_cl(p)
      [d,n] = size(p)
      u=[]
      df = sum(abs(p(:,2:n)-p(:,1:n-1)),1)
      df = df/sum(df)
      u(1)=0;
      for i=2:n
         u(i) = u(i-1)+(df(i-1))
      end
      u=(u/u(n))'
endfunction
```

- **Centripeta:** La parametrizzazione centripeta è definita in modo simile a quella Chord length:

$$\begin{cases} t_0 & = a \\ t_{i+1} & = t_i + (b-a)\dfrac{|P_{i+1} - P_i|^\alpha}{\sum_{j=0}^{n} |P_{j+1-Pj}|^\alpha}, \quad \alpha = \frac{1}{2} \end{cases}$$

Parametrizzazione Centripeta

```
function u=l_par_centrip(p)
        [d,n] = size(p)
        u=[]
        df = sum(abs(p(:,2:n)-p(:,1:n-1)),1)
        df = df.^(0.5)/sum(df)^(0.5)
        u(1)=0;
        for i=2:n
            u(i) = u(i-1)+(df(i-1))
        end
        u=(u/u(n))'
endfunction
```

In generale questi stessi schemi di parametrizzazione si possono applicare nel dominio delle curve SPLINEs per segmentare il dominio della curva nei suoi sottodomini.

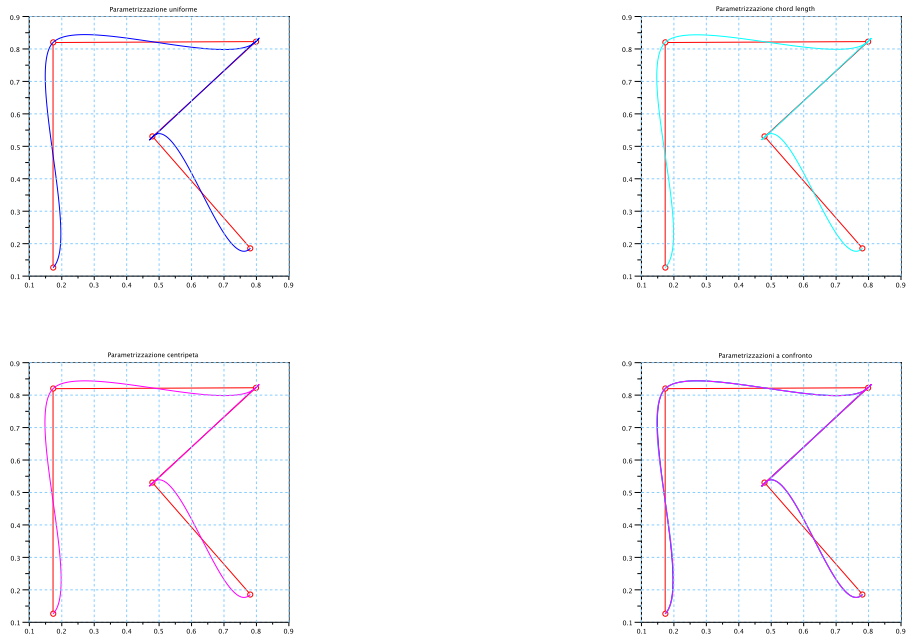


Figura 4.1: Parametrizzazioni a confronto

Interpolazione mediante SPLINE cubica $\mathcal{C}^1$ alla Hermite

Si supponga di avere a disposizione i dati da interpolare in forma di triple:

$$(t_i, f_i, f_i'), \quad i = 0, ..., n$$

in cui f_i rappresenta il valora da interpolare e f_i' la tangente alla curva nel punto di interpolazione; si cerca una funzione spline $f(t) \in \Pi_n$ che interpoli i dati in questo modo:

$$\begin{cases} f(t_i) & = f_i \\ f'(t_i) & = f_i' \end{cases}$$

Ricapitolando, dati:

$$(u_i, P_i, w_i), \quad a = u_0 < ... < u_n = b$$

si può costruire una curva spline di terzo grado con continuità $\mathcal{C}^1$ interpolante a partire dalla base di Bernstein. In particolare, si cerca una curva:

$$X(u) = X_i(u), \quad u \in [u_i, u_{i+1}]$$
$$X_i(u) = \sum_{k=0}^{3} b_{k,i} B_k^3 \left(\frac{u - u_i}{u_{i+1} - u_i} \right)$$

per la quale valgano:

$$\begin{cases} X_i(u_i) & = P_i \\ \dfrac{dX_i(u_i)}{du} & = w_i \\ X_i(u_{i+1}) & = P_{i+1} \\ \dfrac{dX_i(u_{i+1})}{du} & = w_{i+1} \end{cases}$$

Dalle proprietà dei polinomi di Bernstein segue direttamente che per le condizioni agli estremi si devono imporre:

$$\begin{cases} X_i(u_i) & = X_i(0) = b_{0,i} = P_i \\ X_i(u_{i+1}) & = X_i(1) = b_{3,i} = P_{i+1} \end{cases}$$

mentre per le derivate:

$$\begin{cases} \dfrac{dX_i(u_i)}{du} & = \dfrac{1}{u_{i+1} - u_i} \dfrac{dX_i(0)}{du} = \dfrac{3}{u_{i+1} - u_i} (b_{1,i} - b_{0,i}) = w_i \\ \dfrac{dX_i(u_{i+1})}{du} & = \dfrac{1}{u_{i+1} - u_i} \dfrac{dX_i(1)}{du} = \dfrac{3}{u_{i+1} - u_i} (b_{3,i} - b_{2,i}) = w_{i+1} \end{cases}$$

da cui si ricavano direttamente:

$$\begin{cases} b_{1,i} = \dfrac{u_{i+1} - u_i}{3} w_i + b_{0,i} \\ \\ b_{2,i} = -\dfrac{u_{i+1} - u_i}{3} w_{i+1} + b_{3,i} \end{cases}$$

Spline cubiche

```
function [Curve,hodo,hodo2]=Interp_c1(p,w,tparam,npoints)
        [d,n] = size(p)
        Curve=[]
        hodo=[]
        hodo2=[]
        dp=diff(tparam)
        for i=1:n-1
            b=[]
            b(:,1)=p(:,i)
            b(:,2)=dp(i)*w(:,i)/3+b(:,1)
            b(:,4)=p(:,i+1)
            b(:,3)=-dp(i)*w(:,i+1)/3+b(:,4)
            tab = [0:dp(i)/npoints:1]
            Curve=[Curve,Bezier_casteljau(b,tab)]
            hodo=[hodo,1/dp(i)*Bezier_casteljau(Bezier_hodograph(b),tab)]
            hodo2=[hodo2,1/dp(i)^2*Bezier_casteljau(Bezier_hodograph(Bezier_hodograph(b)),tab
                )]
        end
endfunction
```

Schemi di approssimazione delle derivate

Se non si hanno a disposizione triplette di dati ma soltanto coppie (u_i, f_i), si possono definire degli schemi per approssimare i valori f_i' ed utilizzare comunque un approccio alla Hermite per l'interpolazione.

Schema FMILL Scegliendo di approssimare le derivate in questo modo:

$$w_i = \frac{P_{i+1} - P_{i-1}}{h_{i-1} + h_i}, \quad i = 1, ..., n$$
$$w_0 = \frac{P_1 - P_0}{h_0}, w_n = \frac{P_n - P_{n-1}}{h_{n-1}}$$

in cui $h_i = u_{i+1} - u_i$, si ha un'approssimazione al prim'ordine della curva. Infatti, se $F(u)$ fosse la funzione che ha originato le coppie di interpolazione (u_i, f_i), si avrebbe:

$$P_{i+1} = F(u_{i+1}) = F(u_i) + h_i F'(u_i) + \frac{1}{2} h_i^2 F''(\xi)$$
$$P_{i-1} = F(u_{i-1}) = F(u_i) + h_{i-1} F'(u_i) + \frac{1}{2} h_{i-1}^2 F''(\zeta)$$
$$P_{i+1} - P_{i-1} = (h_{i-1} + h_i) F'(u_i) + O(h^2)$$
$$\frac{P_{i+1} - P_{i-1}}{h_{i-1} + h_i} = F'(u_i) + O(h^2)$$

Schema FMILL

```
function w=l_tang_fmill(p,u)
        [d,n] = size(p)
        h=diff(u)
        w = [(p(:,2)-p(:,1))/h(1)]
        for i=2:n-1
            w = [w,(p(:,i+1)-p(:,i-1))/(h(i-1)+h(i))]
        end
        w = [w,(p(:,n)-p(:,n-1))/h(n-1)]
endfunction
```

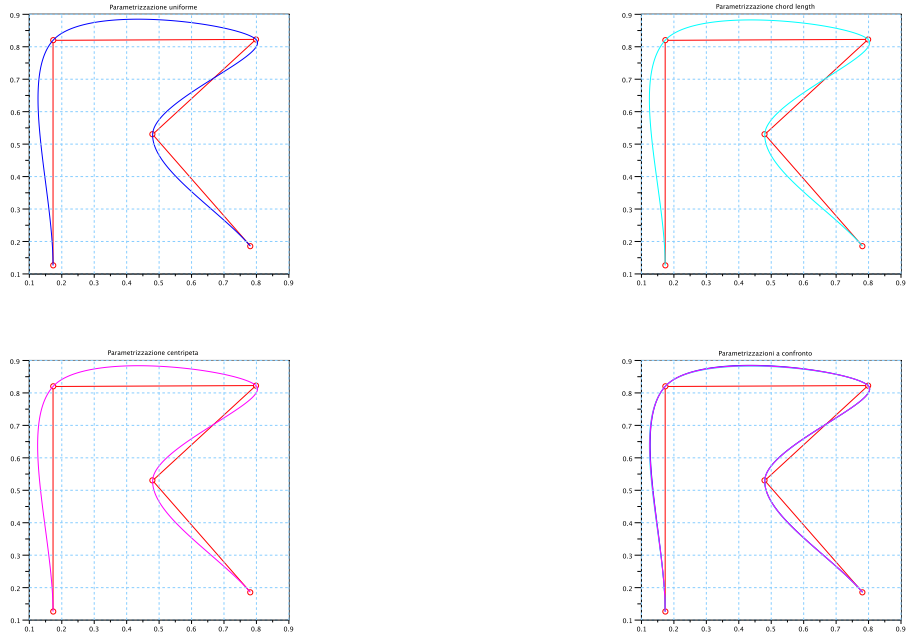


Figura 4.2: Schema FMILL

Tangenti di Bessel L'idea alla base delle tangenti di Bessel è quella di considerare la derivata di un polinomio di secondo grado interpolante tre punti consecutivi come terzo valore per il polinomio interpolante di terzo grado.
Siano $q_i(u)$ curve polinomiali a tratti in Π_2 tali che:

$$q_i(u_j) = P_j, \quad j = i-1, i, i+1 \quad u \in [u_{i-1}, u_{i+1}]$$

definite in questo modo:

$$q_i(u) = \sum_{k=0}^{2} b_{k,i} B_k^2 \left(\frac{u - u_{i-1}}{u_{i+1} - u_{i-1}} \right)$$

definendo $t(u) = \left(\frac{u - u_{i-1}}{u_{i+1} - u_{i-1}}\right)$ la precente può essere scritta:

$$q_i(u_i) = \sum_{k=0}^{3} b_{k,i} B_k^2(t_{u_i}) = P_{i-1}B_0^2(t_{u_i}) + b_{1,1}B_1^2(t_{u_i}) + P_{i+1}B_2^2(t_{u_i})$$
$$= P_{i-1}(1 - t_{u_i})^2 + 2b_{1,i}t_{u_i}(1 - t_{u_i}) + P_{i+1}t_{u_i}^2 =_{def} P_i$$

da cui:

$$b_{1,i} = \frac{P_i - P_{i-1}(1 - t_{u_i})^2 - P_{i+1}t_{u_i}^2}{2t_{u_i}(1 - t_{u_i})}$$

Mantenendo la notazione $h_i = u_{i+1} - u_i$, si ha che:

$$t_{u_i} = \frac{h_{i-1}}{h_{i-1} + h_i}, \quad 1 - t_{u_i} = \frac{h_i}{h_{i-1} + h_i}$$

e si può riscrivere l'equazione precedente in questa forma:

$$b_{1,i} = \frac{1}{2}\left(\frac{P_i(h_{i-1} + h_i)^2}{h_{i-1}h_i} - \frac{P_{i-1}h_i}{h_{i-1}} - \frac{P_{i+1}h_{i-1}}{h_i}\right)$$

Ora che i polinomi q_i sono definiti è possibile ricavare i valori w_i per i punti di interpolazione in questo modo:

$$w_i = \frac{dq_i(u_i)}{du} = \frac{1}{h_{i-1} + h_i}\frac{dq_i(t_i)}{dt} = \frac{h_i\frac{P_i - P_{i-1}}{h_{i-1}} + h_{i-1}\frac{P_{i+1} - P_i}{h_i}}{h_{i-1} + h_i}$$

Agli estremi si fissano:

$$w_0 = \frac{2(P_1 - P_0)}{h_0 + h_1} \quad w_n = \frac{2(P_n - P_{n-1})}{h_n + h_{n-1}}$$

Questo schema fornisce un'approssimazione della derivata della curva, infatti:

$$P_i = F(u_i)$$
$$P_{i-1} = P_i - F'(u_i)h_{i-1} + \frac{1}{2}h_{i-1}^2F''(u_i) + O(h_{i-1}^3)$$
$$P_{i+1} = P_i + F'(u_i)h_i + \frac{1}{2}h_i^2F''(u_i) + O(h_i^3)$$

da cui:

$$\frac{P_{i+1} - P_i}{h_i} = F'(u_i) + \frac{1}{2}h_iF''(u_i) + O(h_i^2)$$
$$\frac{P_i - P_{i-1}}{h_{i-1}} = F'(u_i) - \frac{1}{2}h_{i-1}F''(u_i) + O(h_{i-1}^2)$$

ed infine:

$$w_i = F'(u_i) + O(h^2)$$

Schema di Bessel

```
function w=l_tang_bessel(p,u)
        [d,n] = size(p)
        h=diff(u)
        w = [2*(p(:,2)-p(:,1))/h(1)+h(2)]
        for i=2:n-1
            w = [w,((p(:,i)-p(:,i-1))*h(i)/h(i-1)+(p(:,i+1)-p(:,i))*h(i-1)/h(i))/(h(i-1)+h(i)
                )]
        end
        w = [w,2*(p(:,n)-p(:,n-1))/h(n-1)+h(n-2)]
endfunction
```

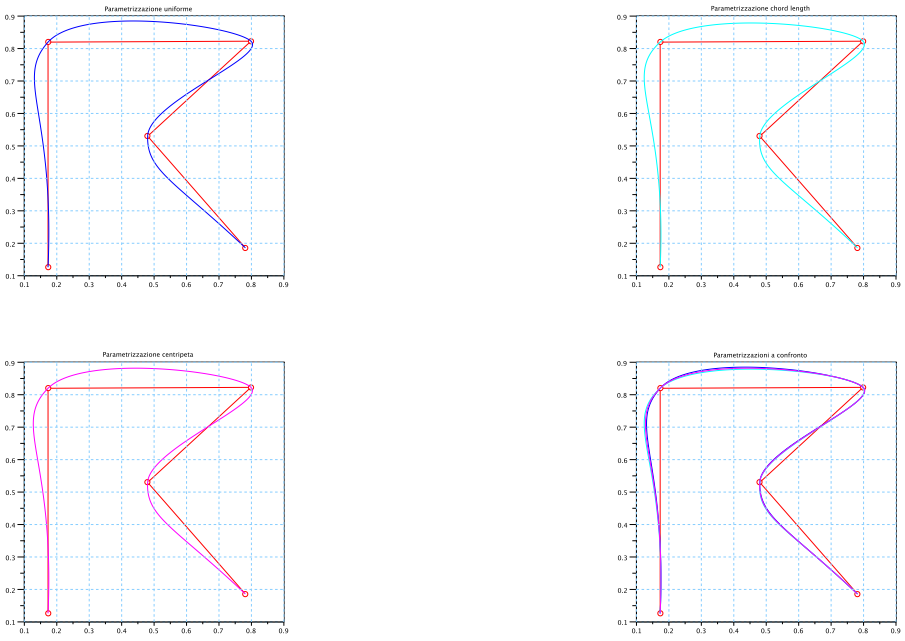


Figura 4.3: Schema di Bessel

Interpolazione mediante SPLINE cubica $\mathcal{C}^2$

Dati:

$$a = u_0 < ... < u_n = b, \quad P_i \in \mathbb{E}^d, \quad i = 0, ..., n$$

si vuole definire una spline $\mathcal{C}^2$ composta dalle curve:

$$X_i(u)^{\ddagger} = \sum_{k=0}^{3} b_{k,i} B_k^3(t(u)), \quad t(u) = \left(\frac{u - u_i}{u_{i+1} - u_i} \right)^{\dagger}$$

per le quali valgono le condizioni:

$$\begin{aligned} X_i(u_i) &= X_{i-1}(u_i) = P_i \\ \frac{dX_i(u_i)}{du} &= \frac{dX_{i-1}(u_i)}{du} \\ \frac{d^2X_i(u_i)}{d^2u} &= \frac{d^2X_{i-1}(u_i)}{d^2u} \end{aligned}$$

Dalle condizioni per le continuità $\mathcal{C}^0$ e $\mathcal{C}^1$ deriva direttamente che i cofficienti devono valere, come ricavato in precedenza:

$$\begin{aligned} b_{0,i} &= P_i \\ b_{1,i} &= P_i + \frac{h_i}{3} w_i \\ b_{2,i} &= P_{i+1} - \frac{h_i}{3} w_{i+1} \\ b_{3,i} &= P_{i+1} \end{aligned}$$

Scelta delle tangenti w_i

Dalle condizioni di continuità della derivata seconda $\frac{d^2X_i(u_i)}{d^2u} = \frac{d^2X_{i-1}(u_i)}{d^2u}$, segue che:

$$\frac{1}{h_i^2} \frac{d^2X_i(0)}{d^2t} = \frac{1}{h_{i-1}^2} \frac{d^2X_{i-1}(1)}{d^2t}$$

ovvero:

$$\frac{1}{h_i^2} \left(b_{2,i} - 2b_{1,i} + b_{0,i}\right) = \frac{1}{h_{i-1}^2} \left(b_{3,i-1} - 2b_{2,i-1} + b_{1,i-1}\right)$$

da cui:

$$\begin{aligned} &\frac{1}{h_i^2} \left[\left(P_{i+1} - \frac{h_i}{3} w_{i+1} \right) - 2 \left(P_i + \frac{h_i}{3} w_i \right) + (P_i) \right] \\ &= \frac{1}{h_{i-1}^2} \left[(P_i) - 2 \left(P_i - \frac{h_{i-1}}{3} w_i \right) + \left(P_{i-1} + \frac{h_{i-1}}{3} w_{i-1} \right) \right] \end{aligned}$$

[†] per brevità si pone $u_{i+1} - u_i = h_i$

[‡] per comodità di notazione talvolta si usa direttamente $X_i(t) = \sum_{k=0}^{3} b_{k,i} B_k^3(t)$, $t \in [0,1]$; l'espressione usata è di volta in volta determinata dal nome del parametro, u oppure t.

la quale può essere manipolata, moltiplicando per $3h_ih_{i-1}$, fino a questa forma:

$$h_iw_{i-1} + 2(h_{i-1} + h_i)w_i + h_{i-1}w_{i+1} = \underbrace{3\left[h_{i-1}\frac{P_{i+1} - P_i}{h_i} + h_i\frac{P_i - P_{i-1}}{h_{i-1}}\right]}_{T_i}$$

I T_n sono tutti termini noti, e si ha un sistema di $n-1$[†] equazioni in $n+1$[‡] incognite, che può essere scritto nella forma:

$$A\mathbf{w} = \mathbf{T} \tag{4.2}$$

in cui:

$$\mathbf{w} = (w_0, ..., w_n)^T, \quad \mathbf{T} = (T_1, ..., T_{n-1})^T$$

ed A è una matrice in $\mathcal{M}^{(n-1)\times(n-1)}$ ad elementi:

$$A = \begin{pmatrix} h_1 & 2(h_0+h_1) & h_0 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_1+h_2) & h_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n-1} & 2(h_{n-2}+h_{n-1}) & h_{n-2} \end{pmatrix}$$

Rimangono da fissare i valori delle derivate agli estrimi w_0 e w_n. Alcune tra le scelte più comuni sono le seguenti:

- **Spline naturale:** scegliendo di avere derivata nulla agli estremi, ovvero:

  $$\frac{d^2X_0(0)}{d^2t} = \frac{d^2X_{n-1}(1)}{d^2t} = 0$$

  si ha, per $i = 0$:

  $$\left(P_1 - \frac{h_0}{3}w_1\right) - 2\left(P_0 + \frac{h_0}{3}w_0\right) + P_0 = 0$$

  da cui:

  $$2h_0w_0 + h_0w_1 = 3(P_1 - P_0)$$

  Per la derivata all'ultimo punto il ragionamento è analogo e si ha:

  $$h_{n-1}w_{n-1} + 2h_{n-1}w_n = 3(P_n - P_{n-1})$$

  La matrice A ed il vettore $\mathbf{T}$ dell'equazione (4.2) possono essere allargati per comprendere i valori agli estremi:

  $$A = \begin{pmatrix} 2h_0 & h_0 & 0 & \cdots & \cdots & 0 \\ h_1 & 2(h_0+h_1) & h_0 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_1+h_2) & h_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & h_{n-1} & 2(h_{n-2}+h_{n-1}) & h_{n-2} \\ 0 & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{pmatrix}$$

[†] considerando n intervalli

[‡] tutti i w_i

$$\mathbf{T} = (3(P_1 - P_0), T_1, ..., T_{n-1}, 3(P_n - P_{n-1}))^T$$

Spline naturali

