Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

MODELLAZIONE GEOMETRICA CON MD-SPLINE

Relatore:
Chiar.mo Prof.
GIULIO CASCIOLA

Presentata da:
ANDREA BENETTI

II Sessione
Anno Accademico 2015/2016

Alla mia famiglia.

Introduzione

La modellazione geometrica studia metodi matematici per descrivere la forma di un oggetto. Costruire il modello di un oggetto significa definire una rappresentazione di questo mediante opportune primitive geometriche. Ogni oggetto può infatti essere scomposto in forme geometriche più semplici, divise in due categorie: classificabili e non classificabili. Alla prima categoria appartengono gli elementi della geometria classica: punti, segmenti, piani, sfere, cilindri, ecc. Per applicazioni diverse (ad esempio areonautica, industria automobilistica, architettura) sono tuttavia necessarie curve e superfici di forma "libera" o *free-form*.

Il problema della rappresentazione di una forma qualsiasi fu affrontato a partire dagli anni '60 con l'avvento dei primi sistemi CAD (*Computer Aided Design*) e portò alla nascita di quella disciplina chiamata *Computer Aided Geometric Design* (CAGD).

I tipi di rappresentazione matematica di curve più comunemente utilizzati nei sistemi CAD sono le curve di Bézier e le curve B-Spline. Le curve di Bézier possono essere semplici o a tratti. Quelle semplici sono definite da un unico polinomio e si rivelano spesso inadeguate per la modellazione geometrica. Quelle a tratti, essendo formate da più segmenti polinomiali, sono più flessibili ma soffrono della mancanza di regolarità nei punti di raccordo tra i segmenti. Le curve B-spline, invece, mantengono la flessibilità dei polinomi a tratti e, allo stesso tempo, permettono ai segmenti si raccordarsi con una certa regolarità. Comunque, il fatto che tutti i segmenti polinomiali che formano una B-Spline siano vincolati ad avere lo stesso grado porta spesso a

complicare inutilmente la fase di modellazione e a dover gestire informazioni superflue che non aggiungono nulla alla descrizione della curva.

Questa tesi tratta, sia dal punto di vista teorico che pratico, le curve MD-spline (*Multi Degree Spline*), una generalizzazione delle curve B-spline composte da segmenti polinomiali di gradi differenti. Queste permettono di modellare una forma in modo rapido ed intuitivo come le curve B-spline, avvantaggiandosi della possibilità di modificare il grado di ogni singolo segmento indipendentemente dagli altri. Ad ogni segmento di una MD-spline potrà quindi essere associato il minimo grado necessario a rappresentarne la forma e, di conseguenza, la descrizione matematica della curva includerà solo le informazioni effettivamente necessarie a rappresentarla.

Il lavoro di tesi ha permesso di ottenere il primo software di modellazione di curve MD-spline: il *Mini-System*.

Il capitolo 1, dopo aver illustrato le differenze tra i diversi tipi di rappresentazioni matematiche di curve, si focalizza sulla rappresentazione in forma parametrica e sui concetti geometrici associati. Tale rappresentazione è infatti la più idonea all'utilizzo nei sistemi CAD ed è alla base delle tipologie di curve descritte in questa tesi.

Nel capitolo 2 vengono descritte le curve di Bézier e le relative proprietà.

I capitoli 3 e 4 sono dedicati, rispettivamente, alla descrizione di funzioni e curve MD-Spline; ne vengono descritti gli aspetti teorici alla base, le proprietà ed i principali algoritmi di editing. Inoltre vengono evidenziate analogie e differenze con il caso *mono-degree* delle curve B-spline.

Infine, il capitolo 5 si occupa di descrivere il software Mini-System e mostra un esempio di modellazione di curve MD-spline che permette di apprezzare le funzionalità implementate ed i vantaggi offerti dalla rappresentazione spline multi-degree.

Indice

Capitolo 1

Curve Parametriche

Una generica curva, nel piano e nello spazio, si può esprimere mediante un'equazione matematica che individua tutti i suoi punti. In base alla forma di quest'equazione si parla di:

- rappresentazione implicita;
- rappresentazione esplicita, parametrica.

In generale, tutte le primitive geometriche come punti, curve e superfici possono essere rappresentate matematicamente in forma **implicita** o **esplicita** [1]. Questo capitolo mostra le differenze tra tali tipi di rappresentazioni ed, in particolare, mostra che la rappresentazione esplicita **parametrica** è quella più adatta per l'utilizzo CAD.

1.1 Curve in forma implicita ed esplicita

Rappresentare singoli punti in forma esplicita significa definire le coordinate degli stessi. In alternativa, la rappresentazione implicita definisce i punti come i valori per cui una certa funzione si annulla:

$$f : \mathbb{R} \to \mathbb{R}, \quad f(x) = 0$$

Ad esempio, in uno spazio 1-dimensionale, i punti $x = 1$ ed $x = -1$ sulla retta reale, possono essere rappresentati implicitamente come i valori per

cui si annulla la funzione $f(x) = x^2 - 1$. Tali punti sono infatti soluzione dell'equazione $x^2 - 1 = 0$.

La **rappresentazione in forma implicita di una curva** si ottiene in modo analogo. Una curva in forma implicita è definita come la soluzione di un'equazione (ovvero l'insieme dei punti che soddisfano):

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x, y) = 0$$

Ad esempio, la funzione $f(x, y) = x^2 + y^2 - 1$ permette di definire in forma implicita la curva chiusa rappresentante il cerchio unitario. Si chiama rappresentazione implicita perchè i punti (x, y) sulla curva devono essere determinati risolvendo l'equazione $x^2 + y^2 - 1 = 0$ e non sono quindi definiti esplicitamente.

La rappresentazione implicita ha le seguenti limitazioni:

- dipende dagli assi cartesiani del sistema di riferimento;
- non dà un'idea intuitiva della forma rappresentata;
- una trasformazione geometrica applicata alla curva ne cambia la forma matematica rappresentativa;
- il calcolo di punti della curva risulta laborioso.

A causa di tali limitazioni, nasce l'idea di esprimere le coordinate dei punti mediante leggi matematiche che le facciano dipendere in modo esplicito da uno o più parametri, quantità variabili che non hanno alcun legame con il sistema di riferimento cartesiano.

La **rappresentazione in forma esplicita di una curva** si può ottenere come grafico di una funzione in una variabile.

Definizione 1.1 (**Grafico di una funzione in una variabile**)**.**
Sia $D \subset \mathbb{R}$, f una funzione in una variabile $x \in D$, allora $f : D \to \mathbb{R}$ associa ad ogni $x \in D$ un unico elemento reale $y = f(x)$. D è il dominio di f ed il range dei valori che f assume è $\{y = f(x) \mid x \in D\}$. Il grafico di f è

l'insieme dei punti $G(f) = \{(x, y) \in \mathbb{R}^2 \mid y = f(x), x \in D\}$, che è una curva di equazione esplicita $y = f(x)$.

La curva in tale forma esplicita è ad un solo valore, cioè definisce un unico valore di y per ogni $x \in D$. Come conseguenza queste curve non possono auto-intersecarsi ma soprattutto non si può con tale forma rappresentare ogni curva. Questo perché non tutte le curve sono grafici di funzione. Ad esempio, il cerchio unitario definito prima in forma implicita non è rappresentabile in tale forma esplicita; con quest'ultima si può infatti rappresentare solo il semicerchio $y = \pm\sqrt{(1 - x^2)}$. Vale infatti la seguente definizione:

Definizione 1.2. Una curva nel piano è il grafico di una funzione se e solo se non esiste una linea verticale che interseca la curva più di una volta.

Inoltre, la forma esplicita a grafico di funzione soffre di parte delle limitazioni viste per la rappresentazione implicita. In particolare:

- dipende dagli assi di riferimento;
- non si possono, in generale, applicare alla curva trasformazioni affini (ad esempio traslazioni, rotazioni, ecc...) mantenendo l'integrità della rappresentazione. Infatti, se applichiamo una rotazione ad una curva rappresentata come grafico di funzione, questa, in generale, non è più un grafico di funzione.
- una curva che ha linee tangenti parallele agli assi comporta valori di tangenza all'infinito per alcuni punti.

La modellazione geometrica richiede che la scelta del sistema di coordinate non influenzi la forma della curva e che le trasformazioni affini non ne modifichino la rappresentazione. Quindi la formulazione esplicita appena vista è inadeguata per la modellazione geometrica.

Per superare tali difficoltà si utilizza una forma esplicita vettoriale, anche detta **forma parametrica** che risulta più semplice da gestire.

1.2 Curve in forma parametrica

Una curva parametrica in $\mathbb{R}^2$ è definita da una funzione vettoriale:

$$\mathbf{C}(t) = (x(t), y(t)) \tag{1.1}$$

dove $x(t)$, $y(t)$, sono funzioni del parametro $t \in I \subset \mathbb{R}$ dette componenti cartesiane del vettore posizione (punto sulla curva); sono funzioni scalari (ad es. polinomi). $\mathbf{C}(t)$ fa corrispondere, a valori dell'intervallo I, punti di $\mathbb{R}^2$, cioè $\mathbf{C} : I \to \mathbb{R}^2$. Al variare di t, le coordinate $(x(t), y(t))$ individuano un punto che si sposta sulla curva. L'intervallo $I = [a, b]$ definisce un insieme di punti in cui si sceglie di visualizzare la curva, anche se essa esiste anche per valori esterni a tale intervallo, cioè in $(-\infty, +\infty)$. In particolare la curva ristretta ad I si definisce segmento di curva.

L'**orientamento di una curva** è definito dal verso di percorrenza della curva stessa, ovvero il verso che si ottiene percorrendo la curva dal valore del parametro $t = a$ dando incrementi positivi al parametro verso $t = b$. La curva si dice **semplice** se non interseca se stessa e **chiusa** se $\mathbf{C}(a) = \mathbf{C}(b)$;

La rappresentazione esplicita parametrica gode dei seguenti vantaggi:

- non dipende dagli assi del sistema di riferimento;
- il calcolo di punti sulla curva è immediato;
- la visualizzazione della curva richiede solo di discretizzare il dominio parametrico 1D e valutare la curva direttamente dalla sua espressione parametrica per ogni punto del dominio. Rispetto alla rappresentazione implicita, tale operazione risulta meno onerosa e più accurata.
- le trasformazioni geometriche come traslazioni e rotazioni possono essere effettuate in modo semplice mediante moltiplicazioni con una matrice.

Tale rappresentazione è quindi la più adatta per l'utilizzo in ambiente CAD e, in generale, per la modellazione geometrica.

A titolo esemplificativo vengono di seguito mostrate l'equazione parametrica di una retta e quella della circonferenza unitaria.

Equazione parametrica di una retta. La retta passante per un punto $\mathbf{P}_0 \equiv (x_0, y_0)$ parallela ad un vettore non nullo $\mathbf{v}$ di componenti (v_x, v_y) ha equazione parametrica:

$$\mathbf{C}(t) = \mathbf{P}_0 + t\mathbf{v} = \begin{pmatrix} x_0 + tv_x \\ y_0 + tv_y \end{pmatrix}, \quad t \in (-\infty, +\infty)$$

Al variare del parametro t si ottengono tutti i punti di tale retta.

Equazione parametrica della circonferenza unitaria. Un cerchio C del piano xy, di centro l'origine O e raggio 1 ha equazione parametrica:

$$\mathbf{C}(t) = \begin{pmatrix} sin(t) \\ cos(t) \end{pmatrix}, \quad t \in [0, 2\pi] \tag{1.2}$$

Infine va sottolineato che l'equazione parametrica che definisce una determinata curva non è unica. Per chiarire questo concetto va innanzitutto data la seguente definizione.

Definizione 1.3 (Immagine o supporto della curva). Chiamiamo immagine o supporto di una curva l'insieme:

$$Im(\mathbf{C}) = \{\mathbf{C}(t) \mid t \in [a, b]\}$$

Il supporto di una curva è l'insieme di punti che la definiscono. È importante distinguere tra supporto della curva e funzioni coordinate $x(t)$ e $y(t)$ che ne definiscono la parametrizzazione. Una stessa curva può infatti essere rappresentata attraverso parametrizzazioni differenti.

Ad esempio la stessa circonferenza identificata dall'equazione (1.2) viene rappresentata anche attraverso l'equazione parametrica:

$$\mathbf{C}(t) = \begin{pmatrix} sin(2t) \\ cos(2t) \end{pmatrix}, \quad t \in [0, 2\pi] \tag{1.3}$$

Le due rappresentazioni parametriche (1.2) e (1.3) rappresentano il medesimo percorso (circonferenza) ma, valutandole per medesimi valori di t, rappresentano punti diversi sulla curva. Questo significa che il supporto della curva (punti della curva) è lo stesso ma la parametrizzazione (funzioni coordinate) è diffferente.

In altre parole, se interpretiamo la curva come la traiettoria di una particella che si muove nel tempo t, mentre nell'equazione (1.2) la curva inizia in $(0, 1)$ e la particella al massimo percorre la circonferenza una volta, nell'equazione (1.3) la particella si muove lungo la curva percorrendo due volte la circonferenza. La stessa curva viene cioè tracciata a velocità differenti.

1.3 Tangente, Normale e Curvatura

Prima di definire i concetti di tangente, normale e curvatura di una curva parametrica, dal momento che questi riguardano la geometria differenziale delle curve, vanno dapprima date le seguenti definizioni.

Sia $\mathbf{C}(t) = (x(t), y(t))$, con $t \in [a, b]$, l'equazione parametrica di una curva.

Definizione 1.4 (**Curva parametrica differenziabile**)**.** La curva parametrica $\mathbf{C}(t)$ è detta differenziabile in t se $x(t)$ e $y(t)$ sono differenziabili in $t \in [a, b]$ (ossia le derivate prime $x'(t)$ e $y'(t)$ sono continue)

Definizione 1.5 (**Curva parametrica regolare**)**.** La curva parametrica $\mathbf{C}(t)$ è detta regolare in t se $\mathbf{C}'(t) = (x'(t), y'(t))$ è diverso dal vettore nullo $(0, 0)$ per ogni $t \in [a, b]$.

Quindi, in una curva regolare, le derivate delle funzioni componenti non si annullano mai contemporaneamente. Una curva non regolare presenta invece dei punti in cui il vettore $\mathbf{C}'(t)$ è nullo, questi sono detti **punti singolari**.

Tangente ad una curva parametrica

Sia $\mathbf{C}(t)$ una curva definita dalle funzioni componenti $(x(t), y(t))$, e sia $\mathbf{P}_0 = \mathbf{C}(t_0)$ un punto regolare di $\mathbf{C}$.

Si chiama **vettore tangente** alla curva $\mathbf{C}(t)$ nel punto $\mathbf{P}_0$ un qualunque vettore parallelo alla retta tangente nel punto

Proposizione 1.1. *Il vettore*

$$\mathbf{C}'(t_0) = (x'(t_0), y'(t_0))$$

è tangente alla curva $\mathbf{C}(t)$ *nel punto* $\mathbf{C}(t_0)$.

