

Fondamenti Numerici della Progettazione

Valentino Pediroda

pediroda@units.it

Tel: 040 558 3231

- **Parametrizzazione geometrica**
 - Curve di Bezier
 - B-spline quadratiche e cubiche
 - B-spline di grado n
 - Bezier e B-Spline razionali (Nurbs)
 - Cenni su superfici parametriche
- **DOE (Design of experiment)**
 - Random, Sobol
 - Fattoriale, fattoriale ridotto
 - Box-Benker, Latin Square
 - Taguchi
 - Analisi statistica dei dati (t-Student)
- **ALGORITMI DI OTTIMIZZAZIONE**
 - Metodologie mono obiettivo (Cauchy, Conjugate Gradient, Newton, Quasi Newton, BFGS, SQP)
 - Simplex
 - Simulated Annealing
 - Metodologie multi obiettivo (Algoritmo genetico)
 - Teoria dei giochi (Nash, Stackelberg, Pareto)
 - MCDM (Multi Criteria Decision Making)
- **SUPERFICI DI RISPOSTA**
 - Superfici lineari, quadratiche, serie Taylor
 - Kriging
 - Reti neurali
 - Processi gaussiani
 - Support Vector Machines
- **ROBUST DESIGN**
- **VISUALIZZAZIONE DI DATI IN SPAZI n -D**
 - Self Organizing Maps
 - Clustering

Libri consigliati

- *Curves and surfaces for CAGD*, Gerald Farin, Rheinbolt, Academic Press, 1997
- *Design of Experiments*, R. J. Del Vecchio, Hanser Publishers, 1997
- *Engineering Optimization*, Singiresu Rao, Wiley 1996
- *Neural Networks*, R. Rojas, Springer, 1996

Introduzione

Ogni problema ingegneristico che coinvolga un progetto si avvale più o meno esplicitamente di un processo di ottimizzazione.

Soprattutto in questi ultimi anni, con il crescente impiego del computer e della progettazione assistita, si è particolarmente sentita la necessità di integrare le tecniche quali il CAD, CAE, CFD con metodi di ottimizzazione numerica sempre più efficienti.

La progettazione di qualunque sistema ingegneristico può essere vista come una successione di decisioni da prendersi in determinati momenti dello sviluppo. Il fine ultimo di queste decisioni è sempre la possibilità di aumentare i benefici prodotto o di diminuire gli inconvenienti indotti; si nota quindi come nello sviluppo di un progetto ci si trova sempre di fronte ad una successione di ottimizzazioni, viste come il mezzo per ottenere il migliore risultato possibile.

Quindi l'ottimizzazione può essere vista come la ricerca del massimo o minimo di una funzione.

Progettazione

variabili \Rightarrow sistema \Rightarrow prestazioni

Ottimizzazione (numerica)

$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \Rightarrow$ solver $\Rightarrow \mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_m$

