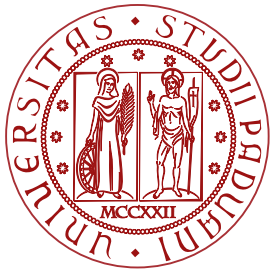


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

Corso di Laurea in Matematica

Curve di Bézier: confronto tra l'algoritmo di De Casteljau e di Woźny-Chudy

Relatore:
Prof. Alberto Viscardi

Laureando: Simone Vettoretto
Matricola: 2017589

Anno Accademico 2025/2026

24/04/2026

Indice

Capitolo 1

Introduzione

Due pilastri fondamentali nella Computer-Aided Design (CAD) sono i polinomi di Bernstein e le curve di Bézier. Grazie a questi oggetti matematici è possibile manipolare curve nello spazio in modo efficiente e stabile, proprietà che risultano essenziali nell'ambiente computazionale.

L'idea dei polinomi di Bernstein è attribuita al matematico sovietico Sergei Natanovich Bernstein nel 1912. Al tempo, la sua attenzione era rivolta allo studio dell'interpolazione polinomiale di funzioni. Egli, per primo, diede una dimostrazione costruttiva al Teorema di Weierstrass, il quale garantiva unicamente l'esistenza di una successione di polinomi in grado di approssimare uniformemente ogni funzione continua nello spazio normato $(\mathcal{C}^0([0,1]), ||\cdot||_\infty)$ ma non dava una classe di polinomi che potesse soddisfarlo. Nella seconda metà del Novecento, col progressivo sviluppo dei calcolatori, si fece evidente la necessità di algoritmi che rappresentassero le curve in modo stabile e in grado di limitare il propagarsi di errori numerici. In questo contesto, nel 1959, l'ingegnere Paul de Casteljau, mentre era dipendente presso Citroën, ideò l'omonimo algoritmo per la valutazione efficiente di curve definite mediante i polinomi di Berstein. Tale algoritmo si distinse negli anni avvenire tanto per la sua stabilità quanto per la sua semplicità ed eleganza. Poco anni più tardi, nel 1962 in modo del tutto indipendente, Pierre Bézier rese famosa la famiglia di curve che ad oggi portano il suo nome, contribuendo ad una rapida diffusione in ambito industriale e applicativo. Per molti anni l'algoritmo di de Casteljau fu lo standard per la valutazione delle curve di Bézier. Solamente in tempi recenti, nel 2020, Paweł Woźny e Filip Chudy proposero un nuovo algoritmo in grado di superare in termini di efficienza computazionale l'algoritmo di De Castelaju, mantenendo buone proprietà numeriche.

In questa tesi ci proponiamo dunque di analizzare i due algoritmi, quello di de Casteljau e quello di Woźny-Chudy, per la valutazione di un punto su una curva di Bézier, trattando per entrambi i metodi sia la parte teorica che quella computazionale-numerica.

La prima parte è dedicata allo studio dei polinomi di Bernstein. Inizieremo

mostrando come possano essere costruiti a partire da un'identità fondamentale e alcune delle proprietà essenziali, che saranno poi utili per i due algoritmi successivi. Concluderemo la prima parte della tesi dando dimostrazione del fatto che, tra tutte le basi stabili dello spazio dei polinomi di grado minore o uguale a $n \in \mathbb{N}$, la base di Bernstein risulti ottimale.

Nella seconda parte introdurremo le curve di Bézier come curve parametriche in $\mathbb{R}^d$. Ne daremo la definizione ed elencheremo le principali proprietà che costituiranno la base per il successivo capitolo. Verranno dati inoltre esempi esplicativi della costruzione delle curve di grado uno e due.

Attingendo alla teoria dei capitoli precedenti, procederemo ad illustrare l'algoritmo di de Casteljau. Questo è uno degli algoritmi cruciali e maggiormente utilizzati, grazie alle sue proprietà. Ci soffermeremo nella costruzione ricorsiva e nella descrizione geometrica. Ne daremo infine una pseudo-implementazione.

Per l'algoritmo di Woźny-Chudy seguiremo la medesima impostazione dell'algoritmo precedente, fornendo la costruzione, l'interpretazione geometrica e uno pseudo-algoritmo.

Infine, la tesi si concluderà con una comparazione numerica tra gli approcci adottati. Verranno valutate una serie di curve di Bézier di gradi variabili col fine di evidenziare l'efficienza computazione e i risultati teorici esposti.

Capitolo 2

Polinomi di Bernstein e proprietà

2.1 Funzioni Polinomiali

Definizione 2.1.1 (Polinomio)**.** Sia $n \in \mathbb{N}$. Un *polinomio di grado al più n* è una funzione $p : \mathbb{R} \to \mathbb{R}$ della forma

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

dove $\{a_k \in \mathbb{R}\}_{k=0}^{n}$.

Definizione 2.1.2 (Spazio dei polinomi)**.** Diremo *spazio dei polinomi* $\mathbb{P}_n$ lo spazio vettoriale di tutti i polinomi di grado al più $n \in \mathbb{N}$.

Osservazione 2.1.1. La base canonica dello spazio dei polinomi $\mathbb{P}_n$ è data dagli $n+1$ elementi $\{1, x, x^2, \dots x^n\}$.

2.2 Costruzione dei polinomi di Bernstein

In questa sezione presentiamo la costruzione dei polinomi di Bernstein di grado $n \in \mathbb{N}$ fissato. Consideriamo l'identità fondamentale

$$1 = (1 - t) + t, \qquad \forall t \in [0, 1].$$

Elevando entrambi i membri alla potenza n, otteniamo

$$1 = ((1 - t) + t)^n. \tag{2.1}$$

Teorema 2.2.1 (Binomio di Newton)**.** *Per ogni intero $n \geq 0$ e per ogni k tale che $0 \leq k \leq n$ vale che*

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k,$$

dove $\binom{n}{k}$ *è il coefficiente binomiale* n *su* k*.*

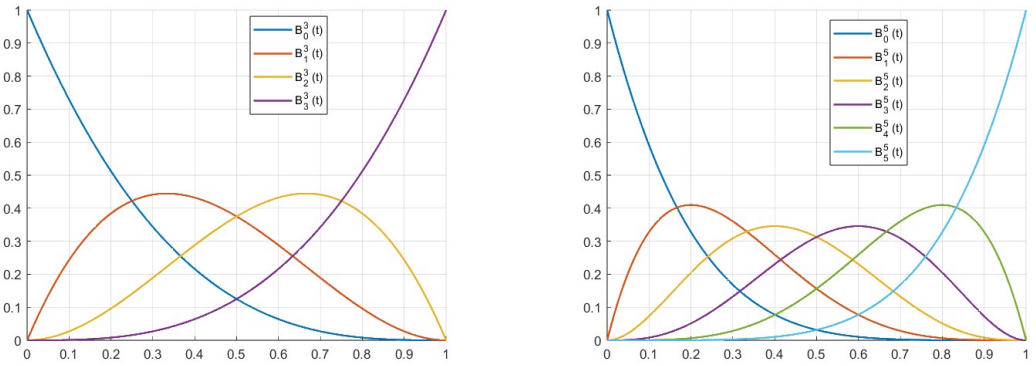
Applicando la formula del binomio di Newton a (2.1), si ha:

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

Definizione 2.2.1 (Polinomio di Bernstein)**.** Sia $t \in [0,1]$ e $n \in \mathbb{N}$. L'i-esimo polinomio di Bernstein di grado n è definito come:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad \text{per } 0 \leq i \leq n \tag{2.2}$$

Da questo punto della tesi in poi, useremo la convenzione che $\forall i \notin \{0, \ldots, n\}$, allora $B_i^n(t) \equiv 0$.



(a) *Polinomi di Bernstein per* $n = 3$ (b) *Polinomi di Bernstein per* $n = 5$

Figura 2.1: Esempi di polinomi di Bernstein

2.3 Proprietà dei polinomi di Bernstein

Enunciamo ora alcune proprietà fondamentali dei polinomi di Bernstein.

Non negatività: Per ogni $t \in [0,1]$ vale $B_i^n(t) \geq 0$. Inoltre valgono agli estremi le seguenti identità:

$$B_i^n(0) = \delta_{i,0}, \quad B_i^n(1) = \delta_{i,n}. \tag{2.3}$$

In particolare $B_i^n(t) > 0$ per $t \in (0,1)$.

Partizione dell'unità: Per ogni $t \in [0,1]$ vale

$$\sum_{i=0}^{n} B_i^n(t) = 1. \tag{2.4}$$

Questa proprietà deriva direttamente dalla costruzione tramite il binomio di Newton.

Combinazione convessa: Poiché $\sum_{0\le i\le n} B_i^n(t) = 1$ e $B_i^n(t) \ge 0$, segue che ogni combinazione del tipo

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

è una combinazione convessa dei coefficienti $\{a_i \in \mathbb{R}\}_{i=0}^n$. Ne segue che $p(t)$ è sempre compreso tra $\min_{0\le i\le n} a_i$ e $\max_{0\le i\le n} a_i$.

Formula ricorsiva: I polinomi di Bernstein soddisfano la relazione:

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t), \tag{2.5}$$

con condizioni iniziali $B_0^0(t) \equiv 1$ e $B_i^n(t) = 0$ se $i < 0$ o $i > n$.

Dimostrazione: Usando le proprietà del coefficiente binomiale

$$\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1},$$

otteniamo

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

□

Derivata: La derivata dell'i-esimo polinomio di Bernstein è data da

$$\frac{d}{dt} B_i^n(t) = n[B_{i-1}^{n-1}(t) - B_i^{n-1}(t)]. \tag{2.6}$$

Dimostrazione: Iniziamo analizzando i casi $i \in \{0, n\}$.
Per $i = 0$:

$$\frac{d}{dt}B_0^n(t) = \frac{d}{dt}(1-t)^n = -n(1-t)^{n-1} = -nB_0^{n-1}(t).$$

La formula continua a valere poiché, per $i = 0$, $B_{-1}^{n-1}(t)$ è identicamente nullo per definizione.
Per $i = n$:

$$\frac{d}{dt}B_n^n(t) = \frac{d}{dt}t^n = nt^{n-1} = nB_{n-1}^{n-1}(t).$$

La formula continua a valere poiché, per $i = n$, $B_n^{n-1}(t)$ è identicamente nullo per definizione.