Il versore tangente è il vettore tangente normalizzato ovvero di lunghezza unitaria. Dato il vettore tangente $\mathbf{C}'(t_0)$, il versore ad esso corrispondente si trova ponendo:

$$\mathbf{T}_{t_0} = \frac{\mathbf{C}'(t_0)}{||\mathbf{C}'(t_0)||_2}$$

Una curva regolare varia con continuità, ammette sempre retta tangente in ogni punto e la tangente varia con continuità.

Tornando al modello fisico del moto di una particella, ad ogni istante t_0, le coordinate $(x(t_0), y(t_0))$ individuano un punto che si sposta sulla curva con velocità $\mathbf{v}$ data dalla tangente alla curva parametrica in t_0:

$$\mathbf{v}(t_0) = \frac{d}{dt}\mathbf{C}(t_0) = \mathbf{C}'(t_0) = (x'(t_0), y'(t_0))$$

In particolare si ha che:

- il *modulo del vettore velocità* rappresenta la velocità istantanea con la quale la particella si muove lungo la curva, ed è dato da:

$$||\mathbf{C}'(t_0)||_2 = \sqrt{x'(t_0)^2 + y'(t_0)^2}$$

- la *direzione del vettore velocità* è data dal vettore tangente $\mathbf{C}'(t_0)$ e punta nel verso delle t crescenti.

Normale ad una curva parametrica

Dato un vettore di componenti (x, y), il vettore ortogonale ad esso è ottenuto scambiando la posizione delle sue componenti e invertendo il segno di una di esse. Pertanto il vettore normale alla curva $\mathbf{C}(t)$ nel punto t_0, che è ortogonale al vettore tangente nello stesso punto, è dato da:

$$\mathbf{N}_{t_0} = \frac{(-y'(t_0), x'(t_0))}{||\mathbf{C}'(t_0)||_2}$$

Si può infatti mostrare la reciproca ortogonalità verificando che il loro prodotto scalare è nullo:

$$< \mathbf{T}_{t_0} , \mathbf{N}_{t_0} > = \frac{-x'y' + y'x'}{||\mathbf{C}'(t_0)||_2} = 0$$

Curvatura di una curva parametrica

Una retta può essere definita come una curva che non cambia mai direzione, ovvero il rispettivo vettore tangente ha direzione costante. Di conseguenza è ragionevole pensare che la variazione della direzione del vettore tangente possa indicare di quanto una curva si discosta dall'essere una retta. Per misurare efficientemente questa variazione e quindi la curvatura della curva va usato il versore tangente $\mathbf{T}$.

In particolare, poiché la curvatura di una curva in generale cambia da punto a punto, quello che vogliamo calcolare è la misura matematica di quanto la curva devii dall'essere retta nell'intorno di un punto. Ad esempio, in una circonferenza la curvatura è la stessa in ogni punto e sarà maggiore quanto il raggio sarà minore. Invece, per la parabola

$$C(t) = \begin{pmatrix} t \\ t^2 \end{pmatrix}, \quad t \in [-1, 1]$$

la curvatura è maggiore attorno al vertice e diventa sempre più piccola allontanandosi da esso, dove la parabola diventa sempre più simile ad una retta. Formalmente possiamo dare la seguente definizione.

Definizione 1.6 (**Curvatura di una curva parametrica**).
Sia $\mathbf{C}(t)$, $\mathbf{C} : I \to \mathbb{R}^2$, una curva regolare di classe C^k, con $k \geq 2$. La curvatura di $\mathbf{C}(t)$ è la funzione $k : I \to \mathbb{R}^+$. di classe C^{k-2}, data da:

$$k(t) = \frac{||\mathbf{T}'(t)||}{||\mathbf{C}'(t)||} \tag{1.4}$$

L'equazione (1.4) esprime la variazione del versore tangente rispetto ad una variazione nella posizione. Nel punto t_0, la curvatura è lo scalare $k(t_0)$.

Ad esempio, data la retta $\mathbf{C}(t) = \mathbf{P}_0 + t\,\mathbf{v}_1$, allora $C'(t) = v_1$, costante, quindi:

$$\mathbf{T}(t) = \frac{\mathbf{C}'(t)}{||\mathbf{C}'(t)||} = \frac{\mathbf{v}_1}{||\mathbf{v_1}||}$$

e $\mathbf{T}'(t) = 0$. Quindi, per la (1.4) una linea retta ha curvatura nulla.

Se consideriamo, invece, una circonferenza di raggio R, la cui equazione parametrica è $\mathbf{C}(t) = (R\,cos(t), R\,sin(t))$, allora $\mathbf{C}'(t) = (-R\,sin(t), R\,cos(t))$ e $||\mathbf{C}'(t)||_2 = R$. Avremo quindi:

$$\mathbf{T}(t) = \frac{\mathbf{C}'(t)}{||\mathbf{C}'(t)||} = (-sin(t), cos(t))$$

$$k(t) = \frac{||\mathbf{T}'(t)||}{R} = \frac{||(-cos(t), -sin(t))||_2}{R} = \frac{1}{R}$$

Quindi, come accennato in precedenza e come intuibile, circonferenze di grande raggio avranno una piccola curvatura mentre circonferenze di piccolo raggio avranno una grande curvatura.

Per concludere, è importante notare che la curvatura dipende dalla curva stessa (supporto della curva), non dal modo in cui la si percorre (la particolare parametrizzazione scelta per rappresentare la curva)

1.4 Continuità parametrica e geometrica

Nella pratica, per ottenere una curva complessa ed un maggior controllo locale della forma, è necessario unire tra loro diversi segmenti di curva. Tali segmenti, per descrivere un'unica forma continua, dovranno come minimo

raccordarsi tra loro e, affinché ciò avvenga in modo dolce (*smooth*), dovranno raccordarsi in un modo opportuno. A tal proposito si deve fare riferimento ai concetti di continuità parametrica e geometrica di curve.

Continuità parametrica

Convenzionalmente, la regolarità di una curva è misurata mediante la continuità delle sue derivate.

Definizione 1.7 (**Continuità parametrica di una curva**)**.** una curva $\mathbf{C}(t)$ è detta essere C^k in u se e solo se:

$$\lim_{t \to u^+} \mathbf{C}^{(i)}(t) = \lim_{t \to u^-} \mathbf{C}^{(i)}(t) \quad i = 0, \ldots, k$$

Analogamente date due curve $\mathbf{C}_1(t)$ e $\mathbf{C}_2(t)$ definite su intervalli parametrici consecutivi $[u_1, u_2]$ e $[u_2, u_3]$ diremo che sono C^k nel punto comune u_2 se e solo se:

$$\lim_{t \to u_2^+} \mathbf{C}_2^{(i)}(t) = \lim_{t \to u_2^-} \mathbf{C}_1^{(i)}(t) \quad i = 0, \ldots, k$$

In particolare, nel punto di raccordo u_2 si possono distinguere tre casi:

- **Continuità di posizione, il raccordo è** C^0. Le curve si raccordano solo attraverso il vettore posizione ($\mathbf{C}_1(u_2) = \mathbf{C}_2(u_2)$). Questa è la condizione più debole: le due curve hanno in comune il vettore posizione nel punto di raccordo; ma le tangenti e le curvature sono diverse.

- **Continuità di tangenza, il raccordo è** C^1. Le curve si raccordano con continuità della tangente. In questo caso le due curve hanno in comune, nel punto di raccordo, sia il vettore posizione che il vettore tangente. In particolare le due tangenti sono uguali sia in direzione che in modulo nel punto di raccordo. Vale cioè $\mathbf{C}_1(u_2) = \mathbf{C}_2(u_2)$, $\mathbf{C}_1'(u_2) = \mathbf{C}_2'(u_2)$ e $||\mathbf{C}_1'(u_2)||_2 = ||\mathbf{C}_2'(u_2)||_2$.

- **Continuità di curvatura, il raccordo è** C^2. È la situazione di miglior "saldatura" dal momento che anche la curvatura non presenta salti nel punto di raccordo. Si raccordano in particolare il vettore posizione, il vettore tangente e si ha la stessa curvatura k.

Continuità geometrica

La continuità geometrica è una generalizzazione della continuità parametrica, invariante sotto riparametrizzazione.

Definizione 1.8 (**Continuità geometrica di una curva**)**.** Una curva $\mathbf{C}(t)$ è detta essere continua geometricamente, e denotata con G^k, se e solo se esiste una funzione di riparametrizzazione $\varphi(\tau)$ tale che $\mathbf{C}(\varphi(\tau))$ è regolare e con continuità parametrica C^k.

Chiaramente tutte le curve regolari C^k sono G^k, ma non viceversa.
Barsky e DeRose [2] introducono la nozione di continuità geometrica e derivano delle condizioni, dette *Beta-condizioni*, sulle componenti della curva $\mathbf{C}(t)$ che valgono se e solo se la curva è geometricamente continua. Di seguito vengono descritte soltanto le Beta-condizioni per G^0, G^1 e G^2.

La Beta-condizione per G^0 è identica alla condizione per la continuità parametrica C^0, cioè:

$$\lim_{t \to u^+} \mathbf{C}^{(0)}(t) = \lim_{t \to u^-} \mathbf{C}^{(0)}(t)$$

Le Beta-condizioni per G^1 dicono che una curva $\mathbf{C}(t)$ è G^1 se e solo se è G^0 e

$$\lim_{t \to u^+} \mathbf{C}^{(1)}(t) = \lim_{t \to u^-} \beta_1 \, \mathbf{C}^{(1)}(t)$$

per un certo $\beta_1 > 0$. Questo significa che, se due segmenti di curva si congiungono in un punto $\mathbf{P}_0$ con continuità G^1, allora le direzioni dei vettori tangenti dei due segmenti nel punto di contatto $\mathbf{P}_0$ sono uguali mentre i moduli differiscono per un fattore scalare positivo (e quest'ultima è la differenza rispetto alla continuità C^1). Detto in altro modo, la particella che si muove lungo la curva varia la sua velocità a sbalzi ma ancora percorre una curva continua siccome la sua direzione cambia in modo continuo.

La curva $\mathbf{C}(t)$ è G^2 se e solo se, in aggiunta alle Beta-condizioni per G^0 e G^1 si ha:

$$\lim_{t \to u^+} \mathbf{C}^{(2)}(t) = \lim_{t \to u^-} (\beta_1^2 \, \mathbf{C}^{(2)}(t) + \beta_2 \, \mathbf{C}^{(1)}(t))$$

per un certo β_2.

Capitolo 2

Curve di Bézier

La rappresentazione di curve nella forma di Bézier è una delle più utilizzate in *computer graphics* e in modellazione geometrica. Le curve di Bézier vennero introdotte da due ingegneri francesi, Paul de Casteljau che le propose nel 1959 e Pierre Étienne Bézier che le rese popolari nel 1962 utilizzandole nel primo sistema CAD (UNISURF) per progettare la carrozzeria delle automobili Renault.

Dal momento che le componenti della rappresentazione parametrica di una curva di Bézier sono polinomi nella base di Bernstein, questo capitolo, prima di discutere le curve di Bézier e le relative proprietà, fornisce una introduzione alle funzioni polinomiali.

2.1 I polinomi di Bernstein

I polinomi sono uno strumento estremamente utile in matematica poiché sono definiti in modo semplice, possono essere valutati velocemente e possono approssimare un'ampia classe di funzioni. Inoltre possono essere differenziati ed integrati facilmente e composti a tratti per formare curve più complesse. I polinomi sono comunemente presentati nella classica base delle potenze, dove un polinomio di grado al più n, cioè $p(x) \in \mathbb{P}_n$, è rappresentato nella

forma seguente:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = \sum_{i=0}^{n} a_i x^i$$

L'insieme dei polinomi $p(x) \in \mathbb{P}_n$ forma uno spazio vettoriale; l'insieme delle funzioni $\{1, x, x^2, \ldots, x^n\}$ forma una base per questo spazio vettoriale, cioè ogni polinomio di grado minore o uguale a n può essere scritto come combinazione lineare di queste funzioni. La dimensione di questo spazio è $n + 1$. Questa base, comunemente detta base delle potenze, è solo una delle infinite basi dello spazio dei polinomi $\mathbb{P}_n$.

Un'altra base comunemente utilizzata per rappresentare un polinomio, che tiene conto dell'intervallo di definizione sul quale si vuole studiare il comportamento dello stesso e che risulta particolarmente utile nell'ambito della progettazione CAD, è la **base di Bernstein**.

I polinomi di Bernstein di grado n nell'intervallo $t \in [0, 1]$ sono definiti dalla formula:

$$B_{i,n}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i}, \quad i = 0, \ldots, n, \quad \begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!} \tag{2.1}$$

In un generico intervallo $[a, b]$ della retta reale l'espressione (2.1) dei polinomi di Bernstein assume la forma:

$$B_{i,n}(x) = \begin{pmatrix} n \\ i \end{pmatrix} \frac{(x-a)^i (b-x)^{n-i}}{(b-a)^n}, \quad i = 0, \ldots, n$$

In questo caso ogni polinomio $p \in \mathbb{P}_n$ si può scrivere come:

$$p(x) = \sum_{i=0}^{n} b_i B_{i,n}(x), \quad x \in [a, b]$$

Poiché i polinomi sono invarianti per traslazione e scala dell'intervallo di definizione, sia $x \in [a, b]$ e $t \in [0, 1]$, allora $x = a + t(b - a)$ e si avrà:

$$B_{i,n}(x) = B_{i,n}(t), \quad i = 0, \ldots, n$$

cioè il cambio di variabile non altera i coefficienti. Se si vuole valutare $p(x)$ in un determinato valore $\underline{x}$, si determina il corrispondente valore di $\underline{t} = (\underline{x} - a)/(b - a)$, poi si valuta $p(\underline{t})$, quindi sarà $p(\underline{x}) = p(\underline{t})$.

Facciamo alcuni esempi di basi di Bernstein.

La base dello spazio $\mathbb{P}_1$ dei polinomi di Bernstein di grado al più 1 è definita dalle funzioni:

$$B_{0,1}(t) = 1 - t, \quad B_{1,1}(t) = t$$

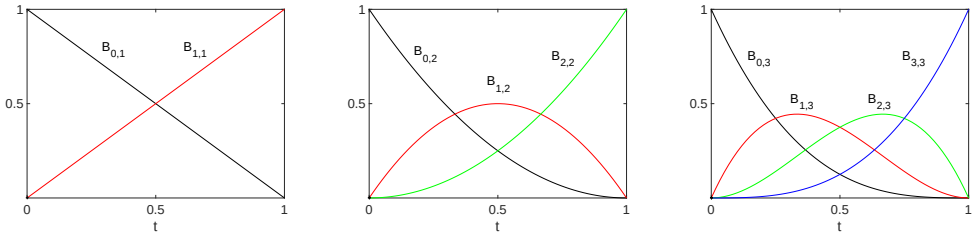
La base dello spazio $\mathbb{P}_2$ dei polinomi di Bernstein di grado al più 2 è definita dalle funzioni:

$$B_{0,2}(t) = (1-t)^2, \quad B_{1,2}(t) = 2t(1-t), \quad B_{2,2}(t) = t^2$$

La base dello spazio $\mathbb{P}_3$ dei polinomi di Bernstein di grado al più 3 è definita dalle funzioni:

$$B_{0,3}(t) = (1-t)^3, \quad B_{1,3}(t) = 3t(1-t)^2, \quad B_{2,3}(t) = 3t^2(1-t), \quad B_{3,3}(t) = t^3$$

I grafici delle suddette funzioni base di Bernstein di grado 1,2 e 3 sono mostrati in figura 2.1.