```
function [w,A,T]=l_tang_c2_nat(p,u)
      [d,n] = size(p)
      h=diff(u)
      A = []
      T = []
      A(1,:) = [2*h(1),h(1),zeros(1,n-2)]
      T(:,1) = 3*(p(:,2)-p(:,1))
      for i=2:n-1
      A(i,:)=[zeros(1,i-2),h(i),2*(h(i)+h(i-1)),h(i-1),zeros(1,n-i-1)]
      T(:,i) = 3*((h(i-1)/h(i))*(p(:,i+1)-p(:,i))+(h(i)/h(i-1))*(p(:,i)-p(:,i-1)))
      end
      A(n,:) = [zeros(1,n-2),h(n-1),2*h(n-1)]
      T(:,n) = 3*(p(:,n)-p(:,n-1))
      w=(A\T')'
endfunction
```

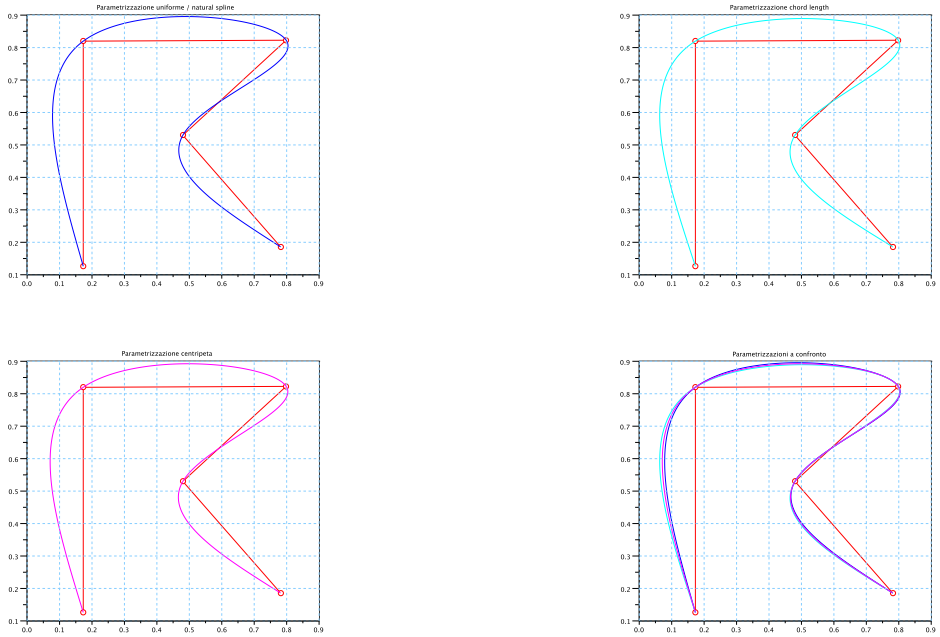


Figura 4.4: Spline $\mathcal{C}^2$ naturale

- **Spline not a knot:** una possibile scelta di w_0 e w_n meno arbitraria si può ottenere

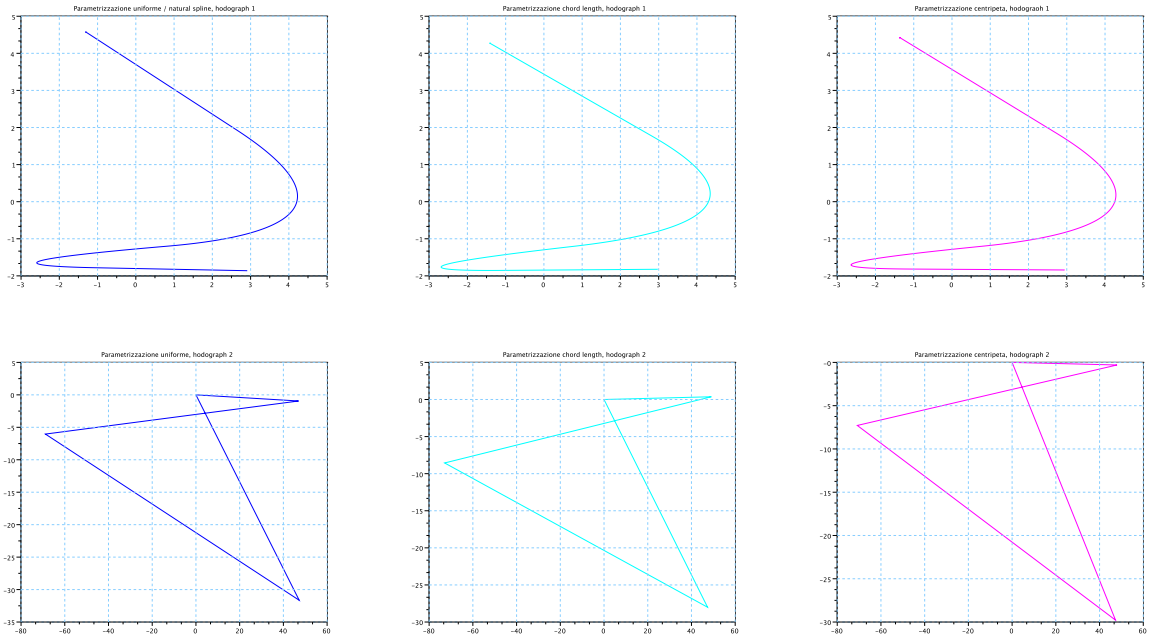


Figura 4.5: Spline $\mathcal{C}^2$ naturale, curve hodograph

forzando due condizioni in più sulle derivate terze agli estremi:

$$\frac{d^3X_0(u_1)}{d^3t} = \frac{d^3X_1(u_1)}{d^3t} \rightarrow \frac{1}{h_0^3}\frac{d^3X_0(1)}{d^3t} = \frac{1}{h_1^3}\frac{d^3X_1(0)}{d^3t}$$
$$\frac{d^3X_{n-2}(u_{n-1})}{d^3t} = \frac{d^3X_{n-1}(u_{n-1})}{d^3t} \rightarrow \frac{1}{h_{n-2}^3}\frac{d^3X_{n-2}(1)}{d^3t} = \frac{1}{h_{n-1}^3}\frac{d^3X_{n-1}(0)}{d^3t}$$

Svolgendo (la prima equazione, la seconda è analoga) si ottiene:

$$\frac{1}{h_0^3}(b_{3,0} - 3b_{2,0} + 3b_{1,0} - b_{0,0}) = \frac{1}{h_1^3}(b_{3,1} - 3b_{2,1} + 3b_{1,1} - b_{0,1})$$
$$\frac{1}{h_0^3}(2(P_0 - P_1) + h_0w_0 + h_0w_1) = \frac{1}{h_1^3}(2(P_1 - P_2) + h_1w_2 + h_1w_1)$$
$$h_1^2w_0 + (h_1^2 - h_0^2)w_1 - h_0^2w_2 = \frac{2h_0^2}{h_1}(P_1 - P_2) - \frac{2h_1^2}{h_0}(P_0 - P_1)$$

Quindi, dando un nome breve per comodità alle parti note delle equazioni che abbiano trovato:

$$\beta = \frac{2h_0^2}{h_1}(P_1 - P_2) - \frac{2h_1^2}{h_0}(P_0 - P_1)$$
$$\gamma = \frac{2h_{n-2}^2}{h_{n-1}}(P_{n-1} - P_n) - \frac{2h_{n-1}^2}{h_{n-2}}(P_{n-2} - P_{n-1})$$

la matrice A ed il vettore $\mathbf{T}$ dell'equazione (4.2) diventano:

$$A = \begin{pmatrix} h_1^2 & (h_1^2 - h_0^2) & -h_0^2 & \cdots & \cdots & 0 \\ h_1 & 2(h_0 + h_1) & h_0 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_1 + h_2) & h_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} \\ 0 & \cdots & \cdots & h_{n-1}^2 & (h_{n-1}^2 - h_{n-2}^2) & -h_{n-2}^2 \end{pmatrix}$$

$$\mathbf{T} = (\beta, T_1, ..., T_{n-1}, \gamma)^T$$

Spline not-a-knot

```
function [w,A,T]=l_tang_c2_nak(p,u)
      [d,n] = size(p)
      h=diff(u)
      A = []
      T = []
      A(1,:) = [h(2)^2,(h(2)^2-h(1)^2),-h(1)^2,zeros(1,n-3)]
      T(:,1) = (2*h(1)^2/h(2)*(p(:,2)-p(:,3))-2*h(2)^2/h(1)*(p(:,1)-p(:,2)))
      for i=2:n-1
      A(i,:)=[zeros(1,i-2),h(i),2*(h(i)+h(i-1)),h(i-1),zeros(1,n-i-1)]
      T(:,i) = 3*((h(i-1)/h(i))*(p(:,i+1)-p(:,i))+(h(i)/h(i-1))*(p(:,i)-p(:,i-1)))
      end
      A(n,:) = [zeros(1,n-3),h(n-1)^2,(h(n-1)^2-h(n-2)^2),-h(n-2)^2]
      T(:,n) = 2*h(n-2)^2/h(n-1)*(p(:,n-1)-p(:,n))-2*h(n-1)^2/h(n-2)*(p(:,n-2)-p(:,n
          -1))
      w=(A\T')'
endfunction
```

- **Spline periodica:** Per ottenere una curva periodica (ovvero che sia affiancabile), si impongono le condizioni:

$$w_0 = w_n, \quad \frac{1}{h_0^2}\frac{d^2X_0(0)}{d^2t} = \frac{1}{h_{n-1}^2}\frac{d^2X_{n-1}(1)}{d^2t}$$

da cui:

$$\frac{1}{h_0^2}\left[\left(P_1 - \frac{h_0}{3}w_1\right) - 2\left(P_0 + \frac{h_0}{3}w_0\right) + P_0\right] = \frac{1}{h_{n-1}^2}\left[P_n - 2\left(P_n + \frac{h_{n-1}}{3}w_n\right) + \left(P_{n-1} - \frac{h_{n-1}}{3}w_{n-1}\right)\right]$$

$$h_0 w_{n-1} + 2(h_0 + h_{n-1})w_0 + h_{n-1}w_1 = 3\left[h_{n-1}\frac{P_1 - P_0}{h_0} + h_0\frac{P_n - P_{n-1}}{h_{n-1}}\right]$$

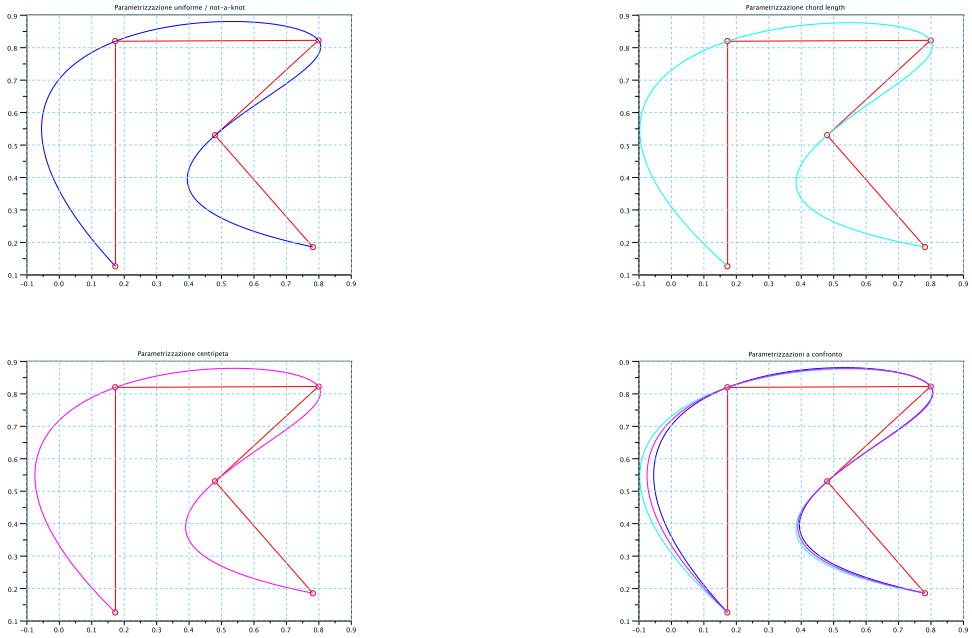


Figura 4.6: Spline $\mathcal{C}^2$ not-a-knot

Chiamando per comodità la parte nota:

$$\beta = 3\left[h_{n-1}\frac{P_1 - P_0}{h_0} + h_0\frac{P_n - P_{n-1}}{h_{n-1}}\right]$$

la matrice A e i vettori $\mathbf{T}$,$\mathbf{w}$ dell'equazione (4.2) diventano:

$$A = \begin{pmatrix} 2(h_0 + h_{n-1}) & h_{n-1} & 0 & \cdots & h_0 & 0 \\ h_1 & 2(h_0 + h_1) & h_0 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_1 + h_2) & h_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} \\ 0 & h_{n-1} & \cdots & 0 & h_0 & 2(h_{n-1} + h_0) \end{pmatrix}$$

$$\mathbf{w} = (w_0, ..., w_{n-1})^T$$

$$\mathbf{T} = (\beta, T_1, ..., T_{n-1}, \beta)^T$$

in cui la condizione $w_0 = w_n$ è imposta implicitamente nella prima e ultima riga della matrice.

Spline periodica

```
1 function [w,A,T]=I_tang_c2_periodic(p,u)
```

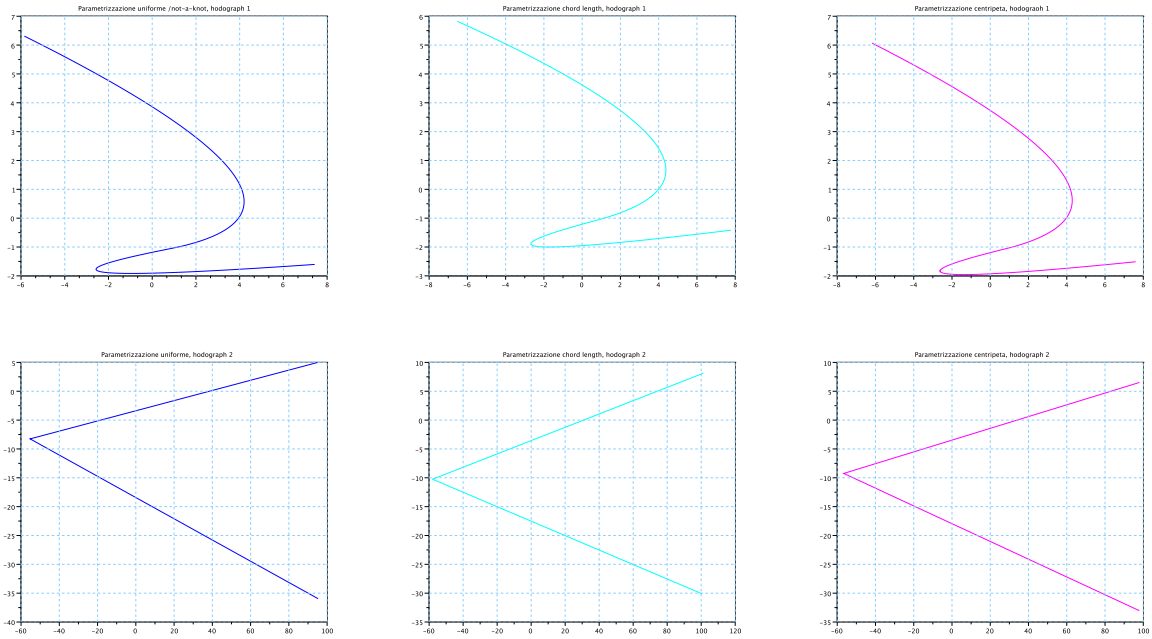


Figura 4.7: Spline $\mathcal{C}^2$ not-a-knot, curve hodograph

```
        [d,n] = size(p)
        h=diff(u)
        A = []
        T = []
        A(1,:) = [2*(h(1)+h(n-1)),h(n-1),zeros(1,n-4),h(1),0]
        T(:,1) = 3*(h(n-1)*((p(:,2)-p(:,1))/h(1))+h(1)*((p(:,n)-p(:,n-1))/h(n-1)))
        for i=2:n-1
        A(i,:)=[zeros(1,i-2),h(i),2*(h(i)+h(i-1)),h(i-1),zeros(1,n-i-1)]
        T(:,i) = 3*((h(i-1)/h(i))*(p(:,i+1)-p(:,i))+(h(i)/h(i-1))*(p(:,i)-p(:,i-1)))
        end
        A(n,:) = [0,h(n-1),zeros(1,n-4),h(1),2*(h(n-1)+h(1))]
        T(:,n)=T(:,1)
        w=(A\T')'
endfunction
```

Considerazioni

Sia $s(u)$ una spline cubica $\mathcal{C}^2$ tale che:

$$s(u_i) = f_i, \quad i = 0, ..., n$$

e $y(u)$ una funzione qualsiasi $\mathcal{C}^2$ a sua volta tale che:

$$y(u_i) = f_i, \quad i = 0, ..., n$$

La quantità:

$$\int_{u_0}^{u_n} \ddot{s}^2(u) du$$

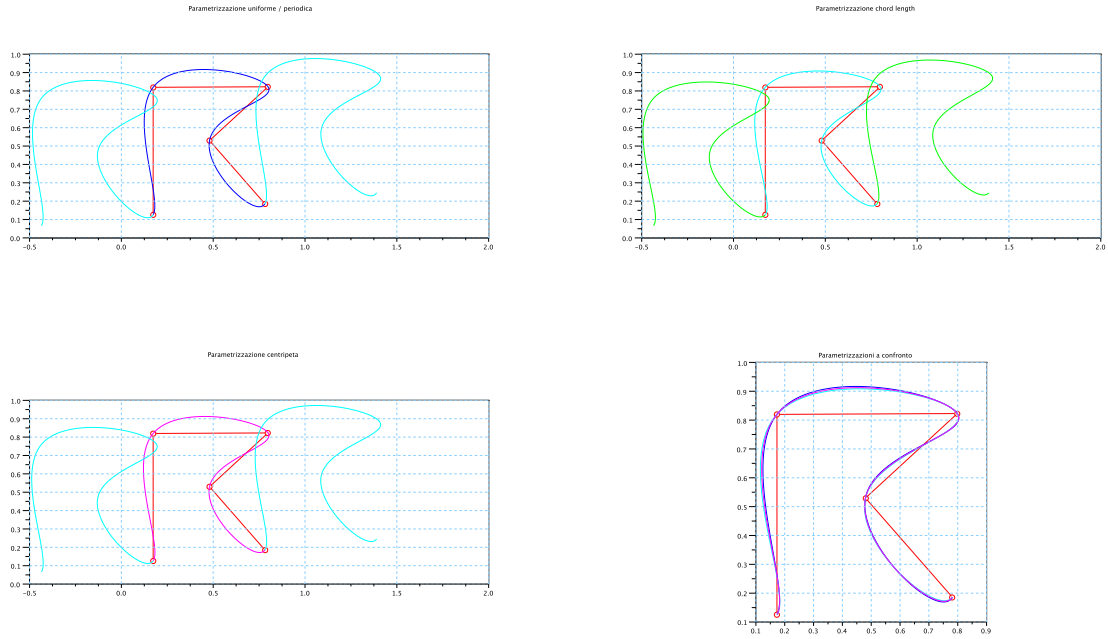


Figura 4.8: Spline $\mathcal{C}^2$ periodica

è una possibile misura del grado di oscillazione della curva.
Si dimostra che vale la relazione:

$$\int_{u_0}^{u_n} \ddot{s}^2(u)du \leq \int_{u_0}^{u_n} \ddot{y}^2(u)du$$

Infatti:

$$\begin{aligned}
\ddot{y}(u) &= (\ddot{y}(u) - \ddot{s}(u)) + \ddot{s}(u) \\
\ddot{y}^2(u) &= \ddot{s}^2(u) + (\ddot{y}(u) - \ddot{s}(u))^2 + 2\ddot{s}(u)(\ddot{y}(u) - \ddot{s}(u)) \\
\int_{u_0}^{u_n} \ddot{y}^2(u)du &= \int_{u_0}^{u_n} \ddot{s}^2(u)du + \int_{u_0}^{u_n} (\ddot{y}(u) - \ddot{s}(u))^2 du + 2\int_{u_0}^{u_n} \ddot{s}(u)(\ddot{y}(u) - \ddot{s}(u))du
\end{aligned}$$

integrando per parti l'ultimo integrale della sommatoria si ha:

$$\int_{u_0}^{u_n} \ddot{s}(u)(\ddot{y}(u) - \ddot{s}(u))du = \underbrace{\ddot{s}(u)(\dot{y}(u) - \dot{s}(u))|_{u_0}^{u_n}}_{\alpha} - \underbrace{\int_{u_0}^{u_n} \dddot{s}(u)(\dot{y}(u) - \dot{s}(u))du}_{\beta}$$

in cui $\alpha = 0$ ed in quanto $\dddot{s}(u)$ è costante a tratti, β può essere trasformato nella somma:

$$\sum_{i=0}^{n-1} s_i \left[y(u) - s(u)\right]_{u_i}^{u_{i+1}}$$

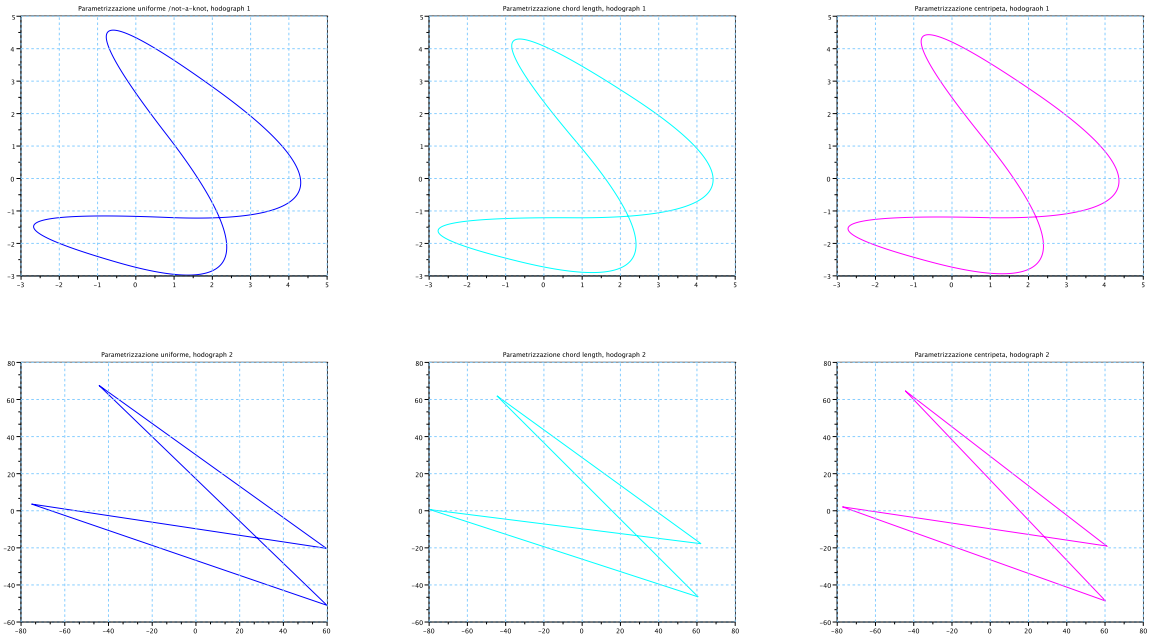


Figura 4.9: Spline $\mathcal{C}^2$ periodica, curve hodograph

che a sua volta è nulla perchè valgono le condizioni $y(u_i) = s(u_i)$.
Un'altra misura della stessa quantità può essere ottenuta dalla curvatura, considerando come parametro la lunghezza dell'arco:

$$\int_0^L k(s)ds = \int_{u_0}^{u_n} k^2 \|\dot{X}(u)\| du, \quad s(t) = \int_{u_0}^{u_n} \|\dot{X}(t)\| dt$$

Rappresentando la funzione vettorialmente:

$$X(u) = \begin{pmatrix} u \\ y(u) \end{pmatrix}, \quad X(u) = \begin{pmatrix} 1 \\ \dot{y}(u) \end{pmatrix}, \quad X(u) = \begin{pmatrix} 0 \\ \ddot{y}(u) \end{pmatrix}$$

la curvatura assume espressione:

$$k(u) = \frac{\dot{X}(u) \times \ddot{X}(u)}{\|\dot{X}(u)\|^3} = \frac{\ddot{y}(u)}{(1 + \dot{y}^2(u))^{\frac{3}{2}}}$$

da cui:

$$\int_{u_0}^{u_n} k^2(u) \|\dot{X}(u)\| du, \quad s(t) = \int_{u_0}^{u_n} \frac{\ddot{y}(u)}{(1 + \dot{y}^2(u))} du$$

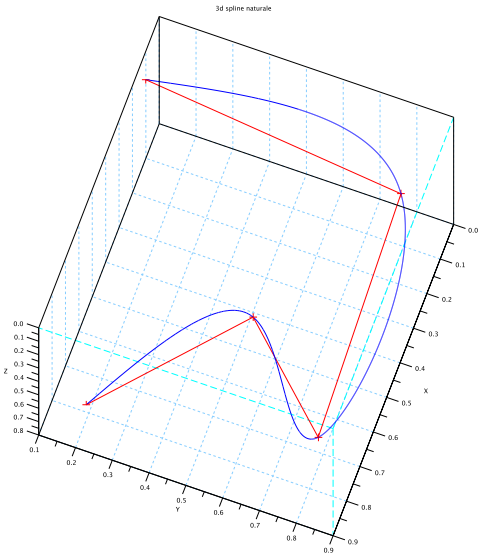


Figura 4.10: Spline $\mathcal{C}^2$ naturale 3d

Spline cubiche $\mathcal{C}^1 \cap \mathcal{G}^2$

Data una spline in forma di Bézier:

$$X(u) = X_i(u), \quad u \in [u_i, u_{i+1}]$$
$$X_i(t) = \sum_{i=0}^{3} b_{k,i} B_k^3(t), \quad t = \left(\frac{u - u_i}{h_i}\right)$$

che ha coefficienti:

$$\begin{cases} b_{0,i} &= P_i \\ b_{1,i} &= P_i + \dfrac{h_i}{3} w_i \\ b_{2,i} &= P_{i+1} - \dfrac{h_i}{3} w_{i+1} \\ b_{3,i} &= P_{i+1} \end{cases}$$

Imponendo le condizioni di raccordo, considerando $\omega_{1,i} = 1$ si ottiene:

$$\frac{d^2 X_i}{du}(u_i) - \frac{d^2 X_{i-1}}{du}(u_i) = \nu_i w_i, \quad w_i = \frac{dX_i}{du}(u_i), \quad \nu_i = \omega_{2,i}$$

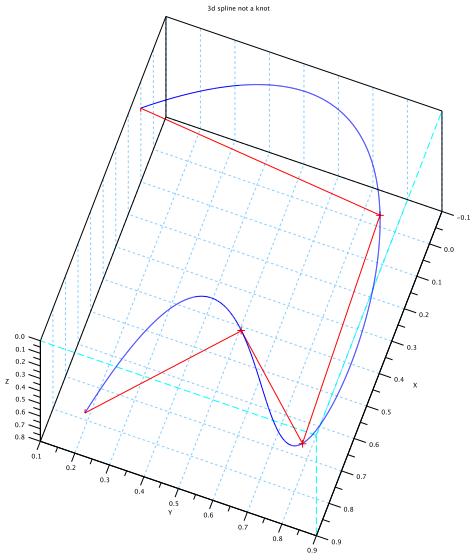


Figura 4.11: Spline $\mathcal{C}^2$ not a knot 3d

Le condizioni di raccordo permettono di scrivere infine le equazioni:

$$\frac{6}{h_i^2}\left[(P_{i+1} - \frac{h_i}{3}w_{i+1}) - 2(P_i + \frac{h_i}{3}w_i) + P_i\right] -$$
$$\frac{6}{h_{i+1}^2}\left[P_i - 2(P_i - \frac{h_{i-1}}{3}w_i) + (P_{i-1} + \frac{h_{i-1}}{3}w_{i-1})\right] - \nu_i w_i = 0$$

manipolando e moltiplicando per $-\frac{h_i h_{i-1}}{2}$ si ottiene infine:

$$h_i w_i + \left[2(h_{i-1} + h_i) + \frac{\nu_i}{2}h_i h_{i-1}\right] w_i + h_{i-1}w_{i+1} = 3\left[h_{i-1}\frac{P_{i+1} - P_i}{h_i} + h_i\frac{P_i - P_{i-1}}{h_{i-1}}\right]$$

ovvero si ottiene un sistema analogo a quello delle spline $\mathcal{C}^2$ con un termine in più sulla diagonale:

$$(A + D)\mathbf{w} = \mathbf{T}, \quad D = \begin{pmatrix} d_0 & & \\ & \ddots & \\ & & d_n \end{pmatrix}$$

in cui d_0, d_n sono assegnati e $d_i = \frac{\nu_i}{2}h_{i-1}h_i$.
Per ottenre una spline chiusa, supponendo di avere il primo punto uguale al primo ($P_0 = P_n$) si ha invece un sistema in n incognite in cui le prime e le ultime equazioni di A sono analoghe

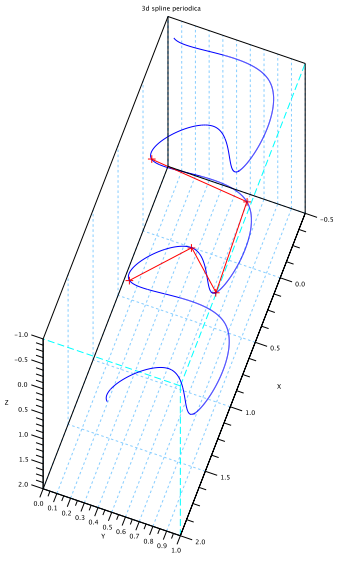


Figura 4.12: Spline $\mathcal{C}^2$ periodica 3d

a quelle per la spline periodica precedentemente ricavate, ovvero del tipo:

$$h_0 w_{n-1} + 2(h_{n-1} + h_0)w_0 + h_{n-1}w_1 = 3\left(h_{n-1}\frac{P_1 - P_0}{h_0} + h_0\frac{P_0 - P_{n-1}}{h_{n-1}}\right)$$

e inoltre

$$d_0 = d_n = \frac{\nu_0}{2}h_{n-1}h_0$$

È interessante osservare cosa succede per $\nu_{min} \to \infty$. Innanzitutto serve calcolare:

$$\begin{aligned}(A + D)\mathbf{w} &= \mathbf{T} \\ (AD^{-1} + I)(D\mathbf{w}) &= \mathbf{T} \\ (I + E)\tilde{\mathbf{w}} &= \mathbf{T}, \quad E = AD, \quad \tilde{\mathbf{w}} = D\mathbf{w}\end{aligned}$$

Se $\nu_{min} \to \infty$ anche $D \to \infty$; AD^{-1} è non singolare e per D sufficientemente grande $\|E\| < 1$. In quanto:

$$\|(I + E)^{-1}\| \leq \frac{1}{1 - \|E\|}$$

si ha che:

$$(I + E)\tilde{\mathbf{w}} \Rightarrow \|\tilde{\mathbf{w}}\| \leq \frac{1}{1 - \|E\|}\|\mathbf{T}\|$$

ovvero se $D \to \infty$, $\mathbf{w} \to 0$.

ν-spline