Consideriamo ora il caso in cui $1 \leq i \leq n-1$:

$$\frac{d}{dt}B_i^n(t) = \frac{d}{dt}\left[\binom{n}{i}t^i(1-t)^{n-i}\right]$$

$$= \binom{n}{i}\left[it^{i-1}(1-t)^{n-i} - (n-i)t^i(1-t)^{n-i-1}\right]$$

$$= \frac{n!}{i!(n-i)!}\left[it^{i-1}(1-t)^{n-i} - (n-i)t^i(1-t)^{n-i-1}\right]$$

$$= \frac{n!}{(i-1)!(n-i)!}t^{i-1}(1-t)^{n-i} - \frac{n!}{i!(n-i-1)!}t^i(1-t)^{n-i-1}$$

$$= n\left[\binom{n-1}{i-1}t^{i-1}(1-t)^{n-i} - \binom{n-1}{i}t^i(1-t)^{n-i-1}\right]$$

$$= \boxed{n\left[B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right]}.$$

□

Base di $\mathbb{P}_n$: Fissato $n \in \mathbb{N}$, l'insieme $\{B_0^n, B_1^n, \ldots, B_n^n\}$ è una base per lo spazio dei polinomi reali di grado minore o uguale a n. Segue che per ogni $p \in \mathbb{P}_n$ esistono unici i coefficienti $\{a_i \in \mathbb{R}\}_{i=0}^n$ tale che

$$p(t) = \sum_{i=0}^{n} a_i B_i^n(t).$$

Dimostrazione: Espandendo $(1-t)^{n-i}$ con il binomio di Newton e facendo il cambio variabile $m = k - i$ si ottiene

$$B_i^n(t) = \binom{n}{i} t^i \sum_{m=0}^{n-i} \binom{n-i}{m} (-1)^m t^m = \sum_{k=i}^{n} \left[\binom{n}{i} \binom{n-i}{k-i} (-1)^{k-i} \right] t^k.$$

Indichiamo il coefficiente del monomio t^k in $B_i^n(t)$ con

$$c_{k,i} = \binom{n}{i} \binom{n-i}{k-i} (-1)^{k-i},$$

ovvero l'entrata (k, i) della matrice di cambio base $\mathcal{M}$. In particolare il coefficiente diagonale è

$$c_{i,i} = \binom{n}{i} \neq 0.$$

Se ordiniamo le basi in ordine crescente di grado (righe corrispondenti a $B_0^n, \ldots, B_n^n$, colonne a $1, t, \ldots, t^n$), la matrice di cambiamento di base $\mathcal{M} = (c_{k,i})$ è triangolare (i coefficienti $c_{k,i}$ sono nulli per $k < i$) e la sua diagonale è formata da $\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}$.
Essendo il determinante di una matrice triangolare uguale al prodotto degli elementi sulla diagonale, si ha

$$\det(\mathcal{M}) = \prod_{i=0}^{n} \binom{n}{i} \neq 0,$$

quindi $\mathcal{M}$ è invertibile. Ne consegue che la trasformazione che manda la base di Bernstein nella base monomiale è un isomorfismo; in particolare $\{B_0^n, \ldots, B_n^n\}$ è una base di $\mathbb{P}_n$. □

Valori notevoli: Sia $p(t) = \sum_{i=0}^{n} a_i B_i^n(t)$ polinomio nella base di Bernstein a coefficienti in $\mathbb{R}$. Valgono allora le seguenti proprietà:

1. $p(0) = a_0$ e $p(1) = a_n$.
2. Ogni $B_i^n(t)$ ammette un unico massimo in $t = i/n$.
3. $\min_i a_i \leq p(t) \leq \max_i a_i$ per ogni $t \in [0, 1]$.

Dimostrazione: (a), (c) sono banali.
(b) Dimostriamo che, per $0 < i < n$, B_i^n ammette un unico massimo in $t = i/n$.

Ricordando la formula

$$\frac{d}{dt} B_i^n(t) = n\big(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\big),$$

valutiamo il segno della differenza $B_{i-1}^{n-1}(t) - B_i^{n-1}(t)$.

Calcoliamo il rapporto usando nella seconda uguaglianza $\frac{\binom{n-1}{i-1}}{\binom{n-1}{i}} = \frac{i}{n-i}$

$$\frac{B_{i-1}^{n-1}(t)}{B_i^{n-1}(t)} = \frac{\binom{n-1}{i-1} t^{i-1}(1-t)^{n-i}}{\binom{n-1}{i} t^{i}(1-t)^{n-i-1}} = \frac{i}{n-i} \cdot \frac{1-t}{t}. \tag{2.7}$$

Da (2.7), prima moltiplichiamo e poi sottraiamo per $B_i^{n-1}(t)$

$$B_{i-1}^{n-1}(t) - B_i^{n-1}(t) = B_i^{n-1}(t) \left(\frac{i}{n-i} \cdot \frac{1-t}{t} - 1 \right) = \frac{B_i^{n-1}(t)}{(n-i)t}\big(i - nt\big).$$

Poiché $B_i^{n-1}(t) > 0$ per $t \in (0,1)$ e $(n-i)t > 0$, il segno della derivata $\frac{d}{dt}B_i^n(t)$ coincide con il segno di $i - nt$. Ne segue che

$$\frac{d}{dt}B_i^n(t) > 0 \quad \text{se } t < \frac{i}{n}, \qquad \frac{d}{dt}B_i^n(t) = 0 \quad \text{se } t = \frac{i}{n}, \qquad \frac{d}{dt}B_i^n(t) < 0 \quad \text{se } t > \frac{i}{n}.$$

Quindi B_i^n è strettamente crescente per $t < i/n$ e strettamente decrescente per $t > i/n$, perciò $t = i/n$ è l'unico punto di massimo.

Separiamo ora i casi in cui $i \in \{0, 1\}$.
Caso $i = 0$:

$$B_0^n(t) = (1-t)^n,$$

è una funzione strettamente decrescente. Segue che il massimo è in $t = 0$.
Caso $i = n$:

$$B_n^n(t) = t^n, \qquad \forall t \in [0,1]$$

è una funzione strettamente crescente. Segue che il massimo è in $t = 1$.
In entrambi i casi il massimo viene raggiunto in $t = i/n$.

□

Simmetria: Vale la relazione

$$B_i^n(t) = B_{n-i}^n(1-t). \tag{2.8}$$

Segue dalle proprietà del binomiale.

Invarianza per trasformazioni affini: Sia

$$\phi(x) = \frac{x-a}{b-a},$$

con $a, b \in \mathbb{R}$, $a < b$. Allora

$$B_i^n(\phi(x)) = \binom{n}{i} \left(\frac{x-a}{b-a} \right)^i \left(1 - \frac{x-a}{b-a} \right)^{n-i}. \tag{2.9}$$

Questo mostra come la base di Bernstein possa essere definita su qualunque intervallo chiuso $[a, b]$ tramite riparametrizzazione lineare [1, pag 37].

2.4 Ottimalità della base di Benstein

In questa sezione faremo ulteriore chiarezza sul motivo per il quale i polinomi di Benrnstein siano numericamente migliori della base dei polinomi. Ci proponiamo quindi di confrontare la base di Bernstein con quella canonica dei polinomi.

Definizione 2.4.1 (Base polinomiale)**.** Diremo che $n+1$ polinomi di grado $\leq n$ linearmente indipendenti

$$\mathbf{\Phi} = \{\phi_0(t), \ldots, \phi_n(t)\}$$

sono una base polinomiale per lo spazio dei polinomi $\mathbb{P}_n$ se possiamo rappresentare ogni $p(t) \in \mathbb{P}_n$ in modo unico come

$$p(t) = \sum_{k=0}^{n} c_k \phi_k(t)\,, \tag{2.10}$$

con coefficienti $c_0, \ldots, c_n \in \mathbb{R}$.

2.4.1 Base canonica vs base di Bernstein

Supponiamo che i coefficienti c_k di (2.10) soffrano di una perturbazione δc_k e, dunque, $p(t)$ sia perturbato nella forma $p(t) + \delta p(t)$. In modulo $\delta p(t)$ dipende da tre fattori:

- Il parametro in cui si calcola il polinomio, ovvero t;
- La distribuzione delle perturbazioni $\delta c_0, \ldots, \delta c_n$;
- La base utilizzata per esprimere $p(t)$.

Essendo interessati a un confronto tra basi, riduciamo il nostro problema ad avere i coefficienti delle perturbazioni uniformemente distribuiti, con un massimo relativo fissato a $\epsilon > 0$, ovvero

$$-\epsilon \leq \delta c_k / c_k \leq +\epsilon \qquad \text{per } k = 0, \ldots, n. \tag{2.11}$$

Con queste assunzioni, il nostro polinomio diventa

$$p(t) + \delta p(t) = \sum_{k=0}^{n} c_k \phi_k(t) + \sum_{k=0}^{n} \delta c_k \phi_k(t),$$

dove la perturbazione $\delta p(t)$ soddisfa il bound

$$-\sum_{k=0}^{n} |\delta c_k \phi_k(t)| \leq \delta p(t) \leq +\sum_{k=0}^{n} |\delta c_k \phi_k(t)|. \tag{2.12}$$

Riscrivendo l'equazione 2.12 in modulo, otteniamo

$$|\delta p(t)| \leq C_{\mathbf{\Phi}}(p(t))\epsilon\,, \quad \text{dove } C_{\mathbf{\Phi}}(p(t)) = \sum_{k=0}^{n} |c_k \phi_k(t)|. \tag{2.13}$$

Definizione 2.4.2 (Numero di Condizionamento)**.** Diremo che $C_{\mathbf{\Phi}}(p(t))$ è il numero di condizionamento per $p(t)$, per ogni t, rispetto alle perturbazioni uniformi (2.11) dei suoi coefficienti nella base $\mathbf{\Phi}$.

Osservazione 2.4.1. Il numero di condizionamento dipende dalla scelta della base utilizzata per rappresentare il polinomio dato.