(a) n=1 *(b)* n=2 *(c)* n=3

Figura 2.1: Funzioni base di Bernstein

2.1.1 Proprietà dei polinomi di Bernstein

I polinomi di Bernstein godono delle seguenti importanti proprietà:

P.2.1 (*Non negatività*) I polinomi di Bernstein sono tutti non negativi, cioè:

$$B_{i,n}(t) \geq 0, \quad i = 0, \ldots, n, \quad \forall t \in [0,1]$$

P.2.2 (*Partizione dell'unità*) Formano una partizione dell'unità, cioè:

$$\sum_{i=0}^{n} B_{i,n}(t) = 1, \quad \forall t \in [0,1]$$

P.2.3 (*Condizione agli estremi*) Vale:

$$B_{i,n}(0) = \delta_{i,0} \text{ e } B_{i,n}(1) = \delta_{i,n} \quad \text{dove} \quad \delta_{i,j} = \begin{cases} 1 \text{ se } i = j \\ 0 \text{ se } i \neq j \end{cases}$$

P.2.4 (*Unicità del massimo*) $B_{i,n}(t)$ ha un solo massimo $[0,1]$ e questo occorre in corrispondenza di $t = i/n$.

P.2.5 (*Simmetria*) Per ogni n l'insieme di funzioni $\{B_{i,n}(t)\}$ è simmetrico rispetto a $t = 1/2$. Vale cioè la relazione $B_{i,n}(t) = B_{i,n}(1-t)$

P.2.6 (*Variazione di segno dei coefficienti*) un polinomio espresso nella base di Bernstein ha un numero di variazioni di segno minore o uguale al numero di variazioni di segno del vettore dei suoi coefficienti (dove coefficienti nulli non vengono considerati)

P.2.7 (*Definizione ricorsiva*) I polinomi di Bernstein di grado n possono essere definiti in termini dei polinomi di Bernstein di grado $n-1$. L'i-esimo polinomio di Bernstein definito in $[0,1]$ è infatti definito dalla seguente formula ricorrente:

$$B_{i,n}(t) = tB_{i-1,n-1}(t) + (1-t)B_{i,n-1}(t)$$

con $B_{0,0}(t) = 1$ e $B_{i,n}(t) = 0$ per ogni $i \notin [0,n]$.

P.2.8 (*Derivata*) La derivata di un polinomio di Bernstein è data da:

$$B'_{i,n}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t))$$

La proprietà di partizione dell'unità è molto importante in modellazione geometrica perché, se questa vale, allora per ogni insieme di punti $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$, e per ogni $t \in [0,1]$, la combinazione lineare:

$$p(t) = \mathbf{P}_0 B_{0,n}(t) + \mathbf{P}_1 B_{1,n}(t) + \ldots + \mathbf{P}_n B_{n,n}(t)$$

è una *combinazione affine* dell'insieme di punti. Inoltre, siccome per la proprietà di non negatività i coefficienti sono tutti non negativi ($B_{i,n}(t) \geq 0$), allora $p(t)$ è anche *combinazione convessa* dei punti.

La proprietà sulla derivata permette di definire la derivata di un polinomio nella base di Bernstein come:

$$p'(t) = \sum_{i=0}^{n-1} d_i B_{i,n-1}(t)$$

dove $d_i = n\,(b_{i+1} - b_i),\ i = 0, \ldots, n-1$, sono i coefficienti del polinomio derivata prima, di grado $n-1$, nella base di Bernstein. Se $p(x)$ è definito su $[a,b]$, si noti che ogni d_i deve essere calcolato come $d_i = n\,(b_{i+1} - b_i)/(b-a)$.

2.2 Curve di Bézier

Il problema fondamentale della modellazione geometrica consiste nella ricerca di una formulazione matematica tale da consentire la costruzione di una curva a partire da un insieme di punti assegnati.

Una curva piana di Bézier di grado n (ordine $n+1$) in forma parametrica può essere definita a partire da un insieme di punti $\mathbf{P}_i = (x_i, y_i)$, $i = 0, \ldots, n$, con parametro $t \in [0,1]$, ed è rappresentata nella forma:

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,n}(t) = \begin{pmatrix} C_x(t) \\ C_y(t) \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n} x_i B_{i,n}(t) \\ \sum_{i=0}^{n} y_i B_{i,n}(t) \end{pmatrix} \qquad (2.2)$$

dove $B_{i,n}(t)$, $i = 0, \ldots, n$ sono i *polinomi di Bernstein* di grado n definiti nell'intervallo $[0, 1]$ dalla formula (2.1). Gli $n + 1$ punti $\mathbf{P}_i = (x_i, y_i)$, $i = 0, \ldots, n$, sono chiamati **punti di controllo** e formano, uniti da segmenti, **il poligono di controllo** che approssima la forma della curva. $C_x(t)$ e $C_y(t)$ sono polinomi nella base di Bernstein.

In figura 2.2 è mostrata una curva di Bézier di grado $n = 3$ il cui poligono di controllo è formato dai punti $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ e $\mathbf{P}_3$.

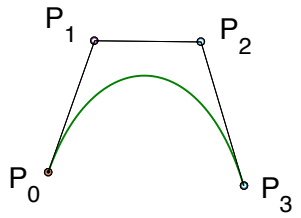


Figura 2.2: Curva di Bézier cubica ($n = 3$)

2.2.1 Proprietà delle curve di Bézier

Una curva di Bézier di grado n in $[0, 1]$ gode delle seguenti proprietà:

P.2.9 la curva interpola il primo e l'ultimo punto di controllo cioè:

$$\mathbf{C}(0) = \mathbf{P}_0, \quad \mathbf{C}(1) = \mathbf{P}_n$$

P.2.10 la curva è continua ed ha derivate continue di tutti gli ordini;

P.2.11 le direzioni delle tangenti alla curva agli estremi sono parallele al primo $(\mathbf{P}_1 - \mathbf{P}_0)$ ed ultimo $(\mathbf{P}_n - \mathbf{P}_{n-1})$ segmento del poligono di controllo;

P.2.12 la curva è invariante rispetto a trasformazioni affini: operando una trasformazione affine al poligono di controllo si ottiene la stessa trasformazione sulla curva;

P.2.13 la curva è contenuta nel *guscio convesso* (*convex hull*) formato dai suoi punti di controllo. Di conseguenza se tutti i punti di controllo

sono allineati lungo una linea, la curva stessa è una retta passante per tali punti di controllo;

P.2.14 la curva è *variation diminishing* cioè il numero di intersezioni della curva con una qualsiasi retta non è mai maggiore del numero di intersezioni di tale retta con il poligono di controllo della curva.

2.3 Valutazione di una curva di Bézier

Per il disegno di una curva di Bézier sarà necessario valutare la curva in un certo numero di valori del parametro t compresi in $[0, 1]$. Per valutare una curva di Bézier di grado n in corrispondenza di un determinato valore del parametro t, si possono seguire due metodi:

- valutazione tramite la formula (2.2);
- valutazione tramite algoritmo di de Casteljau.

Nel primo metodo si valutano in t i polinomi in forma di Bernstein che rappresentano le componenti della curva $\mathbf{C}(t) = (C_x(t), C_y(t))$. Ovvero si valutano prima i polinomi di Bernstein di grado n in t, quindi si moltiplicano per i corrispondenti punti di controllo secondo la combinazione lineare della formula (2.2). Il secondo metodo è descritto di seguito.

2.3.1 Algoritmo di de Casteljau

L'algoritmo introdotto da de Casteljau permette di valutare una curva di Bézier mediante una serie ricorsiva di interpolazioni lineari.

L'interpolazione lineare (*Lerp*) è una tecnica per generare un nuovo valore compreso tra altri due. Consideriamo la funzione di interpolazione tra due punti a e b con parametro t, cioè:

$$Lerp(t, a, b) = (1 - t)a + tb$$

I passi dell'algoritmo vengono descritti attraverso il seguente esempio. Supponiamo che si voglia valutare una curva cubica ($n = 3$) di Bézier $\mathbf{C}(t) = \sum_{i=0}^{3} \mathbf{P}_i B_{i,3}(t)$ in corrispondenza di un valore t. A partire dai 4 punti di controllo iniziali della curva $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ si eseguono i seguenti 3 passi:

(1) si calcola $\mathbf{P}_0^1 = Lerp(t, \mathbf{P}_0, \mathbf{P}_1)$, $\mathbf{P}_1^1 = Lerp(t, \mathbf{P}_1, \mathbf{P}_2)$, $\mathbf{P}_2^1 = Lerp(t, \mathbf{P}_2, \mathbf{P}_3)$

(2) si calcola $\mathbf{P}_0^2 = Lerp(t, \mathbf{P}_0^1, \mathbf{P}_1^1)$, $\mathbf{P}_1^2 = Lerp(t, \mathbf{P}_1^1, \mathbf{P}_2^1)$,

(3) si calcola $\mathbf{P}_0^3 = Lerp(t, \mathbf{P}_0^2, \mathbf{P}_1^2)$,

Il valore della curva in t è dato da $\mathbf{C}(t) = \mathbf{P}_0^3$

In figura 2.3 vengono rappresentati tali passi per $t = 0.4$.

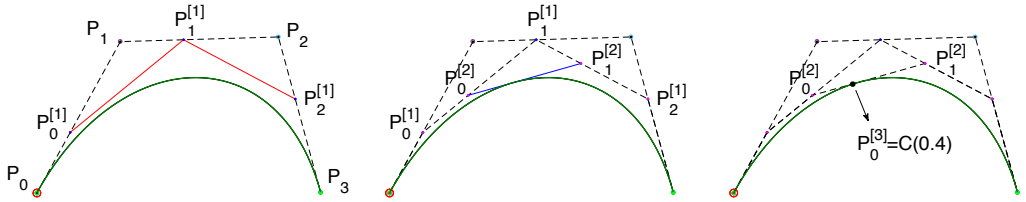


Figura 2.3: Passi dell'algoritmo di de Casteljau per $n = 3$ e $t = 0.4$

L'algoritmo si può generalizzare a curve di Bézier di ogni grado semplicemente applicando più passi.

Per visualizzare la curva si può discretizzare l'intervallo parametrico $[0, 1]$ in modo uniforme, valutare la curva in corrispondenza di tali valori parametrici t, ed unire con segmenti i punti sulla curva così ottenuti.

Capitolo 3

Funzioni MD-Spline

Le componenti $(C_x(t), C_y(t))$ di una curva di Bézier sono polinomi definiti nell'intero intervallo parametrico $[a, b]$ di definizione della curva. Tuttavia curve definite da un unico segmento polinomiale sono spesso inadeguate per la modellazione geometrica. I principali svantaggi sono i seguenti:

- l'utilizzo di molti punti di controllo, spesso necessario per poter accuratamente rappresentare forme complesse, impone un alto grado del polinomio (per generare una curva di Bezier da $n+1$ punti di controllo, è richiesto un polinomio di grado n). Curve di grado elevato sono però inefficienti da valutare e numericamente instabili;

- il controllo della forma della curva non è locale: una modifica ad un punto di controllo influenza in maniera globale la forma della curva.

La soluzione consiste nel lavorare con polinomi di grado relativamente basso e dividere l'intervallo di interesse in intervalli più piccoli, ovvero utilizzare curve le cui componenti siano funzioni polinomiali a tratti (*piecewise polynomial*).

Questo capitolo si occupa di funzioni polinomiali a tratti in spazi Multi-Degree. In particolare, vengono definite le funzioni MD-spline (dove MD sta per Multi-Degree) [3–8] che estendono le tradizionali funzioni NUBS. Queste ultime sono costituite da tratti polinomiali aventi tutti lo stesso grado e

costituiscono le funzioni componenti delle curve B-spline ampiamente usate in modellazione. Le funzioni MD-spline, invece, permettono di avere funzioni a tratti polinomiali di grado differente e sono alla base delle omonime curve MD-spline trattate nel prossimo capitolo. Dal momento che le MD-spline sono una generalizzazione delle spline, durante la trattazione, viene spesso evidenziato come, da una determinata definizione Multi-Degree, sia possibile ricondursi al caso "mono-degree".

3.1 Spazi polinomiali a tratti

Definiamo lo spazio delle funzioni polinomiali a tratti.

Definizione 3.1 (**Spazio Multi-Degree dei polinomi a tratti**). Sia $[a, b]$ un intervallo chiuso e limitato, $\Delta = \{x_i\}_{i=1}^{q}$ un insieme di punti (detti *break-points*) tali che:

$$a \equiv x_0 < x_1 < x_2 < \cdots < x_q < x_{q+1} \equiv b$$

Consideriamo la partizione di $[a, b]$ in $q+1$ sottointervalli indotta dall'insieme Δ, tale che:

$$I_i = [x_i, x_{i+1}) \quad i = 0, \ldots, q-1$$
$$I_q = [x_q, x_{q+1}]$$

Sia $\mathcal{N} := (n_0, \ldots, n_q)$ un vettore di interi positivi, dove n_i è il grado dello spazio polinomiale $\mathbb{P}_{n_i}$ definito sull'intervallo I_i. Definiamo lo spazio MD (Multi-Degree) dei polinomi a tratti come:

$$P\mathbb{P}_{MD}(\Delta) = \{f \mid \exists p_i \in \mathbb{P}_{n_i} \text{ t.c. } f(x) = p_i(x),\ \forall x \in I_i,\ i = 0, \ldots, q\}$$

I polinomi a tratti hanno il vantaggio di essere più flessibili rispetto ai polinomi definiti sull'intero intervallo $[a, b]$. La mancanza di condizioni imposte nei break-points $x_i \in \Delta$ può, però, portare alla perdita di regolarità in tali punti. Allo scopo di mantenere la flessibilità dei polinomi a tratti e allo stesso tempo recuperare alcuni gradi di regolarità devono essere introdotte alcune condizioni aggiuntive che portano alla definizione delle funzioni *spline*.

3.2 Spazi MD-spline polinomiali

Una funzione *Multi-Degree spline* (abbreviata in *MD-spline*) è una funzione costituita da un insieme di polinomi di gradi differenti raccordati tra loro con una certa continuità. Se tali polinomi hanno tutti lo stesso grado allora la funzione è detta *spline*.