$$f : \mathcal{R}^n \Rightarrow \mathcal{R}^m$$

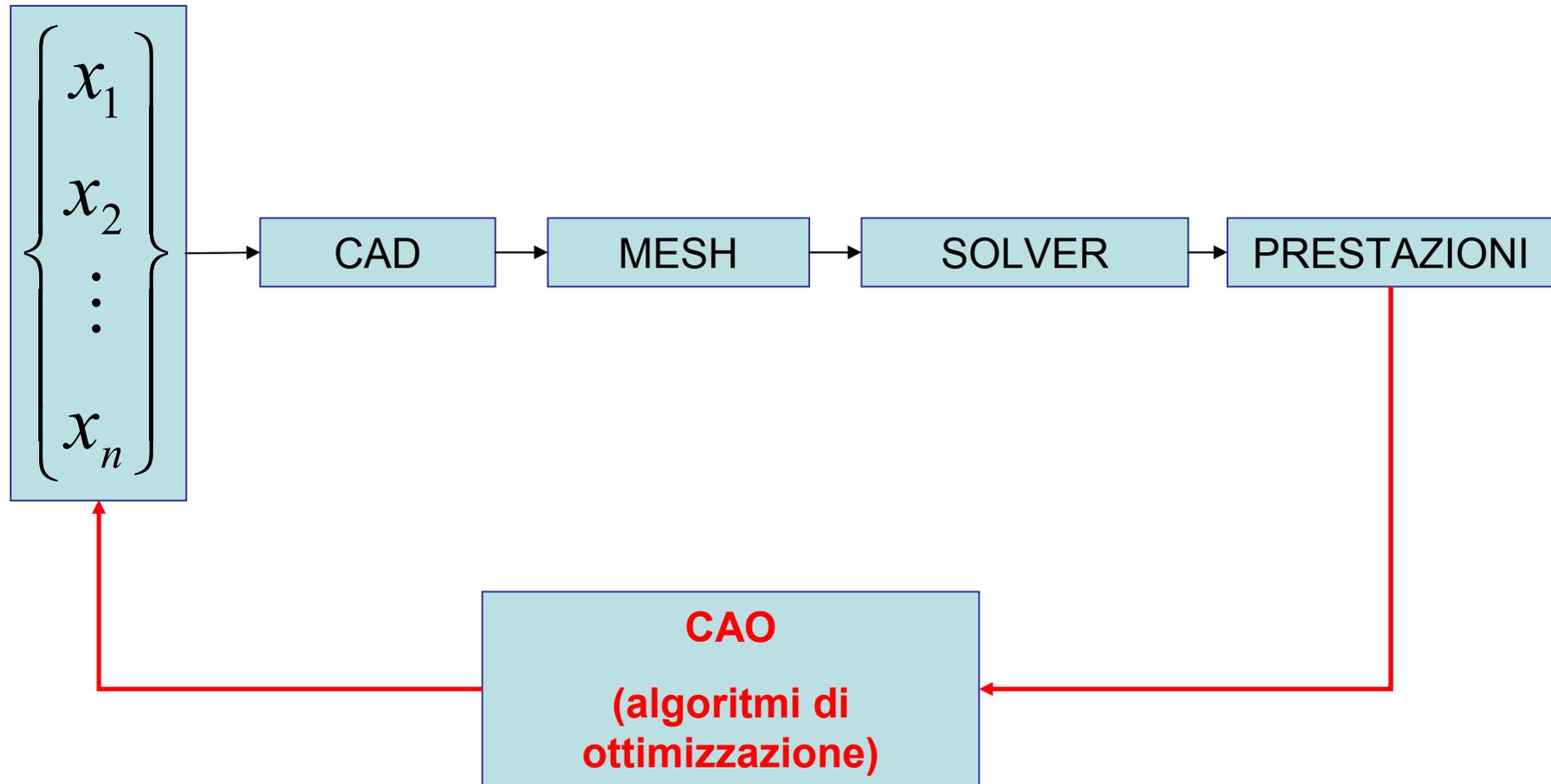
funzione obiettivo

Esempio progettazione motore :

Variabili: geometria cilindro, numero giri, Ciclo Otto-Diesel, numero tempi, etc

Funzione obiettivo: potenza, rendimento, peso, etc.

CAO (Computer Aided Optimisation)



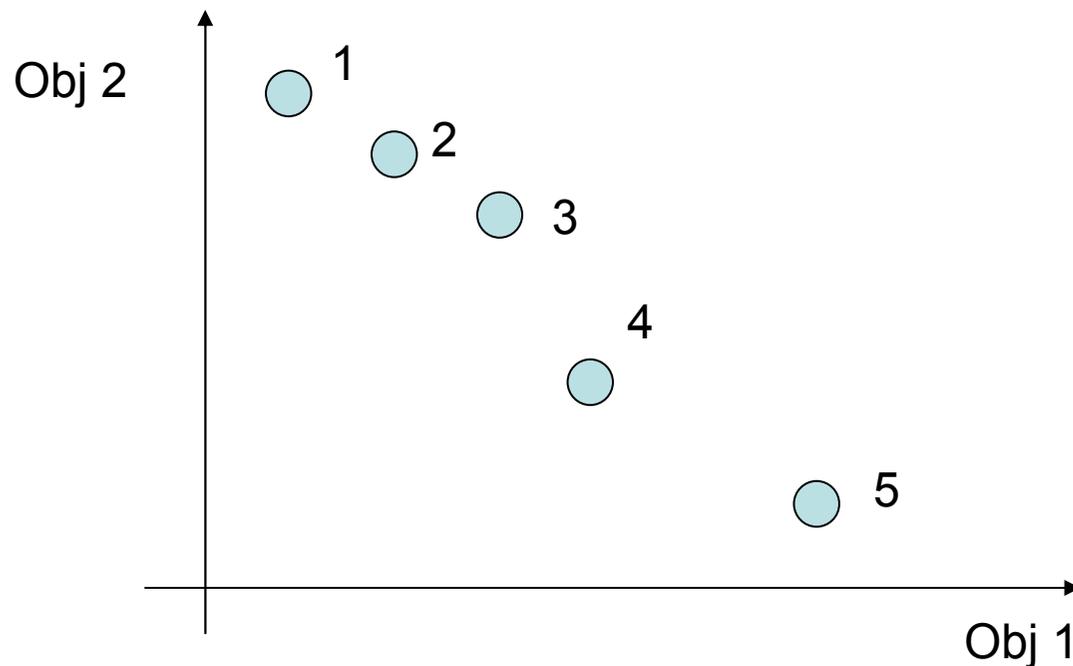
Esempio motore:

Max Potenza

Min Consumo

Min Peso

Come si osserva non sempre è possibile determinare un'unica funzione obiettivo \Rightarrow **PROBLEMA MULTI OBIETTIVO (Teoria Dei Giochi)**



MAX Obj1

MAX Obj2

La soluzione non è unica!

Algoritmi **Multi Criteria Decision Making**: permettono di trovare la migliore soluzione di compromesso.

Molte volte in ingegneria il solver in termini di tempo è molto costoso (calcoli fluidodinamici, strutturali non lineari, sperimentali): l'ottimizzazione numerica potrebbe non diventare interessante in termini industriali)

Superficie di risposta : creare un numerico modello approssimato del modello da studiare (pochi secondi contro ore)

Da un punto di vista numerico significa creare una approssimazione numerica delle funzioni (obiettivo) che stiamo studiando.

$$f_{RS} \approx f_{REALE}$$

Robust design: metodologia numerica che prende in considerazione non solo le prestazioni della configurazione, ma anche la stabilità al variare dei parametri in gioco (condizioni di progetto, tolleranze costruttive)

Parametrizzazione numerica

Algoritmo De Casteljan (formula ricorsiva)

$$\bar{b}_0^1(t) = (1-t)\bar{b}_0 + t\bar{b}_1$$

$$\bar{b}_1^1(t) = (1-t)\bar{b}_1 + t\bar{b}_2$$

$$\begin{aligned}\bar{b}_0^2(t) &= (1-t)\bar{b}_0^1(t) + t\bar{b}_1^1(t) = \\ &= (1-t)^2\bar{b}_0 + 2t(1-t)\bar{b}_1 + t^2\bar{b}_2 \quad (\text{forma quadratica})\end{aligned}$$

Ne consegue che.....

$$\bar{b}_0^2(0) = \bar{b}_0 \quad \bar{b}_0^2(1) = \bar{b}_2 \quad \text{La curva passa per il primo e l'ultimo punto di controllo}$$

La curva è contenuta nel triangolo formato dai punti di controllo

$$\frac{(b_0^1(t), b_0)}{(b_1, b_0)} = \frac{(b_0^2(t), b_0^1(t))}{(b_1^1(t), b_0^1(t))} = \frac{(b_1^1(t), b_1)}{(b_2, b_1)} = \frac{t}{1-t}$$

I rapporti tra i segmenti rimangono costanti (parametrizzati da t)

La precedente costruzione di una parabola può essere generalizzata per costruire una curva polinomiale di grado n