Definizione 2.4.3 (Base non negativa)**.** Diremo che una base $\mathbf{\Phi} = \{\phi_0(t), \ldots, \phi_n(t)\}$ è non negativa nell'intervallo $t \in [a, b]$ se vale che

$$\phi_k(t) \geq 0 \quad \forall t \in [a, b], \quad k = 0, \ldots, n.$$

Osservazione 2.4.2. La compattezza dell'intervallo è necessaria a garantire, in caso della presenza di polinomi di grado dispari, la non negatività.
Nel nostro caso, la base di Bernstein e la base canonica sono basi non negative nell'intervallo $[0, 1]$.

Teorema 2.4.1. *Siano $\mathbf{\Psi} = \{\psi_0(t), \ldots, \psi_n(t)\}$ e $\mathbf{\Phi} = \{\phi_0(t), \ldots, \phi_n(t)\}$ basi non negative per i polinomi di grado n nell'intervallo $t \in [a, b]$ tali che il primo possa essere espresso come combinazione non negativa del secondo nel seguente modo:*

$$\psi_j(t) = \sum_{k=0}^{n} M_{jk}\phi_k(t), \quad j = 0, \ldots, n, \tag{2.14}$$

dove $M_{jk} \geq 0$ per ogni $0 \leq j, k \leq n$. Allora il numero di condizionamento per ogni valore di ogni polinomio $p(t)$ di grado n in ogni punto $t \in [a, b]$ in queste basi soddisfa

$$C_{\mathbf{\Phi}}(p(t)) \leq C_{\mathbf{\Psi}}(p(t)). \tag{2.15}$$

Dimostrazione: Nelle due basi $\mathbf{\Psi}, \mathbf{\Phi}$ possiamo esprimere $p(t)$ nei seguenti modi:

$$p(t) = \sum_{j=0}^{n} a_j\psi_j(t) = \sum_{j=0}^{n} c_k\phi_k(t).$$

Allora, usando (2.14), i coefficienti delle due basi sono legati dalla seguente relazione

$$c_k = \sum_{j=0}^{n} a_j M_{jk} \quad \text{per } k = 0, \ldots, n. \tag{2.16}$$

Per ipotesi, entrambe le basi sono non negative in $[a, b]$. Possiamo allora riscrivere il numero di condizionamento per $p(t)$ come segue:

$$C_{\mathbf{\Phi}}(p(t)) = \sum_{k=0}^{n} |c_k|\phi_k(t) \quad \text{e} \quad C_{\mathbf{\Psi}}(p(t)) = \sum_{k=0}^{n} |a_j|\psi_j(t). \tag{2.17}$$

Sostituiamo ora la (2.16) in $C_{\mathbf{\Phi}}(p(t))$ e, usando la disuguaglianza triangolare per le somme, otteniamo

$$C_{\mathbf{\Phi}}(p(t)) = \sum_{k=0}^{n} \left| \sum_{j=0}^{n} a_j M_{jk} \right| \phi_k(t) \leq \sum_{k=0}^{n} \left[\sum_{j=0}^{n} |a_j M_{jk}| \right] \phi_k(t). \tag{2.18}$$

Poiché $M_{jk} \geq 0$ per ogni $0 \leq j, k \leq n$, possiamo farlo uscire dal modulo. Scambiamo poi le sommatorie e otteniamo

$$C_{\mathbf{\Phi}}(p(t)) \leq \sum_{j=0}^{n} |a_j| \sum_{k=0}^{n} M_{jk}\phi_k(t) = \sum_{j=0}^{n} |a_j| \psi_j(t) = C_{\mathbf{\Psi}}(p(t)) \tag{2.19}$$

in cui abbiamo usato (2.14) nella seconda uguaglianza. □
Nel nostro caso, per 2.4.1, abbiamo che $\mathbf{\Psi} = \mathbf{P} = \{1, t, \ldots, t^n\}$ è la base dei monomi, mentre $\mathbf{\Phi} = \mathbf{B} = \{B_0^n(t), \ldots, B_n^n(t)\}$ è la base di Bernstein di grado n su $[0, 1]$.

Corollario 2.4.2. *I numeri di condizionamento $C_{\mathbf{B}}(p(t))$ e $C_{\mathbf{P}}(p(t))$ per il polinomio $p(t)$ nelle rappresentazioni della base di Bernstein e nella base delle potenze soddisfa*

$$C_{\mathbf{B}}(p(t)) \leq C_{\mathbf{P}}(p(t)), \tag{2.20}$$

per ogni polinomio $p(t)$ e per ogni valore $t \in [0, 1]$.

Dimostrazione: Sia la base di Bernstein che la base delle potenze sono non negative in $[0, 1]$. Inoltre, la matrice di cambio di base dalla base delle potenze alla base di Bernstein è una combinazione non negativa, ma non vale il viceversa. Infatti, per passare dalla base delle potenze a quella di Bernstein, vale per ogni $0 \leq k \leq n$

$$t^k = \sum_{j=k}^{n} \frac{\binom{j}{k}}{\binom{n}{k}} B_j^n(t).$$

Quindi, il corollario segue dal teorema 2.4.1.

□

Osservazione 2.4.3. Il corollario 2.4.2 ci garantisce che la base di Bernstein è *sistematicamente più stabile* della base delle potenze quando si valutano polinomi nell'intervallo $[0, 1]$ soggetti a perturbazioni come in 2.4.1. Bisogna inoltre osservare che il teorema 2.4.1 non fa alcuni ipotesi sulla grandezza del coefficiente di errore relativo ϵ.

Sappiamo ora che la base di Bernstein è non negativa e sistematicamente più stabile della base delle potenze. Ci chiediamo allora se esistano altre basi non negative, nelle quali la base di Bernstein possa essere espressa come una combinazione

non negativa di queste e che siano sistematicamente più stabili. Il problema viene studiato in [2] e [3], nei quali viene mostrato che la base di Bernstein è *"stabilmente ottimale"* nel senso che descriveremo qui di seguito.

Sia $\mathcal{P}_n$ l'insieme di tutte le **basi polinomiali non negative** di grado n sull'intervallo $[0,1]$. La matrice di trasformazione tra basi non negative $\mathbf{M}$ stabilisce un *ordine parziale* sull'insieme $\mathcal{P}_n$: date $\mathbf{\Psi}, \mathbf{\Phi} \in \mathcal{P}_n$ e una matrice di cambio base non negativa $\mathbf{M}$ che rispetti (2.14), scriveremo che $\mathbf{\Psi} \preceq \mathbf{\Psi}$. Poiché il prodotto di due matrici non negative è una matrice non negativa, allora è soddisfatta la proprietà transitiva per $\preceq$. Osserviamo ora che, data una matrice non negativa, essa ha un inversa non negativa se e solo se è il prodotto di una matrice di permutazione e di una matrice con diagonale positiva. Allora valgono contemporaneamente le relazioni $\mathbf{\Psi} \preceq \mathbf{\Phi}$ e $\mathbf{\Phi} \preceq \mathbf{\Psi}$ se e solo se gli elementi di $\mathbf{\Phi}$ sono multipli positivi degli elementi di $\mathbf{\Psi}$. In questo caso, scriveremo $\mathbf{\Phi} \sim \mathbf{\Psi}$. Scriveremo che, se $\mathbf{\Phi} \preceq \mathbf{\Psi}$ ma $\mathbf{\Phi} \not\sim \mathbf{\Psi}$, allora $\mathbf{\Phi} \prec \mathbf{\Psi}$. Diremo infine che $\mathbf{\Phi}$ **precede** $\mathbf{\Psi}$ se $\mathbf{\Phi} \prec \mathbf{\Psi}$ e $\mathbf{\Phi}$ è **simile** a $\mathbf{\Psi}$ se $\mathbf{\Phi} \sim \mathbf{\Psi}$.

Definizione 2.4.4 (Base minimale)**.** Una base non negativa $\mathbf{\Phi}$ è una base minimale di $\mathcal{P}_n$ se non esiste una base $\mathbf{\Psi}$ tale che $\mathbf{\Psi} \prec \mathbf{\Phi}$.

Osservazione 2.4.4. Essendo $\mathcal{P}_n$ solo parzialmente ordinato, possono esserci più basi minimali, a meno di equivalenze di moduli.

Teorema 2.4.3. *Qualsiasi coppia di basi non negative $\mathbf{\Phi}, \mathbf{\Psi} \in \mathcal{P}_n$ soddisfano*

$$\mathbf{\Phi} \preceq \mathbf{\Psi} \quad \Longleftrightarrow \quad C_\phi(p(t)) \leq C_\psi(p(t)),$$

per ogni polinomio $p(t)$ e per ogni valore $t \in [0,1]$.

Consideriamo la base di Bernstein $\mathbf{B} = \{B_0^n(t), \ldots, B_n^n(t)\}$ definita nell'intervallo reale $[a,b]$. Mostriamo che $\mathbf{B}$ è stabile ottimale per $\mathcal{P}_n$.

Teorema 2.4.4. *Siano $\mathbf{\Psi} = \{\psi_0(t), \ldots, \psi_n(t)\}$ e $\tilde{\mathbf{\Psi}} = \{\tilde{\psi}_0(t), \ldots, \tilde{\psi}_n(t)\}$ basi in $\mathcal{P}_n$ che soddisfano*

$$\psi_j^{(i)}(a) = 0 \quad per \quad i = 0, \ldots, j-1 \quad e \quad j = 1, \ldots, n,$$

$$\tilde{\psi}_j^{(i)}(b) = 0 \quad per \quad i = 0, \ldots, n-j-1 \quad e \quad j = 0, \ldots, n-1.$$

Allora, se $\mathbf{\Phi} \in \mathcal{P}_n$ soddisfa sia $\mathbf{\Phi} \preceq \mathbf{\Psi}$ sia $\mathbf{\Phi} \preceq \tilde{\mathbf{\Psi}}$, abbiamo che $\mathbf{\Phi} \sim \mathbf{B}$.