Definizione 3.2 (**Insieme delle spline Multi-Degree**)**.** Siano $[a, b]$, $\Delta = \{x_i\}_{i=1}^{q}$, $\{I_i\}_{i=0}^{q}$ e $\mathcal{N} := (n_0, \ldots, n_q)$ come nella definizione 3.1. Sia, poi, $\mathcal{K} := (k_1, \ldots, k_q)$ un vettore di interi positivi, detto vettore delle continuità, tale che:

$$0 \leq k_i \leq \begin{cases} \min(n_{i-1}, n_i) & \text{se} \quad n_{i-1} \neq n_i \\ n_i - 1 & \text{se} \quad n_{i-1} = n_i \end{cases} \qquad \text{per ogni } i = 1, \ldots, q \tag{3.1}$$

Si definisce l'insieme delle spline multi-degree con break-points $x_1, \ldots, x_q$ ed associati ordini di continuità $k_1, \ldots, k_q$ come:

$$\begin{aligned} S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta) := \{f \,|\, &\text{esiste } p_i \in \mathbb{P}_{n_i},\ i = 0, \ldots, q, \text{tale che:} \\ &\text{i) } f(x) = p_i(x) \text{ per } x \in I_i, i = 0, \ldots, q; \\ &\text{ii) } p_{i-1}^{(\ell)}(x_i) = p_i^{(\ell)}(x_i) \text{ per } \ell = 0, \ldots, k_i, \\ &\quad i = 1, \ldots, q\}. \end{aligned} \tag{3.2}$$

dove $p_i^{(\ell)}(x_i)$ indica la ℓ-esima derivata di p_i valutata in x_i.

Osservazione 3.1. Lo spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ delle funzioni MD-spline si riduce allo spazio delle funzioni spline ponendo $n_i = n$ per ogni $i = 0, \ldots, q$ e $0 \leq k_i \leq n - 1$ per ogni $i = 1, \ldots, q$.

Le funzioni MD-spline correggono la non regolarità dei polinomi a tratti mediante l'aggiunta di particolari condizioni sulla continuità delle funzioni nei punti di raccordo x_i. Tale continuità può essere regolata in maniera diversa in ogni punto di raccordo. In particolare, mentre nelle spline, dove i gradi dei tratti sono tutti uguali, la continuità in ogni punto è compresa tra C^0 e C^{n-1}, nelle MD-spline la massima continuità nel punto x_i è data

da C^{n_i-1} oppure $C^{min(n_{i-1},n_i)}$, a seconda che i gradi dei tratti $(i-1)$-esimo e i-esimo siano uguali o diversi.

Teorema 3.1. *L'insieme delle spline multi-degree $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ è uno spazio di funzioni di dimensione $n_0 + 1 + K_t$, dove:*

$$K_t = \sum_{i=1}^{q} (n_i - k_i)$$

oppure, equivalentemente, di dimensione $n_q + 1 + K_s$ con:

$$K_s = \sum_{i=1}^{q} (n_{i-1} - k_i)$$

Da qui in avanti, per semplicità, denotiamo la dimensione dello spazio S con K, ovvero:

$$K \equiv n_0 + K_t + 1 \equiv n_q + K_s + 1$$

Ogni elemento $f(x)$ dello spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$, cioè ogni funzione MD-spline, può pertanto essere rappresentata come:

$$f(x) = \sum_{i=1}^{K} c_i \, \varphi_i(x), \quad x \in [a, b]$$

dove $\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_K(x)\}$ è una base dello spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$.

Lo spazio delle funzioni MD-spline ha un'opportuna base B-Spline (*basis spline*), stabile dal punto di vista computazionale, descritta nel paragrafo seguente.

3.3 Funzioni B-spline Multi-Degree

Per definire le funzioni base B-spline $\{N_i\}_{i=1}^{K}$ dello spazio spline Multi-Degree è necessario considerare due diverse partizioni estese, $\Delta_t^* = \{t_j\}_{i=1}^{K}$ e $\Delta_s^* = \{s_j\}_{i=1}^{K}$, tali che ogni breakpoint x_i, sia ripetuto precisamente $(n_i - k_i)$ volte in Δ_t^* e $(n_{i-1} - k_i)$ volte in Δ_s^*.

Definizione 3.3 (**Partizioni estese**). L'insieme $\Delta_t^* := \{t_j\}_{j=1}^K$, con K definito come sopra, è chiamato *partizione estesa sinistra* associata a $S(\mathcal{P}_\mathcal{N}, \mathcal{K}, \Delta)$ se e solo se:

(i) $t_1 \leq t_2 \leq \cdots \leq t_K$

(ii) $t_{n_0+1} \equiv a$

(iii) $\{t_{n_0+2}, \ldots, t_{n_0+K_t+1}\} \equiv \{\underbrace{x_1, \ldots, x_1}_{n_1-k_1 \text{ volte}}, \ldots, \underbrace{x_q, \ldots, x_q}_{n_q-k_q \text{ volte}}\}$

Analogamente, l'insieme $\Delta_s^* := \{s_j\}_{j=1}^K$ è chiamato *partizione estesa destra* associata a $S(\mathbb{P}_\mathcal{N}, \mathcal{K}, \Delta)$ se e solo se:

(i) $s_1 \leq s_2 \leq \cdots \leq s_K$

(ii) $s_{K_s+1} \equiv b$

(iii) $\{s_1, \ldots, s_{K_s}\} \equiv \{\underbrace{x_1, \ldots, x_1}_{n_0-k_1 \text{ volte}}, \ldots, \underbrace{x_q, \ldots, x_q}_{n_{q-1}-k_q \text{ volte}}\}$

Esempio 3.1. Se $\Delta = \{1, 2\}$ partizione in $[0, 3]$, $\mathcal{N} = (3, 2, 4)$, $\mathcal{K} = (2, 1)$, allora $K_t = 3$, $K = n_0 + K_t + 1 = 3 + 3 + 1 = 7$ e le partizioni estese sono:

$$\Delta_t^* = \{0, 0, 0, 0, 2, 2, 2\}$$

$$\Delta_s^* = \{1, 2, 3, 3, 3, 3, 3\}$$

Le partizioni estese sono dette vettori nodali (*knot vectors*) ed i rispettivi elementi $\{t_j\}_{i=1}^K$ e $\{s_j\}_{i=1}^K$ sono detti nodi (*knots*).

Come mostrato nel precedente esempio 3.1, per semplicità possiamo considerare le partizioni estese con i due estremi $x_0 \equiv a$ e $x_{q+1} \equiv b$ ripetuti, rispettivamente, n_0+1 volte in Δ_t^* e n_q+1 volte in Δ_s^*, ovvero $k_0 = k_{q+1} = -1$, e quindi $t_1 = \ldots = t_{n_0+1} = x_0$ e $s_{K_s+1} = \ldots = s_{n_q+K_s+1} = x_{q+1}$.

Nel prosieguo della trattazione si userà la notazione p^{t_i} per fare riferimento all'indice del break-point associato al nodo t_i e p^{s_i} per fare riferimento all'indice del break-point associato al nodo s_i.

Osservazione 3.2. Le partizioni estese per il caso Multi-Degree sono costruite in modo tale che alcuni break-point presenti in Δ_t^* possono non essere presenti in Δ_s^* e viceversa. Se però $n_i = n$ per ogni $i = 0, \ldots, q$ allora si avrà che:

$$(t_{n+2}, t_{n+3}, \ldots, t_{n+K_t+1}) \equiv (s_1, s_2, \ldots, s_{K_s})$$

cioè i break-point sono presenti con la stessa molteplicità in entrambe le partizioni e, di conseguenza, si può utilizzare una loro fusione. In particolare, se consideriamo Δ_t^* ed aggiungiamo alla fine gli ultimi $n_q + 1$ elementi di Δ_s^* $(s_{K_s+1} \equiv b, \ldots, s_K)$ allora abbiamo ottenuto la classica partizione estesa associata ad uno spazio spline con stesso grado.

A questo punto possiamo definire le funzioni base B-spline per uno spazio spline Multi-Degree.

Definizione 3.4 (**Funzioni B-spline Multi-Degree**)**.** Consideriamo le partizioni estese Δ_t^* e Δ_s^* associate allo spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ e sia $m = \max_i\{n_i\}$ il massimo tra i gradi in $\mathcal{N}$. Generiamo una sequenza di funzioni $\{N_{i,n}(x)\}$ per $n = 0, \ldots, m$ e $i = m + 1 - n, \ldots, K$ attraverso la seguente relazione di ricorrenza integrale.

Ogni funzione $N_{i,n}$, dove l'indice n è il numero di passo di ricorsione, è definita su ogni intervallo $[x_j, x_{j+1}] \subset [t_i, s_{i-m+n}]$ nel modo seguente:

$$N_{i,n}(x) = \begin{cases} 1 \quad x_j \le x < x_{j+1} & n_j = m - n \\ \int_{-\infty}^{x} \left[\delta_{i,n-1} N_{i,n-1}(u) - \delta_{i+1,n-1} N_{i+1,n-1}(u)\right] du & n_j > m - n \\ 0 & \text{altrimenti} \end{cases}$$

dove $\delta_{i,n} = \left(\int_{-\infty}^{+\infty} N_{i,n}(x)\, dx\right)^{-1}$ e, quando $N_{i,n-1}(x) = 0$, si impone:

$$\int_{-\infty}^{x} \delta_{i,n-1} N_{i,n-1}(u)\, du = \begin{cases} 0 & x < t_i \\ 1 & x \ge t_i \end{cases}$$

La sequenza finale $\{N_{i,m}(x)\}$, $i = 1, \ldots, K$, ottenuta all'ultimo passo ricorsivo m, è la base B-spline Multi-Degree $\{N_i\}_{i=1}^K$ di $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$.

Osservazione 3.3. Per costruzione, l'intervallo nodale di definizione di ogni $N_{i,n}(x)$ è $[t_i, s_{i-m+n}]$. Pertanto le funzioni base B-Spline Multi-Degree $(\{N_{i,m}(x)\}_{i=1}^{K})$ sono definite in $[t_i, s_i]$.

Osservazione 3.4. Il secondo indice m associato alle funzioni base $N_{i,m}$ è usato solo ai fini della suddetta definizione ricorsiva per indicare che queste vengono ottenute al passo m ma, esternamente a tale contesto, le funzioni base Multi-Degree sono indicate semplicemente con N_i. Se lo spazio Multi-Degree fosse ridotto ad uno spazio a grado singolo n allora si avrebbe la notazione classica in cui le funzioni base B-spline vengono indicate con $N_{i,n}$.

In figura 3.1 vengono mostrate le funzioni base B-spline multi-degree N_i associate allo spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ dell'esempio 3.1.

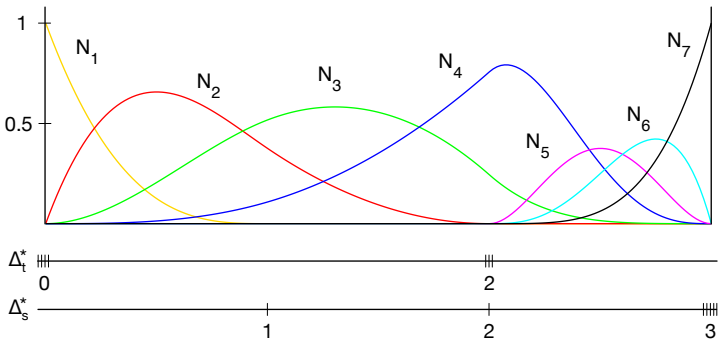


Figura 3.1: Funzioni base B-spline Multi-Degree relative allo spazio dell'esempio 3.1

3.3.1 Proprietà delle B-spline Multi-Degree

Le funzioni B-spline Multi-Degree $\{N_i(x)\}_{i=1}^{K}$ sono una base per lo spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ e godono delle seguenti proprietà:

P.3.1 (*Supporto Locale*)

$$N_i(x) = 0 \quad \forall\, x \notin [t_i, s_i]$$

L'intervallo $[t_i, s_i]$ è detto *intervallo di supporto* della N_i.

P.3.2 (*Non Negatività*)

$$N_i(x) > 0 \quad \forall\, x \in (t_i, s_i)$$

P.3.3 (*Partizione dell'unità*)

$$\sum_i N_i(x) = 1, \quad \forall\, x \in [a,b]$$

P.3.4 (*Continuità agli estremi*) N_i diventa zero esattamente $k_i^t + 1$ volte in t_i e esattamente $k_i^s + 1$ volte in s_i dove:

$$k_i^t := n_{p^{t_i}} - \max\{j \geq 0 \,|\, t_i = t_{i+j}\} - 1$$
$$k_i^s := n_{p^{s_i-1}} - \max\{j \geq 0 \,|\, s_{i-j} = s_i\} - 1$$

P.3.5 In un dato intervallo $[t_\ell, t_{\ell+1})$, con $x \in [x_j, x_{j+1}]$ e $t_\ell \leq x_j < t_{\ell+1}$, al massimo $n_j + 1$ delle N_i sono non nulle.

In particolare queste saranno le $N_{\ell-n_j}, \ldots, N_\ell$.

P.3.6 Partizioni nodali estese della forma:

$$\Delta_t^* = \{\underbrace{0, \ldots, 0}_{m}\}$$
$$\Delta_s^* = \{\underbrace{1, \ldots, 1}_{m}\}$$

generano i polinomi di Bernstein di grado $n = m - 1$. Ovvero, le funzioni B-spline sono una generalizzazione dei polinomi di Bernstein.

Osservazione 3.5. Dalla proprietà P.3.1, ogni nodo di Δ_t^* e Δ_s^* è, rispettivamente, l'estremo iniziale e finale di una funzione base B-spline.

Osservazione 3.6. La proprietà P.3.4 dice che:

$$N_i(t_i) = D^1 N_i(t_i) = \ldots = D^{k_i^t} N_i(t_i) = 0, \quad D^{k_i^t+1} N_i(t_i) > 0$$
$$N_i(s_i) = D^1 N_i(s_i) = \ldots = D^{k_i^s} N_i(s_i) = 0, \quad D^{k_i^s+1} N_i(s_i) > 0$$

cioè che N_i è $C^{k_i^t}$ continua in t_i e $C^{k_i^s}$ continua in s_i. Le molteplicità dei break-point associati a t_i e ad s_i, rispetto ad N_i, saranno quindi date da:

$$\mu_i^t := n_{p^{t_i}} - k_i^t \quad \text{e} \quad \mu_i^s := n_{p^{s_i-1}} - k_i^s$$

dove $\mu_i^t = \max\{j \geq 0 \,|\, t_i = t_{i+j}\} + 1$ e $\mu_i^s = \max\{j \geq 0 \,|\, s_{i-j} = s_i\} + 1$.

3.4 Funzioni MD-Spline

In questo paragrafo vengono definite le funzioni MD-spline, che costituiscono le funzioni coordinate delle curve MD-spline.

Definizione 3.5 (**Funzioni MD-Spline**)**.** Chiamiamo MD-Spline (*Multi-Degree Spline*) le funzioni dello spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ rappresentate nella base delle B-spline Multi-Degree, cioè:

$$f(x) = \sum_{i=1}^{K} c_i \, N_i(x), \quad x \in [a, b] \tag{3.3}$$

dove $c_i \in \mathbb{R}$ sono coefficienti scalari.