```
function [w,A,T]=l_tang_nu_nat(p,u,nu)
        [d,n] = size(p)
        h=diff(u)
        A = []
        T = []
        A(1,:) = [2*h(1),h(1),zeros(1,n-2)]
        T(:,1) = 3*(p(:,2)-p(:,1))
        for i=2:n-1
        A(i,:)=[zeros(1,i-2),h(i),2*(h(i)+h(i-1)),h(i-1),zeros(1,n-i-1)]
        T(:,i) = 3*((h(i-1)/h(i))*(p(:,i+1)-p(:,i))+(h(i)/h(i-1))*(p(:,i)-p(:,i-1)))
        end
        A(n,:) = [zeros(1,n-2),h(n-1),2*h(n-1)]
        T(:,n) = 3*(p(:,n)-p(:,n-1))
        A(1,1)=A(1,1)+nu(1)
        for i=2:n-1
        A(i,i)=A(i,i)+(nu(i)*h(i-1)*h(i)/2)
        end
        A(n,n)=A(1,1)+nu(n)
        w=(A\T')'
endfunction
```

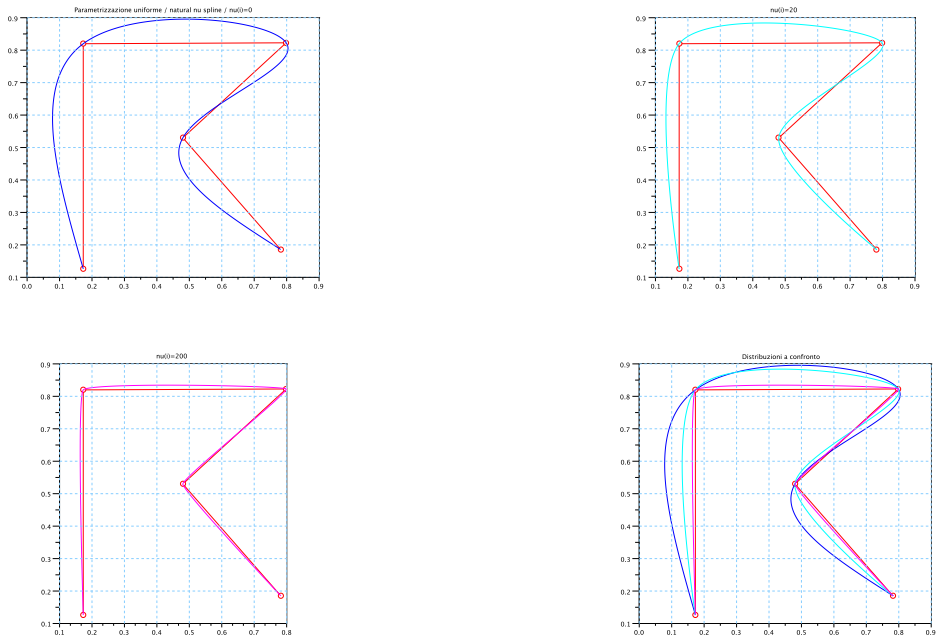


Figura 4.13: ν-spline naturale

ν-spline chiusa

```
function [w,A,T]=l_tang_nu_closed(p,u,nu)
        [d,n] = size(p)
        h=diff(u)
        A = []
        T = []
        A(1,:) = [2*(h(1)+h(n-1)),h(n-1),zeros(1,n-4),h(1),0]
```

```
7         T(:,1) = 3*(h(n-1)*((p(:,2)-p(:,1))/h(1))+h(1)*((p(:,n)-p(:,n-1))/h(n-1)))
8         for i=2:n-1
9         A(i,:)=[zeros(1,i-2),h(i),2*(h(i)+h(i-1)),h(i-1),zeros(1,n-i-1)]
10        T(:,i) = 3*((h(i-1)/h(i))*(p(:,i+1)-p(:,i))+(h(i)/h(i-1))*(p(:,i)-p(:,i-1)))
11        end
12        A(n,:) = [0,h(n-1),zeros(1,n-4),h(1),2*(h(n-1)+h(1))]
13        T(:,n)=T(:,1)
14        A(1,1)=A(1,1)+(nu(1)/2)*h(1)*h(n-1)
15        for i=2:n-1
16        A(i,i)=A(i,i)+(nu(i)*h(i-1)*h(i)/2)
17        end
18        A(n,n)=A(1,1)+(nu(1)/2)*h(1)*h(n-1)
19        w=(A\T')'
20  endfunction
```

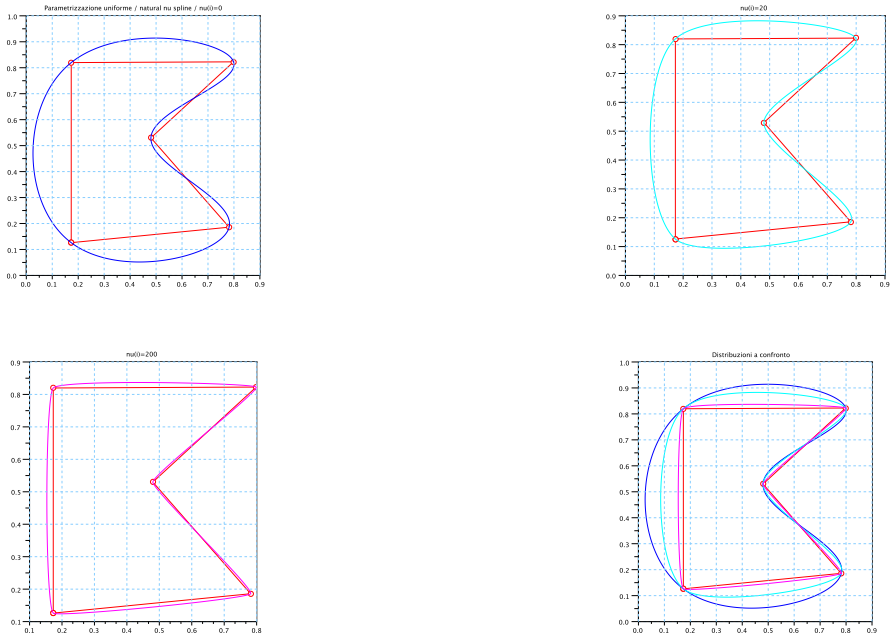


Figura 4.14: ν-spline chiusa

Capitolo 5

Superfici parametriche

Rappresentazione parametrica

Una superficie, in maniera analoga ad una curva, può essere rappresentata in modo parametrico come l'immagine di una funzione continua e iniettiva di due variabili reali $X : (A \in \mathbb{R}^2) \to \mathbb{E}^3$ tale che:

$$X(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \quad (u,v) \in \mathbb{R}^2$$

Definizione 5.1 (Superficie parametrica regolare)**.** La superficie parametrica si dice **regolare** se valgono le condizioni:

- le funzioni $x(u,v), y(u,v), z(u,v) \in \mathcal{C}^1$ ovvero sono derivabili e la derivata prima è continua
- la matrice Jacobiana† $\dfrac{\partial(x,y,z)}{\partial(u,v)} = \begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \end{bmatrix}$ abbia rango 2, ovvero le derivate non si annullano mai entrambe in uno stesso punto
- le funzioni $x(u,v), y(u,v), z(u,v)$ sono iniettive

†La matrice jacobiana è la matrice di tutte le derivate parziali prime di una funzione che ha dominio e codominio in uno spazio euclideo. Una funzione $F : \mathbb{R}^n \to \mathbb{R}^m$ è descritta in modo parametrico attraverso le sue componenti: $(F_1(x_1, ..., x_n), ..., F_m(x_1, ..., x_n))^T$ e la relativa matrice Jacobiana è definita come:

$$J_f(x_1, ..., x_m) = \frac{\partial(F_1, ..., F_m)}{\partial(x_1, ..., x_n)} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}$$

Definizione 5.2 (Vettori associati alla superficie: tangenti e normale)**.** Per comodità nello studio delle proprietà della superficie si definiscono i vettori:

$$X_u = \begin{pmatrix} \dfrac{\partial x(u,v)}{\partial u} \\ \dfrac{\partial y(u,v)}{\partial u} \\ \dfrac{\partial z(u,v)}{\partial u} \end{pmatrix}, \quad X_v = \begin{pmatrix} \dfrac{\partial x(u,v)}{\partial v} \\ \dfrac{\partial y(u,v)}{\partial v} \\ \dfrac{\partial z(u,v)}{\partial v} \end{pmatrix}$$

delle derivate parziali rispetto ai parametri, ed il vettore normale alla superficie di conseguenza:

$$N(u,v) = \frac{X_u \wedge X_v}{|X_u \wedge X_v|}$$

Osservazione 5.1 (Caratteristiche dei vettori $X_u, X_v, N(u,v)$). Se la superficie è regolare si ha che i vettori X_u e X_v sono linearmente indipendenti e non paralleli, ovvero $X_u \wedge X_v \neq 0$; di conseguenza anche $N(u,v) \neq 0 \ \forall (u,v) \in A$.

Esempi di superfici parametriche

Sfera La sfera[†] è definibile come il luogo geometrico dei punti equidistanti dall'origine:

$$x^2 + y^2 + z^2 = r$$

Una possibile parametrizzazione per i punti della superficie sferica si può dare a partire dai parametri $u \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ e $v \in [0, 2\pi]$:

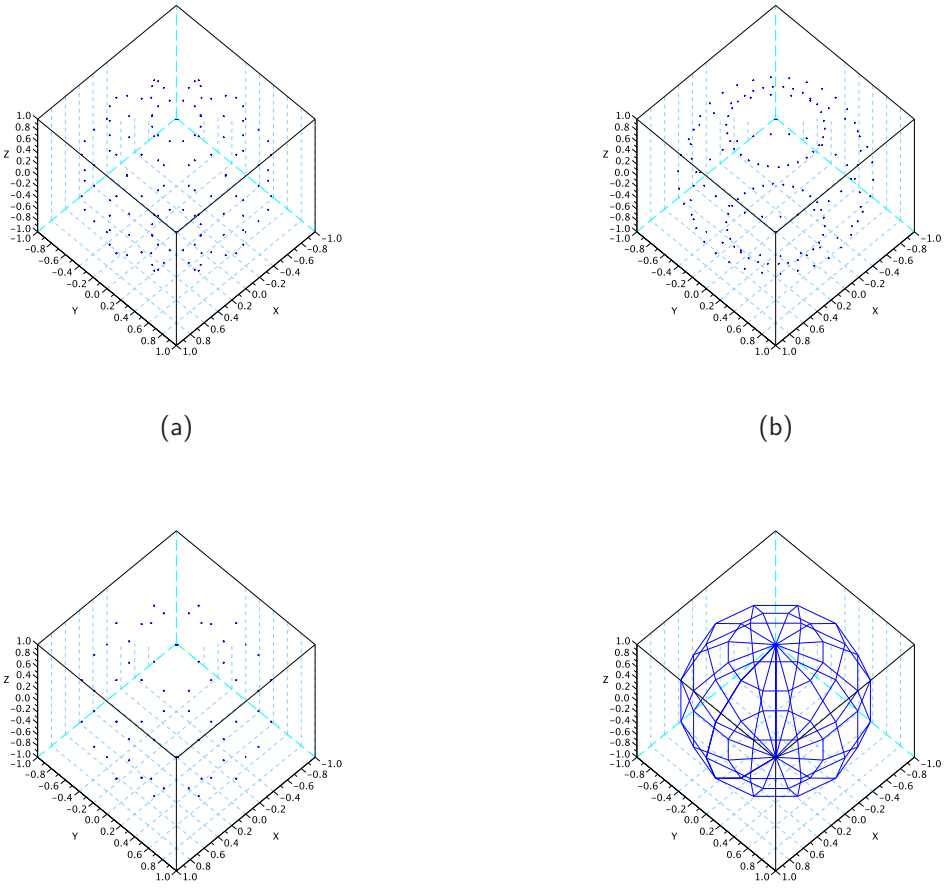
$$S(u,v) = \begin{pmatrix} \cos u \cos v \\ \cos u \sin v \\ \sin u \end{pmatrix}$$

I vettori tangenti alla superficie sono quindi:

$$S_u = \begin{pmatrix} -\sin u \cos v \\ -\sin u \sin v \\ \cos u \end{pmatrix}, \quad S_v = \begin{pmatrix} -\cos u \sin v \\ \cos u \cos v \\ 0 \end{pmatrix}$$

[†]centrata nell'origine degli assi

[‡]A causa di un bug di SCILAB non è attualmente possibile esportare figure vettoriali che contengono più di un centinaio marks, le figure riportate nel testo sono quindi in bassa risoluzione. Le stesse figure con risoluzione migliore sono generabili in qualsiasi momento lanciando lo script esempi_surf.sci allegato.



(a) (b)

(c) (d)

Figura 5.1: Esempio di sfera parametrizzata come descritto nel testo. Nella prima e seconda figura si evidenziano in risoluzione le coordinate verticali od orizzontali, in basso l'unione delle due figure con e senza linee ad unire i punti nelle due direzioni[‡].

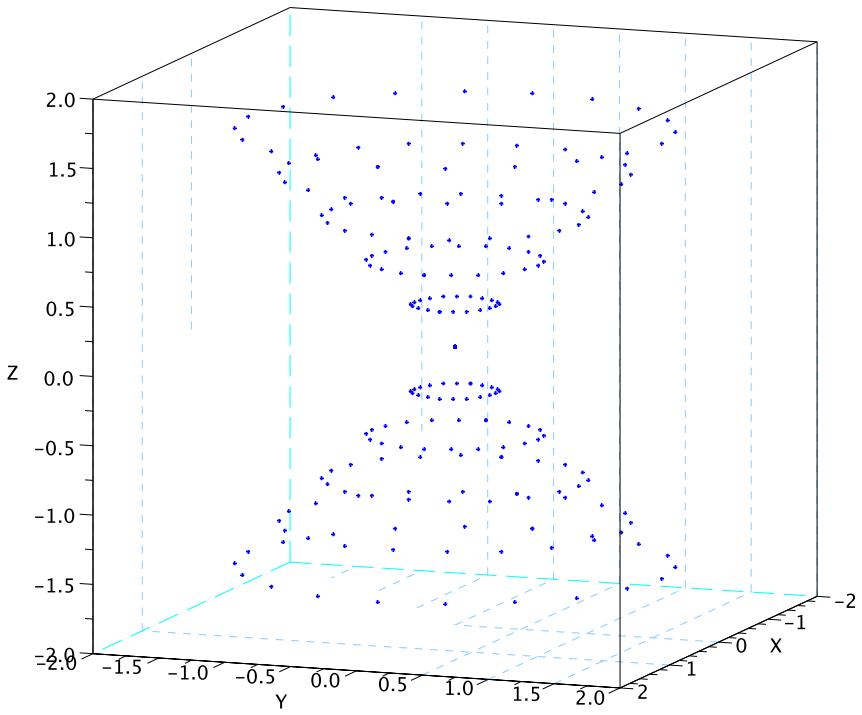


Figura 5.2: Esempio di cono parametrizzato come descritto nel testo.

Cono Un cono è lo spazio geometrico dei punti che soddisfano l'equazione:

$$z^2 = x^2 + y^2$$

Una parametrizzazione in due parametri $u \in [-L, L]$, $v \in [0, 2\pi]$ è:

$$C(u, v) = \begin{pmatrix} cu \cos v \\ cu \sin v \\ cu \end{pmatrix}$$

I vettori tangenti sono, di conseguenza:

$$C_u = \begin{pmatrix} c \cos v \\ c \sin v \\ c \end{pmatrix}, \quad C_v = \begin{pmatrix} -cu \sin v \\ cu \cos v \\ 0 \end{pmatrix}$$

Riparametrizzazione ammissibile

Si definiscono due nuovi parametri in funzione di quelli vecchi:

$$\begin{cases} \tilde{u} = \tilde{u}(u,v) \\ \tilde{v} = \tilde{v}(u,v) \end{cases}$$

Data la matrice:

$$J(u,v) = \begin{pmatrix} \dfrac{\partial \tilde{u}}{\partial u} & \dfrac{\partial \tilde{u}}{\partial v} \\ \dfrac{\partial \tilde{v}}{\partial u} & \dfrac{\partial \tilde{v}}{\partial v} \end{pmatrix}$$

la riparametrizzazione si definisce **ammissibile** se $\det J(u,v) \neq 0$. In particolare si ha che:

$$X_u \wedge X_v = \left(X_{\tilde{u}} \frac{\partial \tilde{u}}{\partial u} + X_{\tilde{v}} \frac{\partial \tilde{v}}{\partial u} \right) \wedge \left(X_{\tilde{u}} \frac{\partial \tilde{u}}{\partial v} + X_{\tilde{v}} \frac{\partial \tilde{v}}{\partial v} \right) = (X_{\tilde{u}} \wedge X_{\tilde{v}}) \underbrace{\left(\frac{\partial \tilde{u}}{\partial u} \frac{\partial \tilde{u}}{\partial v} - \frac{\partial \tilde{v}}{\partial u} \frac{\partial \tilde{v}}{\partial v} \right)}_{\det J(u,v)}$$

Tipologie di superfici

Superfici di rivoluzione

Una superficie di rivoluzione è una superficie ottenuta ruotando una curva (chiamata curva generatrice o profilo) intorno ad un asse (detto asse di rotazione). La circonferenza ottenuta interesecando un piano perpendicolare all'asse di rotazione con la superficie è detta parallelo, la curva ottenuta intersecando la superficie con un piano passate per l'asse di rotazione è detta meridiano.
In generale una superficie di rotazione è rappresentabile in modo parametrico. Sia una curva C definita parametricamente come:

$$C(u) = \begin{cases} C_x(u) \\ C_y(u) \end{cases} \quad , \quad u \in [a,b]$$

La curva C può essere fatta ruotare intorno all'asse z definendo la superficie:

$$X(u,v) = \begin{cases} C_x(u)\cos(v) \\ C_y(u)\sin(v) \\ C_x(u) \end{cases} \quad , \quad v \in [0, 2\pi]$$

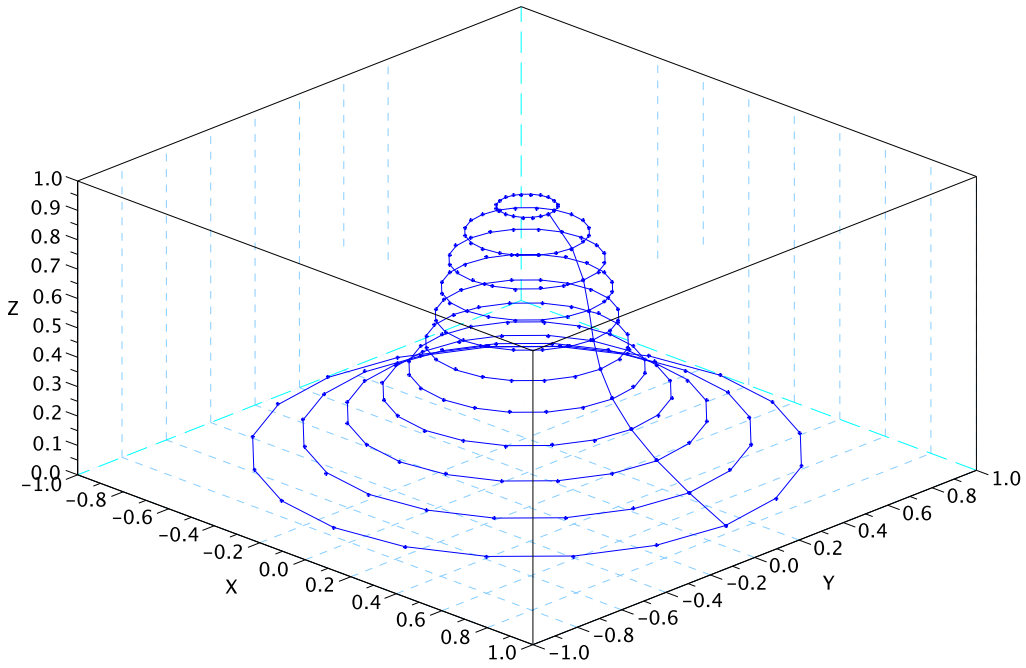


Figura 5.3: Esempio di curva di Bézier utilizzata come curva parametrica di rotazione.

Superfici Rigate

Una superficie rigata è una superficie ottenuta da un'unione di rette.

Definizione 5.3 (Superficie rigata)**.** Una superficie S si dice rigata se esiste una famiglia di rette r_α tale che S è l'unione delle rette di tale famiglia: $S = \cup_\alpha r_\alpha$.[†]

Date due curve $C_0(u), C_1(u)$ e due parametri $u \in [a,b], v \in [0,1]$ una superficie rigata può essere difinita parametricamente come:

$$S(u,v) = (1-v)C_0(u) + vC_1(u)$$

ovvero si sfrutta l'esistenza di una corrispondenza univoca tra i punti di C_0 e quelli di C_1.

Superficie bilineare

Se C_0 e C_1 sono segmenti, ovvero sono parametrizzabili come:

$$\begin{cases} C_0(u) = (1-u)p_{00} + up_{01} \\ C_1(u) = (1-u)p_{10} + up_{11} \end{cases}$$

la superficie rigata S può essere scritta come:

$$S(u,v) = (1-v)C_0(u) + vC_1(u) = (1-u, u)\begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix}\begin{pmatrix} 1-v \\ v \end{pmatrix}$$

Superfici tensor-product

Definizione 5.4 (Tensor product)**.** Date due funzioni monovariate S_1, S_2 tali che:

$$S_1 =< f_0, ..., f_n >, \quad f_0, ..., f_n \text{ funzioni monovariate in } u \in [a,b]$$
$$S_2 =< g_0, ..., g_m >, \quad g_0, ..., g_m \text{ funzioni monovariate in } v \in [c,d]$$

si definisce **tensor product** $S_1 \otimes S_2$ la funzione:

$$S_1 \otimes S_2 =< f_0g_0, ..., f_0g_m, ..., f_ng_0, ..., f_ng_m >$$

$S_1 \otimes S_2$ è pertanto una funzione bivariata, e vale:

$$S(u,v) \in S \Leftrightarrow S(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij}f_i(u)g_j(v), \quad (u,v) \in [a,b] \times [c,d]$$

Il tensor product può quindi essere utilizzato per definire superfici, considerando funzioni del tipo:

$$X(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij}f_i(u)g_j(v), \quad (u,v) \in [a,b] \times [c,d]$$

[†]Una definizione equivalente è che S è rigata se per ogni punto di S passa una retta r_s che sia tutta contenuta in essa.

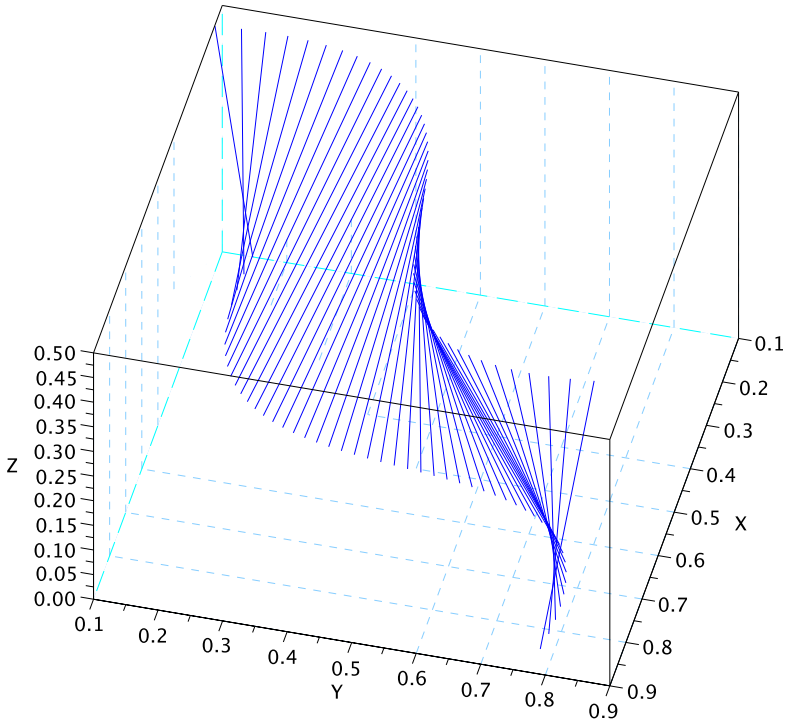


Figura 5.4: Esempio di curva di Bézier utilizzata come bordi per una superficie rigata.

Superfici tensor-product come patch Bézier

La patch di Bézier è definita a partire dalla base omonima nel modo seguente:

$$X(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij} B_i^n(u) B_j^m(v)$$

Proprietà

Curve di contorno La superficie è delimitata da quattro curve di contorno[†]:

$$\begin{cases} u = 0, v \in [0,1], & X(0,v) = \sum_{j=0}^{m} b_{0j} B_j^m(v) \\ u = 1, v \in [0,1], & X(1,v) = \sum_{j=0}^{m} b_{1j} B_j^m(v) \\ v = 0, u \in [0,1], & X(u,0) = \sum_{i=0}^{n} b_{i0} B_i^n(u) \\ v = 1, u \in [0,1], & X(u,1) = \sum_{i=0}^{n} b_{i1} B_i^n(u) \end{cases}$$

Posizione negli angoli Negli angoli la supericie è parallela al piano individuato dai punti di controllo. Infatti, considerando le derivate prime parziali della superficie lungo le due direzioni negli angoli, si ha:

$$\begin{aligned} \dot{X}_u(0,0) &= n\Delta^{(1,0)} b_{0,0} = n(b_{1,0} - b_{0,0}) \\ \dot{X}_u(1,0) &= n\sum_{i=0}^{n-1} \Delta^{(1,0)} b_{i,0} B_i^{n-1}(1) = n\Delta^{(1,0)} b_{n-1,0} = n(b_{n,0} - b_{n-1,0}) \\ \dot{X}_v(0,0) &= n\Delta^{(0,1)} b_{0,0} = m(b_{0,1} - b_{0,0}) \\ \dot{X}_v(0,1) &= m\sum_{j=0}^{m-1} \Delta^{(0,1)} b_{0,j} B_j^{m-1}(1) = m\Delta^{(0,1)} b_{0,m-1} = m(b_{0,m} - b_{0,m-1}) \end{aligned}$$

Altre proprietà In quanto somme di polinomi di Bernstein valgono le seguenti proprietà, dimostrabili in modo analogo a quanto già mostrato in modo esplicito per le curve di Bézier:

- $\sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v) = 1$
- I punti della superficie $X(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij} B_i^n(u) B_j^m(v)$ sono una combinazione convessa dei punti di controllo $b_{i,j}$
- Vale la priprità di convex hull
- Vale la proprietà di invarianza alle trasformazioni affini

[†]boundary

Forma compatta La superficie $X(u,v)$ può essere espressa in forma compatta come:

$$X(u,v) = \left(B_0^n(u), ..., B_n^n(u)\right) \begin{pmatrix} b_{0,0} & \cdots & b_{0,m} \\ \vdots & & \vdots \\ b_{n,0} & \cdots & b_{n,m} \end{pmatrix} \begin{pmatrix} B_0^m(v) \\ \vdots \\ B_m^m(v) \end{pmatrix}$$

Algoritmi

de Casteljau

L'algoritmo di de Casteljau per le superfici funziona in maniera del tutto analoga a quello descritto precedentemente per le curve.

Dati $b_{ij}^{00} = b_{ij}$ come punti iniziali per l'algoritmo $(i = 0, .., n; j = 0, ..., m)$, si definisce il passo dell'algoritmo unidirezionale (nelle due direzioni, rispettivamente) in questo modo:

$$b_{ij}^{k0}(u) = (1-u)b_{ij}^{k-1,0} + ub_{i+1,j}^{k-1,0}, \quad i = 0, ..., n-k$$
$$b_{ij}^{0k}(v) = (1-v)b_{ij}^{0,k-1} + vb_{i,j+1}^{0,k-1}, \quad j = 0, ..., m-k$$

Svolgendo contemporaneamente nelle due direzioni si ha:

$$\begin{aligned} b_{ij}^{kl}(u,v) &= (1-v)b_{ij}^{k,l-1} + vb_{i,j+1}^{k,l-1} \\ &= (1-v)\left[(1-u)b_{ij}^{k-1,l-1} + ub_{i+1,j}^{k-1,l-1}\right] + v\left[(1-u)b_{i,j+1}^{k-1,l-1} + ub_{i+1,j+1}^{k-1,l-1}\right] \end{aligned}$$

e se $m = n$,

$$b_{ij}^{kl}(u,v) = \left((1-u), u\right) \begin{pmatrix} b_{ij}^{k-1,l-1} & b_{i+1,j}^{k-1,l-1} \\ b_{i,j+1}^{k-1,l-1} & b_{i+1,j+1}^{k-1,l-1} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}, \quad i = 0, ..., n-k; j = 0, ..., m-l$$

Algoritmo per il calcolo delle patch di Bézeir