Dimostrazione: Sia M_{ij} per $0 \leq i, j \leq n$ l'elemento della matrice non negativa $\mathbf{M}$ tale che

$$\tilde{\mathbf{\Psi}}^T = \mathbf{M}\mathbf{\Phi}^T. \tag{2.21}$$

Proviamo per induzione che la base $\mathbf{\Phi}$ può essere ordinata in modo tale che la matrice $\mathbf{M}$ risulti triangolare inferiore. Supponiamo che, per un certo ordine di $\mathbf{\Phi}$, ci sia una riga k tale che

$$M_{ij} = 0 \quad \text{per} \quad j \geq i+1 \quad \text{e} \quad i = 0, \ldots, k-1.$$

Per $k = 0$ l'ipotesi perde significato. Notiamo che, per avere la lineare indipendenza di $\tilde{\psi}_0, \ldots, \tilde{\psi}_{k-1}$ sotto questa ipotesi, deve valere $M_{ij} > 0$ per $i = 0, \ldots, k-1$. Ora, per $k \leq n-1$,

$$0 = \tilde{\psi}_k(b) = \sum_{j=0}^{n} M_{kj}\phi_j(b),$$

e poiché M_{kj} e $\phi_j(b)$ sono non negativi, dobbiamo avere che $M_{kj}\phi_j(b) = 0$ per $j = 0, \ldots, n$. Se avessimo supposto che $M_{kj} > 0$ per $j = k, \ldots, n$, allora $\phi_j(b) = 0$ per $j = k, \ldots, n$. Ad ogni modo, poiché $\tilde{\psi}_j(b) = 0$ per $j = 0, \ldots, k-1$, possiamo dedurre che $\psi_j(b) = 0$ per $j = 0, \ldots, k-1$ dalle ipotesi. Notiamo che i valori $\phi_0(b), \ldots, \phi_n(b)$ non possono annullarsi tutti poiché è una base. Allora dev'essere che la nostra assunzione, $M_{jk} > 0$ per $j = k, \ldots, n$ dev'essere falsa. Quindi, riordinando se necessario $\phi_k, \ldots, \phi_n$, possiamo supporre che $M_{kn} = 0$.
Ora, se $k = n-1$, per induzione possiamo terminare. Altrimenti, per $k \leq n-2$, notiamo che

$$0 = \tilde{\psi}'_k(b) = \sum_{j=0}^{n-1} M_{kj}\phi'_j(b) \tag{2.22}$$

e, di nuovo, $\sum_{j=0}^{n-1} M_{kj}\phi_j(b) = 0$. Supponiamo ora che $M_{kj} > 0$ per $j = k, \ldots, n-1$. Come prima, avremo che $\phi_j(b) = 0$ per $j = k, \ldots, n-1$. Poiché $\mathbf{\Phi}$ è una base non negativa sull'intervallo $[a, b]$, ciò implica che $\phi'_j(b) \leq 0$ per $j = k, \ldots, n-1$. Ora abbiamo che, $\tilde{\psi}_j(b) = \tilde{\psi}'_j(b) = 0$ per $j = 0, \ldots, k-1$ e, per ipotesi, abbiamo anche che $\phi_j(b) = \phi'_j(b) = 0$ per $j = 0, \ldots, k-1$. Per soddisfare la (2.22) sotto queste condizioni, dobbiamo avere che $\phi'_j(b) = 0$ per $j = k \ldots, n-1$. Quindi, $\phi_j(b) = \phi'_j(b) = 0$ per $j = 0, \ldots, n-1$, ovvero i polinomi $\phi_0, \ldots, \phi_{n-1}$ giacciono nello spazio $(n-1)$-dimensionale $\{p \in \mathbb{P}_n : p(b) = p(b)' = 0\}$. Ciò contraddice la lineare indipendenza della base $\mathbf{\Phi}$. La supposizione che $M_{kj} > 0$ per $j = k \ldots, n-1$ deve quindi essere falsa e, riordinando $\phi_k, \ldots, \phi_{n-1}$ se necessario, possiamo supporre che $M_{k,n-1} = 0$. Con ragionamenti analoghi, si mostra che $M_{kj} = 0$ per $j = k+1, \ldots, n$. Dimostriamo ora per induzione che

$$\phi_j(t) = c_j(b-t)^{n-j}(t-a)^j \tag{2.23}$$

per delle costanti $c_0, \ldots, c_n \in \mathbb{R}$ diverse da zero. Come ipotesi induttiva, assumiamo che per un certo $0 < k \leq n$, l'espressione (2.23) sia valida per $j = n-k+1, \ldots, n$. Sia $\mathbf{\Psi}^T = \mathbf{N}\mathbf{\Phi}^T$, con $\mathbf{N}$ matrice non negativa. Supponiamo inizialmente che $k \leq n-1$.

Allora

$$0 = \psi_{n-k}(a) = \sum_{j=0}^{n} N_{n-k,\,j}\,\psi_j(a),$$

e quindi $N_{n-k,\,j}\,\psi_j(a) = 0$ per $j = 0, \ldots, n$. Se supponiamo che $N_{n-k,\,j} > 0$ per $j = 0, \ldots, n-k$, allora $\phi_j(a) = 0$ per $j = 0, \ldots, n-k$. Ma, per ipotesi, ϕ_j ha la forma (2.23) per $j = n-k, \ldots, n$. Segue che anche $\phi_j(a) = 0$ per $j = n-k+1, \ldots, n$. In ogni caso, i valori $\phi_0(a), \ldots, \phi_n(a)$ non possono essere tutti zero, poiché $\lneqq$ è base per $\mathbb{P}_n$. Possiamo allora concludere che la supposizione $N_{n-k,\,j} > 0$ per $j = 0, \ldots, n-k$ deve essere falsa. Quindi, dobbiamo avere $N_{n-k,\,l} = 0$ per qualche $0 \le l \le n-k$. Supponiamo ora che $k \le n-2$. Allora abbiamo

$$0 = \psi'_{n-k}(a) = \sum_{j=0}^{n} N_{n-k,\,j}\,\phi'_j(a), \tag{2.24}$$

allo stesso modo $\sum_{j=0}^{n} N_{n-k,\,j}\,\phi_j(a) = 0$. Supponiamo che $N_{n-k,\,j} > 0$ per $j = 0, \ldots, n-k$, $j \neq l$. Allora, come prima, abbiamo $\phi_j(a) = 0$ per $j = 0, \ldots, n-k$, $j \neq l$. Poiché $\lneqq$ è base non negativa su $[a, b]$, ciò implica che $\phi'_j(a) \ge 0$ per $j = n-k+1, \ldots, n$, $j \neq l$. Ora, per ipotesi, ϕ_j ha la forma (2.23) per $j = n-k+1, \ldots, n$ e quindi $\phi_j(a) = \phi'_j(a) = 0$ per $j = n-k+1, \ldots, n$, $j \neq l$. Per soddisfare l'equazione (2.24) sotto queste condizioni, dobbiamo avere che $\phi'_j(a) = 0$ per $j = 0, \ldots, n-k$, $j \neq l$. Ciò implica che l'insieme $\{\phi_j : 0 \le j \le n, j \neq l\ \}$ dei n polinomi giace nello spazio $(n-1)$-dimensionale $\{p \in \mathbb{P}_n : p(a) = p'(a) = 0\}$, il che contraddice l'ipotesi di indipendenza lineare. Segue che la supposizione $N_{n-k\,j} > 0$ per $j = 0, \ldots, n-k$, $j \neq l$ è falsa. Deduciamo allora che $N_{n-k,\,m} = 0$ per qualche $0 \le m \le n-k$, $m \neq l$. Continuando con lo stesso ragionamento, si può dimostrare che solo uno degli elementi di $N_{n-k\,j}$ per $j = 0, \ldots, n-k$ è diverso da zero. Allora, per qualche $0 \le r \le n-k$, facendo uso di (2.23), deduciamo che

$$\psi_{n-k} = c_r\phi_r + \sum_{j=n-k+1}^{n} c_j(b-t)^{n-j}(t-a)^j \tag{2.25}$$

per delle costanti non negative c_r e $c_{n-k+1}, \ldots, c_n$. Ora, poiché $\tilde{\psi}^i_r(b) = 0$ per $i = 0, ..., n-r-1$, la relazione (2.21), in cui M è triangolare inferiore, mostra che

$$\phi^{(i)}_r(b) = 0 \quad \text{per} \quad i = 0, ..., n-r-1.$$

Inoltre, l'equazione (2.25) e l'ipotesi che $\psi^{(i)}_{n-k}(a) = 0$ per $i = 0, ..., n-k-1$ ci indicano che

$$\phi^{(i)}_r(a) = 0 \quad \text{per} \quad i = 0, ..., n-k-1.$$

Dalle condizioni sopra, vediamo che, per far sì che ϕ_r non sia identicamente nullo, dobbiamo avere $r = n-k$. Allora, deduciamo che $\phi_{n-k}(t)$ è una costante positiva

multiplo di $(b-t)^k(t-a)^{n-k}$. Quindi vale (2.23) per $j = n-k$, il che dimostra il passo induttivo. Segue che la forma (2.23) vale per $j = 0, ..., n$ e abbiamo che $\mathbf{\Phi} \sim \mathbf{B}$.

Corollario 2.4.5. *La base* $\mathbf{B}$ *è minimale.*

Dimostrazione: Supponiamo $\mathbf{\Phi} \in \mathcal{P}_n$ sia tale che $\mathbf{\Phi} \preceq \mathbf{B}$. Poiché la base $\mathbf{B}$ soddisfa le condizioni di entrambe $\psi, \tilde{\psi}$ del teorema 2.4.4, ciò garantisce che $\mathbf{\Phi} \sim \mathbf{B}$. Allora, non esiste un'altra base $\mathbf{\Phi} \in \mathcal{P}_n$ tale che $\mathbf{\Phi} \prec \mathbf{B}$.

□

Teorema 2.4.6. *La base di Bernstein è un elemento minimo di* $\mathcal{P}_n$ *ed è inoltre l'unico elemento minimo per il quale le funzioni di base non hanno radici in* $(0, 1)$.

La conseguenza immediata del teorema 2.4.6 è che la base di Bernstein è l'unica base conosciuta, stabile e ottimale. Si potrebbero costruire altre basi minimali per $\mathcal{P}_n$ (un esempio si trova in [2]), ma godono di proprietà di stabilità più deboli rispetto alla base di Bernstein.