Se i gradi dei tratti polinomiali sono tutti uguali allora la funzione è detta semplicemente NUBS (*Non Uniform B-spline*).

Osservazione 3.7. Come si può vedere, ad ogni coefficiente c_i è associata una ben precisa funzione base N_i.

Le proprietà delle funzioni MD-spline sono una diretta conseguenza delle proprietà delle B-spline Multi-Degree. In particolare:

P.3.7 Per la proprietà P.3.5, risulta che:

$$f(x) = \sum_{i=\ell-n_j}^{\ell} c_i \, N_i(x), \quad x \in [x_j, x_{j+1}], \quad t_\ell \leq x_j < t_{\ell+1} \tag{3.4}$$

P.3.8 Per le proprietà P.3.2 e P.3.3, una funzione MD-spline $f(x)$ è una combinazione convessa di coefficienti c_i, per cui il valore assunto dalla funzione MD-spline in un punto $x \in [a, b]$ è sempre compreso tra il valore minimo e quello massimo dei c_i, ovvero:

$$min_{i=1,\dots,K}(c_i) \leq f(x) \leq max_{i=1,\dots,K}(c_i) \quad x \in [a, b]$$

che, per la proprietà P.3.5, è equivalente a:

$$min_{i=\ell-n_j,\dots,\ell}(c_i) \leq f(x) \leq max_{i=\ell-n_j,\dots,\ell}(c_i) \quad x \in [x_j, x_{j+1}]$$

Nota che la proprietà P.3.7 mette in evidenzia il comportamento di tipo locale delle funzioni spline: cambiando un qualsiasi coefficiente c_i la forma della MD-spline cambia solo in corrispondenza degli $n_j + 1$ intervalli nodali interessati.

3.5 Valutazione delle funzioni MD-Spline

Il problema della valutazione consiste nel determinare il valore che una determinata funzione MD-spline assume in corrispondenza di un certo punto $\overline{x} \in [a, b]$. Il processo si può descrivere in tre passi:

1. ricerca dell'intervallo $[x_j, x_{j+1}]$ contenente $\overline{x}$;
2. ricerca dell'intervallo $[t_\ell, t_{\ell+1})$ contenente x_j;
3. valutazione in $\overline{x}$ del polinomio di grado n_j che definisce $f(x)$ in $[x_j, x_{j+1}]$ mediante l'espressione:

$$f(\overline{x}) = \sum_{i=\ell-n_j}^{\ell} c_i \, N_i(\overline{x}) \tag{3.5}$$

Per valutare quest'ultima espressione, non è possibile servirsi del noto algoritmo di *de Boor* che viene usato per la valutazione delle NUBS. Si potrebbe, allora, considerare di calcolare le $N_i(x)$ attraverso la formulazione ricorrente integrale della definizione 3.4 e poi valutarne la combinazione lineare con i c_i, ma tale metodo è poco efficiente.

La soluzione consiste nell'aggiungere un passo di calcolo intermedio, in cui si ottengono le cosiddette *funzioni di transizione* dello spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$, mediante le quali è possibile calcolare le funzioni base Multi-Degree $N_i(x)$.

3.5.1 Funzioni di transizione

Le funzioni di transizione sono uno strumento efficiente per calcolare le funzioni base B-spline Multi-Degree $\{N_i\}_{i=1}^{K}$ e sono definite come segue.

Definizione 3.6 (**Funzioni di transizione**)**.** Sia $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ uno spazio spline Multi-Degree di dimensione K con $\Delta = \{x_i\}_{i=1}^{q}$ una partizione di $[a, b]$ e $\Delta_t^* = \{t_i\}_{i=1}^{K}$ e $\Delta_s^* = \{s_i\}_{i=1}^{K}$ le associate partizioni estese sinistra e destra. Si supponga inoltre che $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ abbia una base B-Spline $\{N_i\}_{i=1}^{K}$. Si dicono funzioni di transizione, le funzioni f_i a tratti, date da:

$$f_i = \sum_{j=i}^{K} N_j, \quad i = 1, \ldots, K \tag{3.6}$$

Assumendo $f_{K+1} \equiv 0$ si può invertire la relazione (3.6) esprimendo le funzioni base B-spline attraverso le funzioni di transizione come:

$$N_i = f_i - f_{i+1}, \quad i = 1, \ldots, K \tag{3.7}$$

Dalla proprietà P.3.3 delle funzioni base B-spline si può dedurre che $f_1(x) = 1$, per ogni $x \in [a, b]$ e che le funzioni f_i, $i = 2, \ldots, K$ hanno le seguenti proprietà:

(a)
$$f_i(x) = \begin{cases} 0, & x \leq t_i \\ 1, & x \geq s_{i-1} \end{cases} \tag{3.8}$$

(b) f_i diventa zero esattamente $k_i^t + 1$ volte in t_i e $1 - f_i$ diventa zero esattamente $k_{i-1}^s + 1$ volte in s_{i-1}, con K_i^t e k_{i-1}^s dati in P.3.4.

Le suddette proprietà (a) e (b) forniscono un pratico modo per calcolare le funzioni di transizione per lo spazio spline $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$. In particolare, da (a) segue che ogni funzione di transizione f_i, $i = 2, \ldots, K$ ha un'espressione non banale (cioè non è la funzione costante zero o uno) in $[t_i, s_{i-1}]$. La restrizione di f_i a tale intervallo può essere determinata usando la proprietà (b).

Siano $x_{p^{t_i}}, \ldots, x_{p^{s_{i-1}}}$ i break-point di Δ contenuti in $[t_i, s_{i-1}]$. Allora la funzione f_i è formata da $p^{s_{i-1}} - p^{t_i} - 1$ tratti non banali e soddisfa le seguenti condizioni di continuità:

$$\begin{aligned} &D^r f_{i,p^{t_i}}(x_{p^{t_i}}) = 0, && r = 0, \ldots, k_i^t, \\ &D^r f_{i,j-1}(x_j) = D^r f_{i,j}(x_j), && r = 0, \ldots, k_j, \quad j = p^{t_i} + 1, \ldots, p^{s_{i-1}} - 1, \\ &D^r f_{i,p^{s_{i-1}}-1}(x_{p^{s_{i-1}}}) = \delta_{r,0}, && r = 0, \ldots, k_{i-1}^s \end{aligned} \tag{3.9}$$

dove $f_{i,j}$, $j = p^{t_i}, \ldots, p^{s_{i-1}} - 1$, è la restrizione di f_i all'intervallo $[x_j, x_{j+1}]$, k_i^t e k_{i-1}^s sono dati in P.3.4 e $\delta_{r,0}$ è così definito;

$$\delta_{r,0} = \begin{cases} 1 & \text{se} \quad r = 0 \\ 0 & \text{se} \quad r \neq 0 \end{cases} \tag{3.10}$$

Quindi è possibile determinare f_i come una funzione dello spazio $S_{[t_i,s_{i-1}]}(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$, restrizione di $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ a $[t_i, s_{i-1}]$. Per appartenere a tale spazio le funzioni di transizione devono soddisfare le condizioni di continuità nei break-point x_j che sono contenuti in $[t_i, s_{i-1}]$.

In pratica, per calcolare la funzione di transizione i-esima f_i si procede nel modo seguente. Si supponga che, in $[x_j, x_{j+1}]$, $j = p^{t_i}, \ldots, p^{s_{i-1}} - 1$, lo spazio spline sia localmente generato dalla base di Bernstein $\{B_{h,n_j}\}_{h=0,\ldots,n_j}$ e siano $b_{i,\ell}$, $\ell = 1, \ldots, H_i$ con $H_i = \sum_{j=p^{t_i}}^{p^{s_{i-1}}-1}(n_j + 1)$, i coefficienti delle espansioni locali di $f_{i,j}$ in tali basi. Sia cioè:

$$\begin{aligned} & & & x \in [x_j, x_{j+1}] \\ f_{i,j}(x) = \sum_{h=0}^{n_j} b_{i,h_{i,j}+h} B_{h,n_j}(x), & & & j = p^{t_i}, \ldots, p^{s_{i-1}} - 1 \\ & & & h_{i,j} = \sum_{\ell=p^{t_i}}^{j-1} (n_\ell + 1) \end{aligned} \tag{3.11}$$

Quindi, in base alla (3.9), i primi $k_i^t + 1$ coefficienti $b_{i,h_{i,j}+h}$ saranno uguali a 0 mentre gli ultimi $k_{i-1}^s + 1$ saranno uguali a 1. I rimanenti coefficienti in (3.11) possono essere determinati risolvendo un sistema lineare creato imponendo unicamente le condizioni di continuità di (3.9). Si noti che, nel caso di funzioni di transizione definite su un unico intervallo (cioè $[t_i, s_{i-1}]$ composto dall'unico tratto $[x_{p^{t_i}}, x_{p^{s_{i-1}}}]$), allora non è necessario risolvere alcun sistema lineare.

In figura 3.2 sono mostrate le funzioni di transizione associate allo spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ dell'esempio 3.1.

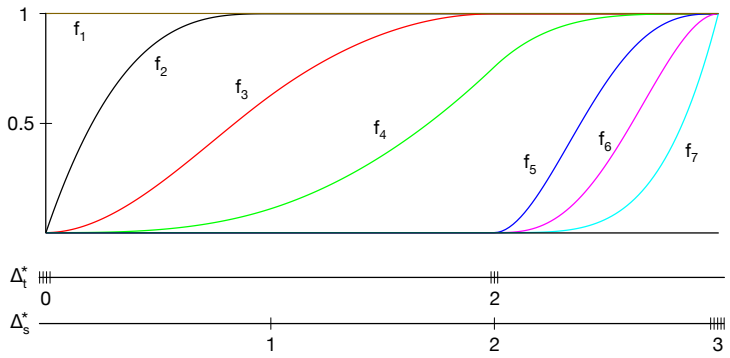


Figura 3.2: Funzioni di transizione relative allo spazio dell'esempio 3.1

Riprendendo la (3.4) e la (3.11), per $x \in [x_j, x_{j+1}]$, $t_\ell \leq x_j < t_{\ell+1}$, sarà:

$$\begin{aligned} f(x) = \sum_{i=\ell-n_j}^{\ell} c_i\, N_i(x) = \sum_{i=\ell-n_j}^{\ell} c_i (f_{i,j}(x) - f_{i+1,j}(x)) = \\ \sum_{i=\ell-n_j}^{\ell} c_i \sum_{h=0}^{n_j} (b_{i,h_{i,j}+h} - b_{i+1,h_{i+1,j}+h}) B_{h,n_j}(x) \end{aligned} \tag{3.12}$$

Osservazione 3.8. Si noti che, per costruzione, si possiede la relazione fra la base B-spline Multi-Degree e le basi di Bernstein locali di gradi differenti.

Derivata ed integrale di una funzione MD-spline $f(x)$ possono, allora, essere calcolati in modo semplice, ricordando che:

$$DB_{h,n_j}(x) = \frac{n_j}{x_{j+1} - x_j} \left(B_{h-1,n_j-1}(x) - B_{h,n_j-1}(x)\right)$$

$$\int_{x_j}^{x_{j+1}} B_{h,n_j}(x)\, dx = \frac{x_{j+1} - x_j}{n_j + 1}$$

3.5.2 Esempio numerico di valutazione

Riprendiamo l'esempio 3.1. Sia, cioè, $\Delta = \{1, 2\}$ partizione in $[0, 3]$, $\mathcal{N} = (3, 2, 4)$, $\mathcal{K} = (2, 1)$ e $K = 7$. Le partizioni estese associate sono $\Delta_t^* = \{0, 0, 0, 0, 2, 2, 2\}$ e $\Delta_s^* = \{1, 2, 3, 3, 3, 3, 3\}$.

Una volta definito lo spazio multi-degree $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ si devono calcolare i coefficienti delle relative funzioni di transizione $\{f_i(x)\}_{i=2}^{K}$ (la $f_1(x)$ non viene considerata perché vale 1 su ogni tratto, ovvero è banale ovunque).

Nel nostro spazio di dimensione 7 andranno pertanto calcolati i coefficienti delle $\{f_i(x)\}_{i=2}^{7}$ ($f_1(x) = 1$, $f_8(x) = 0$). Di seguito, a titolo esemplificativo, viene mostrata la procedura per ottenere quelli delle funzioni f_3 ed f_4.

Come detto in 3.5.1, i coefficienti si ottengono risolvendo opportuni sistemi lineari su cui sono imposte le condizioni di continuità di (3.9). In particolare, per ricavare tutti i coefficienti della f_3, si procede nel modo seguente. Innanzitutto va imposta la prima condizione, cioè:

$$D^r f_{3,p^{t_3}}(x_{p^{t_3}}) = 0, \quad r = 0, \ldots, k_3^t$$

dove $t_3 = 0$ e quindi $x_{p^{t_3}} = x_0$ e $k_3^t = 1$. Ovvero deve valere $f_{3,0}(0) = 0$ e $D' f_{3,0}(0) = 0$. Dalla (3.11), $f_{3,0}(x)$ può essere espresso come:

$$f_{3,0}(x) = \sum_{h=0}^{3} b_{3,h_{3,0}+h} B_{h,3}(x), \quad \text{con } h_{3,0} = 0 \tag{3.13}$$

Siccome:

$$\begin{array}{llll} B_{0,3}(0) = 1 & B_{1,3}(0) = 0, & B_{2,3}(0) = 0, & B_{3,3}(0) = 0 \\ D'B_{0,3}(0) = -3, & D'B_{1,3}(0) = 3, & D'B_{2,3}(0) = 0, & D'B_{3,3}(0) = 0 \end{array}$$

allora questa condizione si impone con:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \end{pmatrix} \begin{pmatrix} b_{3,0} \\ b_{3,1} \\ b_{3,2} \\ b_{3,3} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

dove i $\{b_{3,i}\}_{i=0}^{3}$ sono i primi 4 coefficienti associati alla f_3 (cioè i 4 coefficienti della $f_{3,0}$, restrizione di f_3 all'intervallo $[0, 1]$).