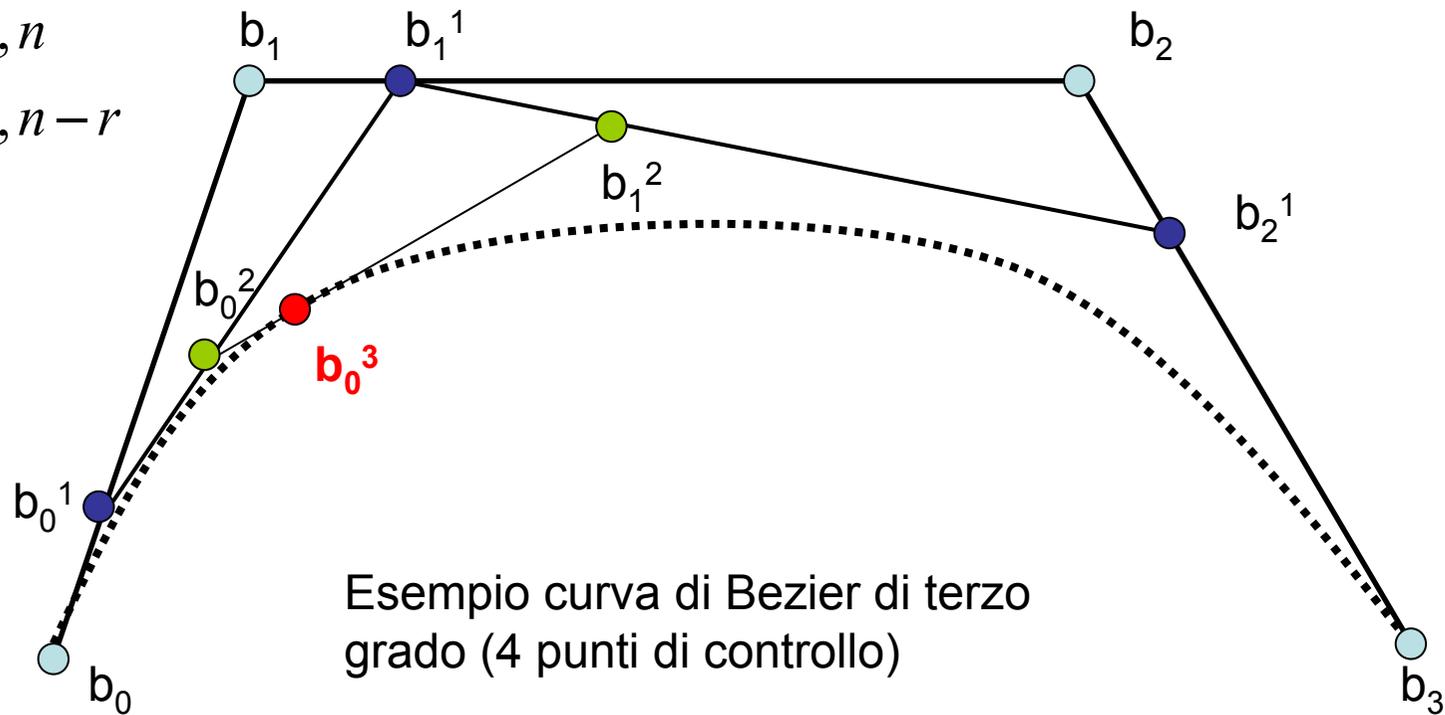
$b_0, b_1, b_2, \dots, b_n$ (grado curva $n \Rightarrow n+1$ punti di controllo)

$$\bar{b}_i^r(t) = (1-t)\bar{b}_i^{r-1}(t) \quad 0 \leq t \leq 1$$

$$\bar{b}_i^0(t) = \bar{b}_i \quad \text{punto di controllo}$$

$$r = 1, \dots, n$$

$$i = 0, \dots, n-r$$



L'algoritmo de Casteljano formula la curva di Bezier in modo ricorsivo, meglio se forma esplicita \Rightarrow FORMA DI BERNSTEIN

POLINOMI DI BERNSTEIN

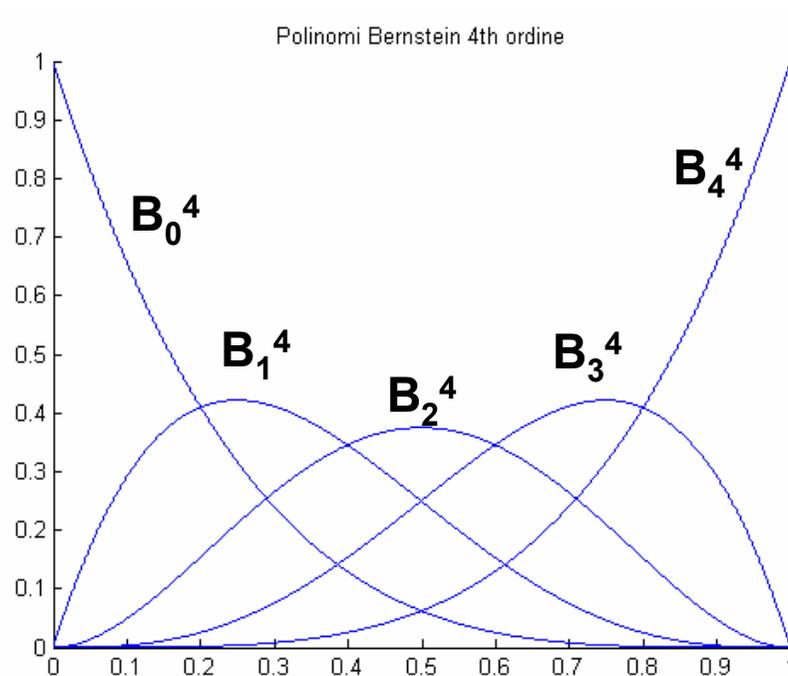
$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad 0 \leq t \leq 1$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \quad 0 \leq i \leq n$$