```
function Curve = Bezier_patch_casteljau(b,u,v) //b poligono, u,v punti di valutazione
Lu = length(u)
Lv = length(v)
[d,n,m] = size(b)
Curve =[]
tcurve=[]
for im=1:m
      tcurve(:,:,im) = Bezier_casteljau(b(:,:,im),v);
end
for iv=1:Lv
      Curve(:,:,iv)=Bezier_casteljau(matrix(tcurve(:,iv,:),3,-1),u)
end
endfunction
```

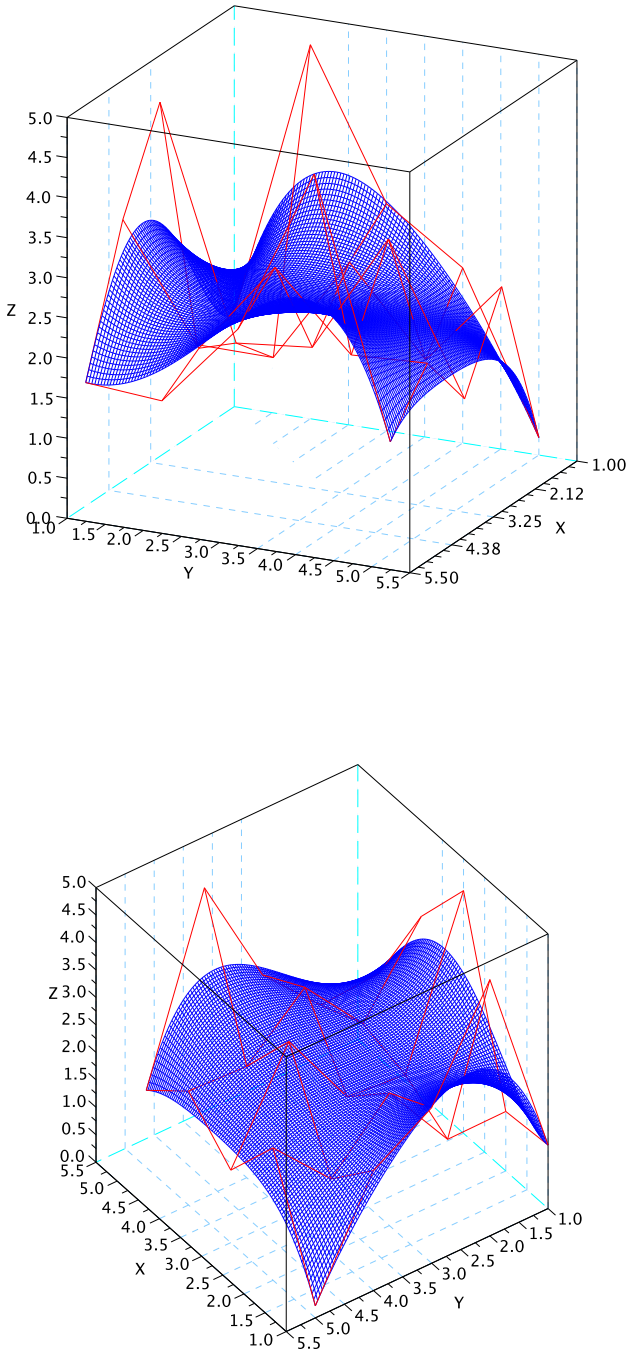


Figura 5.5: Esempio di Patch di Bézier da due angolazioni di verse.

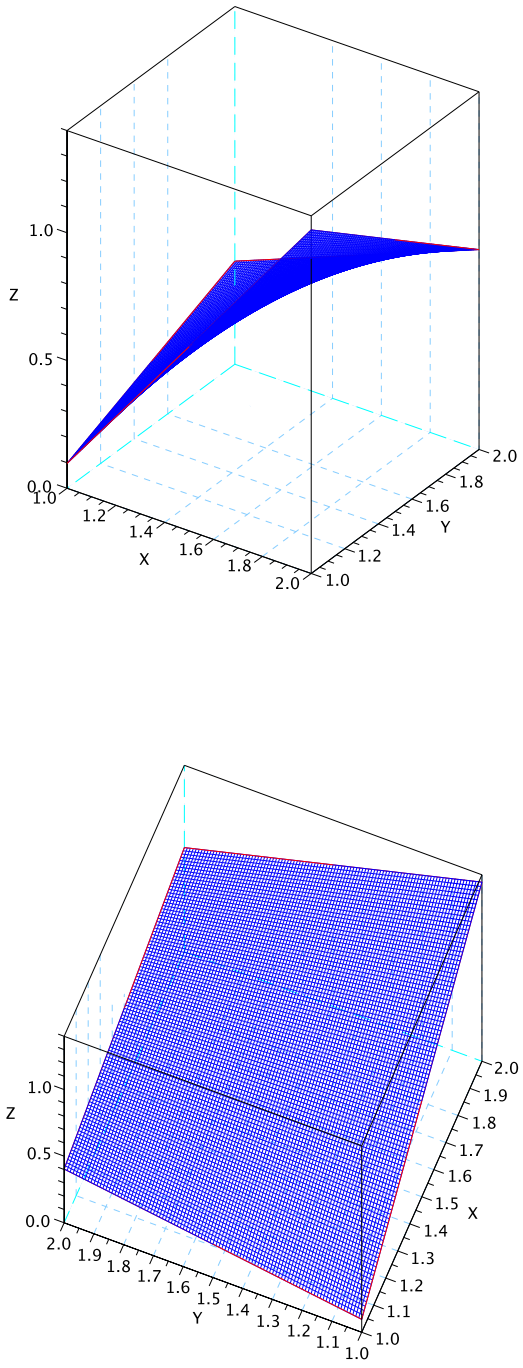


Figura 5.6: Esempio di Patch di Bézier con soli quattro punti di controllo: risalta la proprietà di tangenza agli angoli.

Degree-elevation

L'algoritmo di degree elevation consente di elevare il grado della patch di Bézier lungo una delle due direzioni aggiungendo una riga di punti di controllo e modificando quelli esistenti in modo opportuno. In particolare, data la superficie:

$$X_{n,m}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij}B_i^n(u)B_j^m(v)$$

si vuole aumentarne il grado lungo una direzione, ovvero ottenere:

$$X_{n+1,m}(u,v) = \sum_{i=0}^{n+1}\sum_{j=0}^{m} c_{ij}B_i^{n+1}(u)B_j^m(v)$$

oppure

$$X_{n,m+1}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m+1} d_{ij}B_i^n(u)B_j^{m+1}(v)$$

Per le proprietà dei polinomi di Bernstein segue direttamente che:

$$\begin{aligned} X_{n,m}(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij}B_i^n(u)B_j^m(v) \\ &= X_{n+1,m}(u,v) = \sum_{i=0}^{n+1}\sum_{j=0}^{m} c_{ij}B_i^{n+1}(u)B_j^m(v) = \sum_{j=0}^{m}\left(\sum_{i=0}^{n+1} c_{ij}B_i^{n+1}(u)\right)B_j^m(v) \end{aligned}$$

da cui:

$$\sum_{j=0}^{m}\left(\sum_{i=0}^{n} b_{ij}B_i^n(u)\right)B_j^m(v) = \sum_{j=0}^{m}\left(\sum_{i=0}^{n+1} c_{ij}B_i^{n+1}(u)\right)B_j^m(v)$$

che implica:

$$\sum_{i=0}^{n} b_{ij}B_i^n(u) = \sum_{i=0}^{n+1} c_{ij}B_i^{n+1}(u), \quad \forall j$$

I nuovi punti di controllo sono pertanto:

$$c_{ij} = \frac{n+1-i}{n+1}b_{ij} + \frac{i}{n+1}b_{i-1,j}, \quad \forall j,\ i = 0,...,n+1$$

Gli stessi ragionamenti si applicano in modo perfettamente analogo anche all'altro caso.

In generale si preferisce aumentare il grado contemporaneamente in entrambe le direzioni. Applicando lo stesso ragionamento anche nell'altra direzione si ottiene:

$$\sum_{i=0}^{n+1}\left(\sum_{j=0}^{m} c_{ij}B_j^m(v)\right)B_i^{n+1}(u) = \sum_{i=0}^{n+1}\left(\sum_{j=0}^{m+1} d_{ij}B_j^{m+1}(v)\right)B_i^{n+1}(u)$$

da cui, come prima:

$$\sum_{j=0}^{m} c_{ij} B_j^m(v) = \sum_{j=0}^{m+1} d_{ij} B_j^{m+1}(v)$$

che implica:

$$d_{ij} = \frac{m+1-j}{m+1} c_{ij} + \frac{j}{m+1} c_{i,j-1}, \quad \forall i,\ j = 0, ..., m+1$$

Esplicitando i c_{ij} si ottiene:

$$d_{ij} = \frac{m+1-j}{m+1}\left(\frac{n+1-i}{n+1} b_{ij} + \frac{i}{n+1} b_{i-1,j}\right) + \frac{j}{m+1}\left(\frac{n+1-i}{n+1} b_{i,j-1} + \frac{i}{n+1} b_{i-1,j-1}\right)$$

$$= \left(\frac{i}{n+1}, \frac{n+1-i}{n+1}\right) \begin{pmatrix} b_{i-1,j-1} & b_{i-1,j} \\ b_{i,j-1} & b_{ij} \end{pmatrix} \begin{pmatrix} \frac{j}{m+1} \\ \frac{m+1-j}{m+1} \end{pmatrix}$$

Degree elevation nelle due direzioni

```
function B = Bezier_patch_degelev(b)
[d,n,m] = size(b)
B=b

for i=1:n
 B(:,i,1:m+1)=matrix(Bezier_degelev(matrix(b(:,i,:),3,-1)),3,1,m+1)
end
b=B
for i=1:m+1
 B(:,1:n+1,i)=Bezier_degelev(b(:,:,i))
end

endfunction
```

Subdivision

L'algoritmo di subdivision lungo una delle due direzioni è analogo a quello per le curve. Data la superficie:

$$X(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} B_i^n(u) B_j^m(v)$$

la si vuole suddividere in due superfici lungo una delle due direzioni:

$$\begin{cases} X_L(\tau, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{ij} B_i^n(\tau) B_j^m(v) = X(t,v),\ t \in [0, \hat{t}],\ \tau = \dfrac{t}{\hat{t}} \in [0,1] \\ X_R(\tau, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} d_{ij} B_i^n(\tau) B_j^m(v) = X(t,v),\ t \in [\hat{t}, 1],\ \tau = \dfrac{t-\hat{t}}{1-\hat{t}} \in [0,1] \end{cases}$$

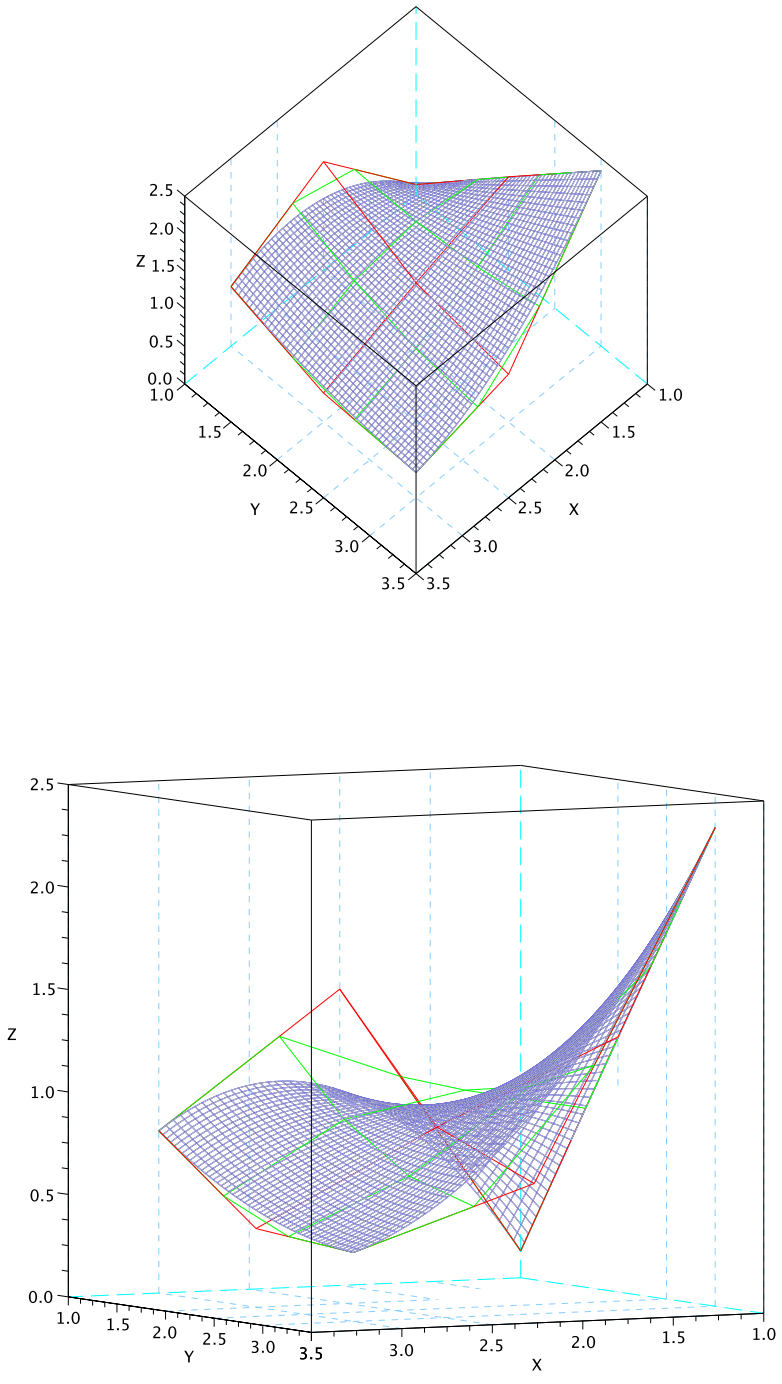


Figura 5.7: Esempio di applicazione della degree elevation, da due angolazioni di verse. In rosso il poligono di controllo originale, in verde quello di grado elevato, in grigio la sovrapposizione esatta della vecchia superficie (in blu) e della nuova (grigia).

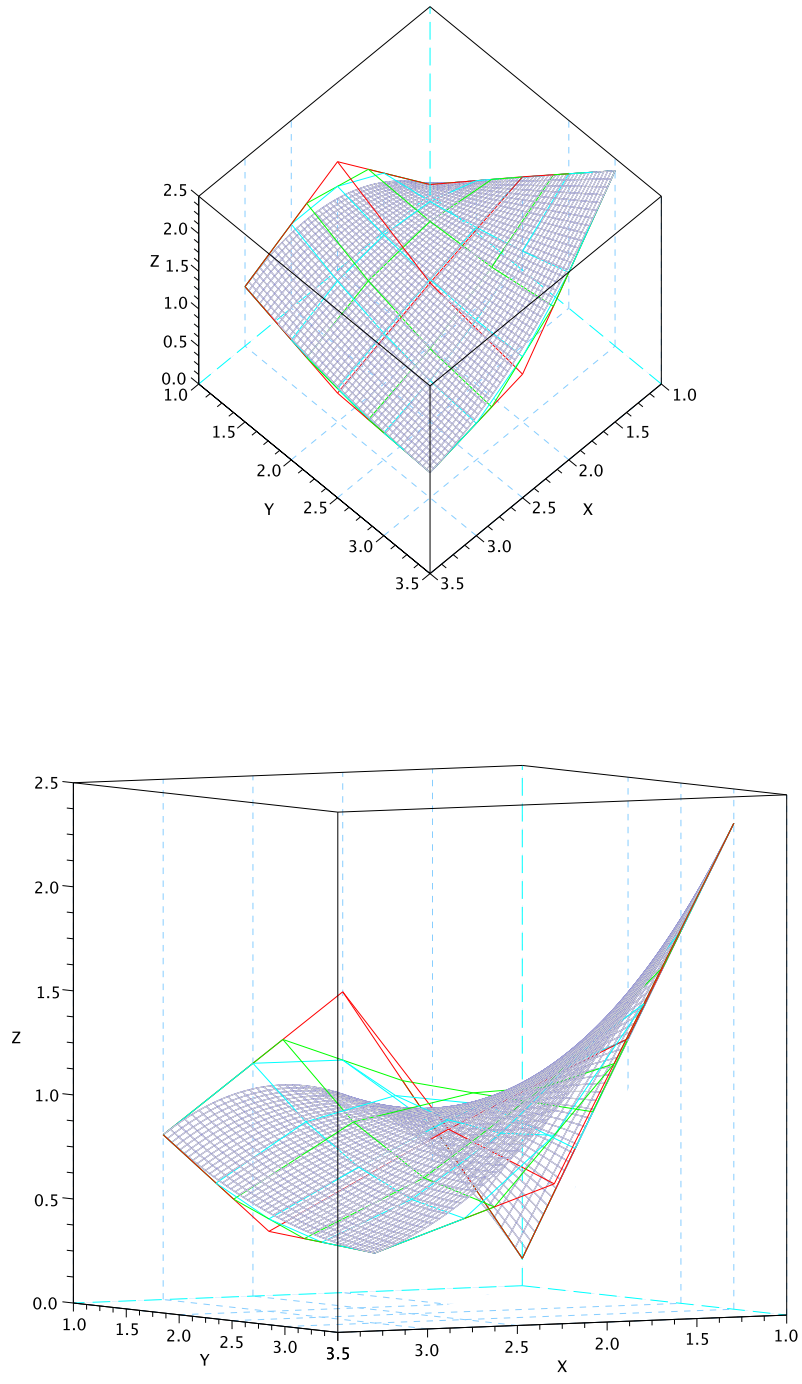


Figura 5.8: Esempio di applicazione della degree elevation, da due angolazioni di verse. In rosso il poligono di controllo originale, in verde quello di grado elevato, in ciano una seconda elevazione. In grigio la sovrapposizione esatta della vecchia superficie (in blu) e delle nuova (grigia).

oppure lungo l'altra:

$$\begin{cases} X_L(u,\tau) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{ij}B_i^n(u)B_j^m(\tau) = X(u,t),\ t\in[0,\hat{t}],\ \tau = \dfrac{t}{\hat{t}} \in [0,1] \\ X_R(u,\tau) = \sum_{i=0}^{n}\sum_{j=0}^{m} d_{ij}B_i^n(u)B_j^m(\tau) = X(u,t),\ t\in[\hat{t},1],\ \tau = \dfrac{t-\hat{t}}{1-\hat{t}} \in [0,1] \end{cases}$$

In entrambi i casi, in modo analogo al caso bidimensionale i coefficienti si ottengono a partire da quelli dell'algoritmo di de Casteljau:

$$c_{kj} = b_{0j}^{k,0}(\tau), \quad d_{kj} = b_{k,j}^{n-k,0}(\tau)$$

per la direzione u e:

$$c_{ik} = b_{i0}^{0,k}(\tau), \quad d_{ik} = b_{i,k}^{0,m-k}(\tau)$$

lungo v.

Subdivision nelle due direzioni