La seconda condizione da imporre è:

$$D^r f_{3,j-1}(x_j) = D^r f_{3,j}(x_j), \quad r = 0, \ldots, k_j, \quad j = p^{t_3} + 1, \ldots, p^{s_2} - 1$$

ovvero $f_{3,0}(1) = f_{3,1}(1)$, $D'f_{3,0}(1) = D'f_{3,1}(1)$ e $D^2 f_{3,0}(1) = D^2 f_{3,1}(1)$.
$f_{3,0}(x)$ è definito in (3.13) ed $f_{3,1}(x)$ è dato da:

$$f_{3,1}(x) = \sum_{h=0}^{2} b_{3,h_{3,1}+h} B_{h,2}(x), \quad \text{con } h_{3,1} = 3 \tag{3.14}$$

Siccome:

$$\begin{array}{llll}
B_{0,3}(1) = 0 & B_{1,3}(1) = 0, & B_{2,3}(1) = 0, & B_{3,3}(1) = 1 \\
D'B_{0,3}(1) = 0, & D'B_{1,3}(1) = 0, & D'B_{2,3}(1) = -3, & D'B_{3,3}(1) = 3 \\
D^2B_{0,3}(1) = 0, & D^2B_{1,3}(1) = 6, & D^2B_{2,3}(1) = -12, & D^2B_{3,3}(1) = 6 \\
B_{0,2}(1) = 1 & B_{1,2}(1) = 0, & B_{2,2}(1) = 0 & \\
D'B_{0,2}(1) = -2, & D'B_{1,2}(1) = 2, & D'B_{2,2}(1) = 0 & \\
D^2B_{0,2}(1) = 2, & D^2B_{1,2}(1) = -4, & D^2B_{2,2}(1) = 2 &
\end{array}$$

allora questa condizione si impone con:

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -3 & 3 \\ 0 & 6 & -12 & 6 \end{pmatrix} \begin{pmatrix} b_{3,0} \\ b_{3,1} \\ b_{3,2} \\ b_{3,3} \end{pmatrix} - 1 \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 2 & -4 & 2 & 0 \end{pmatrix} \begin{pmatrix} b_{3,4} \\ b_{3,5} \\ b_{3,6} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

dove i $\{b_{3,i}\}_{i=0}^{3}$ sono i coefficienti della $f_{3,0}$, restrizione di f_3 all'intervallo $[0, 1]$ mentre i $\{b_{3,i}\}_{i=4}^{6}$ sono i coefficienti di $f_{3,1}$, restrizione di f_3 all'intervallo $[1, 2]$.

L'ultima condizione da imporre è:

$$D^r f_{3,p^{s_2}-1}(x_{p^{s_2}}) = \delta_{r,0}, \quad r = 0, \ldots, k_2^s$$

dove $s_2 = 2$ e quindi $k_2^s = 1$. Ovvero $f_{3,1}(2) = 1$ e $D'f_{3,1}(2) = 0$.
Dalla (3.14), siccome:

$$\begin{array}{lll}
B_{0,2}(2) = 0 & B_{1,2}(2) = 0, & B_{2,2}(2) = 1 \\
D'B_{0,2}(2) = 0, & D'B_{1,2}(2) = -2, & D'B_{2,2}(2) = 2
\end{array}$$

allora questa condizione si impone con:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & -2 & 2 \end{pmatrix} \begin{pmatrix} b_{3,4} \\ b_{3,5} \\ b_{3,6} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Infine, mettendo tutte le precedenti condizione insieme si ha il sistema seguente:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -3 & 3 & 2 & -2 & 0 \\ 0 & 6 & -12 & 6 & -2 & 4 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -2 & 2 \end{pmatrix} \begin{pmatrix} b_{3,0} \\ b_{3,1} \\ b_{3,2} \\ b_{3,3} \\ b_{3,4} \\ b_{3,5} \\ b_{3,6} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

da cui vengono ricavati: $b_{3,0} = 0$, $b_{3,1} = 0$, $b_{3,2} = 0.375$, $b_{3,3} = 0.625$, $b_{3,4} = 0.625$, $b_{3,5} = 1$, $b_{3,6} = 1$.

Ragionamento analogo vale per la f_4, i cui coefficienti vengono ottenuti risolvendo il seguente sistema:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & -12 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 3 & 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & -12 & 6 & -2 & 4 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 2 & 4 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & -24 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -24 & 72 & -72 & 24 \end{pmatrix} \begin{pmatrix} b_{4,0} \\ b_{4,1} \\ b_{4,2} \\ b_{4,3} \\ b_{4,4} \\ b_{4,5} \\ b_{4,6} \\ b_{4,7} \\ b_{4,8} \\ b_{4,9} \\ b_{4,10} \\ b_{4,11} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Si ottiene: $b_{4,0} = 0$, $b_{4,1} = 0$, $b_{4,2} = 0$, $b_{4,3} = 0.108$, $b_{4,4} = 0.108$, $b_{4,5} = 0.270$, $b_{4,6} = 0.757$, $b_{4,7} = 0.757$, $b_{4,8} = 1$, $b_{4,9} = 1$, $b_{4,10} = 1$, $b_{4,11} = 1$.

Osservazione 3.9. La f_4 ha 5 coefficienti in più della f_3 dal momento che è non banale (cioè la funzione non è né 1 né 0) in un intervallo aggiuntivo, $[x_2, x_3]$ (cioè $[2, 3]$), che ha proprio ordine 5 (grado 4).

Supponiamo ora che siano dati i coefficienti $\{c_i\}_{i=1}^{K}$ e si voglia valutare la funzione MD-spline $f(x)$ in $\overline{x} = 1.4$. I passi dell'algoritmo di valutazione sono i seguenti:

1. si determina j tale che $\overline{x} \in [x_j, x_{j+1}]$. Nel nostro caso j vale 1.

2. si determina ℓ tale che $t_\ell \leq x_j < t_{\ell+1}$. Nel nostro caso ℓ vale 4.

Per la proprietà P.3.7, per valutare la funzione $f(x)$ nel caso in esame dobbiamo calcolare le $\{N_i(\overline{x})\}_{i=\ell-n_j}^{\ell}$, con $n_j = n_1 = 2$ ed $\ell = 4$. Dalla (3.12), ognuna di queste N_i è data dalla differenza $f_{i,1} - f_{i+1,1}$, per cui, per calcolarle, ci serve valutare in $\overline{x}$ le $\{f_{i,1}\}_{i=2}^{5}$, ovvero le $\{f_i\}_{i=2}^{5}$ ristrette all'intervallo $[x_1, x_2]$. Come si vede dalla figura 3.1, $f_{2,1}(x) = 1$, $f_{5,1}(x) = 0$ e $\{f_{i,1}(x)\}_{i=3}^{4}$ sono le rimanenti funzioni non nulle nell'intervallo $[1, 2]$ considerato. Ora, siccome in $[1, 2]$ lo spazio spline è localmente generato dalla base di Bernstein $\{B_{h,2}\}_{h=0}^{2}$, dalla (3.11) con $j = 1$, si ha:

$$\begin{aligned} f_{3,1}(\overline{x}) &= \sum_{h=0}^{2} b_{3,h_{3,1}+h} B_{h,2}(\overline{x}), \qquad \text{con } h_{3,1} = 4 \\ f_{4,1}(\overline{x}) &= \sum_{h=0}^{2} b_{4,h_{4,1}+h} B_{h,2}(\overline{x}), \qquad \text{con } h_{4,1} = 4 \end{aligned} \tag{3.15}$$

dove $h_{3,1}$ e $h_{4,1}$ sono, rispettivamente, la somme degli ordini dei tratti di f_3 e di quelli di f_4 precedenti al tratto $[x_1, x_2]$. Questi valgono entrambi 4 perché, come si può vedere in figura 3.1, le due funzioni di transizione decollano entrambe da $x_0 = 0$ e per cui l'unico tratto polinomiale precedente a $[1, 2]$ è $[0, 1]$ che è proprio di ordine 4 (grado 3). I passi successivi dell'algoritmo pertanto riguardano la valutazione delle suddette funzioni di transizione.

3. si calcolano le $\{f_{i,1}(\overline{x})\}_{i=3}^{4}$ con la (3.15): i polinomi base di Bernstein $\{B_{h,2}(x)\}_{h=0}^{2}$ vengono valutati in $\overline{x}$ tramite la formula (2.2) ed i coefficienti $\{b_{i,h_{i,1}+h}\}_{i=3}^{4}$ sono quelli che sono stati precalcolati all'inizio (che quindi non devono essere ricalcolati per ogni valutazione in un punto);

4. ora che abbiamo tutti i valori delle $\{f_{i,1}(\overline{x})\}_{i=2}^{5}$, si può applicare la (3.12):

$$f(\overline{x}) = \sum_{i=2}^{4} c_i \, N_i(\overline{x}) = \sum_{i=2}^{4} c_i (f_{i,1}(\overline{x}) - f_{i+1,1}(\overline{x}))$$

3.6 Algoritmi geometrici per MD-Spline

In questo paragrafo vengono definiti formalmente gli algoritmi di knot-insertion/removal e degree elevation/reduction applicati a funzioni MD-spline. Questi, estendono gli omonimi algoritmi su NUBS, ampiamente discussi in letteratura, prendendo in considerazione la possibilità di avere tratti polinomiali di gradi differenti.

3.6.1 Knot insertion

Siano Δ_t^* e Δ_s^* le partizioni nodali estese associate a $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$. Inserire un nodo $\hat{t}$ in Δ_t^* equivale all'inserzione dello stesso anche in Δ_s^*, che equivale anche, se il nodo è nuovo, all'inserimento di un breakpoint mentre, se il nodo già esiste in Δ_t^* o Δ_s^*, a richiedere una continuità inferiore in corrispondenza del breakpoint associato. In ogni caso, se $t_\ell \leq \hat{t} < t_{\ell+1}$ ed indichiamo con $S(\mathbb{P}_{\hat{\mathcal{N}}}, \hat{\mathcal{K}}, \hat{\Delta})$ lo spazio multi-degree dopo il knot-insertion, si avrà $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta) \subset S(\mathbb{P}_{\hat{\mathcal{N}}}, \hat{\mathcal{K}}, \hat{\Delta})$ e

$$dimensione(S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)) = dimensione(S(\mathbb{P}_{\hat{\mathcal{N}}}, \hat{\mathcal{K}}, \hat{\Delta})) + 1 = K + 1$$

Denotiamo con $\{N_j\}_{i=1}^{K}$ e $\{\hat{N}_j\}_{i=1}^{K+1}$ le funzioni base B-Spline MD sulle partizioni nodali estese sinistra e destra, Δ_t^*, Δ_s^* e $\hat{\Delta}_t^*$, $\hat{\Delta}_s^*$, rispettivamente.

Il termine *knot insertion* si riferisce al procedimento usato per determinare la rappresentazione di una funzione MD-spline $f(x) \in S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$, nella nuova base $\{\hat{N}_j\}_{i=1}^{K+1}$, conoscendone la rappresentazione nella vecchia $\{N_j\}_{i=1}^{K}$. È importante notare che il knot insertion è solamente una modifica della base dello spazio spline; la funzione MD-spline non viene modificata né geometricamente né parametricamente.

Teorema 3.2. *(Knot-insertion) Siano* Δ_t^*, Δ_s^* *e* $\hat{\Delta}_t^*$, $\hat{\Delta}_s^*$ *le partizioni nodali definite, allora esistono i coefficienti* $\alpha_1, \ldots, \alpha_{K+1}$ *con* $0 \leq \alpha_i \leq 1$, $i = 1, \ldots, K+1$ *tali che:*

$$N_i(x) = \alpha_i \hat{N}_i(x) + (1 - \alpha_{i+1})\hat{N}_{i+1}(x), \qquad \begin{matrix} i = 1, \ldots, K \\ x \in \mathbb{R} \end{matrix} \tag{3.16}$$

In particolare:

$$\alpha_i = \begin{cases} 1, & i \leq \ell - n_j, \quad con \quad x_j \leq \hat{t} < x_{j+1} \\ \dfrac{D_+^{k_i^t+1} f_{i,p^{t_i}}(t_i)}{D_+^{k_i^t+1} \hat{f}_{i,p^{t_i}}(t_i)}, & \ell - n_j + 1 \leq i \leq \ell - r + 1 \\ 0, & i \geq \ell - r + 2 \end{cases} \tag{3.17}$$

dove $f_{i,p^{t_i}}$ *e* $\hat{f}_{i,p^{t_i}}$ *sono i primi tratti non banali delle funzioni di transizione che definiscono rispettivamente le funzioni base B-spline multi-degree* N_i *e* $\hat{N}_i$, k_i^t *è dato in P.3.4, e* $1 \leq r < n_j$ *dove* $n_j - r$ *rappresenta l'ordine di continuità richiesto nel breakpoint associato a* $\hat{t}$ *nel nuovo spazio* $S(\mathcal{P}_{\hat{\mathcal{N}}}, \hat{\mathcal{K}}, \hat{\Delta})$.

Osservazione 3.10. Dalle espressioni (3.16) e (3.17), risulta che:

$$\begin{aligned} N_i(x) &= \hat{N}_i(x) \qquad i = 1, \ldots, \ell - n_j - 1 \\ N_i(x) &= \hat{N}_{i+1}(x) \qquad i = \ell - r - 2, \ldots, K \end{aligned}$$

Osservazione 3.11. I coefficienti α_i in (3.17) sono determinati come il rapporto delle derivate delle funzioni di transizione, quindi è richiesto che queste siano diverse da zero nel punto t_i di valutazione. Siccome in t_i, sia f_i che $\hat{f}_i$ hanno lo stesso ordine di continuità k_i^t, allora è evidente che vanno considerate le loro derivate di ordine $k_i^t + 1$.

La funzione $f(x)$ avrà, nel nuovo spazio $S(\mathbb{P}_{\hat{\mathcal{N}}}, \hat{\mathcal{K}}, \hat{\Delta})$, una rappresentazione della forma:

$$f(x) = \sum_{i=1}^{K+1} \hat{c}_i \, \hat{N}_i(x) \tag{3.18}$$

dove le $\hat{N}_i(x)$ sono le funzioni base definite su $\hat{\Delta}_t^*$ e $\hat{\Delta}_s^*$.

Corollario 3.1. *Siano Δ_t^*, Δ_s^* e $\hat{\Delta}_t^*$, $\hat{\Delta}_s^*$ le partizioni nodali già definite, allora:*

$$\hat{c}_i = \begin{cases} c_i & i \leq \ell - n_j \\ \alpha_i c_i + (1-\alpha_i) c_{i-1} & \ell - n_j + 1 \leq i \leq \ell - r + 1 \\ c_{i-1} & i \geq \ell - r + 2 \end{cases} \tag{3.19}$$

dove gli α_i sono stati definiti in (3.17).

Infine, è importante ripetere che, se il nodo da inserire è coincidente con uno dei breakpoint, ovvero $x_j \equiv \hat{t}$, non si tratterà di inserire un nuovo nodo ma di ridurre la continuità di x_j, ossia verrà richiesto un ordine di continuità $min\{n_{j-1}, n_j\} - r$ minore delle precedente continuità k_j.

3.6.2 Knot removal

Il *knot-removal* è il procedimento inverso del knot-insertion e consiste nel rappresentare una MD-spline con un numero inferiore di nodi. Tale rappresentazione può essere esatta oppure approssimata, a seconda che la rimozione del nodo lasci o meno inalterata la forma della funzione.