Importanti proprietà dei polinomi

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

$$\sum_{j=0}^n B_j^n(t) = 1 \quad \text{funzione pesata}$$



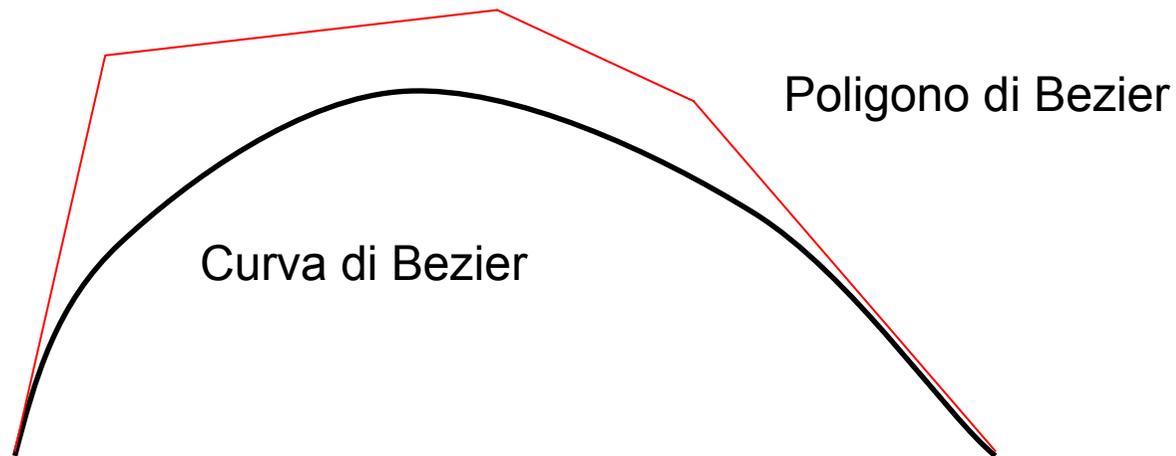
Qualsiasi curva di Bezier rappresentata mediante l'algoritmo di De Casteljan può essere rappresentata mediante polinomi di Bernstein in forma esplicita.

b_0, \dots, b_n bezier grado n

$$b_i^r(t) = \sum_{j=0}^r \bar{b}_{i+j} B_j^r(t)$$

il caso più interessante è $r = n$

$$b^n(t) = \sum_{j=0}^n \bar{b}_j B_j^n(t)$$



Proprietà

1. La curva passa per il primo e ultimo punto di controllo;
2. La curva è tangente al primo e ultimo segmento del poligono di Bezier (controllo sulla tangente iniziale e finale);
3. I punti della Bezier si accumulano dove la curvatura è maggiore

Dimostrazione punto 2)

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

ricordando che

$$b^n(t) = \sum_{j=0}^n b_j B_j^n(t)$$

$$\frac{d}{dt} b^n(t) = n \sum_{j=0}^n \left[B_{i-1}^{n-1}(t) - B_i^{n-1}(t) \right] b_j$$

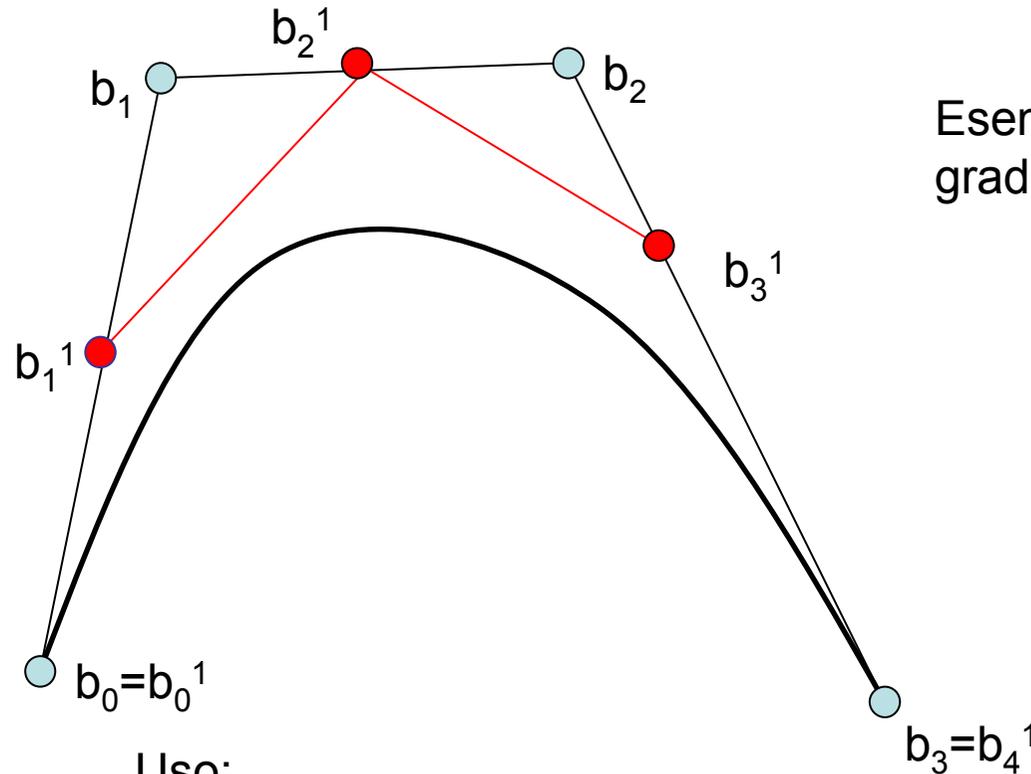
$$= n \sum_{j=0}^{n-1} (b_{j+1} - b_j) B_j^{n-1}$$

Tangente al primo ed
ultimo segmento

$$\begin{cases} \left. \frac{\partial b^n(t)}{\partial t} \right|_0 = n\Delta b_1 \\ \left. \frac{\partial b^n(t)}{\partial t} \right|_1 = n\Delta b_{n-1} \end{cases}$$

Degree elevation:

Una curva di Bezier di grado n può essere rappresentata da una curva di grado $n+1$ senza nessuna deformazione geometrica.