```
function [B1,B2,B3,B4] = Bezier_patch_subdiv(B,t)
[d,n,m] = size(B)

for i=1:n
      [a,b] = Bezier_subdiv(matrix(B(:,i,:),3,-1),t)
      B5(1:3,i,1:m)=matrix(a,3,1,m)
      B6(1:3,i,1:m)=matrix(b,3,1,m)
end

for i=1:m
      [a,b] = Bezier_subdiv(B5(:,:,i),t)
      B1(1:3,1:n,i)=a
      B2(1:3,1:n,i)=b
      [a,b] = Bezier_subdiv(B6(:,:,i),t)
      B3(1:3,1:n,i)=a
      B4(1:3,1:n,i)=b
end
endfunction
```

Interpolazione di superfici con patch di Bézier

Una superficie tensor-product di Bézier, come definita in precedenza, è una funzione del tipo:

$$X(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij}B_i^n(u)B_j^m(v), \quad (u,v)\in[0,1]\times[0,1]$$

In questo caso, definiti:

$$S_1 = < B_0^n, ..., B_n^n >, \quad S_2 = < B_0^m, ..., B_m^m >, \quad S = S_1 \otimes S_2$$

si ha che si sta lavorando su uno spazio di dimensione:

$$N = \dim S = (n+1)(m+1)$$

Il problema dell'interpolazione di superfici quindi è completamente determinanto se si hanno a disposizione N campionamenti della superficie da interpolare; i dati del problema sono rappresentabili come:

$$X_{in}(u_i, v_j) = P_k, \quad k = 0, ..., N-1, \quad P_k \in \mathbb{E}^3$$

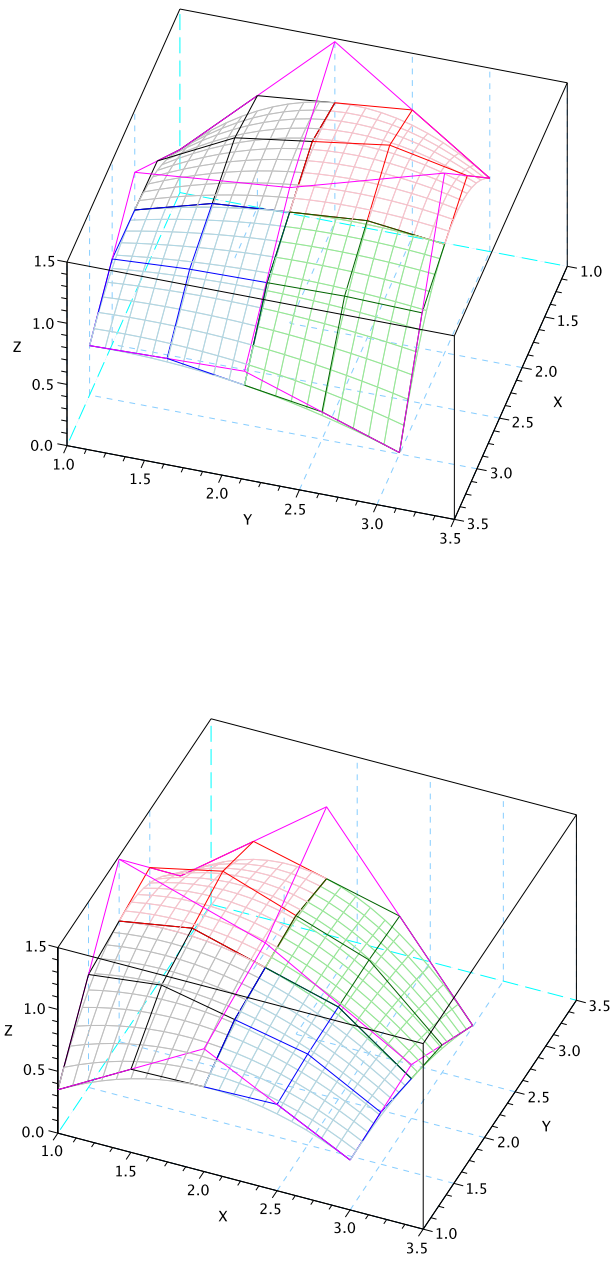


Figura 5.9: Esempio di subdivision applicata a t=0.5, vista da due angolazioni diverse. In magenta i punti di controllo originali; i nuovi punti di controllo sono colorati e le relative superfici hanno colori concordi.

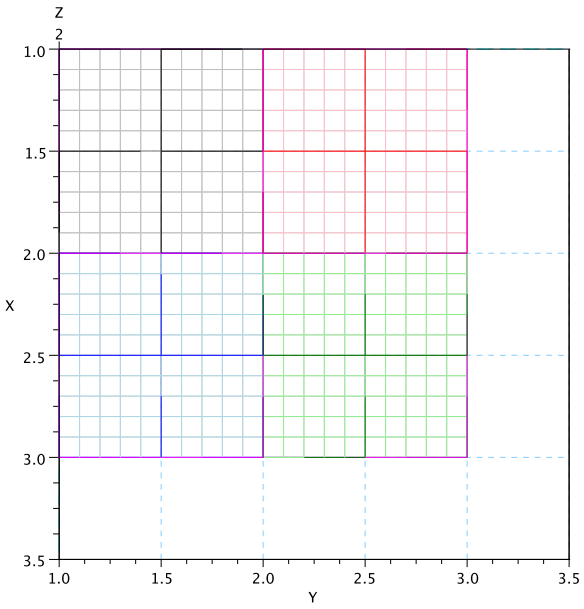


Figura 5.10: Esempio di subdivision applicata a t=0.5, vista dall'alto si nota bene come ogni quadrante sia diviso in quattro quadranti uguali.

Interpolazione su punti sparsi

Se si hanno a disposizione campionamenti sparsi della superficie che si vuole interpolare:

$$X_{in}(u_i, v_i) = P_k, \quad i = 0, ..., N;$$

si va a risolvere il sistema:

$$X(u_i, v_i) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} B_i^n(u_i) B_j^m(v_i) = X_{in}(u_i, v_i) = P_k$$

il quale può essere scritto nella forma:

$$\begin{pmatrix} B_0^n(u_1)B_0^m(v_1) & \cdots & B_n^n(u_1)B_m^m(v_1) \\ \vdots & & \vdots \\ B_0^n(u_n)B_0^m(v_n) & \cdots & B_n^n(u_n)B_m^m(v_n) \end{pmatrix} \begin{pmatrix} b_{00} \\ \vdots \\ b_{0m} \\ b_{10} \\ \vdots \\ b_{nm} \end{pmatrix} = \begin{pmatrix} P_1 \\ \vdots \\ P_k \end{pmatrix}$$

Interpolazione su griglia rettangolare

Se i campionamenti sono distribuiti su una griglia rettangolare:

$$X(u_i, v_i) = P_{ij}, \quad i = 0, ..., n; \; j = 0, ..., m$$

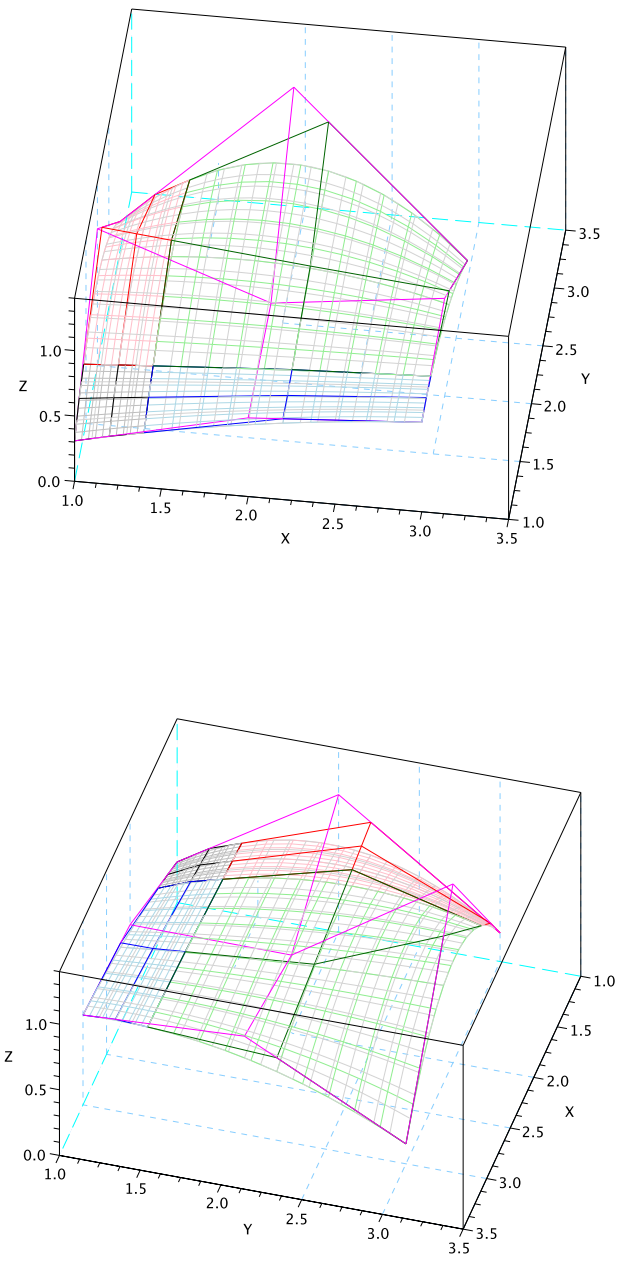


Figura 5.11: Esempio di subdivision applicata a t=0.2, vista da due angolazioni diverse. In magenta i punti di controllo originali, in grigio la superficie originale; i nuovi punti di controllo sono colorati e le relative superfici hanno colori concordi.

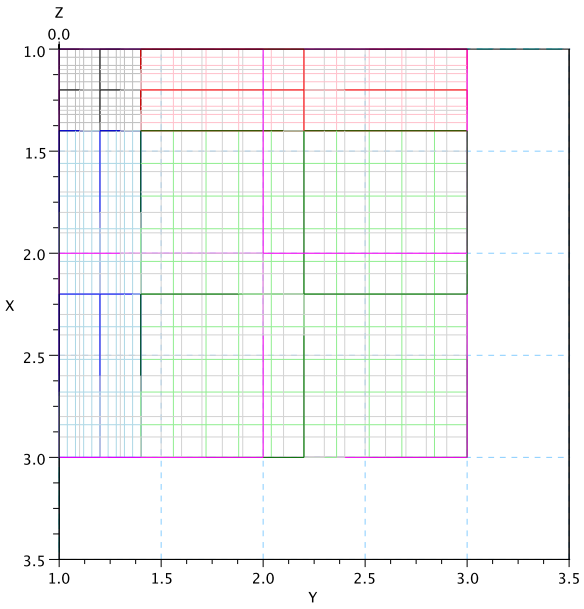


Figura 5.12: Esempio di subdivision applicata a t=0.2, vista dall'alto si nota la distribuzione della suddivisione; le griglie delle nuove superfici, campionate uniformemente come in tutte gli esempi precedenti, enfatizzano la diversa scalatura delle quattro suddivisioni.

si vuole risolvere, come prima, il sistema:

$$X(u_i, v_j) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} B_i^n(u_i) B_j^m(v_j) = X_{in}(u_i, v_j) = P_{ij}$$

Per ogni coppia di punti si ha:

$$X(u_k, v_r) = \left(B_0^n(u_k), ..., B_n^n(u_k)\right) \underbrace{\begin{pmatrix} b_{00} & \cdots & b_{0m} \\ \vdots & & \vdots \\ b_{n0} & \cdots & b_{nm} \end{pmatrix}}_{B} \begin{pmatrix} B_0^m(v_r) \\ \vdots \\ B_m^m(v_r) \end{pmatrix} = P_{kr}$$

Definita la matrice dei punti noti:

$$P = \begin{pmatrix} P_{00} & \cdots & P_{0m} \\ \vdots & & \vdots \\ P_{n0} & \cdots & P_{nm} \end{pmatrix}$$

e le matrici:

$$F = \begin{pmatrix} B_0^n(u_0) & \cdots & B_0^n(u_n) \\ \vdots & & \vdots \\ B_n^n(u_0) & \cdots & B_n^n(u_n) \end{pmatrix}, \quad G = \begin{pmatrix} B_0^m(v_0) & \cdots & B_0^m(v_m) \\ \vdots & & \vdots \\ B_m^m(v_0) & \cdots & B_m^m(v_m) \end{pmatrix}$$

il problema può essere rappresentato in forma matriciale come

$$F^T B G = P$$

Patch triangolari di Bézeir

L'idea alla base delle patch triangolari è quella di usare una combinazioni baricentrica di coordinate come parametri per muoversi su una superficie. In particolare in questo caso si considera come dominio un triangolo non degenere T,di vertici $\widehat{ABC}$; i punti dell'area sono rappresentabili come combinazione baricentrica dei vertici:

$$P \in T, \quad P = uA + vB + wC, \quad u + v + w = 1$$

e la superficie come una funzione:

$$X : T \to \mathbb{E}^3$$

Coordinate baricentriche

Le coordinate baricentriche descritte sopra sono in relazione biunivoca con quelle cartesiane. Si ha:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} x_A & x_B & x_C \\ y_A & y_B & y_C \\ 1 & 1 & 1 \end{pmatrix}}_{T} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Affinchè la relazione sia biunivoca (ovvero il sistema ha soluzione unica) se e solo se la matrice T è non singolare, ovvero il triangolo $\widehat{ABC}$ non è degenere.

Polinomi di Bernstein generalizzati

I polinomi di Bernstein possono essere generalizzati alle coordinate baricentriche. Per comodità si utilizza la seguente notazione:

- $I = (i, j, k)$ è un multiindice
- $I! = i!j!k!$
- $|I| = i + j + k = n$
- $\mu = (u, v, w)^T$
- $\mu^I = u^i v^j w^k$

Definizione 5.5 (polinomio di Bernstein generalizzato)**.** Si definisce polinomio di Bernstein generalizzato il polinomio:

$$B_I^n(\mu) = \frac{n!}{I!}\mu^I$$

Tale polinomio è bivariato e di grado n, infatti $\mu^I = u^i v^j w^k$ e $i + j + k = n$.
Fissare $|I| = n$ riduce il numero di indici distinti a $(n+1)(n+2)/2$.

Proprietà

Non negatività Tutti i fattori sono positivi o nulli, e la proprità è banalmente soddisfatta.

Partizione dell'unità Vale che:

$$\sum_{|I|=n} B_I^n(\mu) \equiv 1, \quad \forall \mu \in T$$

Dimostrazione. si ha infatti che:

$$\begin{aligned}
1 = 1^n = (u+v+w)^n &= (u+(v+w))^n \\
&= \sum_{i=0}^{n} \binom{n}{i} u^i (v+w)^{n-i} \\
&= \sum_{i=0}^{n} \binom{n}{i} u^i \sum_{j=0}^{n-i} \binom{n-i}{j} v^j w^{\overbrace{n-i-k}^{k}} \\
&= \sum_{i=0}^{n} \sum_{j=0}^{n-i} \binom{n}{i} \binom{n-i}{j} u^i v^j w^k \\
&= \sum_{i=0}^{n} \sum_{j=0}^{n-i} \frac{n!}{i!j!k!} u^i v^j w^k \\
&= \sum_{|I|=n} \frac{n!}{I!} \mu^I \\
&= B_I^n(\mu)
\end{aligned}$$

□

Linear Precision Si può dimostrare che vale:

$$\mu = \sum_{|I|=n} \frac{I}{n} B_I^n(\mu)$$

la dimostrazione è componente per componente e si riconduce direttamente alla proprietà analoga già dimostrata nel caso monovariato.

Patch triangolari

Ora è possibile dare una definizione più rigorosa delle patch triangolari di Bézier.

Definizione 5.6 (Patch triangolare di Bézier)**.** Dato un dominio triangolare T, una patch triangolare di Bézeire di grado n è una funzione $X : T \to \mathbb{E}^3$ del tipo:

$$X(\mu) = \sum_{|I|=n} b_I B_I^n(\mu)$$

in cui $b_I \in \mathbb{E}^3$ sono chiamati punti di controllo.

Proprietà

Comportamento ai bordi Lungo uno dei bordi (ovvero quando uno dei tre parametri in μ è nullo si ha:

$$\begin{aligned} X\begin{pmatrix} u \\ v \\ 0 \end{pmatrix} &= \sum_{\substack{|I|=n \\ k=0}} b_I B_I^n \begin{pmatrix} u \\ v \\ 0 \end{pmatrix} \\ &= \sum_{i=0}^{n} b_{(i,n-i,0)} \frac{n!}{(n-i)!i!0!} u^i v^{n-i} w^0 \\ &= \sum_{i=0}^{n} b_{(i,n-i,0)} \underbrace{\binom{n}{i} u^i v^{n-i} w^0}_{B_i^n(n),\ v=1-u} \end{aligned}$$

ovvero lungo i bordi si ha una curva di Bézier. Gli altri casi sono analoghi.

Controllo pseudolocale Spostando un punto di controllo:

$$\tilde{b}_I = b_I + \delta b$$

si ha:

$$\tilde{X}(\mu) = X(\mu) + \delta b B_I^n(\mu)$$

ovvero come nel caso delle curve di Bézier la modifica di un punto di controllo ha effetti su tutta la curva, in maniera proporzionale al valore del polinomio di Bernstein corrispondente.

End-point interpolation

Ai vertici del triangolo si ha:

$$X\begin{pmatrix}1\\0\\0\end{pmatrix} = \sum_{\substack{|I|=n\\k=0}} b_I B_I^n \sum_{i=0}^{n} b_{(i,0,0)}\begin{pmatrix}1\\0\\0\end{pmatrix} = \sum_{\substack{|I|=n\\j=0\\k=0}} b_I B_I^n \begin{pmatrix}1\\0\\0\end{pmatrix}$$

$$= \sum_{i=n}^{n} b_{(i,0,0)} B^n_{(i,0,0)} \begin{pmatrix}1\\0\\0\end{pmatrix}$$

$$= b_{(n,0,0)} \frac{n!}{n!0!0!} 1^n v^0 w^0 = b_{(n,0,0)}$$

La dimostrazione per gli altri due casi è analoga.

Forma ricorsiva

I polinomi di Bérnstein generalizzati posso essere espressi in forma ricorsiva:

$$B_I^n(\mu)^\dagger = uB^{n-1}_{I-e_1}(\mu) + vB^{n-1}_{I-e_2}(\mu) + wB^{n-1}_{I-e_3}(\mu) \tag{5.1}$$

Infatti:

$$B^{n-1}_{I-e_1}(\mu) = \frac{(n-1)!}{(i-1)!j!k!} u^{i-1}v^j w^k$$
$$B^{n-1}_{I-e_2}(\mu) = \frac{(n-1)!}{(j-1)!i!k!} u^i v^{j-1} w^k$$
$$B^{n-1}_{I-e_3}(\mu) = \frac{(n-1)!}{(k-1)!i!j!} u^i v^j w^{k-1}$$

e svolgendo la somma (5.1) si ottiene:

$$\left[\frac{1}{jk} + \frac{1}{ik} + \frac{1}{ij}\right] \frac{(n-1)!}{(i-1)!(j-1)!(k-1)!} u^i v^j w^k = \frac{n!}{i!j!k!} u^i v^j w^k =_{def} B_I^n(\mu)$$

Algoritmi

de Casteljau

Come nel caso delle patch quadrate di Bézeir, l'algoritmo di de Casteljau si definisce a partire dalla definizione ricorsiva del polinomi di Bernstein.
Si definiscono i punti di controllo al passo zeresimo:

$$b_I^0 = b_I \quad \forall\, |I| = n$$

†in cui con e_1, e_2, e_3 si indicano i versori unitari lungo le direzioni u, v, w

La superficie è definita dai punti:

$$\begin{aligned} X(\mu) &= \sum_{|I|=n} b_I^0 B_I^n(\mu) \\ &= \sum_{|I|=n} b_I^0 \left[u B_{I-e_1}^{n-1}(\mu) + v B_{I-e_2}^{n-1}(\mu) + w B_{I-e_3}^{n-1}(\mu) \right] \\ &= \sum_{|J|=n-1} (u b_{J+e_1}^0 + v b_{J+e_2}^0 + w b_{J+e_3}^0) B_J^{n-1}(\mu) \end{aligned}$$

e definendo:

$$b_J^n = (u b_{J+e_1}^{n-1} + v b_{J+e_2}^{n-1} + w b_{J+e_3}^{n-1})$$

in generale la superficie può essere rappresentata come:

$$X(\mu) = \sum_{|J|=n-r} b_J^r B_J^{n-r}(\mu)$$

i cui singoli punti sono quindi in:

$$X(\mu) = b_{(0,0,0)}^n$$

de Casteljau per patch triangolari