Siano $\hat{\Delta}_t^*$ e $\hat{\Delta}_s^*$ le partizioni nodali estese dopo il knot-insertion del nodo $\hat{x}$, Δ_t^* e Δ_s^* le partizioni nodale originarie e $\{\hat{N}_i(x)\}_{i=1}^{K+1}$ e $\{N_i(x)\}_{i=1}^{K}$ le rispettive funzioni base multi-degree. Si vuole determinare $f(x) = \sum_{i=1}^{K} c_i\, N_i(x)$ nella base originaria. Il problema consiste nel ricavare i c_i conoscendo i $\hat{c}_i$ dell'espressione 3.19 e può essere interpretato come un sistema di $K+1$ equazioni e K incognite come mostrato di seguito.

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & 0 & \\ & (1-\alpha_{\ell-n_j+1}) & \alpha_{\ell-n_j+1} & & \\ & & & (1-\alpha_{\ell-n_j+2}) & \alpha_{\ell-n_j+2} \\ & 0 & & \ddots & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ c_K \end{pmatrix} = \begin{pmatrix} \hat{c}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hat{c}_{K+1} \end{pmatrix}$$

In pratica si può procedere alla determinazione dei c_i a partire dalla prima ed ultima equazione procedendo verso le equazioni centrali; se l'equazione

centrale (o le due equazioni centrali, a seconda che la matrice abbia un numero dispari o pari di righe) è soddisfatta (a meno di una certa tolleranza) allora il sistema ammette un'unica soluzione ed i c_i trovati sono la soluzione; è quindi possibile procedere alla rimozione esatta del nodo $\hat{x}$.

Se l'equazione centrale (o le equazioni centrali) non sono soddisfatte allora la soluzione va approssimata. Infatti, dal momento che i calcoli effettuati a partire dalla prima ed ultima equazione non sono andati a convergere su un'unica soluzione per il c_i (o i c_i) centrali, questi devono essere approssimati. L'approssimazione può essere fatta eseguendo una media tra i corrispondenti differenti valori ottenuti dalle computazioni oppure, come descritto in 4.3.3, utilizzando la base duale.

3.6.3 Degree elevation

L'algoritmo di elevamento di grado classico su NUBS permette di incrementare di uno il grado di tutti i tratti polinomiali della funzione. Nel caso Multi-Degree, invece, l'algoritmo permette di limitare la richiesta di incremento ad un singolo tratto $[x_j, x_{j+1}]$, piuttosto che a tutti i tratti contemporaneamente. Questo comporta che lo spazio MD-spline con un tratto elevato di 1 grado, abbia una dimensione di un'unità in più rispetto allo spazio iniziale. Ogni funzione B-spline MD iniziale risulta combinazione di al massimo due nuove funzioni B-spline MD dello spazio con grado elevato. Per la precisione tutte le B-spline MD iniziali che non contengono l'intervallo $[x_j, x_{j+1}]$ soggetto al *degree elevation* resteranno uguali nello spazio finale mentre le rimanenti saranno rappresentabili come una combinazione di al più due nuove.

Teorema 3.3. *(Degree elevation) Siano Δ_t^* e Δ_s^* le partizioni nodali estese sinistra e destra per $S(\mathcal{P}_\mathcal{N}, \mathcal{K}, \Delta)$. Eseguendo il degree-elevation dello spazio spline Multi-Degree S nell'intervallo $[x_j, x_{j+1}]$, otteniamo il nuovo spazio $S(\hat{\mathbb{P}_\mathcal{N}}, \mathcal{K}, \Delta)$ e le nuove partizioni nodali $\hat{\Delta}_t^*$ e $\hat{\Delta}_s^*$. Siano $m = \max\{n_i\}$, $\hat{m} = \max\{\hat{n}_j\}$ ed ℓ tale che $t_\ell \leq x_j < t_\ell + 1$. Denotando con $N_{i,m}$ e*

$\hat{N}_{i,\hat{m}}$ le rispettive funzioni base B-spline MD, esistono i coefficienti $\alpha_{i,m}$ con $0 \leq \alpha_{i,m} \leq 1$, $i = 1, \ldots, K+1$, tali che:

$$N_{i,m}(x) = \alpha_{i,m}\hat{N}_{i,\hat{m}}(x) + (1 - \alpha_{i+1,m})\hat{N}_{i+1,\hat{m}}(x), \qquad \begin{matrix} i = 1, \ldots, K \\ x \in \mathbb{R} \end{matrix} \tag{3.20}$$

In particolare:

$$\alpha_{i,m} = \begin{cases} 1, & i \leq \ell - n_j \\ \dfrac{D_+^{k_i^t+1} f_{i,p^{t_i}}(t_i)}{D_+^{k_i^t+1} \hat{f}_{i,p^{t_i}}(t_i)}, & \ell - n_j + 1 \leq i \leq \ell \\ 0, & i \geq \ell + 1 \end{cases} \tag{3.21}$$

dove $f_{i,p^{t_i}}$ e $\hat{f}_{i,p^{t_i}}$ sono i primi tratti non banali delle funzioni di transizione che definiscono, rispettivamente, le funzioni base B-spline multi-degree $N_{i,m}$ e $\hat{N}_{i,\hat{m}}$ e k_i^t è dato in in P.3.4.

La funzione MD-spline $f(x)$ in seguito all'elevamento di grado, avrà, nel nuovo spazio $S(\hat{\mathbb{P}}_{\mathcal{N}}, \mathcal{K}, \Delta)$, una rappresentazione della forma:

$$f(x) = \sum_{i=1}^{K+1} \hat{c}_i \, \hat{N}_{i,\hat{m}}(x) \tag{3.22}$$

dove le $\hat{N}_{i,\hat{m}}(x)$ sono le funzioni base definite su $\hat{\Delta}_t^*$ e $\hat{\Delta}_s^*$.

Corollario 3.2. *Siano Δ_t^*, Δ_s^* e $\hat{\Delta}_t^*$, $\hat{\Delta}_s^*$ le partizioni nodali già definite, allora:*

$$\hat{c}_i = \begin{cases} c_i & i \leq \ell - n_j \\ \alpha_i c_i + (1 - \alpha_i) c_{i-1} & \ell - n_j + 1 \leq i \leq \ell \\ c_{i-1} & i \geq \ell + 1 \end{cases} \tag{3.23}$$

In [10] viene descritto un diverso approccio all'elevamento di grado su MD-spline che, invece di utilizzare le funzioni di transizione, sfrutta la relazione di ricorrenza integrale vista in 3.4.

3.6.4 Degree reduction

Il *Degree reduction* è l'operazione inversa del degree elevation. Come abbiamo visto in precedenza, dal punto di vista dei calcoli, gli algoritmi di knot-insertion e di degree elevation sono molto simili. Entrambi, infatti, incrementano lo spazio (ed il numero di coefficienti) di 1 e calcolano i nuovi valori mediante combinazioni lineari con gli α_i. È quindi ovvio pensare che knot-removal e degree reduction, che rappresentano le controparti dei suddetti algoritmi, possano comportarsi in maniera analoga. Ed infatti, non solo entrambi riducono di 1 la dimensione dello spazio e possono essere eseguiti in maniera esatta o approssimata, ma il metodo usato dal degree reduction per calcolare i nuovi coefficienti, può essere ricondotto alla soluzione di un sistema di $K + 1$ equazioni in K incognite, esattamente come visto per il knot-removal. Pertanto, per non essere ripetitivi, tale algoritmo non verrà qui ulteriormente descritto.

Capitolo 4

Curve MD-Spline

Le tradizionali curve B-spline [11,12] usate in modellazione, essendo generate mediante funzioni NUBS, sono costituite da segmenti polinomiali aventi tutti lo stesso grado. Nella pratica comunque sono spesso necessarie curve formate da segmenti polinomiali di gradi differenti e, l'utilizzo delle curve B-spline per modellarle, ha i seguenti svantaggi:

- costruire una curva composta da segmenti di gradi diversi, ad esempio m ed n con $m < n$, richiede ai progettisti CAD di scegliere la base polinomiale di grado più alto, cioè n, ed utilizzare le funzioni base B-spline di tale grado per ottenere la curva. Questo fa si che, per ogni segmento di grado m o n, verranno utilizzati $(n+1)$ punti di controllo. Dal momento che i segmenti di curva di grado m ne necessiterebbero solamente di $(m+1)$, è chiaro che questo metodo di operare introduce una quantità superflua di punti di controllo.

- modellare segmenti polinomiali con grado maggiore di quello effettivamente richiesto, è poco pratico. Si pensi, ad esempio, di dover modellare un segmento rettilineo, ovvero di grado 1, di una curva B-spline di grado 4. Nonostante per definire tale segmento basterebbe posizionare due soli punti di controllo in corrispondenza degli estremi, il grado quattro della B-spline obbliga il progettista a dover posizionare un totale di

cinque punti di controllo collineari. La modellazione della curva viene cioè complicata inutilmente.

La soluzione consiste nell'utilizzare curve spline Multi-Degree, formate cioè da segmenti polinomiali di gradi diversi, dette *curve MD-spline*.
Queste, introdotte per la prima volta in [13], sono funzioni vettoriali le cui componenti sono le funzioni MD-spline discusse nel precedente capitolo. Tali curve offrono i seguenti vantaggi:

- permettono di rappresentare una qualsiasi forma specificando solo i punti di controllo strettamente necessari a definirla;
- i gradi dei singoli segmenti della curva possono essere modificati in maniera indipendente l'uno dall'altro;
- il vantaggio più importante, diretta conseguenza dei precedenti due, è la possibilità, offerta ai progettisti, di modellare interattivamente in un modo più semplice e rapido rispetto alle curve B-spline.

È infine opportuno sottolineare che le curve MD-spline includono, come casi speciali, i due tipi di rappresentazioni di curve maggiormente usati nel CAD, cioè le curve B-spline e le curve di Bézier. In particolare, estendendo le curve B-spline, ne ereditano le importanti proprietà di modellazione, tra cui, ad esempio, controllo locale, guscio convesso, continuità nei punti di raccordo e invarianza per trasformazioni affini.

Questo capitolo formalizza la definizione di curve MD-spline in forma parametrica e ne illustra proprietà ed algoritmi.

4.1 Curve MD-Spline parametriche

Riprendendo quanto detto nel Capitolo 1, una curva $\mathbf{C}(x) \in \mathbb{R}^2$ in forma parametrica è una funzione vettoriale composta da 2 componenti/funzioni scalari, che può essere scritta come:

$$\mathbf{C}(x) = \begin{pmatrix} C_1(x) \\ C_2(x) \end{pmatrix}, \quad x \in [a, b]$$

Questa definizione può essere specializzata per definire formalmente le curve MD-spline in forma parametrica.

Definizione 4.1 (**Curva MD-spline in forma parametrica**)**.**
Siano $\mathbf{P}_i \equiv (P_{x_i}, P_{y_i})$, $i = 1, \ldots, K$, punti in $\mathbb{R}^2$. Una curva $\mathbf{C}(x) \in \mathbb{R}^2$, le cui componenti sono funzioni MD-spline, ovvero $C_1(x)$ e $C_2(x) \in S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$, è detta *curva MD-spline in forma parametrica*. La sua espressione è:

$$\mathbf{C}(x) = \begin{pmatrix} \sum_{i=1}^{K} P_{x_i} N_i(x) \\ \sum_{i=1}^{K} P_{y_i} N_i(x) \end{pmatrix} = \sum_{i=1}^{K} \mathbf{P}_i N_i(x) \tag{4.1}$$

dove i $\mathbf{P}_i$ sono i punti di controllo della curva. La spezzata che si ottiene congiungendo coppie consecutive di $\mathbf{P}_i$ è la poligonale di controllo della curva.

In figura 4.1 è mostrata una curva MD-spline le cui funzioni coordinate appartengono allo spazio multi-degree dell'esempio 3.1. La relativa poligonale di controllo è formata da 7 punti di controllo (la dimensione $K = 7$ dello spazio). I pallini rossi sulla curva rappresentano i punti della curva valutati in corrispondenza dei breakpoint e pertanto identificano inizio e fine dei segmenti polinomiali che formano la curva. I segmenti sono tre, ognuno con corrispondente grado n_i, $i = 0, \ldots, 2$.

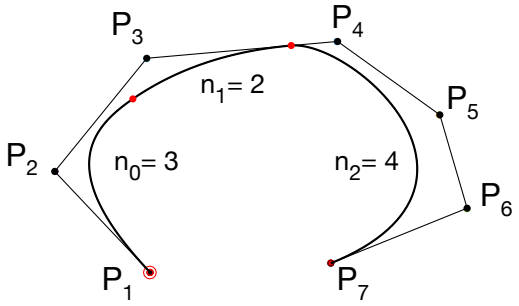


Figura 4.1: Curva MD-spline associata allo spazio dell'esempio 3.1

4.2 Le proprietà delle curve MD-spline

Le proprietà delle curve MD-spline derivano direttamente da quelle delle funzioni MD-spline e delle funzioni base B-Spline multi-degree. Le più importanti sono elencate di seguito.

P.4.1 (*Polinomiale a tratti*) $\mathbf{C}(x)$ è una curva polinomiale a tratti: su ogni segmento $[x_j, x_{j+1})$ è una curva polinomiale di grado n_j.

P.4.2 La curva interpola il primo e l'ultimo punto di controllo cioè:

$$\mathbf{C}(a) = \mathbf{P}_1, \quad \mathbf{C}(b) = \mathbf{P}_K$$

dove $[a, b]$ è l'intervallo di definizione della curva.

P.4.3 (*Controllo locale*) Lo spostamento di un punto di controllo P_i modifica la curva $\mathbf{C}(x)$ solo nell'intervallo $[t_i, s_i]$. Questo è conseguenza della proprietà P.3.1 delle B-spline multi-degree ($N_i(x) = 0$ per $x \notin [t_i, s_i]$).

P.4.4 (*Guscio convesso*) La curva MD-spline $\mathbf{C}(x)$ cade interamente all'interno del poligono di controllo generato dai relativi punti di controllo.

P.4.5 (*Invarianza per trasformazioni affini*) La curva MD-spline non cambia forma operando una trasformazione affine (traslazione, rotazione e scala) al suo poligono di controllo.

P.4.6 (*Continuità nei punti di raccordo*)
Dati due segmenti di grado m ed n, nel punto di raccordo tra i due, la curva $\mathbf{C}(x)$ ha, al massimo, continuità C^k con:

$$\begin{aligned} k &= \min(n, m) \quad \text{se} \quad n \neq m \\ k &= m - 1 \qquad \text{se} \quad n = m \end{aligned}$$

P.4.7 (*Curva B-spline come caso speciale*) Se tutti i segmenti della curva hanno grado n allora la curva MD-spline si riduce ad una curva B-spline di grado n.

P.4.8 (*Curva di Bézier come caso speciale*)
Dal momento che, per la proprietà P.3.6, un'opportuna configurazione delle partizioni nodali estese fa si che le funzioni base B-spline multi-degree si riducano ai polinomi di Bernstein, allora una curva MD-spline associata a tali partizioni si riduce ad una curva di Bézier.

P.4.9 La poligonale di controllo approssima la forma della curva. Come si vedrà in 4.3.1, tale approssimazione può essere migliorata mediante l'algoritmo di knot-insertion.