Capitolo 3

Curve di Bézier

3.1 Curve in $\mathbb{R}^d$

Non è facile indicare in prima battuta cosa sia una curva nel piano (o nello spazio). Ci serve una definizione precisa che inserisca all'interno della categoria tutti gli oggetti che sottendono a determinate caratteristiche. Essenzialmente i matematici hanno trovato due modi per rappresentare le curve:

1. **Rappresentazione parametrica;**

2. **Rappresentazione cartesiana.**

La prima è la più completa e ci restituisce l'insieme delle proprietà della curva e la descrive in tutti i suoi punto evitando di introdurre più funzioni e ambiguità in determinati punti. La seconda ci permette di avere informazioni riguardo una o più variabili che determinano la curva grazie al *Teorema della funzione implicita*.

Esempio 3.1.1. Consideriamo la circonferenza di raggio $R > 0$ centrata nell'origine $(0, 0)$. Si ha allora che:

Equazioni parametriche:

$$\begin{cases} x(t) &= R\cos(t), \\ y(t) &= R\sin(t), \end{cases} \qquad t \in [0, 2\pi).$$

Equazioni cartesiane:

(1) $x^2 + y^2 = R^2$, (forma implicita)

(2) $y = \pm\sqrt{R^2 - x^2}, \quad x \in [-R, R]$. (forma esplicita)

Le problematiche nei punti $(R, 0)$ e $(-R, 0)$ ci impongono di utilizzare due equazioni per descrivere nel piano la circonferenza, oltre ad avere una doppia rappresentazione data sia dal $+$ che dal $-$. Al contrario, nella rappresentazione parametrica ogni punto viene descritto una sola volta.

Definizione 3.1.1 (Curva parametrica)**.** Dato $k \in \mathbb{N} \cup \{\infty\}$ e $n \geq 2$, una *curva parametrizzata di classe* C^k è un'applicazione $\phi : I \to \mathbb{R}^d$ di classe C^k, dove $I \subseteq \mathbb{R}$ è un intervallo. L'immagine $\phi(I)$ sarà detta *sostegno* della curva; la variabile $t \in I$ *parametro* della curva. Se $I = [a, b]$ e $\phi(a) = \phi(b)$ diremo che la curva è *chiusa*.

3.2 Curve di Bézier

Partiamo da un paio di esempi che, una volta generalizzati, porteranno alla costruzione delle curve di Bézier.

Esempio 3.2.1 (Curve di Bézier di grado 1)**.** Dati b_0 e b_1 nel piano, allora la curva di Bézier passante per i due punti si ottiene dalla loro combinazione convessa

$$s(t) = (1 - t)b_0 + tb_1 \,, \qquad t \in [0, 1]. \tag{3.1}$$

La curva risultante è, come mostrato dalla figura 3.1, un segmento che congiunge i due punti b_0, b_1.

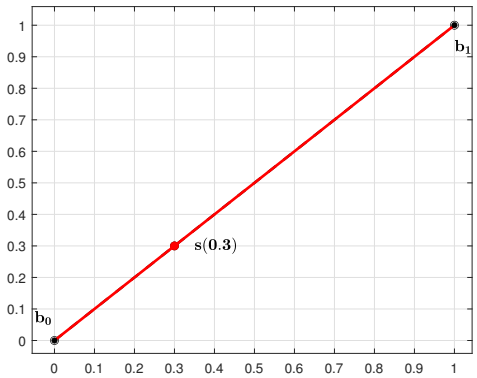


Figura 3.1: **Curva di Bézier di grado 1**: Abbiamo usato $b_0 = (0, 0)$ e $b_1 = (1, 1)$. La curva in rosso è quindi una parametrizzazione dell'intervallo $[0, 1]$ in un segmento.

Esempio 3.2.2 (Curve di Bézier di grado 2)**.** Consideriamo tre punti nel piano b_0, b_1, b_2. Mostriamo che, a partire da questi tre punti e tramite combinazioni convesse, siamo in grado di ottenere una curva di Bézier di grado due.

Fissiamo $t \in [0,1]$ e definiamo i due punti intermedi $r_0(t)$, $r_1(t)$ tramite le seguenti relazioni:

$$\begin{aligned} r_0(t) &= (1-t)b_0 + tb_1 \\ r_1(t) &= (1-t)b_1 + tb_2. \end{aligned} \tag{3.2}$$

Osservazione 3.2.1. Al variare di $t \in [0,1]$, $r_0(t)$ e $r_1(t)$ sono due segmenti parametrizzati come nell'esempio 3.2.1.

A questo punto, definiamo il punto $p(t)$ come la combinazione convessa dei due punti trovati in (3.2)

$$p(t) = (1-t)r_0(t) + tr_1(t). \tag{3.3}$$

Sostituiamo infine i punti $r_0(t)$ e $r_1(t)$ con l'equazione trovata in (3.3):

$$p(t) = (1-t)^2 b_0 + 2t(1-t)b_1 + t^2 b_2. \tag{3.4}$$

Notiamo ora che, facendo variare t nell'intervallo $[0,1]$, otteniamo una curva parametrizzata di grado 2. Inoltre, la curva ottenuta facendo variare t può essere scritta usando la base di Bernstein. Infatti,

$$\begin{aligned} p(t) &= (1-t)^2\, b_0 && +2t\,(1-t)\, b_1 && +t^2\, b_2 \\ &= B_0^2(t)\; b_0 && +B_1^2(t)\; b_1 && +B_2^2(t)\; b_2. \end{aligned}$$

Un esempio visivo di come si genera una curva di grado 2 a partire da un punto fissato è mostrato in figura 3.2.

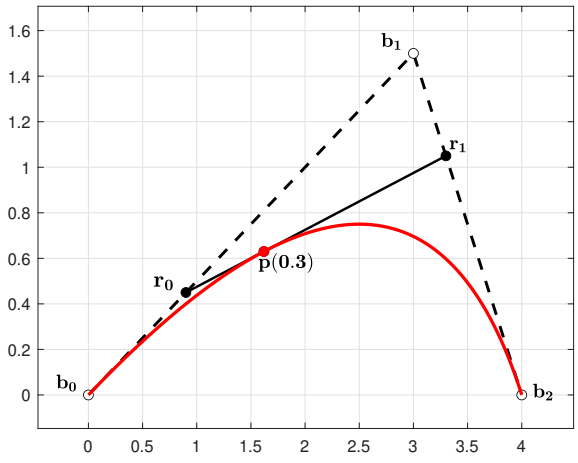


Figura 3.2: **Costruzione visiva della parabola**: abbiamo usato $b_0 = (0,0)$, $b_1 = (3, 1.5)$ e $b_3 = (4,0)$. Il punto sulla curva corrisponde alla valutazione della parabola in $t = 0.3$.
In rosso è rappresentata la parabola risultante.

Definizione 3.2.1 (Curva di Bézier)**.** Dati $n+1$ *punti di controllo*, $b_0, \ldots, b_n$ in $\mathbb{R}^d$, la Curva di Bézier di grado n determinata dai punti dati è la curva

$$b(t) = \sum_{i=0}^{n} b_i B_i^n(t) \qquad \text{per } t \in [0,1],$$

dove i coefficienti B_i^n sono i *polinomi di Bernstein di grado n*, per $i = 0, \ldots, n$.

Osservazione 3.2.2. Per ogni $t^* \in [0,1]$ fissato, il punto sulla curva di Bézier $b(t^*)$ è una combinazione convessa dei punti di controllo $b_0, \ldots, b_n \in \mathbb{R}^d$, i cui pesi sono dati dalle valutazioni dei polinomi di Bernstein nel parametro t^*.

3.3 Proprietà delle curve di Bézier

La maggior parte delle proprietà delle curve di Bézier sono ereditate dalla base dei polinomi di Bernstein. Riassumiamo qui le più importanti.

1. **Invarianza affine sotto trasformazioni parametriche.** Data una trasformazione affine con $a, b \in \mathbb{R}$, a<b, e una curva di Bézier, rispettivamente

   $$\phi(x) = \frac{x-a}{b-a}, \qquad b(t) = \sum_{i=0}^{n} b_i B_i^n(t),$$

   allora

   $$b(\phi(x)) = \sum_{i=0}^{n} b_i B_i^n \left(\frac{x-a}{b-a} \right).$$

   Ciò consente di riparametrizzare la curva in un qualsiasi intervallo reale $[a, b]$.

2. **Inviluppo convesso.** La curva disegnata per ogni $t \in [0,1]$ rimane all'interno dell'inviluppo convesso formato dai punti di controllo della curva.
   Questo segue dalle proprietà (2.3) e (2.4) dei polinomi di Bernstein.

3. **Interpolazione dei punti estremi.** Dalla proprietà 2.3 dei polinomi di Bernstein segue che la curva inizia esattamente nel primo punto di controllo b_0 e termina nell'ultimo b_n.

4. **Derivata.** La derivata di una curva di Bézier $b(t) = \sum_{i=0}^{n} b_i B_i^n(t)$ di *grado n*, dove $\{b_i \in \mathbb{R}^d\}_{i=0}^n$, è data da

   $$\frac{d}{dt} b(t) = n \sum_{i=0}^{n-1} (b_{i+1} - b_i) B_i^{n-1}(t). \tag{3.5}$$

   Introduciamo un concetto che ci sarà utile poi nella dimostrazione della proprietà 4.