```
function Curve = Bezier_tripatch_casteljau(b,u,v)
Lu = length(u)
Lv = length(v)

for cu=1:Lu
      tcurve = []
      for cv=1:Lv
         tcurve = [tcurve, tripatch_point(b,u(cu),v(cv))]
      end
      Curve(:,:,cu) = tcurve
end
endfunction

function P = tripatch_point(b,cu,cv)
[d,n,m] = size(b) //n,m must be the same!
tb1=b
tb2=b
cw = 1-cu-cv
for r=1:n-1
      for cn=1:n-1-r+1
         for cm=1:m-cn
            tb2(:,cn,cm) = cw*tb1(:,cn,cm)+cu*tb1(:,cn+1,cm)+cv*tb1(:,cn,cm+1)
         end
      tb1=tb2
      end
end
P = tb1(:,1,1)
endfunction
```

Degree-elevation

Data la superficie:

$$X(\mu) = \sum_{|I|=n} b_I B_I^n(\mu), \quad \mu \in T$$

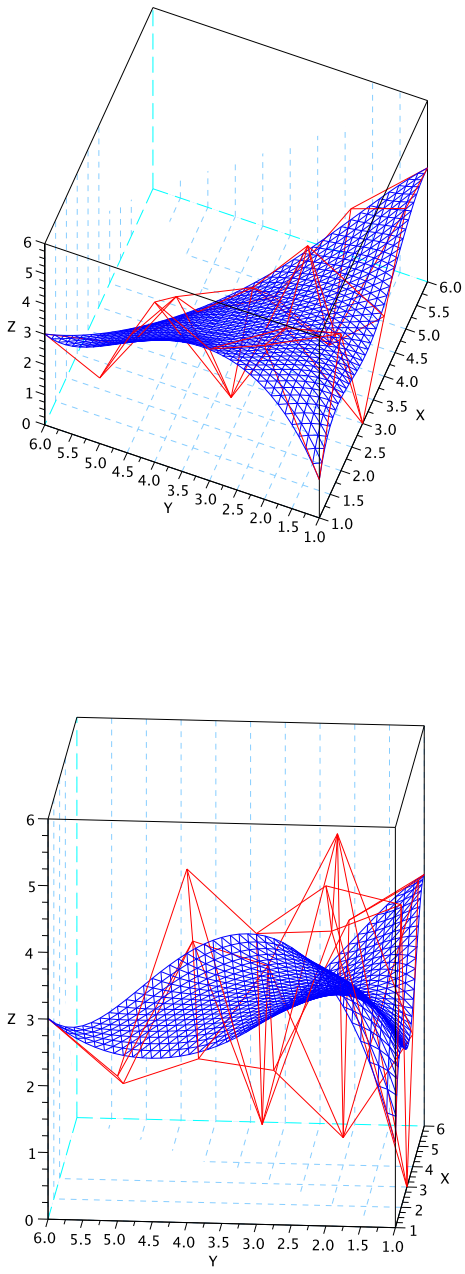


Figura 5.13: Esempio di patch di bezier triangolare, da due angolazioni diverse.

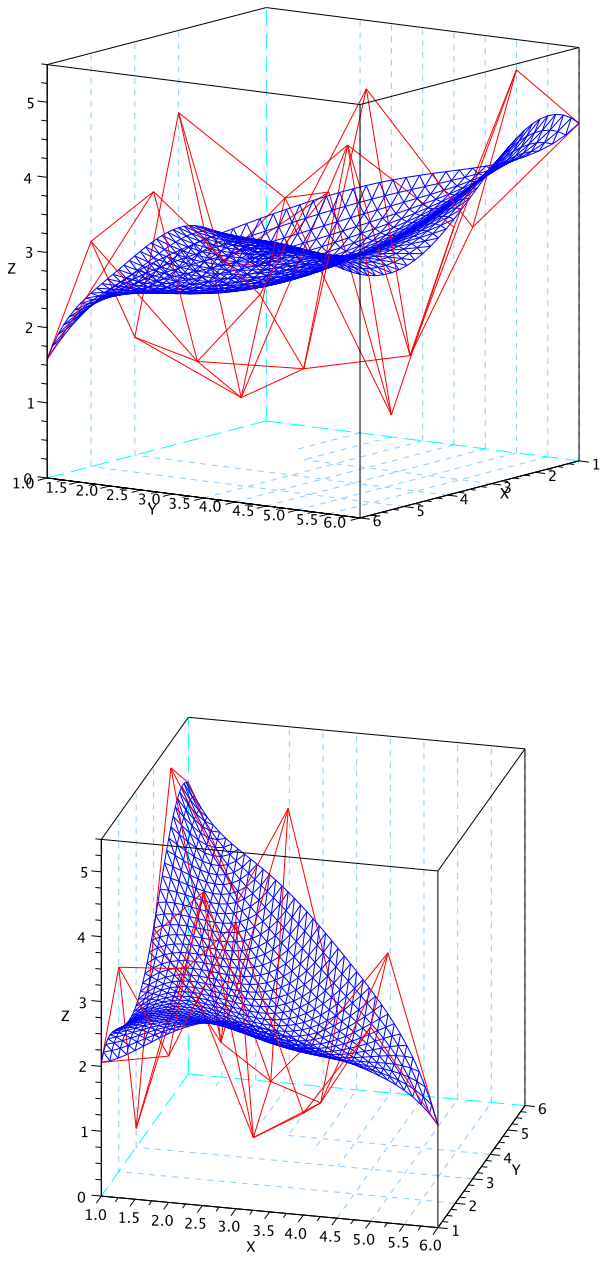


Figura 5.14: Esempio di patch di bezier triangolare, da due angolazioni diverse.

la si vuole rappresentare con una patch di bezier di grado più elevato:

$$X_e(\mu) = \sum_{|I|=n+1} c_I B_I^{n+1}(\mu) = X(\mu), \quad \mu \in T$$

In quanto $u+v+w=1$ si può scrivere:

$$\begin{aligned}
X(\mu) &= \sum_{|I|=n} b_I B_I^n(\mu) = (u+v+w) \sum_{|I|=n} b_I B_I^n(\mu) \\
&= \sum_{|I|=n} b_I \frac{n!}{i!j!k!} u^{i+1} v^j w^k + \sum_{|I|=n} b_I \frac{n!}{i!j!k!} u^i v^{j+1} w^k + \sum_{|I|=n} b_I \frac{n!}{i!j!k!} u^i v^j w^{k+1} \\
&= {}^{\dagger} \sum_{|I|=n} b_I \frac{i+1}{n+1} \frac{(n+1)!}{(I+e_1)!} U^{I+e_1} + \sum_{|I|=n} b_I \frac{i+1}{n+1} \frac{(n+1)!}{(I+e_2)!} U^{I+e_2} + \sum_{|I|=n} b_I \frac{i+1}{n+1} \frac{(n+1)!}{(I+e_3)!} U^{I+e_3} \\
&= {}^{\ddagger} \sum_{|J|=n+1} \left(\frac{i}{n+1} b_{J-e_1} + \frac{j}{n+1} b_{J-e_2} + \frac{k}{n+1} b_{J-e_3} \right) B_J^{n+1}(\mu) \\
&= \sum_{|J|=n+1} c_I B_J^{n+1}(\mu) asd
\end{aligned}$$

ovvero i coefficienti c_I possono essere ricavati direttamente:

$$c_I = \left(\frac{i}{n+1} b_{J-e_1} + \frac{j}{n+1} b_{J-e_2} + \frac{k}{n+1} b_{J-e_3} \right)$$

Degree elevation per patch triangolari

```
function d = Bezier_tripatch_degelev(b)
[dim,n,m] = size(b)
d = hypermat([dim,n+1,m+1])



for i=2:n
      for j=2:n+2-i
         k = (n-j+1-i+1)
         d(1:3,i,j)=(1/(n))*((i-1)*b(1:3,i-1,j)+(j-1)*b(1:3,i,j-1)+(k)*b(1:3,i,j))
      end
end

for j=2:n
      k=n+1-1-j+1
      d(1:3,1,j) = (1/(n)) * ((j-1)*b(1:3,1,j-1)+(k)*b(1:3,1,j))
end

for i=2:n
      k=n+1-1-i+1
      d(1:3,i,1) = (1/(n)) * ((i-1)*b(1:3,i-1,1)+(k)*b(1:3,i,1))
```

[†] Chiamando $u^i v^j w^k = U^I$

[‡] Considerato che $\frac{(n+1)!}{(I+e_1)!} U^{I+e_1} = B_{I+e_1}^{n+1}$ e così via

```
22 end
23
24 d(1:3,1,1) = b(1:3,1,1)
25 d(1:3,n+1,1) = b(1:3,n,1)
26 d(1:3,1,m+1) = b(1:3,1,m)
27
28 endfunction
```

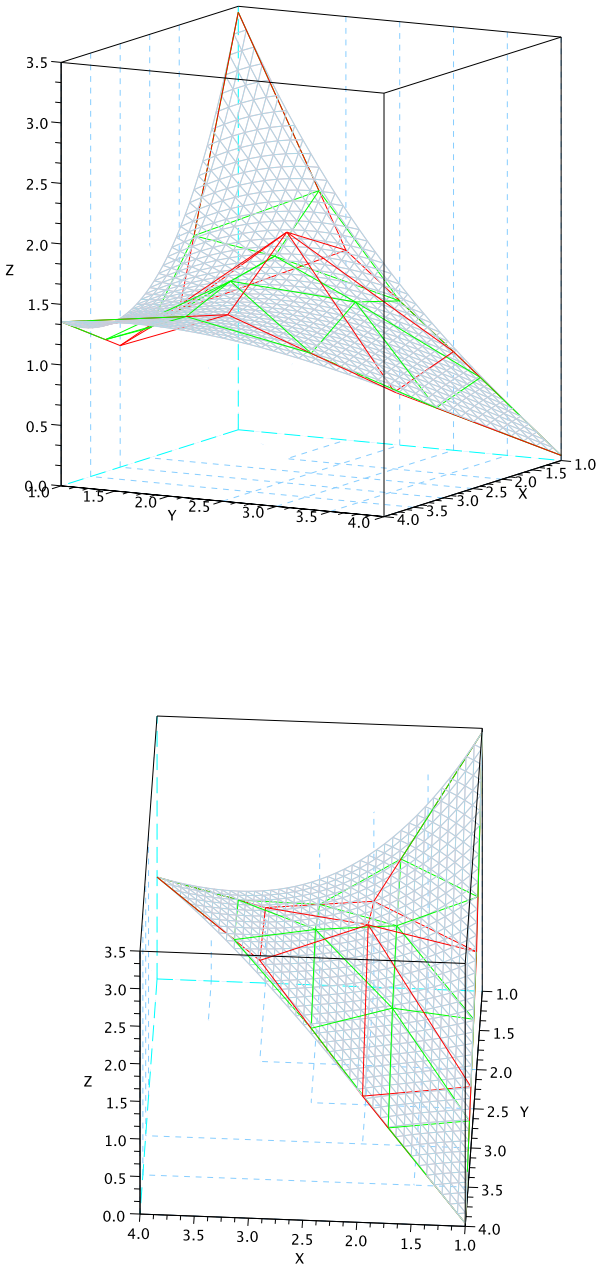


Figura 5.15: Esempio di degree elevation applicata ad una patch di bezier triangolare, da due angolazioni diverse.

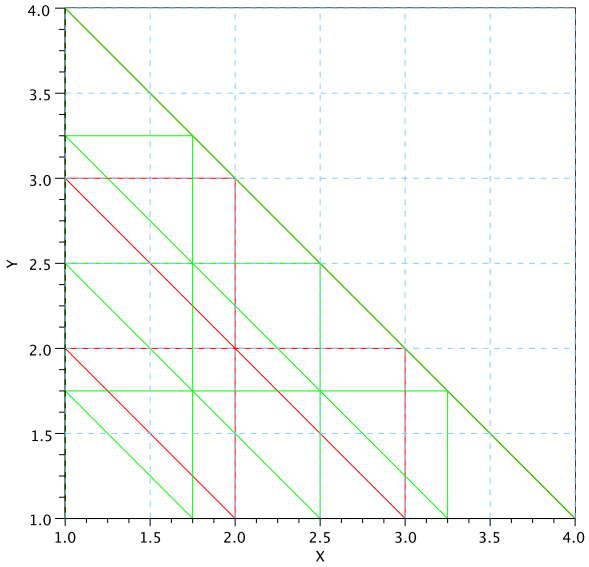


Figura 5.16: Esempio di degree elevation applicata ad una patch di bezier triangolare, su un griglia piana.

Appendice A

Codici

Qui sono riportati i codici troppo lunghi o non abbastanza importanti per essere interlacciati al testo.
Per provare i vari codici e sufficiente avviare scilab nella directory che contiene tutti gli script e lanciare uno qualsiasi degli script di esempio con il comando

```
exec script.sci
```

All'inizio di ogni script di esempio, tutti denominati `esempi_*.sci`, c'è una lista di selettori per selezionare quali figure si vogliono prodotte.

A.1 Bézier SPLINE: acquisizione punti condizionale e calcolo

Acquisizione e calcolo Bézier spline C0

```
exec glib.sci;
xcol = [0:0.001:1];
gcount=1;
g = input('numero e grado curve? ')

po(:,:,1) = mouse_input_2d(g(2));
plot(po(1,:,1),po(2,:,1),'Color', [1/g(1),0.5,1-1/g(1)],'LineStyle','--','Marker','o');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
for i=1:g(1)-1
po(:,:,i+1) = [po(:,size(po,2),i),mouse_input_2d(g(2)-1)];
plot(po(1,:,i+1),po(2,:,i+1),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','
    Marker','o');
end

scf(gcount);gcount=gcount+1;
subplot(1,2,1)
for i=1:g(1)
```

```
 c(:,:,i) = Bezier_casteljau(po(:,:,i),xcol);
 plot(po(1,:,i),po(2,:,i),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o')
     ;
plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
end
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(1,2,2)
for i=1:g(1)
rpo = Bezier_hodograph(po(:,:,i));
c(:,:,i) = Bezier_casteljau(rpo,xcol);
plot(rpo(1,:),rpo(2,:),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o');
plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
end
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
```

Acquisizione e calcolo Bézier spline C1

```
exec glib.sci;
xcol = [0:0.001:1];
gcount=1;
g = input('numero e grado curve? ')

po(:,:,1) = mouse_input_2d(g(2));
plot(po(1,:,1),po(2,:,1),'Color',[1/g(1),0.5,1-1/g(1)],'LineStyle','--','Marker','o');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
for i=1:g(1)-1
//tutte le curve hanno lo stesso grado, i coefficienti si semplificano
tpo = [po(:,size(po,2),i),(2*po(:,size(po,2),i)-po(:,size(po,2)-1,i))];
plot(tpo(1,:),tpo(2,:),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','Marker','o'
    );
po(:,:,i+1)=[tpo,mouse_input_2d(g(2)-2)]
plot(po(1,:,i+1),po(2,:,i+1),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','
    Marker','o');
end
scf(gcount);gcount=gcount+1;
subplot(1,2,1)
for i=1:g(1)
 c(:,:,i) = Bezier_casteljau(po(:,:,i),xcol);
 plot(po(1,:,i),po(2,:,i),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o')
     ;
plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
end
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(1,2,2)
for i=1:g(1)
rpo = Bezier_hodograph(po(:,:,i));
c(:,:,i) = Bezier_casteljau(rpo,xcol);
plot(rpo(1,:),rpo(2,:),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o');
plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
end
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
```

Acquisizione e calcolo Bézier spline C2

```
exec glib.sci;
xcol = [0:0.001:1];
gcount=1;
g = input('numero e grado curve? ')
```

```
5
6  po(:,:,1) = mouse_input_2d(g(2));
7  plot(po(1,:,1),po(2,:,1),'Color',[1/g(1),0.5,1-1/g(1)],'LineStyle','--','Marker','o');
8  handler=get("current_axes");
9  handler.isoview="on";handler.grid=[12,12];
10 for i=1:g(1)-1
11 //i coefficienti si semplificano se le curve hanno tutte lo stesso grado
12 tpo = [po(:,size(po,2),i)]
13 tpo = [tpo(:,1),(2*tpo(:,1)-po(:,size(po,2)-1,i))]
14 tpo1 = -2*po(:,size(po,2)-1,i) + po(:,size(po,2)-2,i) +2*tpo(:,2)
15 tpo = [tpo,tpo1]
16
17
18 plot(tpo(1,:),tpo(2,:),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','Marker','o'
       );
19 po(:,:,i+1)=[tpo,mouse_input_2d(g(2)-3)]
20 plot(po(1,:,i+1),po(2,:,i+1),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','
       Marker','o');
21 end
22
23 scf(gcount);gcount=gcount+1;
24 subplot(1,3,1)
25 for i=1:g(1)
26  c(:,:,i) = Bezier_casteljau(po(:,:,i),xcol);
27  plot(po(1,:,i),po(2,:,i),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o')
       ;
28 plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
29 end
30 handler=get("current_axes");
31 handler.isoview="on";handler.grid=[12,12];
32 subplot(1,3,2)
33 for i=1:g(1)
34 rpo = Bezier_hodograph(po(:,:,i));
35 c(:,:,i) = Bezier_casteljau(rpo,xcol);
36 plot(rpo(1,:),rpo(2,:),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o');
37 plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
38 end
39 handler=get("current_axes");
40 handler.isoview="on";handler.grid=[12,12];
41 subplot(1,3,3)
42 for i=1:g(1)
43 rrpo = Bezier_hodograph(Bezier_hodograph(po(:,:,i)));
44 c(:,:,i) = Bezier_casteljau(rrpo,xcol);
45 plot(rrpo(1,:),rrpo(2,:),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o');
46 plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
47 end
48 handler=get("current_axes");
49 handler.isoview="on";handler.grid=[12,12];
```

Acquisizione e calcolo Bézier spline G1

```
1  exec glib.sci;
2  xcol = [0:0.001:1];
3  gcount=1;
4  g = input('numero, grado, scala curve? ')
5  k = g(3)
6
7  po(:,:,1) = mouse_input_2d(g(2));
8  plot(po(1,:,1),po(2,:,1),'Color',[1/g(1),0.5,1-1/g(1)],'LineStyle','--','Marker','o');
9  handler=get("current_axes");
10 handler.isoview="on";handler.grid=[12,12];
11 for i=1:g(1)-1
12 //tutte le curve hanno lo stesso grado, i coefficienti si semplificano
13 tpo = [po(:,size(po,2),i),((k+1)*po(:,size(po,2),i)-po(:,size(po,2)-1,i))/k];
```

```
14
15 plot(tpo(1,:),tpo(2,:),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','Marker','o'
       );
16 po(:,:,i+1)=[tpo,mouse_input_2d(g(2)-2)]
17 plot(po(1,:,i+1),po(2,:,i+1),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','
       Marker','o');
18 end
19
20 scf(gcount);gcount=gcount+1;
21 subplot(1,2,1)
22 for i=1:g(1)
23  c(:,:,i) = Bezier_casteljau(po(:,:,i),xcol);
24  plot(po(1,:,i),po(2,:,i),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o')
       ;
25 plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
26 end
27 handler=get("current_axes");
28 handler.isoview="on";handler.grid=[12,12];
29 subplot(1,2,2)
30 for i=1:g(1)
31 rpo = Bezier_hodograph(po(:,:,i));
32 c(:,:,i) = Bezier_casteljau(rpo,xcol);
33 plot(rpo(1,:),rpo(2,:),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o');
34 plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
35 if i>1 then
36 plot(k*c(1,:,i),k*c(2,:,i),'Color',[i/g(1),0.7,(1-i/g(1))]);
37 end
38 end
39 handler=get("current_axes");
40 handler.isoview="on";handler.grid=[12,12];
```

Acquisizione e calcolo Bézier spline G2

```
1  exec glib.sci;
2  xcol = [0:0.001:1];
3  gcount=1;
4  g = input('numero, grado, scala1, scala2 curve? ')
5  k = g(3)
6  s = g(4)
7
8  po(:,:,1) = mouse_input_2d(g(2));
9  plot(po(1,:,1),po(2,:,1),'Color',[1/g(1),0.5,1-1/g(1)],'LineStyle','--','Marker','o');
10 handler=get("current_axes");
11 handler.isoview="on";handler.grid=[12,12];
12 for i=1:g(1)-1
13 //tutte le curve hanno lo stesso grado, i coefficienti si semplificano
14 b1 = ((k+1)*po(:,size(po,2),i)-po(:,size(po,2)-1,i))/k
15 b2=((-s+1)*po(:,size(po,2),i)-2*po(:,size(po,2)-1,i)+po(:,size(po,2)-2,i)+2*s*b1)/s
16
17 tpo = [po(:,size(po,2),i),b1,b2];
18
19 plot(tpo(1,:),tpo(2,:),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','Marker','o'
       );
20 po(:,:,i+1)=[tpo,mouse_input_2d(g(2)-3)]
21 plot(po(1,:,i+1),po(2,:,i+1),'Color',[(i+1)/g(1),0.5,1-(i+1)/g(1)],'LineStyle','--','
       Marker','o');
22 end
23
24 scf(gcount);gcount=gcount+1;
25 subplot(1,3,1)
26 for i=1:g(1)
27  c(:,:,i) = Bezier_casteljau(po(:,:,i),xcol);
28  plot(po(1,:,i),po(2,:,i),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o')
       ;
```

```
plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
end
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(1,3,2)
for i=1:g(1)
rpo = Bezier_hodograph(po(:,:,i));
c(:,:,i) = Bezier_casteljau(rpo,xcol);
plot(rpo(1,:),rpo(2,:),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o');
plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
if i>1 then
plot(k*c(1,:,i),k*c(2,:,i),'Color',[i/g(1),0.7,i/g(1)]);
end
end
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(1,3,3)
for i=1:g(1)
rrpo = Bezier_hodograph(Bezier_hodograph(po(:,:,i)));
c(:,:,i) = Bezier_casteljau(rrpo,xcol);
plot(rrpo(1,:),rrpo(2,:),'Color',[(i)/g(1),0.5,1-(i)/g(1)],'LineStyle','--','Marker','o');
plot(c(1,:,i),c(2,:,i),'Color',[i/g(1),0.5,1-i/g(1)]);
if i>1 then
plot(s*c(1,:,i),s*c(2,:,i),'Color',[i/g(1),0.7,i/g(1)]);
end
end
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
```

A.2 Curve di Bézeir: esempi del testo

Listing:

```
//ALMENO 5 PUNTI ALTRIMENTI CRASH SU DERIVATA 3!
exec glib.sci;
xcol = [0:0.001:1];
gcount=1;

//attiva disattiva grafici
castel = 1;
hodo = 1;
td = 1;
degelev = 1;
subdiv = 1;
ratbez = 1;
coniche = 1;

p = mouse_input_2d(input('Numero di punti? '));
//CASTELJAU
c = Bezier_casteljau(p,xcol);
if castel then
scf(gcount);gcount=gcount+1;
xtitle('de Casteljau')
plot(p(1,:),p(2,:),'ro-');
plot(c(1,:),c(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
end
//HODOGRAPH E CURVATURA
pc1 = Bezier_hodograph(p)
```

```
pc2 = Bezier_hodograph(pc1)
c1 = Bezier_casteljau(pc1,xcol);
c2 = Bezier_casteljau(pc2,xcol);
curv = curvature(c1,c2);
if hodo then
scf(gcount);gcount=gcount+1;
subplot(2,2,1);
xtitle('de Casteljau')
plot(p(1,:),p(2,:),'ro-');
plot(c(1,:),c(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(2,2,2);
xtitle('Hodograph 1')
plot(c1(1,:),c1(2,:),'b-');
plot(pc1(1,:),pc1(2,:),'r-o');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(2,2,3);
xtitle('Hodograph 2');
plot(c2(1,:),c2(2,:),'b-');
plot(pc2(1,:),pc2(2,:),'r-o');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(2,2,4);
xtitle('Curvatura');
plot(curv);
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
end
//3d, CURVATURA E TORSIONE
[d,n] = size(p);
z = [0:(1/n):1];
ptd = [p;z(1:n)];
tdc = Bezier_casteljau(ptd,xcol);
ptdd = Bezier_hodograph(ptd);
ptddd = Bezier_hodograph(ptdd);
ptdddd = Bezier_hodograph(ptddd);
tdcd = Bezier_casteljau(ptdd,xcol);
tdcdd = Bezier_casteljau(ptddd,xcol);
tdcddd = Bezier_casteljau(ptdddd,xcol);
tor3d = torsione(tdcd,tdcdd,tdcddd);
cur3d = curvature(tdcd,tdcdd);
if td then
scf(gcount);gcount=gcount+1;
subplot(3,1,1);
xtitle('3d de Casteljau');
param3d(ptd(1,:),ptd(2,:),ptd(3,:),20,-20);
handler = gce();
handler.foreground=color('red') //red
handler.mark_mode='on' //enable marks
handler.mark_size=1
handler.mark_foreground=color('red') //red marks
handler.mark_style=1 //cross
param3d(tdc(1,:),tdc(2,:),tdc(3,:),20,-20);
handler = gce();
handler.foreground=color('blue') //blue
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(3,1,2);
xtitle('Curvatura');
plot(cur3d);
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(3,1,3);
```