Esempio: passaggio da curva di grado 3 a curva di grado 4

Uso:

1. Creazioni di superfici (tutte le curve di contorno devono avere lo stesso grado)
2. CAD diversi (passaggio dati senza perdita di accuratezza)
- 3. Migliore accuratezza locale (ottimizzazione)**

Proprietà polinomi Bernstein

- 1) $(1-t)B_i^n(t) = \frac{n+1-i}{n+1} B_i^{n+1}(t)$
- 2) $tB_i^n(t) = \frac{i+1}{n+1} B_{i+1}^{n+1}(t)$
- 3) $B_i^n(t) = \frac{n+1-i}{n+1} B_i^{n+1}(t) + \frac{i+1}{n+1} B_{i+1}^{n+1}(t)$

Dimostrazione Degree Elevation

$$x(t) = (1-t)x(t) + tx(t)$$

$$\begin{aligned} 1) \quad &= \sum_{i=0}^n \frac{n+1-i}{n+1} b_i B_i^{n+1} + \sum_{i=0}^n \frac{i+1}{n+1} b_i B_{i+1}^{n+1} = 2) \\ &= \sum_{i=0}^{n+1} \frac{n+1-i}{n+1} b_i B_i^{n+1} + \sum_{i=0}^{n+1} \frac{i}{n+1} b_{i-1} B_i^{n+1} 3) \end{aligned}$$

$$\Rightarrow b_i^{n+1} = \frac{i}{n+1} b_{i-1}^n + \left(1 - \frac{i}{n+1}\right) b_i^n$$

I nuovi punti sono
un'interpolazione lineare dei
precedenti

Importante:

Aumentare il grado di una curva di Bezier è **sempre** possibile;

Viceversa **non sempre è possibile** diminuire il grado di una curva di Bezier **senza** perdere di accuratezza; ciò è **possibile** se la curva di partenza è data da un degree elevation.

Esistono metodologie per la degree reduction, ma portano approssimazioni geometriche, il più delle volte non accettabili.

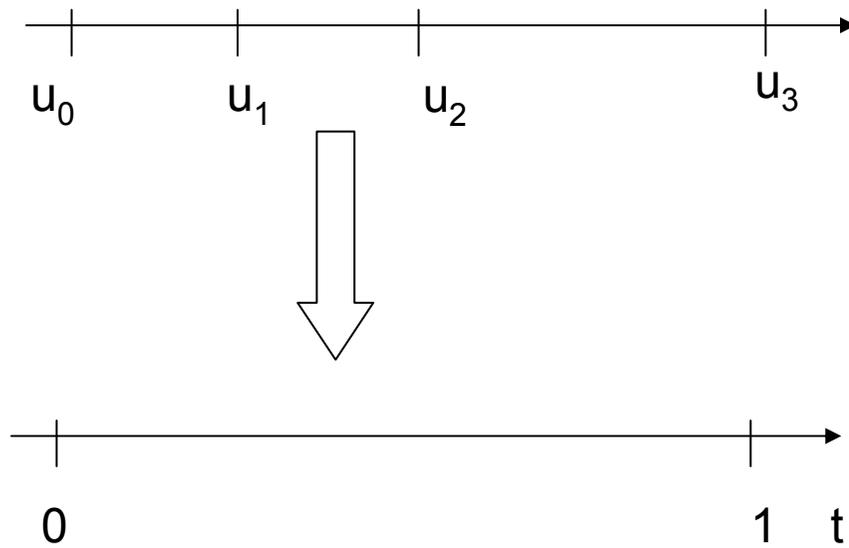
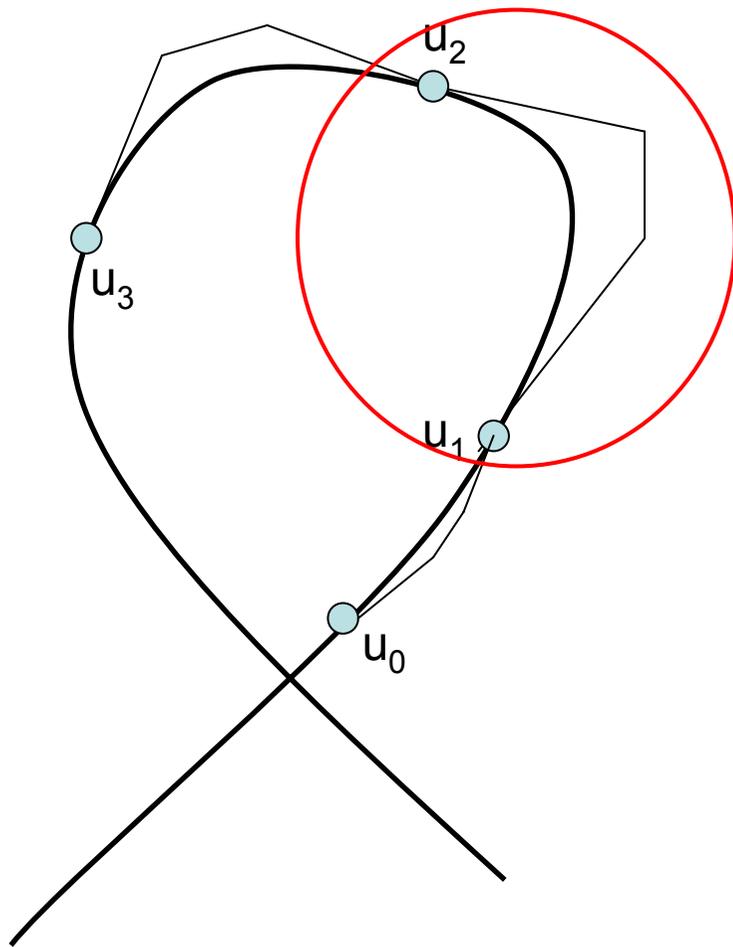
Le curve di Bezier sono sicuramente molto interessanti ed utilizzate, ma hanno limiti quando la geometria è molto complessa; in questi casi bisognerebbe aumentare notevolmente il grado della curva.