Definizione 3.3.1. Siano $b_0, \ldots, b_n$ punti in $\mathbb{R}^d$. Definiamo per $0 \leq i \leq n$

$$\begin{cases} \Delta^0 b_i &= b_i, \\ \Delta^r b_i &= \Delta^{r-1} b_{i+1} - \Delta^{r-1} b_i. \end{cases} \qquad r \in \mathbb{N}$$

Consideriamo, ad esempio, i casi per $r = 0, 1, 2, 3$:

$$\begin{aligned} \Delta^0 b_i &= b_i, \\ \Delta^1 b_i &= b_{i+1} - b_i, \\ \Delta^2 b_i &= b_{i+2} - 2b_{i+1} + b_i, \\ \Delta^3 b_i &= b_{i+3} - 3b_{i+2} + 3b_{i+1} - b_i. \end{aligned}$$

Si osserva che i coefficienti di destra sono quelli di un triangolo di Tartaglia. In generale, la formula segue il seguente schema:

$$\Delta^r b_i = \sum_{j=0}^{r} \binom{r}{j} (-1)^{r-j} b_{i+j} \qquad r \in \mathbb{N},\ 0 \leq i \leq n.$$

Diamo ora la dimostrazione di 4.
Dimostrazione: Partiamo dalla derivata formale nella quale abbiamo utilizzato la proprietà (2.5) dei polinomi di Bernstein:

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

Riscriviamo ora l'equazione facendo uso delle proprietà del binomiale sul secondo membro:

$$\frac{d}{dt} b(t) = n \sum_{i=1}^{n} B_{i-1}^{n-1}(t) b_i - n \sum_{i=0}^{n-1} B_i^{n-1}(t) b_i.$$

Ora, applicando un cambio d'indice nel primo termine ($i \mapsto i+1$), otteniamo:

$$\frac{d}{dt} b(t) = n \sum_{i=0}^{n-1} B_i^{n-1}(t) b_{i+1} - n \sum_{i=0}^{n-1} B_i^{n-1}(t) b_i.$$

Infine, raccogliendo i termini simili:

$$\boxed{\frac{d}{dt} b(t) = n \sum_{i=0}^{n-1} (b_{i+1} - b_i) B_i^{n-1}(t)}.$$

□

5. **Derivata di ordine superiore.** La derivata di ordine r della curva di Bézier $b(t) = \sum_{i=0}^{n} b_i B_i^n(t)$ è data da:

$$\frac{d^r}{dt^r} b(t) = \frac{n!}{(n-r)!} \sum_{i=0}^{n-r} \Delta^r b_i B_i^{n-r}(t). \tag{3.6}$$

Dimostrazione: Usando la notazione (3.3.1)

$$\frac{d}{dt} b(t) = n \sum_{j=0}^{n-1} (b_{j+1} - b_j)\, B_j^{n-1}(t) = n \sum_{j=0}^{n-1} \Delta b_j\, B_j^{n-1}(t).$$

Ripetendo lo stesso procedimento r volte (oppure procedendo per induzione), ad ogni derivazione il grado dei polinomi di Bernstein diminuisce di 1 e compare un fattore moltiplicativo $n, (n-1), \ldots, (n-r+1)$. In dettaglio, applicando nuovamente la relazione della derivata ai polinomi $B_j^{n-1}, B_j^{n-2}, \ldots$, otteniamo

$$\begin{aligned} \frac{d^r}{dt^r} b(t) &= n(n-1)\cdots(n-r+1) \sum_{j=0}^{n-r} \Delta^r b_j\; B_j^{n-r}(t) \\ &= \boxed{\frac{n!}{(n-r)!} \sum_{j=0}^{n-r} \Delta^r b_j\; B_j^{n-r}(t)}\,. \end{aligned}$$

□

Capitolo 4

L'algoritmo di de Casteljau

4.1 Costruzione

Dato un insieme di punti di controllo $\{b_0, \ldots, b_n\} \in \mathbb{R}^d$ e un parametro fissato $t^* \in [0, 1]$, l'algoritmo di de Casteljau fornisce una costruzione geometrica ricorsiva del punto della curva di Bézier di grado n corrispondente a t^*.
Inizializzazione:

$$b_i^{(0)} = b_i \qquad i = 0, \ldots, n. \tag{4.1}$$

I punti $b_i^{(0)}$ coincidono con i punti di controllo iniziali.
Passo iterativo: Per $k = 1, \ldots, n$ e $i = 0, \ldots, n - k$:

$$b_i^{(k)} = (1 - t^*)b_i^{(k-1)} + t^* b_{i+1}^{(k-1)}. \tag{4.2}$$

Ad ogni livello k, i punti $b_i^{(k)}$ sono ottenuti mediante *combinazioni convesse* delle coppie di punti precedenti, seguendo lo schema triangolare come segue:

$$\begin{array}{ccccccccc}
b_0^{(0)} & & b_1^{(0)} & & b_2^{(0)} & \ldots & b_{n-1}^{(0)} & & b_n^{(0)} \\
& b_0^{(1)} & & b_1^{(1)} & & b_2^{(1)} & \ldots & b_n^{(1)} & \\
& & \ldots & & \ldots & & & & \\
& & b_0^{(n-2)} & & b_1^{(n-2)} & & b_2^{(n-2)} & & \\
& & & b_0^{(n-1)} & & b_1^{(n-1)} & & & \\
& & & & b_0^{(n)} & & & &
\end{array}$$

Si distinguono quindi due indici nell'iterazione $b_i^{(k)}$:

- k indica la **profondità** del livello, ovvero il numero di combinazioni convesse successive effettuate dall'algoritmo;
- i scorre sui **punti** del livello corrente.

Ad ogni riga k, il numero di punti coinvolti nell'algoritmo diminuisce di uno, fino ad arrivare a un singolo punto $b_0^{(n)}$.
Un esempio di curva costruita tramite il metodo di de Casteljau è visibile nella figura 4.1.

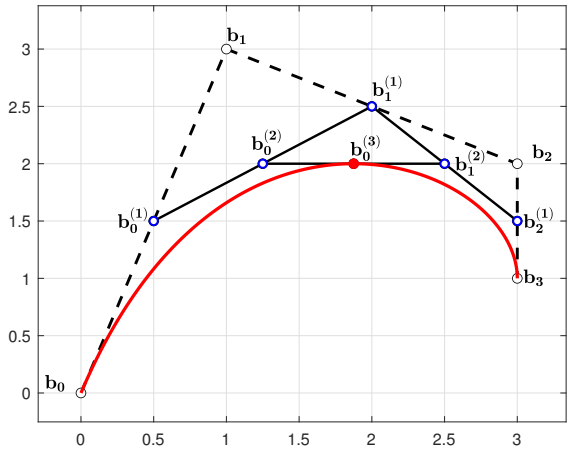


Figura 4.1: **Costruzione tramite algoritmo di de Casteljau**: abbiamo utilizzato come punti di controllo $b_0 = (0, 0)$, $b_1 = (1, 3)$, $b_2 = (3, 2)$, $b_3 = (3, 1)$.
I segmenti che congiungono i punti ad ogni livello di iterazione mostrano come agiscono i vari passi, generando un poligono di controllo sempre più piccolo, sino a convergere nel punto appartenente alla curva.
In figura è rappresentata in rosso la curva di Bézier risultante, in nero con linea tratteggiata il poligono di controllo e in nero con linea continua i passi dell'algoritmo di de Casteljau.

Algoritmo 1 Pseudo-algoritmo di de Casteljau

function DECASTELJAU($b_0, \ldots, b_n, t$)
  for $i = 0, n$ **do**
    $b_i^{(0)} = b_i$
  end for
  for $k = 1, \ldots, n$ **do**
    for $i = 0, n - k$ **do**
      $b_i^{(k)} = (1 - t)b_i^{(k-1)} + tb_{i+1}^{(k-1)}$
    end for
  end for
  return $b_0^{(n)}$
end function

Mostriamo ora che il punto finale rappresenta proprio il *punto della curva di Bézier* calcolato nel parametro fissato t^*.

Teorema 4.1.1. *Sia $b(t) = \sum_{i=0}^{n} b_i B_i^n(t)$ curva di Bézier di grado n, definita per $t \in [0,1]$, dove $b_0, \dots, b_n$ sono i punti di controllo in $\mathbb{R}^d$ e B_i^n polinomi di Bernstein di grado n per $i = 0, \dots, n$. Sia poi $t^* \in [0,1]$ parametro fissato. Allora l'algoritmo di de Casteljau applicato ai punti di controllo termina nel punto $b_0^{(n)}$, cioè vale che $b(t^*) = b_0^{(n)}$.*

Dimostrazione: Usiamo la proprietà (2.5) dei polinomi di Bernstein

$$b(t^*) = \sum_{i=0}^{n} b_i B_i^n(t^*) \overset{\text{prop. (4)}}{=} \sum_{i=0}^{n} b_i t^* B_{i-1}^{n-1}(t^*) + \sum_{i=0}^{n} b_i (1-t^*) B_i^{n-1}(t^*)\,. \tag{4.3}$$

Poiché $B_{-1}^{n-1}(t^*) = B_n^{n-1}(t^*) = 0$ otteniamo da (4.3)

$$b(t^*) = \sum_{i=0}^{n-1} b_{i+1} t^* B_i^{n-1}(t^*) + \sum_{i=0}^{n-1} b_i (1-t^*) B_i^{n-1}(t^*).$$

Raccogliamo i termini simili

$$b(t^*) = \sum_{i=0}^{n-1} [b_{i+1} t^* + b_i (1-t^*)] B_i^{n-1}(t^*) = \sum_{i=0}^{n-1} b_i^{(1)} B_i^{n-1}(t^*). \tag{4.4}$$

Iteriamo (4.4) per $k = 2, \dots, n$, otteniamo

$$b(t^*) = \sum_{i=0}^{n-k} b_i^{(k)} B_i^{n-k}(t^*).$$

Dunque, per $k = n$, rimane un unico termine ed essendo che $B_0^0(t^*) = 1$, segue

$$\boxed{b(t^*) = b_0^{(n)}}.$$

□

4.2 Interpretazione Geometrica

Ad un valore fissato di $t^* \in [0,1]$ vale $b_i^{(k)} = (1-t^*) b_i^{(k-1)} + t^* b_{i+1}^{(k-1)}$, dove

- il primo livello ($k = 1$) individua i valori sui *segmenti* che uniscono i punti di controllo consecutivi;

- il secondo livello ($k = 2$) esegue combinazioni convesse dei *valori trovati al primo livello*, restituendo nuovi segmenti e nuovi punti intermedi.
  Come visto, ad ogni passo i punti intermedi trovati sono uno in meno rispetto al numero di punti precedenti;

- il processo continua fino a ottenere un singolo punto.

In ogni passo, i punti restituiti sono dati da *combinazioni convesse* dei punti precedenti. Ciò è la diretta conseguenza della proprietà 2.3 dei polinomi di Bernstein. Questo consente, passo dopo passo, di ottenere un poligono di controllo ridotto, formato da nuovi punti trovati all'interno del poligono di controllo precedente, come in figura 4.1.

Osservazione 4.2.1. In [4], gli autori, oltre a suggerire dei pratici consigli per la stabilità dell'algoritmo di de Casteljau, ci forniscono anche una sua implementazione basica e l'esatto numero di FLOPs, vedi 5.1, pari a $\mathcal{O}(dn^2)$, dove d è la dimensione spaziale ed n è il grado dei polinomi di Bernstein.