```
93  xtitle('Torsione');
94  plot(tor3d);
95  handler=get("current_axes");
96  handler.isoview="on";handler.grid=[12,12];
97  end
98  //Degree elevation
99  pdegev1 = Bezier_degelev(p);
100 cdeg1 = Bezier_casteljau(pdegev1,xcol);
101 pdegev2 = Bezier_degelev(pdegev1);
102 cdeg2 = Bezier_casteljau(pdegev2,xcol);
103 if degelev then
104 scf(gcount);gcount=gcount+1;
105 xtitle('Degree elevation')
106 subplot(1,2,1);
107 plot(p(1,:),p(2,:),'ro-');
108 plot(c(1,:),c(2,:),'b-');
109 plot(pdegev1(1,:),pdegev1(2,:),'go-');
110 plot(cdeg1(1,:),cdeg1(2,:),'b-');
111 handler=get("current_axes");
112 handler.isoview="on";handler.grid=[12,12];
113 subplot(1,2,2)
114 plot(p(1,:),p(2,:),'ro-');
115 plot(c(1,:),c(2,:),'b-');
116 plot(pdegev2(1,:),pdegev2(2,:),'mo-');
117 plot(pdegev1(1,:),pdegev1(2,:),'go--');
118 plot(cdeg2(1,:),cdeg2(2,:),'b-');
119 handler=get("current_axes");
120 handler.isoview="on";handler.grid=[12,12];
121 end
122 //subdivision
123 [psub1,psub2] = Bezier_subdiv(p,0.5);
124 csub1 = Bezier_casteljau(psub1,xcol);
125 csub2 = Bezier_casteljau(psub2,xcol);
126 if subdiv then
127 scf(gcount);gcount=gcount+1;
128 xtitle('Subdivision');
129 plot(p(1,:),p(2,:),'ro-');
130 plot(c(1,:),c(2,:),'b-');
131 plot(psub1(1,:),psub1(2,:),'go-');
132 plot(psub2(1,:),psub2(2,:),'co-');
133 plot(csub1(1,:),csub1(2,:),'g--');
134 plot(csub2(1,:),csub2(2,:),'c--');
135 handler=get("current_axes");
136 handler.isoview="on";handler.grid=[12,12];
137 end
138 //rational bezier
139 beta0 = ones(1,size(p,2));
140 if pmodulo(size(p,2),2)==1 then
141 n=(size(p,2)+1)/2;
142 beta1 = [[1:1:n],[n-1:-1:1]];
143 beta2 = [[1:1:1*n],[1*(n-1):-1:1]]^4;
144 else
145 n=size(p,2)/2;
146 beta1 = [[1:1:n],[n:-1:1]];
147 beta2 = [[1:1:n],[(n):-1:1]]^2;
148 end
149 beta2c = beta2^-1
150 beta3 = [1:1:size(p,2)]^4;
151 beta3c = beta3^-1;
152 rb0 = RATBezier_casteljau(p,beta0,xcol);
153 rb1 = RATBezier_casteljau(p,beta1,xcol);
154 rb2 = RATBezier_casteljau(p,beta2,xcol);
155 rbc2 = RATBezier_casteljau(p,beta2c,xcol);
156 rb3 = RATBezier_casteljau(p,beta3,xcol);
157 rbc3 = RATBezier_casteljau(p,beta3c,xcol);
```

```
158 if ratbez then
159 scf(gcount);gcount=gcount+1;
160 xtitle('Bezier razionali')
161 subplot(2,2,1)
162 xtitle('Razionale pesi omogenei - normale 1 2 3 2 1')
163 plot(p(1,:),p(2,:),'ro-');
164 plot(c(1,:),c(2,:),'b-');
165 plot(rb0(1,:),rb0(2,:),'k-');
166 handler=get("current_axes");
167 handler.isoview="on";handler.grid=[12,12];
168 subplot(2,2,2)
169 xtitle('modifica pesi 1')
170 plot(p(1,:),p(2,:),'ro-');
171 plot(c(1,:),c(2,:),'b-');
172 plot(rb1(1,:),rb1(2,:),'m-');
173 handler=get("current_axes");
174 handler.isoview="on";handler.grid=[12,12];
175 subplot(2,2,3)
176 xtitle('modifica pesi 2: esponenziale (g) e reciproco (k)')
177 plot(p(1,:),p(2,:),'ro-');
178 plot(c(1,:),c(2,:),'b-');
179 plot(rb1(1,:),rb1(2,:),'m--');
180 plot(rb2(1,:),rb2(2,:),'g-');
181 plot(rbc2(1,:),rbc2(2,:),'k-');
182 handler=get("current_axes");
183 handler.isoview="on";handler.grid=[12,12];
184 subplot(2,2,4)
185 xtitle('modifica pesi 3: asimmetrici exp (1 2 3 ... size(p))^4 e reciproco')
186 plot(p(1,:),p(2,:),'ro-');
187 plot(c(1,:),c(2,:),'b-');
188 plot(rb2(1,:),rb2(2,:),'g--');
189 plot(rb1(1,:),rb1(2,:),'m--');
190 plot(rb3(1,:),rb3(2,:),'r-');
191 plot(rbc3(1,:),rbc3(2,:),'k-');
192 handler=get("current_axes");
193 handler.isoview="on";handler.grid=[12,12];
194 end
195
196 pc = [-1,0,1;1,-1,1];
197 ellipse1 = RATBezier_coniche(pc,0.5,xcol);
198 ellipse2 = RATBezier_coniche(pc,-0.5,xcol);
199 parabola = RATBezier_coniche(pc,1,xcol);
200 iperbole = RATBezier_coniche(pc,3,xcol);
201 pc2 = [.5,0,1;1,0,.5];
202 circ1 = RATBezier_coniche(pc2,0.5,xcol);
203 circ2 = RATBezier_coniche(pc2,-0.5,xcol);
204 if coniche then
205 scf(gcount);gcount=gcount+1;
206 subplot(2,2,1)
207 xtitle('ellisse, beta = 0.5 (g), -0.5 (m)')
208 plot(pc(1,:),pc(2,:),'ro-');
209 plot(ellipse1(1,:),ellipse1(2,:),'g');
210 plot(ellipse2(1,:),ellipse2(2,:),'m');
211 handler=get("current_axes");
212 handler.isoview="on";handler.grid=[12,12];
213 subplot(2,2,2)
214 xtitle('parabola, beta = 1, x^2 (c--)')
215 plot(pc(1,:),pc(2,:),'ro-');
216 plot(parabola(1,:),parabola(2,:),'b');
217 plot(xcol,xcol^2,'c--');
218 handler=get("current_axes");
219 handler.isoview="on";handler.grid=[12,12];
220 subplot(2,2,4)
221 xtitle('iperbole beta = 3')
222 plot(pc(1,:),pc(2,:),'ro-');
```

```
plot(iperbole(1,:),iperbole(2,:),'b');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
subplot(2,2,3)
xtitle('ellisse 2')
plot(pc2(1,:),pc2(2,:),'ro-');
plot(circ1(1,:),circ1(2,:),'g');
plot(circ2(1,:),circ2(2,:),'m');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
end
```

Disegno della base di Bernstein

```
function somma=Bezier_plot_base(n,t)
somma = []
for i=0:n
      c = exp ( gammaln ( n +1) - gammaln ( i +1) - gammaln (n - i +1))
      curva = c*(t^i'.*(1-t)^(n-i)')
      somma = somma+curva
      plot(t,curva,'Color',[i/n,0.5,1-i/n]);
      handler=get("current_axes");
      handler.isoview="on";handler.grid=[12,12];
end
endfunction
```

A.3 Curve BSpline: esempi del testo

Listing:

```
exec glib.sci;
gcount=1;

fig_noep=1;
fig_ep=1;
fig_mult=1;
fig_norm=1;
fig_closed=1;
fig_knotins=1;
fig_rat=1;
fig_circ=1;

k = input('Grado Spline? ');
b = mouse_input_2d(input('Numero di punti? '));
[d,np]=size(b);

T=zeros(k,1)';
for i=k+1:np
  T=[T,(T(i-1)+1)];
end
T=[T,(ones(k,1)'*(T(np)+1))];

Tnoep = [0:1:(k+np-1)];

Tmult=zeros(k,1)';
for i=k+1:floor(np/2)
  Tmult=[Tmult,(Tmult(i-1)+1)];
end
Tmult=[Tmult,(ones(k-1,1)'*(Tmult(floor(np/2))+1))];
```

```
for i=floor(np/2)+k:np
  Tmult=[Tmult,(Tmult(i-1)+1)];
end
Tmult=[Tmult,(ones(k,1)'*(Tmult(np)+1))];

Cnoep = Bspline_deboor(k,Tnoep,b,[k-1:0.01:np]);
Cep = Bspline_deboor(k,T,b,[T(1):0.01:T(length(T))]);
Cmult = Bspline_deboor(k,Tmult,b,[0:0.01:Tmult(length(Tmult))]);

if fig_noep then
scf(gcount);gcount=gcount+1;
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(Cnoep(1,:),Cnoep(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Bspline senza endpoint interpolation')
end

if fig_ep then
scf(gcount);gcount=gcount+1;
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(Cep(1,:),Cep(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Bspline con endpoint interpolation')
end

if fig_mult then
scf(gcount);gcount=gcount+1;
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(Cep(1,:),Cep(2,:),'c--');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(Cmult(1,:),Cmult(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Bspline con endpoint interpolation e nodo con molteplicitÃ  massima')
end

// Cepnorm = Bspline_curve(k,T,b,[T(1):0.01:T(length(T))]);
Cepnorm = Bspline_deboor(k,T,b,[T(1):0.01:T(length(T))]);
//Rimettere il normale!

if fig_norm then
scf(gcount);gcount=gcount+1;
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(Cepnorm(1,:),Cepnorm(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Bspline con endpoint interpolation, algoritmo normale')
// C = Bspline_deboor(b,[0,0,0,1,1,1,2,2,2,3,3,3],3,[0:0.01:3]);
end

bclosed=b
for i=1:k-1
     bclosed=[bclosed,bclosed(:,i)]
end
```

```
95
96  Tclosed = [0:1:(k+size(bclosed,2)-1)];
97  Cclosed = Bspline_deboor(k,Tclosed,bclosed,[k-1:0.01:size(bclosed,2)]);
98
99  if fig_closed then
100 scf(gcount);gcount=gcount+1;
101 plot(bclosed(1,:),bclosed(2,:),'ro-');
102 handler=get("current_axes");
103 handler.isoview="on";handler.grid=[12,12];
104 plot(Cclosed(1,:),Cclosed(2,:),'b-');
105 handler=get("current_axes");
106 handler.isoview="on";handler.grid=[12,12];
107 xtitle('Bspline chiusa')
108 // C = Bspline_deboor(b,[0,0,0,1,1,1,2,2,2,3,3,3],3,[0:0.01:3]);
109 end
110
111 nb=b; nT=T;
112 for i=1:length(T)-1
113       if T(i)~=T(i+1) then
114          [nb,nT] = Bspline_knotins(k,nT,nb,((T(i)+T(i+1))/2));
115       end
116 end
117
118 Cknotins = Bspline_deboor(k,nT,nb,[nT(1):0.01:nT(length(nT))]);
119
120 if fig_knotins then
121 scf(gcount);gcount=gcount+1;
122 plot(b(1,:),b(2,:),'ro-');
123 handler=get("current_axes");
124 handler.isoview="on";handler.grid=[12,12];
125 plot(Cep(1,:),Cep(2,:),'b-');
126 handler=get("current_axes");
127 handler.isoview="on";handler.grid=[12,12];
128 plot(nb(1,:),nb(2,:),'go-');
129 handler=get("current_axes");
130 handler.isoview="on";handler.grid=[12,12];
131 plot(Cknotins(1,:),Cknotins(2,:),'c--');
132 handler=get("current_axes");
133 handler.isoview="on";handler.grid=[12,12];
134 xtitle('Bspline knot insertion')
135 end
136
137
138 beta1 = ones(np,1)'
139 beta2 = ones(np,1)'*np
140
141 beta1(floor(np/2))=beta1(floor(np/2))+np-1
142 beta2(floor(np/2))=beta2(floor(np/2))-np+1
143
144 Crat1 = Bspline_rational(k,T,b,beta1,[T(1):0.01:T(length(T))]);
145 Crat2 = Bspline_rational(k,T,b,beta2,[T(1):0.01:T(length(T))]);
146
147 if fig_rat then
148 scf(gcount);gcount=gcount+1;
149 plot(b(1,:),b(2,:),'ro-');
150 handler=get("current_axes");
151 handler.isoview="on";handler.grid=[12,12];
152 plot(Cep(1,:),Cep(2,:),'b-');
153 handler=get("current_axes");
154 handler.isoview="on";handler.grid=[12,12];
155 plot(Crat1(1,:),Crat1(2,:),'c-');
156 handler=get("current_axes");
157 handler.isoview="on";handler.grid=[12,12];
158 plot(Crat2(1,:),Crat2(2,:),'g-');
159 handler=get("current_axes");
```

```
handler.isoview="on";handler.grid=[12,12];
end

Tcirc = [0,0,0,1/4,1/4,1/2,1/2,3/4,3/4,1,1,1];
betacirc = [sqrt(2),1,sqrt(2),1,sqrt(2),1,sqrt(2),1,sqrt(2)];
bcirc = [[1;0],[1;1],[0;1],[-1;1],[-1;0],[-1;-1],[0;-1],[1;-1],[1;0]];
Circ = Bspline_rational(3,Tcirc,bcirc,betacirc,[0:0.01:1]);

if fig_circ then
scf(gcount);gcount=gcount+1;
plot(bcirc(1,:),bcirc(2,:),'ro-');
plot(bcirc(1,:)*1.2,bcirc(2,:)*-2,'--');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(Circ(1,:),Circ(2,:),'b');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
end
```

Disegno della base delle bspline

```
function base = Bspline_plot_base(k,T,t)
lt = length(t)
lT = length(T)

base = []
for p=1:lt
      for i=1:lT-k
      base(1,p,i) = t(p)
      base(2,p,i) = Bspline_base(i,k,T,t(p))
      end
end

for i=1:lT-k
plot(base(1,:,i),base(2,:,i),'Color',[(i/(lT-k)),0.5,(i/(lT-k))])
end
plot(base(1,:,1),(sum(base(2,:,:),3)),'r')
plot(0,1.1)
handler=get("current_axes");
handler.isoview="off";handler.grid=[12,12];
endfunction
```

A.4 Interpolazione: esempi del testo

Listing:

```
exec glib.sci;
gcount=1;

fig_param=1
fig_fmill=1
fig_bessel=1
// fig_fmillvsbessel=1
fig_c2_nat=1
fig_c2_nat_hodo=1
fig_c2_nak=1
fig_c2_nak_hodo=1
fig_c2_periodic=1
fig_c2_periodic_hodo=1
```

```
fig_c2_3d=1
fig_c2_nu_nat=1
fig_c2_nu_closed=1

b = mouse_input_2d(input('Numero di punti? '));
[d,np]=size(b);

// w = [(b(:,2:np)-b(:,1:np-1))]
w=ones(d,np)
C_unif = Interp_c1(b,w,l_par_unif(b),500);
C_cl = Interp_c1(b,w,l_par_cl(b),500);
C_ce = Interp_c1(b,w,l_par_centrip(b),500);

if fig_param then
scf(gcount);gcount=gcount+1;
subplot(2,2,1)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif(1,:),C_unif(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione uniforme')
subplot(2,2,2)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_cl(1,:),C_cl(2,:),'c-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione chord length')
subplot(2,2,3)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_ce(1,:),C_ce(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione centripeta')
subplot(2,2,4)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif(1,:),C_unif(2,:),'b-');
plot(C_cl(1,:),C_cl(2,:),'c-');
plot(C_ce(1,:),C_ce(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazioni a confronto')
end

C_unif_fmill = Interp_c1(b, l_tang_fmill(b,l_par_unif(b)),l_par_unif(b),500);
C_cl_fmill = Interp_c1(b, l_tang_fmill(b,l_par_cl(b)),l_par_cl(b),500);
C_ce_fmill = Interp_c1(b, l_tang_fmill(b,l_par_centrip(b)),l_par_centrip(b),500);
if fig_fmill then
scf(gcount);gcount=gcount+1;
xtitle('fmill')
subplot(2,2,1)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_fmill(1,:),C_unif_fmill(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione uniforme')
```

```
subplot(2,2,2)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_cl_fmill(1,:),C_cl_fmill(2,:),'c-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione chord length')
subplot(2,2,3)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_ce_fmill(1,:),C_ce_fmill(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione centripeta')
subplot(2,2,4)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_fmill(1,:),C_unif_fmill(2,:),'b-');
plot(C_cl_fmill(1,:),C_cl_fmill(2,:),'c-');
plot(C_ce_fmill(1,:),C_ce_fmill(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazioni a confronto')
end

C_unif_bessel = Interp_c1(b, l_tang_bessel(b,l_par_unif(b)),l_par_unif(b),500);
C_cl_bessel = Interp_c1(b, l_tang_bessel(b,l_par_cl(b)),l_par_cl(b),500);
C_ce_bessel = Interp_c1(b, l_tang_bessel(b,l_par_centrip(b)),l_par_centrip(b),500);
if fig_bessel then
scf(gcount);gcount=gcount+1;
xtitle('bessel')
subplot(2,2,1)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_bessel(1,:),C_unif_bessel(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione uniforme')
subplot(2,2,2)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_cl_bessel(1,:),C_cl_bessel(2,:),'c-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione chord length')
subplot(2,2,3)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_ce_bessel(1,:),C_ce_bessel(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione centripeta')
subplot(2,2,4)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_bessel(1,:),C_unif_bessel(2,:),'b-');
plot(C_cl_bessel(1,:),C_cl_bessel(2,:),'c-');
plot(C_ce_bessel(1,:),C_ce_bessel(2,:),'m-');
```

```
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazioni a confronto')
end

[C_unif_c2_nat,h1_c2_nat,h2_c2_nat] = Interp_c1(b, l_tang_c2_nat(b,l_par_unif(b)),
    l_par_unif(b),500);
[C_cl_c2_nat,h1_c2_cl,h2_c2_cl] = Interp_c1(b, l_tang_c2_nat(b,l_par_cl(b)),l_par_cl(b)
    ,500);
[C_ce_c2_nat,h1_c2_ce,h2_c2_ce] = Interp_c1(b, l_tang_c2_nat(b,l_par_centrip(b)),
    l_par_centrip(b),500);
if fig_c2_nat then
scf(gcount);gcount=gcount+1;
xtitle('c2_nat')
subplot(2,2,1)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nat(1,:),C_unif_c2_nat(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione uniforme / natural spline')
subplot(2,2,2)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_cl_c2_nat(1,:),C_cl_c2_nat(2,:),'c-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione chord length')
subplot(2,2,3)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_ce_c2_nat(1,:),C_ce_c2_nat(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione centripeta')
subplot(2,2,4)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nat(1,:),C_unif_c2_nat(2,:),'b-');
plot(C_cl_c2_nat(1,:),C_cl_c2_nat(2,:),'c-');
plot(C_ce_c2_nat(1,:),C_ce_c2_nat(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazioni a confronto')
end

if fig_c2_nat_hodo then
scf(gcount);gcount=gcount+1;
xtitle('c2_nat')
subplot(2,3,1)
plot(h1_c2_nat(1,:),h1_c2_nat(2,:),'b-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione uniforme / natural spline, hodograph 1')
subplot(2,3,2)
plot(h1_c2_cl(1,:),h1_c2_cl(2,:),'c-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione chord length, hodograph 1')
subplot(2,3,3)
plot(h1_c2_ce(1,:),h1_c2_ce(2,:),'m-');
```

```
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione centripeta, hodograoh 1')
subplot(2,3,4)
plot(h2_c2_nat(1,:),h2_c2_nat(2,:),'b-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione uniforme, hodograph 2')
subplot(2,3,5)
plot(h2_c2_cl(1,:),h2_c2_cl(2,:),'c-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione chord length, hodograph 2')
subplot(2,3,6)
plot(h2_c2_ce(1,:),h2_c2_ce(2,:),'m-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione centripeta, hodograph 2')
end


[C_unif_c2_nak,h1_c2_nak,h2_c2_nak] = Interp_c1(b, l_tang_c2_nak(b,l_par_unif(b)),
    l_par_unif(b),500);
[C_cl_c2_nak,h1_c2_cl,h2_c2_cl] = Interp_c1(b, l_tang_c2_nak(b,l_par_cl(b)),l_par_cl(b)
    ,500);
[C_ce_c2_nak,h1_c2_ce,h2_c2_ce] = Interp_c1(b, l_tang_c2_nak(b,l_par_centrip(b)),
    l_par_centrip(b),500);
if fig_c2_nak then
scf(gcount);gcount=gcount+1;
xtitle('c2_nak')
subplot(2,2,1)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nak(1,:),C_unif_c2_nak(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione uniforme  / not-a-knot')
subplot(2,2,2)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_cl_c2_nak(1,:),C_cl_c2_nak(2,:),'c-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione chord length')
subplot(2,2,3)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_ce_c2_nak(1,:),C_ce_c2_nak(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione centripeta')
subplot(2,2,4)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nak(1,:),C_unif_c2_nak(2,:),'b-');
plot(C_cl_c2_nak(1,:),C_cl_c2_nak(2,:),'c-');
plot(C_ce_c2_nak(1,:),C_ce_c2_nak(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazioni a confronto')
end
```

```
268
269 if fig_c2_nak_hodo then
270 scf(gcount);gcount=gcount+1;
271 xtitle('c2_nak')
272 subplot(2,3,1)
273 plot(h1_c2_nak(1,:),h1_c2_nak(2,:),'b-');
274 handler=get("current_axes");
275 handler.grid=[12,12];
276 xtitle('Parametrizzazione uniforme /not-a-knot, hodograph 1')
277 subplot(2,3,2)
278 plot(h1_c2_cl(1,:),h1_c2_cl(2,:),'c-');
279 handler=get("current_axes");
280 handler.grid=[12,12];
281 xtitle('Parametrizzazione chord length, hodograph 1')
282 subplot(2,3,3)
283 plot(h1_c2_ce(1,:),h1_c2_ce(2,:),'m-');
284 handler=get("current_axes");
285 handler.grid=[12,12];
286 xtitle('Parametrizzazione centripeta, hodograoh 1')
287 subplot(2,3,4)
288 plot(h2_c2_nak(1,:),h2_c2_nak(2,:),'b-');
289 handler=get("current_axes");
290 handler.grid=[12,12];
291 xtitle('Parametrizzazione uniforme, hodograph 2')
292 subplot(2,3,5)
293 plot(h2_c2_cl(1,:),h2_c2_cl(2,:),'c-');
294 handler=get("current_axes");
295 handler.grid=[12,12];
296 xtitle('Parametrizzazione chord length, hodograph 2')
297 subplot(2,3,6)
298 plot(h2_c2_ce(1,:),h2_c2_ce(2,:),'m-');
299 handler=get("current_axes");
300 handler.grid=[12,12];
301 xtitle('Parametrizzazione centripeta, hodograph 2')
302 end
303
304 [C_unif_c2_periodic,h1_c2_periodic,h2_c2_periodic] = Interp_c1(b, l_tang_c2_periodic(b,
      l_par_unif(b)),l_par_unif(b),500);
305 [C_cl_c2_periodic,h1_c2_cl,h2_c2_cl] = Interp_c1(b, l_tang_c2_periodic(b,l_par_cl(b)),
      l_par_cl(b),500);
306 [C_ce_c2_periodic,h1_c2_ce,h2_c2_ce] = Interp_c1(b, l_tang_c2_periodic(b,l_par_centrip(b))
      ,l_par_centrip(b),500);
307 if fig_c2_periodic then
308 scf(gcount);gcount=gcount+1;
309 xtitle('c2_periodic')
310 subplot(2,2,1)
311 plot(b(1,:),b(2,:),'ro-');
312 handler=get("current_axes");
313 handler.isoview="on";handler.grid=[12,12];
314 plot(C_unif_c2_periodic(1,:),C_unif_c2_periodic(2,:),'b-');
315 plot(C_unif_c2_periodic(1,:)+(b(1,np)-b(1,1)),C_unif_c2_periodic(2,:)+(b(2,np)-b(2,1)),'c-
      ');
316 plot(C_unif_c2_periodic(1,:)-(b(1,np)-b(1,1)),C_unif_c2_periodic(2,:)-(b(2,np)-b(2,1)),'c-
      ');
317 handler=get("current_axes");
318 handler.isoview="on";handler.grid=[12,12];
319 xtitle('Parametrizzazione uniforme / periodica')
320 subplot(2,2,2)
321 plot(b(1,:),b(2,:),'ro-');
322 handler=get("current_axes");
323 handler.isoview="on";handler.grid=[12,12];
324 plot(C_cl_c2_periodic(1,:),C_cl_c2_periodic(2,:),'c-');
325 plot(C_cl_c2_periodic(1,:)+(b(1,np)-b(1,1)),C_cl_c2_periodic(2,:)+(b(2,np)-b(2,1)),'g-');
326 plot(C_cl_c2_periodic(1,:)-(b(1,np)-b(1,1)),C_cl_c2_periodic(2,:)-(b(2,np)-b(2,1)),'g-');
327 handler=get("current_axes");
```