Problemi: mancanza di controllo locale (modificando un punto di controllo si modifica **TUTTA** la curva)

Normalmente si utilizzano le curve di Bezier fino a grado 4.

Per geometrie più complesse:

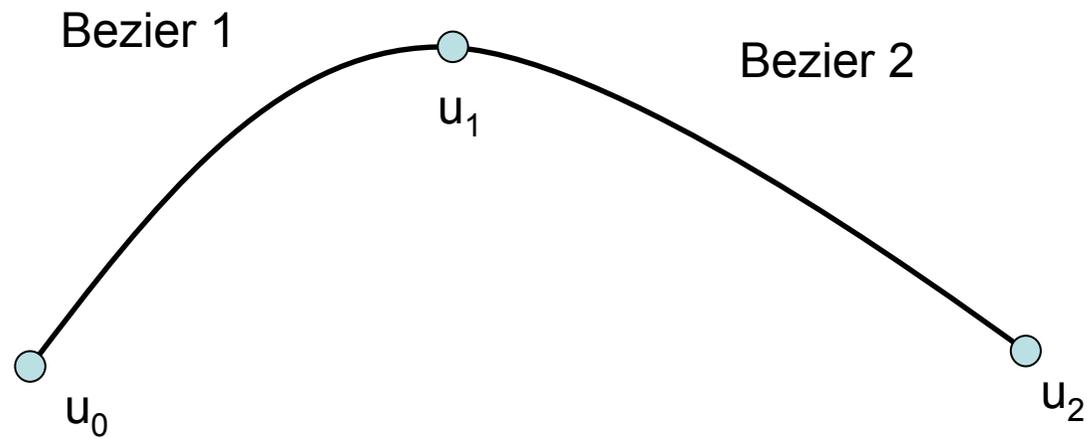
Curve di Bezier composite (B-SPLINE)



$u_0 < \dots < u_L$ nodi

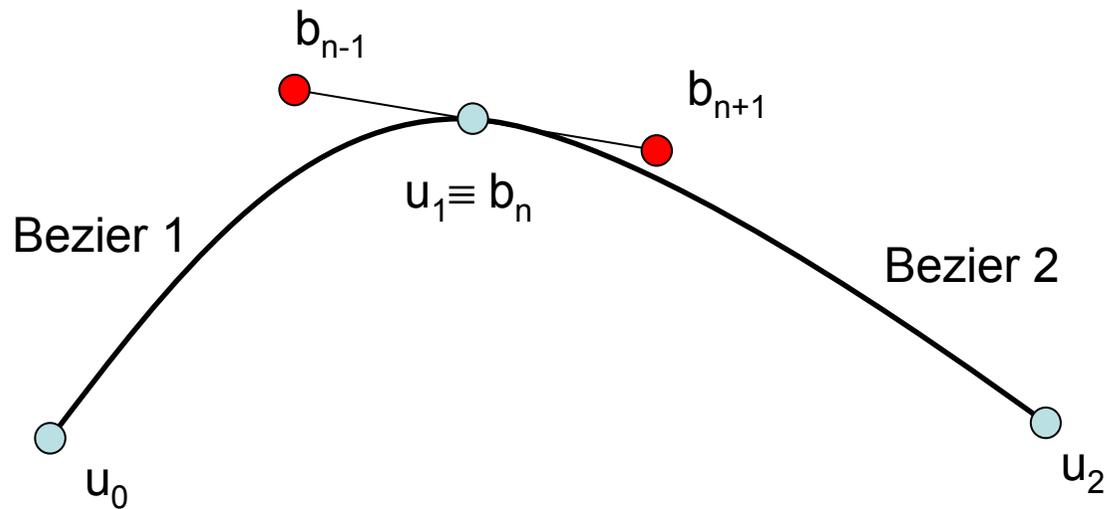
$$t \in [u_i, u_{i+1}] \quad t = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{\Delta_i}$$

Tra gli intervalli dei nodi viene definita una curva di Bezier
 \Rightarrow pezzi di curve polinomiali