Capitolo 5

L'algoritmo di Woźny-Chudy

5.1 Spiegazione dell'algoritmo

L'algoritmo di Woźny-Chudy nasce come alternativa all'algoritmo di de Casteljau per la valutazione di un punto su una curva di Bézier. Analogamente a quest'ultimo, il metodo si basa sull'uso di *combinazioni convesse* dei punti di controllo e dei punti aggiornati, assicurando che il punto calcolato rimanga all'interno dell'inviluppo convesso del poligono di controllo. La differenza fondamentale risiede nella struttura geometrica: invece di basarsi su una costruzione ricorsiva a vari livelli come de Casteljau, adotta una *formula iterativa diretta.* L'algoritmo inizializza il primo punto di aggiornamento ponendo $q_0 = b_0$ e il peso $h_0 = 1$. Successivamente calcola il nuovo coefficiente h_1 e aggiorna il punto intermedio calcolando una combinazione convessa tra q_0 e il successivo punto di controllo b_1. L'algoritmo procede in modo analogo: ad ogni passo, $k = 1 \ldots, n$, si determina il nuovo peso h_k e si aggiorna il punto q_k combinandolo con il precedente q_{k-1} e con il punto di controllo b_k. Al termine delle n iterazioni, il punto ottenuto q_n coincide esattamente con il punto della curva valutato in t. Questa formulazione conserva le proprietà geometriche, ma riduce la *complessità computazionale* da $\mathcal{O}(dn^2)$ dell'algoritmo di de Casteljau a $\mathcal{O}(dn)$. La costruzione di una curva mediante l'algoritmo di Woźny-Chudy si può vedere dalla figura 5.1.

Con il prossimo teorema mostreremo che l'algoritmo termina effettivamente sul punto della curva valutato in $t^* \in [0, 1]$.

Teorema 5.1.1. *Sia $b(t) = \sum_{k=0}^{n} b_k B_k^n(t)$ la curva di Bézier in base di Bernstein di grado n definita dai punti di controllo $b_0, \ldots, b_n \in \mathbb{R}^d$. Fissiamo un parametro $t^* \in [0, 1]$. Allora, l'algoritmo 2 applicato ai punti di controllo e al parametro scelto termina nel punto della curva.*

Dimostrazione: Definiamo le seguenti quantità

$$\tilde{h}_k := \frac{B_k^n(t^*)}{\sum_{0 \le j \le k} B_j^n(t^*)}, \qquad \tilde{q}_k = \frac{\sum_{0 \le j \le k} b_j B_j^n(t^*)}{\sum_{0 \le j \le k} B_j^n(t^*)} \qquad k = 0, \ldots, n.$$

Algoritmo 2 Pseudo-algoritmo di Woźny-Chudy

function WOŹNYCHUDY($b_0, \dots, b_n, t$)
  $q_0 = b_0$
  $h_0 = 1$
  for $k = 1, \dots, n$ **do**
    $h_k = \dfrac{h_{k-1} t (n-k+1)}{k(1-t) + h_{k-1} t (n-k+1)}$
    $q_k = (1 - h_k) q_{k-1} + h_k b_k$
  end for
  return q_n
end function

Allora abbiamo che $\tilde{h}_k \in [0,1]$, poiché rapporto di termini positivi e il termine al numeratore compare una volta al denominatore, e $\tilde{q}_k \in \mathbb{R}^d$ poiché somme di punti in $\mathbb{R}^d$ (il denominatore è uno scalare).
Inoltre, per $k = 0$

$$\tilde{h}_0 = \frac{B_0^n(t^*)}{B_0^n(t^*)} = 1, \quad \tilde{q}_0 = \frac{b_0 B_0^n(t^*)}{B_0^n(t^*)} = b_0$$

e per $k = n$

$$\tilde{q}_n = \frac{\sum_{0 \le j \le n} b_j B_j^n(t^*)}{\sum_{0 \le j \le n} B_j^n(t^*)} = \sum_{0 \le j \le n} b_j B_j^n(t^*) = b(t^*).$$

Essendo $\tilde{q}_k$ una combinazione baricentrica, ciò segue dalle proprietà della base di Bernstein, quindi appartiene all'inviluppo convesso dei punti di controllo.
Per concludere la dimostrazione, mostriamo che, per come abbiamo definito $\tilde{h}_k$ e $\tilde{q}_k$, allora vale ancora l'algoritmo 2:

$$\begin{cases} \tilde{q}_k = (1 - \tilde{h}_k)\tilde{q}_{k-1} + \tilde{h}_k b_k \,, \\ \\ \tilde{h}_k = \dfrac{\tilde{h}_{k-1} t^* (n-k+1)}{k(1-t^*) + \tilde{h}_{k-1} t^* (n-k+1)} \,. \end{cases}$$

Per semplicità di notazione, definiamo

$$S_m := \sum_{0 \le j \le m} B_j^n(t^*) \,, \qquad N_m := \sum_{0 \le j \le m} b_j B_j^n(t^*) \quad \text{per } 0 \le m \le n.$$

Osserviamo che, con la notazione introdotta

$$\tilde{h}_k = \frac{B_k^n(t^*)}{S_k} \,, \tag{5.1}$$

$$\tilde{q}_k = \frac{N_{k-1} + b_k B_k^n(t^*)}{S_k} \,. \tag{5.2}$$

(5.1) e (5.2) sono ben definite, in quanto per $t^* = 0$ l'algoritmo restituisce il primo punto di controllo e per $t^* = 1$ l'ultimo. Inoltre, per $t^* \in (0,1)$, i polinomi di Bernstein non hanno zeri.
Notiamo preliminarmente che, poiché $S_m \neq 0$ per ogni $t \in (0,1)$, allora $\tilde{h}_k^{-1} \neq 0$ per ogni $t \in (0,1)$.
Verifichiamo la prima relazione:

$$(1-\tilde{h}_k)\,\tilde{q}_{k-1} + \tilde{h}_k\, b_k = \left(1 - \frac{B_k^n(t^*)}{S_k}\right)\frac{N_{k-1}}{S_{k-1}} + b_k\,\frac{b_k^n(t^*)}{S_k} \qquad \text{abbiamo sostituito } \tilde{h}_k, \tilde{q}_{k-1}$$

$$= \frac{S_k - B_k^n(t^*)}{S_k}\frac{N_{k-1}}{S_{k-1}} + b_k\,\frac{B_k^n(t^*)}{S_k} \qquad (S_k - B_k^n(t^*) = S_{k-1})$$

$$= \frac{S_{k-1}}{S_k}\frac{N_{k-1}}{S_{k-1}} + b_k\frac{B_k^n(t^*)}{S_k} \qquad \text{semplifico } S_{k-1}$$

$$= \frac{N_{k-1} + b_k B_k^n(t^*)}{S_k} \qquad \text{ricordo com'è definito } \tilde{q}_k$$

$$= \tilde{q}_k\,.$$

Mostriamo ora che vale $h_k = \dfrac{h_{k-1}t^*(n-k+1)}{k(1-t^*) + h_{k-1}t^*(n-k+1)}$ per $\tilde{h}_k := \dfrac{B_k^n(t^*)}{\sum_{0\le j\le k} B_j^n(t^*)}$.

Dalla definizione di $\tilde{h}_k$

$$\tilde{h}_k^{-1} = \frac{S_k}{B_k^n(t^*)} = \frac{S_{k-1} + B_k^n(t^*)}{B_k^n(t^*)} = \frac{S_{k-1}}{B_k^n(t^*)} + 1$$

$$= \frac{B_{k-1}^n(t^*)}{\tilde{h}_{k-1}B_k^n(t^*)} + 1 \qquad \text{abbiamo usato che } S_{k-1} = \frac{B_{k-1}^n(t^*)}{\tilde{h}_{k-1}}$$

$$= \frac{1}{\tilde{h}_{k-1}}\,\frac{B_{k-1}^n(t^*)}{B_k^n(t^*)} + 1 \qquad \text{abbiamo usato (2.7)}$$

$$= \frac{k(1-t^*) + \tilde{h}_{k-1}t^*(n-k+1)}{\tilde{h}_{k-1}t(n-k+1)} \qquad \text{abbiamo fatto denominatore comune}$$

a questo punto è sufficiente invertire $\tilde{h}_k^{-1}$. $\square$

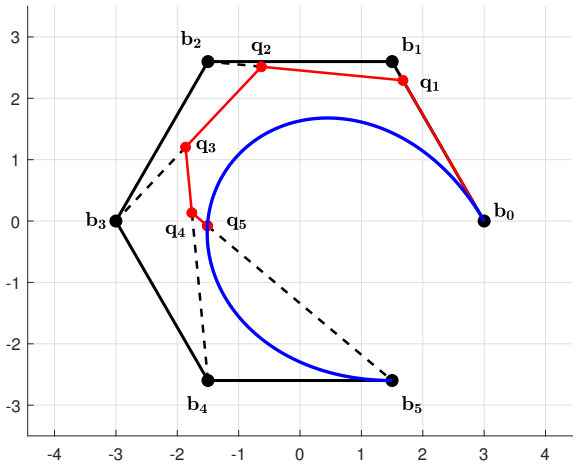


Figura 5.1: **Costruzione tramite algoritmo di Woźny-Chudy:** in questo esempio sono stati utilizzati come punti di controllo $b_0 = (3, 0)$, $b_1 = (1.5, 2.6)$, $b_2 = (-1.5, 2.6)$, $b_3 = (0, -3)$, $b_4 = (-1.5, -2.6)$ e $b_5 = (1.5, -2.6)$. La curva di Bézier risultante è visualizzata in blu, in nero con linea continua il poligono di controllo, in rosso i segmenti generati dai passi dell'algoritmo di Woźny-Chudy.

5.2 Implementazione di Woźny-Chudy

Riportiamo gli algoritmi presentati da Woźny e Chudy in [4], come alternativa all'algoritmo di de Casteljau. Notare che nella seconda implementazione, algoritmo 4, viene aggiunto un controllo nel caso $t \in [0, \frac{1}{2}]$ oppure $t \in (\frac{1}{2}, 1]$ col fine di garantire maggiore stabilità, come spiegato nella sezione 5.4.