```
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione chord length')
subplot(2,2,3)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_ce_c2_periodic(1,:),C_ce_c2_periodic(2,:),'m-');
plot(C_ce_c2_periodic(1,:)+(b(1,np)-b(1,1)),C_ce_c2_periodic(2,:)+(b(2,np)-b(2,1)),'c-');
plot(C_ce_c2_periodic(1,:)-(b(1,np)-b(1,1)),C_ce_c2_periodic(2,:)-(b(2,np)-b(2,1)),'c-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione centripeta')
subplot(2,2,4)
plot(b(1,:),b(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_periodic(1,:),C_unif_c2_periodic(2,:),'b-');
plot(C_cl_c2_periodic(1,:),C_cl_c2_periodic(2,:),'c-');
plot(C_ce_c2_periodic(1,:),C_ce_c2_periodic(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazioni a confronto')
end

if fig_c2_periodic_hodo then
scf(gcount);gcount=gcount+1;
xtitle('c2_periodic')
subplot(2,3,1)
plot(h1_c2_periodic(1,:),h1_c2_periodic(2,:),'b-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione uniforme /not-a-knot, hodograph 1')
subplot(2,3,2)
plot(h1_c2_cl(1,:),h1_c2_cl(2,:),'c-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione chord length, hodograph 1')
subplot(2,3,3)
plot(h1_c2_ce(1,:),h1_c2_ce(2,:),'m-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione centripeta, hodograoh 1')
subplot(2,3,4)
plot(h2_c2_periodic(1,:),h2_c2_periodic(2,:),'b-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione uniforme, hodograph 2')
subplot(2,3,5)
plot(h2_c2_cl(1,:),h2_c2_cl(2,:),'c-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione chord length, hodograph 2')
subplot(2,3,6)
plot(h2_c2_ce(1,:),h2_c2_ce(2,:),'m-');
handler=get("current_axes");
handler.grid=[12,12];
xtitle('Parametrizzazione centripeta, hodograph 2')
end


z = [0:(1/np):1];
b3d = [b;z(1:np)];
[C_unif_c2_3d_3d_nat,h1_c2_nat,h2_c2_nat] = Interp_c1(b3d, l_tang_c2_nat(b3d,l_par_unif(
    b3d)),l_par_unif(b3d),500);
```

```
[C_unif_c2_3d_nak,h1_c2_nak,h2_c2_nak] = Interp_c1(b3d, I_tang_c2_nak(b3d,I_par_unif(b3d))
    ,I_par_unif(b3d),500);
[C_unif_c2_3d_periodic,h1_c2_periodic,h2_c2_periodic] = Interp_c1(b3d, I_tang_c2_periodic(
    b3d,I_par_unif(b3d)),I_par_unif(b3d),500);

if fig_c2_3d then
scf(gcount);gcount=gcount+1;
param3d(b3d(1,:),b3d(2,:),b3d(3,:),20,-20);
handler = gce();
handler.foreground=color('red') //red
handler.mark_mode='on' //enable marks
handler.mark_size=1
handler.mark_foreground=color('red') //red marks
handler.mark_style=1 //cross
param3d(C_unif_c2_3d_3d_nat(1,:),C_unif_c2_3d_3d_nat(2,:),C_unif_c2_3d_3d_nat(3,:),20,-20)
    ;
handler = gce();
handler.foreground=color('blue') //blue
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('3d spline naturale');

scf(gcount);gcount=gcount+1;
param3d(b3d(1,:),b3d(2,:),b3d(3,:),20,-20);
handler = gce();
handler.foreground=color('red') //red
handler.mark_mode='on' //enable marks
handler.mark_size=1
handler.mark_foreground=color('red') //red marks
handler.mark_style=1 //cross
param3d(C_unif_c2_3d_nak(1,:),C_unif_c2_3d_nak(2,:),C_unif_c2_3d_nak(3,:),20,-20);
handler = gce();
handler.foreground=color('blue') //blue
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('3d spline not a knot');

scf(gcount);gcount=gcount+1;
param3d(b3d(1,:),b3d(2,:),b3d(3,:),20,-20);
handler = gce();
handler.foreground=color('red') //red
handler.mark_mode='on' //enable marks
handler.mark_size=1
handler.mark_foreground=color('red') //red marks
handler.mark_style=1 //cross
param3d(C_unif_c2_3d_periodic(1,:),C_unif_c2_3d_periodic(2,:),C_unif_c2_3d_periodic(3,:)
    ,20,-20);
handler = gce();
handler.foreground=color('blue') //blue
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
param3d(C_unif_c2_3d_periodic(1,:)+(b3d(1,np)-b3d(1,1)),C_unif_c2_3d_periodic(2,:)+(b3d(2,
    np)-b3d(2,1)),C_unif_c2_3d_periodic(3,:)+(b3d(3,np)-b3d(3,1)),20,-20);
handler = gce();
handler.foreground=color('blue') //blue
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
param3d(C_unif_c2_3d_periodic(1,:)-(b3d(1,np)-b3d(1,1)),C_unif_c2_3d_periodic(2,:)-(b3d(2,
    np)-b3d(2,1)),C_unif_c2_3d_periodic(3,:)-(b3d(3,np)-b3d(3,1)),20,-20);
handler = gce();
handler.foreground=color('blue') //blue
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('3d spline periodica');
end
```

```
450
451 nu=[]
452 nu1=[]
453 nu2=[]
454 for i=1:np
455 nu(i)=0;
456 nu1(i)=20;
457 nu2(i)=200;
458 end
459 bclosed = [b,b(:,1)]
460
461 [C_unif_c2_nat,h1_c2_nat,h2_c2_nat] = Interp_c1(b, l_tang_nu_nat(b,l_par_unif(b),nu),
    l_par_unif(b),500);
462 [C_unif_c2_nat1,h1_c2_nat,h2_c2_nat1] = Interp_c1(b, l_tang_nu_nat(b,l_par_unif(b),nu1),
    l_par_unif(b),500);
463 [C_unif_c2_nat2,h1_c2_nat,h2_c2_nat2] = Interp_c1(b, l_tang_nu_nat(b,l_par_unif(b),nu2),
    l_par_unif(b),500);
464 if fig_c2_nu_nat then
465 scf(gcount);gcount=gcount+1;
466 xtitle('c2_nat')
467 subplot(2,2,1)
468 plot(b(1,:),b(2,:),'ro-');
469 handler=get("current_axes");
470 handler.isoview="on";handler.grid=[12,12];
471 plot(C_unif_c2_nat(1,:),C_unif_c2_nat(2,:),'b-');
472 handler=get("current_axes");
473 handler.isoview="on";handler.grid=[12,12];
474 xtitle('Parametrizzazione uniforme / natural nu spline / nu(i)=0')
475 subplot(2,2,2)
476 plot(b(1,:),b(2,:),'ro-');
477 handler=get("current_axes");
478 handler.isoview="on";handler.grid=[12,12];
479 plot(C_unif_c2_nat1(1,:),C_unif_c2_nat1(2,:),'c-');
480 handler=get("current_axes");
481 handler.isoview="on";handler.grid=[12,12];
482 xtitle('nu(i)=20')
483 subplot(2,2,3)
484 plot(b(1,:),b(2,:),'ro-');
485 handler=get("current_axes");
486 handler.isoview="on";handler.grid=[12,12];
487 plot(C_unif_c2_nat2(1,:),C_unif_c2_nat2(2,:),'m-');
488 handler=get("current_axes");
489 handler.isoview="on";handler.grid=[12,12];
490 xtitle('nu(i)=200')
491 subplot(2,2,4)
492 plot(b(1,:),b(2,:),'ro-');
493 handler=get("current_axes");
494 handler.isoview="on";handler.grid=[12,12];
495 plot(C_unif_c2_nat(1,:),C_unif_c2_nat(2,:),'b-');
496 plot(C_unif_c2_nat1(1,:),C_unif_c2_nat1(2,:),'c-');
497 plot(C_unif_c2_nat2(1,:),C_unif_c2_nat2(2,:),'m-');
498 handler=get("current_axes");
499 handler.isoview="on";handler.grid=[12,12];
500 xtitle('Distribuzioni a confronto')
501 end
502
503
504
505
506 nu=[]
507 nu1=[]
508 nu2=[]
509 for i=1:np
510 nu(i)=0;
511 nu1(i)=20;
```

```
nu2(i)=200;
end
[C_unif_c2_nat,h1_c2_nat,h2_c2_nat] = Interp_c1(bclosed, l_tang_nu_closed(bclosed,
    l_par_unif(bclosed),nu),l_par_unif(bclosed),500);
[C_unif_c2_nat1,h1_c2_nat,h2_c2_nat1] = Interp_c1(bclosed, l_tang_nu_closed(bclosed,
    l_par_unif(bclosed),nu1),l_par_unif(bclosed),500);
[C_unif_c2_nat2,h1_c2_nat,h2_c2_nat2] = Interp_c1(bclosed, l_tang_nu_closed(bclosed,
    l_par_unif(bclosed),nu2),l_par_unif(bclosed),500);
if fig_c2_nu_closed then
scf(gcount);gcount=gcount+1;
xtitle('c2_nat')
subplot(2,2,1)
plot(bclosed(1,:),bclosed(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nat(1,:),C_unif_c2_nat(2,:),'b-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Parametrizzazione uniforme / natural nu spline / nu(i)=0')
subplot(2,2,2)
plot(bclosed(1,:),bclosed(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nat1(1,:),C_unif_c2_nat1(2,:),'c-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('nu(i)=20')
subplot(2,2,3)
plot(bclosed(1,:),bclosed(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nat2(1,:),C_unif_c2_nat2(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('nu(i)=200')
subplot(2,2,4)
plot(bclosed(1,:),bclosed(2,:),'ro-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
plot(C_unif_c2_nat(1,:),C_unif_c2_nat(2,:),'b-');
plot(C_unif_c2_nat1(1,:),C_unif_c2_nat1(2,:),'c-');
plot(C_unif_c2_nat2(1,:),C_unif_c2_nat2(2,:),'m-');
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
xtitle('Distribuzioni a confronto')
end
```

A.5 Superfici parametriche: esempi dal testo

Listing:

```
exec glib.sci;
gcount=1;

fig_sphere=0
fig_cone=0
get_input=0
fig_rot=0
fig_rig=0
fig_bez=0
```

```
fig_degelev=0
fig_subdiv=0
fig_subdiv2=0
fig_cast_tri=0
fig_tri_degelev=1


Sx1 = [];
Sy1 = [];
Sz1 = [];
for v=0:%pi/12:2*%pi
      for u=-%pi/2:%pi/36:%pi/2
         Sx1 = [Sx1, cos(u)*cos(v)];
         Sy1 = [Sy1, cos(u)*sin(v)];
         Sz1 = [Sz1, sin(u)];
      end
end

Sx2 = [];
Sy2 = [];
Sz2 = [];
for u=-%pi/2:%pi/12:%pi/2
      for v=0:%pi/36:2*%pi
         Sx2 = [Sx2, cos(u)*cos(v)];
         Sy2 = [Sy2, cos(u)*sin(v)];
         Sz2 = [Sz2, sin(u)];
      end
end

if fig_sphere then
scf(gcount);gcount=gcount+1;
param3d1(Sx1,Sy1,Sz1);
handler = gce();
// handler.foreground=color('red') //red
handler.line_mode='off'
handler.mark_mode='on' //enable marks
// handler.mark_size=1
handler.mark_foreground=color('blue') //red marks
handler.mark_style=0 //dot
handler.mark_size=0
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
scf(gcount);gcount=gcount+1;
param3d1(Sx2,Sy2,Sz2);
handler = gce();
// handler.foreground=color('red') //red
handler.line_mode='off'
handler.mark_mode='on' //enable marks
// handler.mark_size=1
handler.mark_foreground=color('blue') //red marks
handler.mark_style=0 //dot
handler.mark_size=0
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
scf(gcount);gcount=gcount+1;
param3d1(Sx2,Sy2,Sz2);
handler = gce();
// handler.foreground=color('red') //red
handler.line_mode='off'
handler.mark_mode='on' //enable marks
// handler.mark_size=1
handler.mark_foreground=color('blue') //red marks
handler.mark_style=0 //dot
handler.mark_size=0
handler=get("current_axes");
```

```
handler.isoview="on";handler.grid=[12,12];
param3d1(Sx1,Sy1,Sz1);
handler = gce();
// handler.foreground=color('red') //red
handler.line_mode='off'
handler.mark_mode='on' //enable marks
// handler.mark_size=1
handler.mark_foreground=color('blue') //red marks
handler.mark_style=0 //dot
handler.mark_size=0
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
scf(gcount);gcount=gcount+1;
param3d1(Sx1,Sy1,Sz1);
handler = gce();
handler.foreground=color('blue') //red marks
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
param3d1(Sx2,Sy2,Sz2)
handler = gce();
handler.foreground=color('blue') //red marks
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
end

Cx1 = [];
Cy1 = [];
Cz1 = [];
for v=0:%pi/50:2*%pi
      for u=-%pi/2:%pi/50:%pi/2
         Cx1 = [Cx1, (u)*cos(v)];
         Cy1 = [Cy1, (u)*sin(v)];
         Cz1 = [Cz1, (u)];
      end
end


if fig_cone then
scf(gcount);gcount=gcount+1;
param3d1(Cx1,Cy1,Cz1)
handler = gce();
// handler.foreground=color('red') //red
handler.line_mode='off'
handler.mark_mode='on' //enable marks
// handler.mark_size=1
handler.mark_foreground=color('blue') //red marks
handler.mark_style=0 //dot
handler.mark_size=0
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
end

if get_input then
p = mouse_input_2d(input('Numero di punti? '));
else
p = [[0,1];[1,0]]
end
//faster but wrong order for nice plot
// Cu = Bezier_casteljau(p,[0:0.02:1]);
// Crx = []
// Cry = []
// Crz = []
// for v=0:%pi/20:2*%pi
//       Crx = [Crx, Cu(1,:)*cos(v)];
//       Cry = [Cry, Cu(1,:)*sin(v)];
```

```
//        Crz = [Crz, Cu(2,:)];
// end

Crx = [];
Cry = [];
Crz = [];
for u=0:0.1:1
      for v=0:%pi/10:2*%pi
         Cu = Bezier_casteljau(p,u);
         Crx = [Crx, Cu(1,:)*cos(v)];
         Cry = [Cry, Cu(1,:)*sin(v)];
         Crz = [Crz, Cu(2,:)];
      end
end

if fig_rot then
scf(gcount);gcount=gcount+1;
param3d1(Crx,Cry,Crz);
handler = gce();
handler.foreground=color('blue') //red
handler.line_mode='on'
handler.mark_mode='off' //enable marks
// handler.mark_size=1
handler.mark_foreground=color('blue') //red marks
handler.mark_style=0 //dot
handler.mark_size=0
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
end

Crigx = [];
Crigy = [];
Crigz = [];

for u=0:0.02:1
      for v=0:1:1
         Cu1 = [Bezier_casteljau(p,u);0.5];
         Cu2 = [Cu1(2,:);Cu1(1,:);0];
         Crigx = [Crigx, (1-v)*Cu1(1,:)+v*Cu2(1,:) ];
         Crigy = [Crigy, (1-v)*Cu1(2,:)+v*Cu2(2,:) ];
         Crigz = [Crigz, (1-v)*Cu1(3,:)+v*Cu2(3,:) ];
      end
end

if fig_rig then
scf(gcount);gcount=gcount+1;
for i=1:2:length(Crigx)-1
      param3d1(Crigx(i:i+1),Crigy(i:i+1),Crigz(i:i+1));
      handler = gce();
      handler.foreground=color('blue') //red;
      handler.line_mode='on';
      handler.mark_mode='off' //enable marks
      // handler.mark_size=1
end
//       param3d1()
      handler=get("current_axes");
      handler.isoview="on";handler.grid=[12,12];
end

d=create_grid(6,6);
c=Bezier_patch_casteljau(d,[0:0.01:1],[0:0.01:1]);
if fig_bez then
scf(gcount);gcount=gcount+1;
plot_3d_grid(d,'red');
plot_3d_grid(c,'blue');
```

```
end


delev=create_grid(3,3);
delevated=Bezier_patch_degelev(delev);
delevated2=Bezier_patch_degelev(delevated);
cdegev=Bezier_patch_casteljau(delev,[0:0.02:1],[0:0.02:1]);
cdelev1=Bezier_patch_casteljau(delevated,[0:0.02:1],[0:0.02:1]);
cdelev2=Bezier_patch_casteljau(delevated2,[0:0.02:1],[0:0.02:1]);
if fig_degelev then
scf(gcount);gcount=gcount+1;
plot_3d_grid(delev,'red');
plot_3d_grid(delevated,'green');
plot_3d_grid(cdegev,'blue');
plot_3d_grid(cdelev1,'grey');

scf(gcount);gcount=gcount+1;
plot_3d_grid(delev,'red');
plot_3d_grid(delevated,'green');
plot_3d_grid(delevated2,'cyan');
plot_3d_grid(cdegev,'blue');
plot_3d_grid(cdelev1,'grey');
plot_3d_grid(cdelev2,'lightgrey');
end

subdiv=create_grid(3,3);
[s1,s2,s3,s4]=Bezier_patch_subdiv(subdiv,0.5);
cs0 = Bezier_patch_casteljau(subdiv,[0:0.05:1],[0:0.05:1]);
cs1=Bezier_patch_casteljau(s1,[0:0.1:1],[0:0.1:1]);
cs2=Bezier_patch_casteljau(s2,[0:0.1:1],[0:0.1:1]);
cs3=Bezier_patch_casteljau(s3,[0:0.1:1],[0:0.1:1]);
cs4=Bezier_patch_casteljau(s4,[0:0.1:1],[0:0.1:1]);
if fig_subdiv then
scf(gcount);gcount=gcount+1;
plot_3d_grid(subdiv,'magenta');
plot_3d_grid(s1,'black');
plot_3d_grid(s2,'blue');
plot_3d_grid(s3,'red');
plot_3d_grid(s4,'darkgreen');

plot_3d_grid(cs0,'lightgrey');
plot_3d_grid(cs1,'grey');
plot_3d_grid(cs2,'lightblue');
plot_3d_grid(cs3,'pink');
plot_3d_grid(cs4,'lightgreen');
end

subdiv=create_grid(3,3);
[s1,s2,s3,s4]=Bezier_patch_subdiv(subdiv,0.2);
cs0 = Bezier_patch_casteljau(subdiv,[0:0.05:1],[0:0.05:1]);
cs1=Bezier_patch_casteljau(s1,[0:0.1:1],[0:0.1:1]);
cs2=Bezier_patch_casteljau(s2,[0:0.1:1],[0:0.1:1]);
cs3=Bezier_patch_casteljau(s3,[0:0.1:1],[0:0.1:1]);
cs4=Bezier_patch_casteljau(s4,[0:0.1:1],[0:0.1:1]);
if fig_subdiv2 then
scf(gcount);gcount=gcount+1;
plot_3d_grid(subdiv,'magenta');
plot_3d_grid(s1,'black');
plot_3d_grid(s2,'blue');
plot_3d_grid(s3,'red');
plot_3d_grid(s4,'darkgreen');

plot_3d_grid(cs0,'lightgrey');
plot_3d_grid(cs1,'grey');
plot_3d_grid(cs2,'lightblue');
```

```
plot_3d_grid(cs3,'pink');
plot_3d_grid(cs4,'lightgreen');
end


u = [0:0.025:1];
Ctrip =  Bezier_tripatch_casteljau(d,u,u);

if fig_cast_tri then
scf(gcount);gcount=gcount+1;
plot_3d_tripatch(d,'red');
plot_3d_tripatch(Ctrip,'blue');
end

u=[0:0.025:1];
d = create_grid(4,4)
Ctrip = Bezier_tripatch_casteljau(d,u,u);
d_tri_deg = Bezier_tripatch_degelev(d);
Cdeg =  Bezier_tripatch_casteljau(d_tri_deg,u,u);

if fig_tri_degelev then
scf(gcount);gcount=gcount+1;
plot_3d_tripatch(d,'red');
plot_3d_tripatch(d_tri_deg,'green');
plot_3d_tripatch(Ctrip,'pink');
plot_3d_tripatch(Cdeg,'lightblue');
end
```

Plot di una griglia in 3d

```
function plot_3d_grid(g,colore)
[x,y,z] = size(g)
gt = []

for i=1:z
 param3d1(g(1,:,i),g(2,:,i),g(3,:,i));
 handler = gce();
handler.foreground=color(colore)
end

for iz=1:z
       gt(:,:,iz) = matrix(g(:,iz,:),3,-1)
end

 for i=1:y
       param3d1(gt(1,:,i),gt(2,:,i),gt(3,:,i));
       handler = gce();
       handler.foreground=color(colore)
 end

handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];

endfunction
```

Creazione di una griglia random per gli esempi

```
function d=create_grid(x,y)
d = []
for iy=1:y
       for ix=1:x
       d(:,ix,iy) = [ix,iy,rand()*(x+y)/2]
       end
```

```
end
endfunction
```

Plot griglia triangolare in 3D

```
function plot_3d_tripatch(g,colore)
[x,y,z] = size(g)
gt = []

for i=1:z
      param3d1(g(1,1:y-i+1,i),g(2,1:y-i+1,i),g(3,1:y-i+1,i));
      handler = gce();
      handler.foreground=color(colore)
end

for iz=1:z
      gt(:,:,iz) = matrix(g(:,iz,:),3,-1)
end

for i=1:y
       param3d1(gt(1,1:y-i+1,i),gt(2,1:y-i+1,i),gt(3,1:y-i+1,i));
       handler = gce();
       handler.foreground=color(colore)
end

gtransp = g

for i=1:y
      for j=1:z-i+1
         gtransp(1:3,i,j) = g(1:3,i,z-i+1-j+1)
      end
end
g

for i=1:y
      param3d1(gtransp(1,1:y-i+1,i),gtransp(2,1:y-i+1,i),gtransp(3,1:y-i+1,i));
      handler = gce();
      handler.foreground=color(colore)
end

handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
// handler.view="2d"
endfunction
```

A.6 Mouse input

Listing:

```
function points = mouse_input_2d(n)
//plot([0,2],[0,2],'.')
//square(0,0,1,1);
plot([0,1],[0,1],'w.')
handler=get("current_axes");
handler.isoview="on";handler.grid=[12,12];
points=locate(n,1);
endfunction
```

A.7 Lista di import per tutti gli script

Listing:

```
exec mouse_input_2d.sci;
exec crossprod.sci;
exec curvature.sci;
exec torsione.sci;
exec Bezier_casteljau.sci;
exec Bezier_degelev.sci;
exec Bezier_hodograph.sci;
exec Bezier_subdiv.sci;
exec RATBezier_casteljau.sci;
exec RATBezier_scalebeta.sci;
exec RATBezier_coniche.sci;
exec Bspline_deboor.sci;
exec Bspline_base.sci;
exec Bspline_curve.sci;
exec Bspline_plot_base.sci;
exec Bspline_knotins.sci;
exec Bspline_rational.sci;
exec Bezier_plot_base.sci;
exec I_par_unif.sci;
exec I_par_cl.sci;
exec I_par_centrip.sci;
exec Interp_c1.sci;
exec I_tang_fmill.sci;
exec I_tang_bessel.sci;
exec I_tang_c2_nat.sci;
exec I_tang_c2_nak.sci;
exec I_tang_c2_periodic.sci;
exec I_tang_nu_nat.sci;
exec I_tang_nu_closed.sci;
exec Bezier_patch_casteljau.sci;
exec plot_3d_grid.sci;
exec create_grid.sci;
exec Bezier_patch_degelev.sci;
exec Bezier_patch_subdiv.sci;
exec Bezier_tripatch_casteljau.sci;
exec plot_3d_tripatch.sci;
exec Bezier_tripatch_degelev.sci;
```

Elenco delle figure

Bibliografia

[1] David Salomon, Curves and Surfaces for Computer Graphics, Springer-Verlag

[2] Les Piegl and Wayne Tiller, The NURBS Book, Springer-Verlag