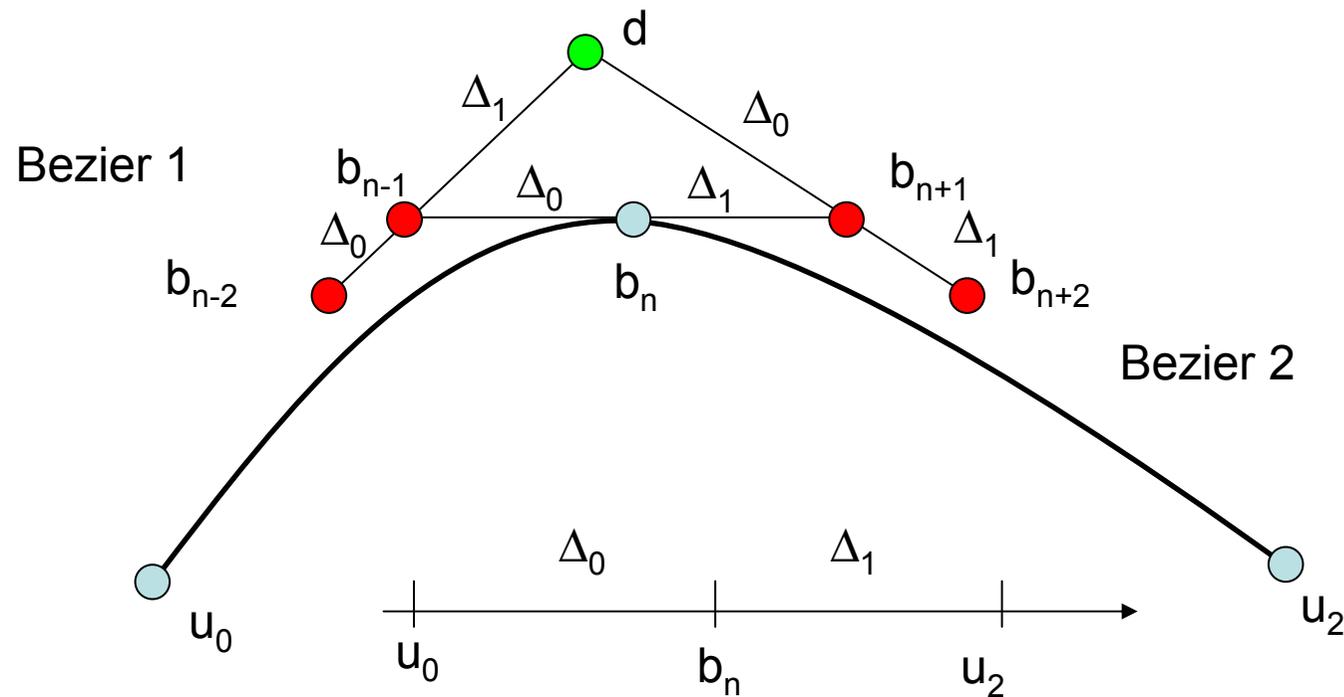


Le diverse tipologie di B-Spline si diversificano per la condizione di continuità nei nodi di connessione u_i

B-Spline C_1 : i punti di controllo immediatamente prima e dopo del nodo di connessione devono stare sulla stessa retta (stessa derivate destra e sinistra)



B-Spline di grado C2



Deve esistere un punto d tale che

$$b_{n-1} = (1-t_1)b_{n-2} + t_1d$$

$$b_{n+1} = (1-t_1)d + t_1b_{n+2}$$

$$t_1 = \frac{\Delta_0}{u_2 - u_0} \quad \text{cioè è il parametro } u_1 \text{ in coordinate locali}$$

Si nota che la condizione per il punto d porta ad avere determinate condizioni non solo ai punti di controllo b_{n+1} e b_{n-1} ma anche ai punti b_{n+2} e b_{n-2} .

Attenzione: è importante la condizione di unicità del punto d , perché solo le condizioni di proporzionalità tra i punti con le dimensioni Δ_0 e Δ_1 portano alla creazione di un flesso.

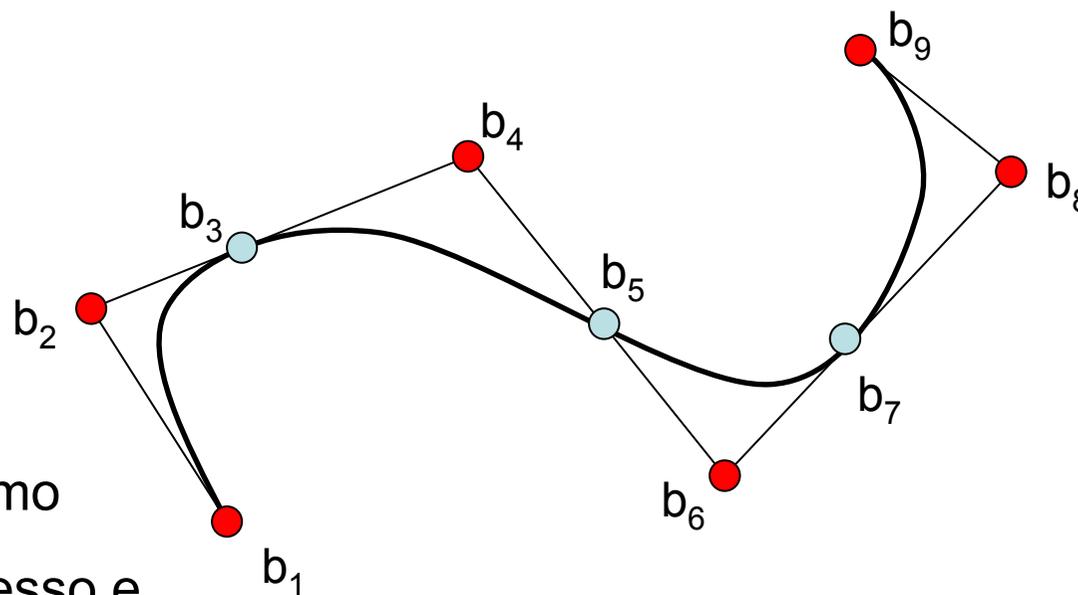
B-Spline quadratiche C_1

I punti “fondamentali” sono:

$$b_0, b_1, b_3, \dots, b_{2i+1}, \dots, b_{2L-1}, b_{2L}$$

Si vede come in tutti i casi abbiamo

il controllo sulla tangenza in ingresso e uscita della curva.



I punti pari (che basta siano sulla stessa retta) possono essere calcolati con la seguente relazione:

$$b_{2i} = \frac{\Delta_i}{\Delta_{i-1} - \Delta_i} b_{2i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i} b_{2i+1}$$

Importante

CONTROLLO LOCALE: con le B-spline muovendo un punto di controllo si muovono al massimo 3 segmenti di curva (precedente, appartenente al punto, seguente); questo al contrario delle curva di Bezier globali dove muovendo un punto si muove tutta la curva.