Algoritmo 3 Prima implementazione di Woźny-Chudy

function WOZNYCHUDY$(b_0, \ldots, b_n, t)$
  $h = 1$
  $u = 1 - t$
  $n_1 = n + 1$
  $Q = b_0$

  for $k = 1, \ldots, n$ **do**
    $h = h \cdot t \cdot (n_1 - k)$
    $h = \dfrac{h}{k \cdot u + h}$
    $Q = (1 - h)\, Q + h\, b_k$
  end for
return Q
end function

Algoritmo 4 Seconda implementazione di Woźny-Chudy

function WOZNYCHUDY($b_0, \ldots, b_n, t$)
$h = 1$
$u = 1 - t$
$n_1 = n + 1$
$Q = b_0$

if $t \leq 0.5$ **then**
$u = \dfrac{t}{1-t}$
for $k = 1, \ldots, n$ **do**
$h = h \cdot u \cdot (n_1 - k)$
$h = \dfrac{h}{k + h}$
$Q = (1 - h)\, Q + h\, b_k$
end for

else
$u = \dfrac{1-t}{t}$
for $k = 1, \ldots, n$ **do**
$h = h \cdot (n_1 - k)$
$h = \dfrac{h}{k\,(u + h)}$
$Q = (1 - h)\, Q + h\, b_k$
end for
end if
return Q
end function

5.3 Significato geometrico

Dati i punti distinti $b_0, \ldots, b_n \in \mathbb{R}^d$ e un parametro $t \in [0, 1]$, l'algoritmo di Woźny-Chudy genera una successione di punti intermedi ottenuti tramite combinazioni baricentriche. Partendo da b_0, ciascun passo restituisce un nuovo punto ottenuto come combinazione convessa tra quello precedente e il successivo punto di controllo.

In questo modo, ad ogni iterazione si crea un segmento che unisce un punto interno al poligono di controllo a uno dei suoi vertici. Si può allora immaginare che la curva spezzata derivante da questo ciclo si avvolga all'interno del poligono, garantendo inoltre una limitatezza globale, come in 5.1.

A differenza di de Casteljau, che ad ogni passo costruisce un livello intermedio di punti, uno per segmento del poligono di controllo rimanente, l'algoritmo 2 aggiorna

un *solo punto alla volta*. Possiamo allora vedere questo metodo come una variante più compatta della costruzione classica, che mantiene l'interpretazione geometrica baricentrica, ma con un numero ridotto di operazioni.

Anche i coefficienti h_k svolgono un ruolo fondamentale nella costruzione progressiva dei punti q_k. Possiamo infatti interpretarli come pesi baricentrici che determinano quanto il punto corrente q_{k-1} e il successivo punto di controllo b_k contribuiscono alla formazione del nuovo punto intermedio q_k. In particolare, se h_k è prossimo allo zero, allora q_k si avvicinerà a q_{k-1}, mentre se è prossimo a uno, sarà vicino a b_k.

5.4 Proprietà dell'algoritmo

L'algoritmo proposto da Woźny-Chudy 5.2 gode di ottime proprietà di stabilità numerica ed efficienza, riuscendo a mantenere i vantaggi della geometria dell'algoritmo di de Casteljau.

1. **Stabilità:** L'algoritmo di Woźny–Chudy può essere implementato in due varianti principali. La prima, *Algoritmo 2.1,* 3 non tiene conto di accorgimenti per la stabilità numerica e si applica indistintamente per $t \in [0, 1]$. La seconda versione, *Algoritmo 2.2* 4, introduce una scelta condizionata nel caso $t \in [0, 1/2]$ oppure $t \in (1/2, 1]$, con l'obiettivo di ridurre l'instabilità dovuta all'aritmetica in virgola mobile. In entrambi i casi, l'algoritmo è costruito in modo da evitare casi di cancellazione numerica, che si verificano per $h_k \approx 1$. La formula

$$1 - h_k = \frac{h_k}{h_{k-1}} \frac{k(1-t)}{t(n-k+1)} \qquad 1 \leq k \leq n,$$

   consente di calcolare $1 - h_k$ in modo numericamente stabile. In questo modo si evitano differenze tra numeri quasi uguali, che sarebbero soggette a significative perdite di precisione dovute agli errori di arrotondamento, e si utilizzano invece rapporti ben condizionati.

2. **Complessità computazionale:** La complessità computazionale dell'algoritmo è lineare $\mathcal{O}(dn)$, dove *d è la dimensione dello spazio*. Per calcolare il costo effettivo dell'algoritmo consideriamo le implementazioni presentate in precedenza. Per l'algoritmo 3 il numero di FLOPs è $(3d + 6)n + 1$, mentre per 4, che tiene conto della stabilità, $(3d + 5)n + 2$.
   Il confronto dei FLOPs tra l'algoritmo ottimizzato e quello di de Casteljau sono riportati nella tabella 5.1.

Tabella 5.1: Confronto diretto sul numero di FLOPs

	Nuovo metodo	de Casteljau
FLOPs totali	$(3d+5)n+2$	$\frac{3dn(n+1)}{2}+1$
Addizioni	$(d+2)n+1$	$\frac{dn(n+1)}{2}+1$
Moltiplicazioni	$2(d+1)n$	$dn(n+1)$
Divisioni	$n+1$	0

Capitolo 6

Comparazione tra gli algoritmi

In questa sezione confronteremo i tempi di esecuzione degli algoritmi di de Casteljau e Woźny–Chudy in MatLab. Il programma utilizzato valuta 501 punti equidistanti nell'intervallo $[0, 1]$ per 10.000 curve di Bézier tramite la funzione `rand`, considerando prima curve nel piano e poi nello spazio. Il grado delle curve viene quindi fatto variare da 1 a 6 e, infine, assunto pari a 10, 15 e 20.
Il test è stato eseguito con un calcolatore dotato di processore **AMD Ryzen 7 7730U** da **2 GHz e 16 GB di RAM**, utilizzando la versione **MatLab R2024b**. Abbiamo indicato con d la dimensione dello spazio in cui vivono i punti di controllo e con n il grado della curva.

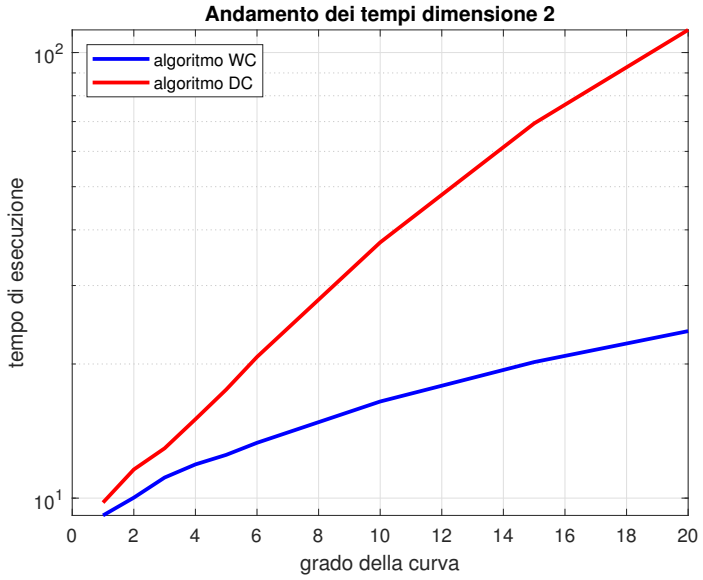


Figura 6.1: Andamento in scala semilogaritmica della tabella 6.1.

Come mostrano la tabella 6.1 e l'andamento 6.1, i risultati per la valutazione

delle curve di Bézier calcolate mediante l'algoritmo di Woźny-Chudy risultano più efficienti rispetto ai tempi impiegati dall'algoritmo di de Casteljau. In particolare, per gradi bassi le differenze sono minimali, con tempi di pochi secondi, sottolineando come de Casteljau sia comunque molto efficiente. Invece, per gradi alti i tempi di esecuzione dell'algoritmo di Woźny–Chudy sono nettamente migliori, confermando il vantaggio che ci si aspettava dalla teoria: mentre l'algoritmo di de Casteljau ha una complessità dell'ordine $\mathcal{O}(dn^2)$, Woźny–Chudy ha un costo computazionale che cresce linearmente col grado della curva. L'alternativa presentata conferma la sua validità in contesti in cui si ha a che fare con curve di alto grado.

Tabella 6.1: Confronto dei tempi in secondi. Woźny-Chudy vs de Casteljau

n	d	**Woźny–Chudy**	**de Casteljau**
1	2	**9.1451**	9.7736
	3	**9.4476**	9.8272
2	2	**10.0237**	11.5942
	3	**10.4234**	11.3537
3	2	**11.1221**	12.9358
	3	**11.0277**	13.0654
4	2	**11.9023**	15.0311
	3	**11.9884**	15.2593
5	2	**12.5052**	17.5231
	3	**12.7461**	17.7264
6	2	**13.3098**	20.7427
	3	**13.7103**	20.5063
10	2	**16.4750**	37.4890
	3	**16.3263**	37.7117
15	2	**20.1980**	69.3344
	3	**20.1358**	69.6592
20	2	**23.7030**	112.4148
	3	**23.8635**	113.6652

Bibliografia

[1] Gerald Farin. *Curves and Surfaces for CAGD – A Practical Guide (Fifth Edition)*. Elsevier, 2002. ISBN: 978-1-55860-737-8. DOI: https://doi.org/10.1016/B978-1-55860-737-8.X5000-5.

[2] Rida Farouki e Tim Goodman. «On the optimal stability of the Bernstein basis». In: *Math. Comput.* 65 (ott. 1996), pp. 1553–1566. ISSN: 1088-6842. DOI: 10.1090/S0025-5718-96-00759-4.

[3] Rida T. Farouki. «The Bernstein polynomial basis: A centennial retrospective». In: *Computer Aided Geometric Design* 29.6 (2012), pp. 379–419. ISSN: 0167-8396. DOI: https://doi.org/10.1016/j.cagd.2012.03.001.

[4] Paweł Woźny e Filip Chudy. «Linear-time geometric algorithm for evaluating Bézier curves». In: *Computer-Aided Design* 118 (2020), p. 102760. ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2019.102760.