CUBIC SPLINE INTERPOLATION IN B-SPLINE (NURBS) (cenni)

In questo caso la curva, a differenza delle Bezier o B-spline generiche) passano per i punti dati

x_0, x_1, \dots, x_L punti

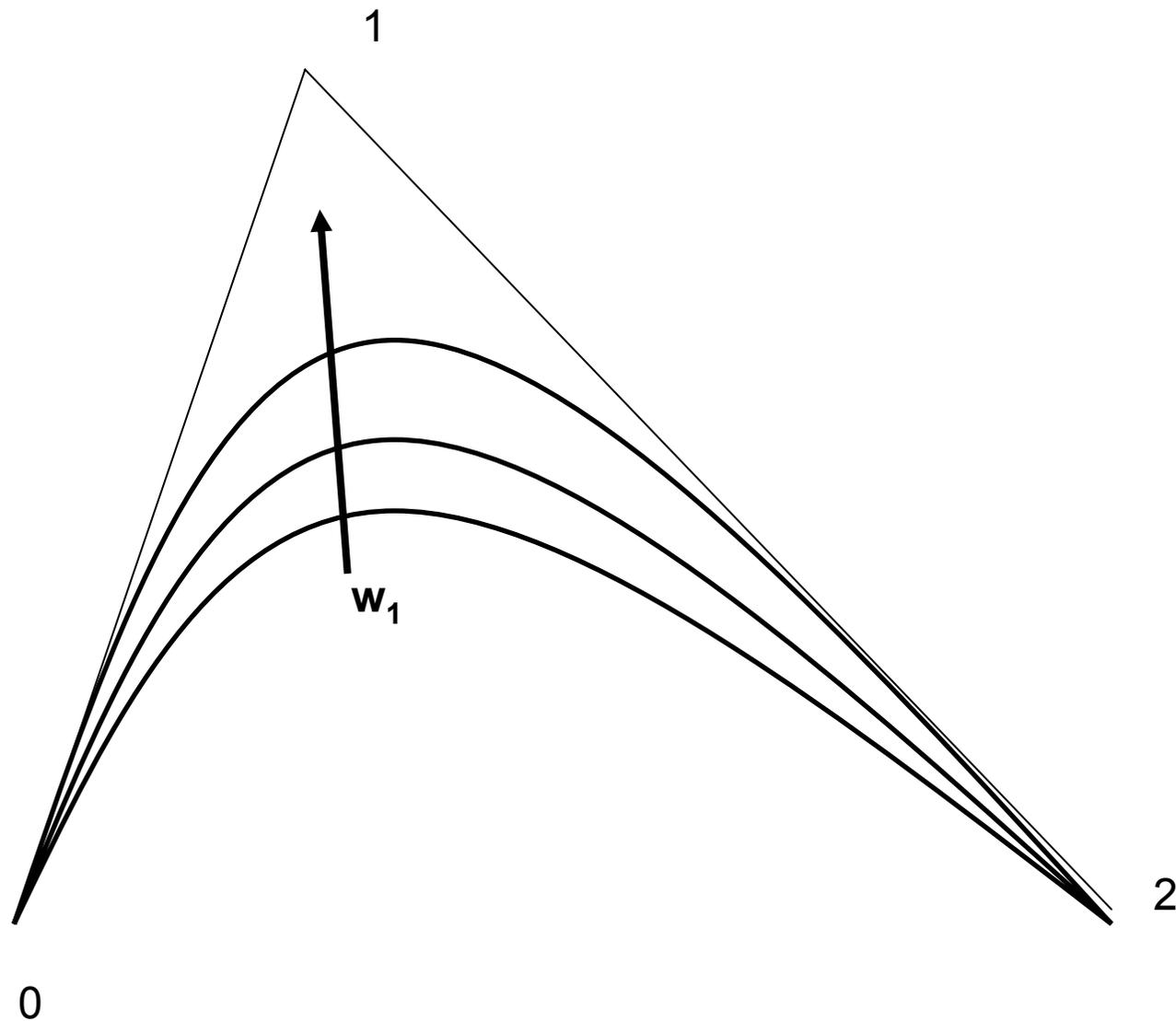
u_0, u_1, \dots, u_L nodi

$s(u_i) = x_i$ interpolazione

B-SPLINE razionali (cenni)

$$x(t) = \frac{w_0 b_0 B_0^n + w_1 b_1 B_1^n + \dots + w_n b_n B_n^n}{w_0 B_0^n + w_1 B_1^n + \dots + w_n B_n^n}$$

Oltre alla definizione dei punti di controllo b_i c'è la presenza associata dei pesi $w_i \Rightarrow$ concetto di **tensione**



Aumentando il peso w_1 la curva si **tende** verso il punto di controllo 1 (concetto presente in CATIA V5)

Superfici (Cenni)

Come la curva lineare è la più semplice curva tra 2 punti, così la superficie bilineare è la più semplice superficie tra 4 punti.

$b_{0,0}, b_{0,1}, b_{1,0}, b_{1,1}$ Punti di controllo

$$x(u, v) = \sum_{i=0}^1 \sum_{j=0}^1 b_{i,j} B_i^1(u) B_j^1(v)$$

B funzioni lineari

in forma matriciale

$$x(u, v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

E' possibile utilizzare l'algoritmo de Casteljan per la creazione di **superfici di Bezier**

$$b_{i,j}^{r,r} = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} b_{i,j}^{r-1,r-1} & b_{i,j+1}^{r-1,r-1} \\ b_{i+1,j}^{r-1,r-1} & b_{i+1,j+1}^{r-1,r-1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

Come nel caso del curve di Bezier è più utilizzata la forma **esplicita (matriciale)**

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

$$= \begin{bmatrix} u^0 & \dots & u^m \end{bmatrix} M^T \begin{bmatrix} b_{0,0} & \dots & b_{0,n} \\ \vdots & \ddots & \vdots \\ b_{m,0} & \dots & b_{m,n} \end{bmatrix} N \begin{bmatrix} v^0 \\ \vdots \\ v^n \end{bmatrix}$$

$$m_{i,j} = (-1)^{j-1} \binom{m}{j} \binom{j}{i}$$

$$n_{i,j} = (-1)^{j-1} \binom{n}{j} \binom{j}{i}$$

Come si nota ci può essere un numero di punti di controllo diverso nelle 2 direzioni ($m \neq n$)

$$m = n = 3$$

$$M = N = \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Per le superfici di Bezier rimangono valide tutte le considerazioni fatte per le curve (controllo pendenza ingresso = controllo delle derivate parziali all'ingresso; degree elevation: indipendente nelle due direzioni u,v)