P.4.10 (*Variation Diminishing*) Il numero di intersezioni della curva con una qualsiasi retta non è mai maggiore del numero di intersezioni di tale retta con il poligono di controllo della curva.

P.4.11 Per la proprietà P.3.5, risulta che:

$$\mathbf{C}(x) = \sum_{i=\ell-n_j}^{\ell} \mathbf{P}_i \, N_i(x), \quad x \in [x_j, x_{j+1}], \quad t_\ell \leq x_j < t_{\ell+1} \tag{4.2}$$

4.3 Algoritmi per curve MD-Spline

In questa sezione vengono descritti i principali algoritmi per curve MD-spline.

Dal momento che una curva MD-spline $\mathbf{C}(x)$ in $\mathbb{R}^2$ è costituita da due funzioni MD-spline, $C_1(x)$ e $C_2(x)$, gli algoritmi di knot-insertion, knot-removal,

degree-elevation e degree-reduction, visti nel paragrafo 3.6, possono essere applicati alle curve MD-spline semplicemente eseguendo due volte i calcoli che riguardano i coefficienti c_i, una utilizzando i coefficienti P_{x_i} di $C_1(x)$ e una con i P_{y_i} di $C_2(x)$. Ovvero, gli algoritmi geometrici su curve MD-spline operano, contemporaneamente, su entrambe le coordinate dei punti di controllo della curva.

Vediamo, a titolo esemplificativo, come l'algoritmo di knot-insertion per curve MD-spline calcola i nuovi punti di controllo. Se $\{\mathbf{P}_i\}_{i=1}^{K}$ sono i punti di controllo della curva prima del knot-insertion del nodo $\hat{t}$, con $t_\ell \leq \hat{t} < t_{\ell+1}$, allora i punti di controllo $\{\mathbf{Q}_i\}_{i=1}^{K+1}$ della curva dopo l'inserimento del nuovo nodo sono dati da:

$$\mathbf{Q}_i = \begin{cases} \mathbf{P}_i & i \leq \ell - n_j \quad \text{con} \quad x_j \leq \hat{t} < x_{j+1} \\ \alpha_i \mathbf{P}_i + (1-\alpha_i)\mathbf{P}_{i-1} & \ell - n_j + 1 \leq i \leq \ell - r + 1 \\ \mathbf{P}_{i-1} & i \geq \ell - r + 2 \end{cases} \tag{4.3}$$

dove gli α_i sono quelli definiti in (3.17) ed $n_j - r$ è l'ordine di continuità richiesto nel breakpoint associato a $\hat{t}$ nel nuovo spazio.

Come si vede l'unica differenza tra questa espressione e la (3.19) è il fatto che i coefficienti c_i e $\hat{c}_i$ sono stati rimpiazzati dai punti di controllo $\mathbf{P}_i$ e $\mathbf{Q}_i$. I suddetti algoritmi visti per le funzioni MD-spline possono infatti essere estesi al caso di curve MD-spline semplicemente sostituendo, nelle relative espressioni, i coefficienti c_i e $\hat{c}_i$ con, rispettivamente, i punti di controllo $\mathbf{P}_i$ e $\mathbf{Q}_i$. Per tale motivo non ci dilungheremo ulteriormente nelle loro descrizione.

I due paragrafi seguenti illustreranno, invece, altri tre algoritmi geometrici legati a questi ma che non sono stati trattati nel capitolo 3.

Gli ultimi tre paragrafi di questa sezione trattano, invece, l'algoritmo di valutazione di una curva MD-spline e gli algoritmi di conversione tra rappresentazione MD-spline e Bézier a tratti. Abbiamo infatti visto che le curve spline, grazie soprattutto alla loro proprietà di supporto locale, si prestano meglio alla modellazione rispetto alle curve di Bézier. Queste ultime però godono di algoritmi di computazione molto efficienti. Pertanto, per combi-

nare i vantaggi offerti dalle due rappresentazioni, è spesso utile poter passare da una all'altra.

4.3.1 Knot refinement e Knot simplify

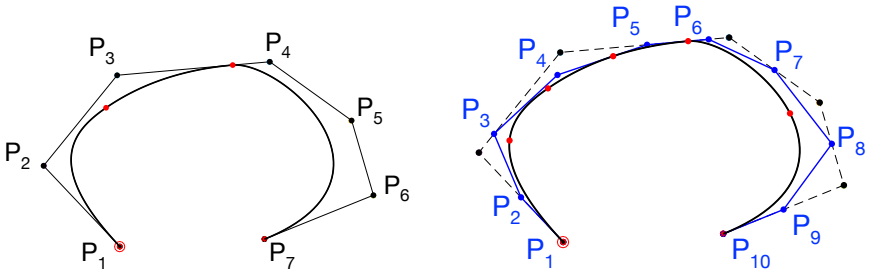
Sia $\mathbf{C}(x)$ la curva MD-spline le cui componenti sono funzioni dello spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$. Sia $a \equiv x_0 < \ldots < x_{q+1} \equiv b$, la partizione di $[a, b]$ indotta da Δ. Sia poi $\hat{X} = \{\hat{x}_0, \ldots, \hat{x}_q\}$ con $x_i < \hat{x}_i < x_{i+1}$ per tutti gli i.

L'algoritmo di *knot-refinement* consiste in un knot-insertion dei nodi in $\hat{X}$ nelle partizioni Δ_t^* e Δ_s^* associate a $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$. Comunemente tali nuovi nodi $\hat{x}_i$ vengono scelti in modo da essere posizionati esattamente al centro del rispettivo intervallo nodale $[x_i, x_{i+1})$.

Ad ogni esecuzione dell'algoritmo di knot-refinement il poligono di controllo si avvicina sempre di più alla forma della curva fino a convergere alla curva stessa. Per tale motivo, nella pratica, tale algoritmo è spesso utilizzato per ottenere un'approssimazione poligonale della curva. In figura 4.2b è mostrato il risultato di un passo di esecuzione dell'algoritmo di knot-refinement sulla curva di figura 4.2a. In particolare, come si nota osservando i pallini rossi, sono stati aggiunti tre nuovi breakpoint al centro di ogni segmento esistente e, di conseguenza, la dimensione dello spazio (ed il numero di punti di controllo) è aumentata di tre.

Osservazione 4.1. Dal momento che il knot-refinement non è altro che l'esecuzione di più knot-insertion, per quanto detto in 3.6.1, tale algoritmo non andrà ad alterare la forma della curva (come si vede in figura 4.2).

L'operazione inversa del knot-refinement è il *knot-simplify* che permette di rimuovere i nodi che risultano superflui ai fini della definizione della forma della curva. In particolare, questo algoritmo continua ad eseguire un knot-removal esatto sui nodi delle partizioni estese Δ_t^* e Δ_s^* associate alla curva finché non viene raggiunto uno stato in cui la rimozione esatta di un qualsiasi nodo non è più possibile, ovvero una rimozione in più modificherebbe la forma della curva. La figura 4.2, se vista come il passaggio da 4.2b alla 4.2a, è un esempio di knot-simplify.

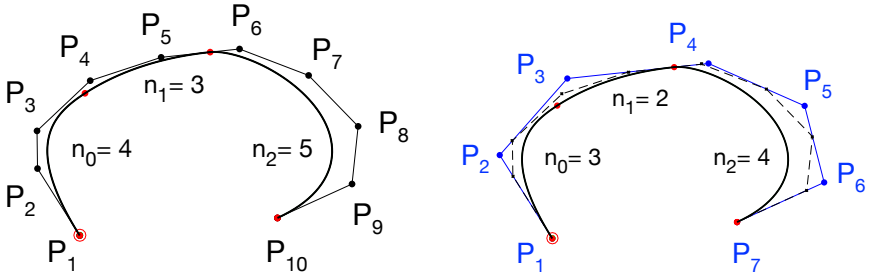


(a) Prima *(b)* Dopo

Figura 4.2: Knot-refinement

4.3.2 Degree simplify

L'algoritmo di *Degree-simplify* opera in maniera simile all'algoritmo di knot-simplify, ma invece di rimuovere i nodi superflui, riduce il grado di ogni segmento fino al minimo grado che permette di mantenere invariata la forma della curva. In figura 4.3a viene mostrata una curva i cui tre segmenti possono essere ridotti di grado in maniera esatta ed in figura 4.3b vi è il risultato dell'applicazione dell'algoritmo di degree-simplify su tale curva; come si vede, dal momento che ognuno dei tre segmenti è stato ridotto di un grado, il numero di punti di controllo viene ridotto di tre. Si noti che tale curva anche dopo il degree-simplify ha mantenuto la stessa forma; l'algoritmo infatti smette di ridurre il grado di un segmento non appena rileva che la riduzione andrebbe a modificarne la forma.



(a) Prima *(b)* Dopo

Figura 4.3: Degree simplify

Ora che abbiamo definito gli algoritmi di Knot-simplify ed Degree-simplify, è possibile dare la seguente definizione correlata.

Definizione 4.2 (**Curva MD-spline in forma minima**)**.** Si definisce *curva MD-spline in forma minima* una curva per cui non sia possibile rimuovere nessun nodo né ridurre il grado di nessun intervallo senza lasciare la forma della curva inalterata.

4.3.3 Degree Simplify e Knot Simplify non esatti

Sia degree-simplify, in 4.3.2, che knot-simplify, in 4.3.1, comportano una semplificazione esatta, cioè vengono rimosse solo le informazioni superflue ai fini della definizione della forma.

Spesso risulta utile una versione non esatta, ovvero approssimata, degli algoritmi di Degree Simplify e Knot Simplify. Questi partendo dalla curva MD-spline da semplificare, eseguono i seguenti due passi:

1. data una curva MD-spline in forma minima si definisce uno spazio MD-spline di dimensione inferiore richiedendo una continuità maggiore in alcuni nodi o un grado inferiore in alcuni intervalli;

2. si applica un algoritmo di approssimazione che, dato un insieme di punti campionati sulla MD-spline iniziale, restituisce, nello spazio definito al punto 1, la MD-spline che meglio approssima quella iniziale.

Al termine del primo passo si è in grado di calcolare la dimensione K del nuovo spazio, di creare le partizioni estese Δ_t^* e Δ_s^* e di calcolare le B-spline multi-degree $\{N_i\}_{i=1}^K$ associate, che saranno usate dall'algoritmo di approssimazione nel passo successivo.

L'algoritmo di approssimazione è basato sul calcolo della cosiddetta base B-spline duale $\{D_i\}_{i=1}^K$, una base costruita a partire dalla base B-spline multi-degree $\{N_i\}_{i=1}^K$ in modo che siano soddisfatte le seguenti condizioni:

- le funzioni B-spline duali $\{D_i(x)\}_{i=1}^K$ generano lo stesso spazio generato dalle $\{N_i(x)\}_{i=1}^K$;

- $\langle N_i, D_j \rangle = \delta_{ij}, \quad 1 \leq i, j \leq K,$
  dove δ_{ij} è il *delta di Kronecker*, cioè uguale a 1 se $i = j$ e 0 altrimenti, e $\langle \cdot, \cdot \rangle$ indica un prodotto interno (in questo lavoro si è usato

$$\langle f, g \rangle = \sum_{i=1}^{n} f(\tau_i)\, g(\tau_i)$$

  con τ_i opportuni parametri in [a,b]).

In [14] viene dettagliatamente spiegato il procedimento di costruzione della base B-spline duale e ne vengono mostrate le proprietà fondamentali che ne permettono l'utilizzo in diversi problemi di approssimazione legati all'analisi numerica e alla *computer graphics*. Tra queste, in questo lavoro si sono usate le seguenti:

- sia $f(x) \in S$, la sua rappresentazione nella base delle B-spline multi-degree $\{N_i\}_{i=1}^{K}$ avrà come coefficienti i c_i dati da:

$$c_i = \langle f(x), D_i(x) \rangle \quad i = 1, \ldots, K \tag{4.4}$$

  Questo significa che una volta calcolata la base duale (che verrà precalcolata una volta), i coefficienti c_i sono semplicemente dati dal prodotto interno fra la $f(x)$ e le funzioni base duali D_i.

- data una funzione $f(x) \in S$ e uno spazio $\hat{S} \subset S$, allora la funzione $g(x) \in \hat{S}$ che costituisce la miglior approssimazione in $\hat{S}$ di $f(x)$ è data da:

$$g(x) = \sum_{i=1}^{\hat{K}} c_i \hat{N}_i(x) \quad \text{dove } c_i = \langle f(x), \hat{D}_i(x) \rangle, \quad i = 1, \ldots, \hat{K} \tag{4.5}$$

  con $\hat{N}_i$ e $\hat{D}_i$ le basi B-spline e duale di $\hat{S}$ che ha dimensione $\hat{K} < K$.

In pratica l'algoritmo di approssimazione si servirà dell'espressione (4.5) per ottenere le coordinate dei punti di controllo della curva MD-spline semplificata.

4.3.4 Conversione da MD-Spline a Bézier

Se supponiamo che lo spazio spline sia localmente generato dalla base di Bernstein, abbiamo visto che una funzione MD-spline $f(x)$ può essere espressa con l'equazione (3.12). Da quest'ultima è possibile ricavare la corrispondente equazione per curve MD-spline:

$$\mathbf{C}(x) = \sum_{i=\ell-n_j}^{\ell} \mathbf{P}_i \sum_{h=0}^{n_j} (b_{i,h_{i,j}+h} - b_{i+1,h_{i+1,j}+h}) B_{h,n_j}(x) \tag{4.6}$$

dove i $\mathbf{P}_i$ sono i punti di controllo della curva MD-spline.

Tale relazione è alla base dell'algoritmo di conversione di una curva MD-spline nella corrispondente curva di Bézier a tratti.

In particolare, l'algoritmo itera sui segmenti della curva MD-spline eseguendo, per ogni segmento polinomiale, i seguenti passi:

1. dall'intervallo $[x_j, x_{j+1})$ di grado n_j della curva MD-spline viene creato il corrispondente tratto della curva di Bézier di egual grado ed ampiezza parametrica.

2. vengono calcolati gli $n_j + 1$ punti di controllo del tratto di Bézier. In particolare, dalla (4.6), se indichiamo con $\mathbf{P}_i$ i punti di controllo della MD-spline e con $\{\mathbf{Q}_h\}_{h=0}^{n_j}$ i punti di controllo della curva di Bézier definita sul tratto considerato, la relazione che li lega è espressa da:

$$\sum_{h=0}^{n_j} \underbrace{\sum_{i=\ell-n_j}^{\ell} \mathbf{P}_i (b_{i,h_{i,j}+h} - b_{i+1,h_{i+1,j}+h})}_{\mathbf{Q}_h} B_{h,n_j}(x), \qquad \begin{aligned} & t_\ell \leq x_j < t_{\ell+1} \\ & h_{i,j} = \sum_{\ell=p^{t_i}}^{j-1} (n_\ell + 1) \end{aligned} \tag{4.7}$$

Osservazione 4.2. Al termine dell'algoritmo la forma della curva sarà rimasta la stessa. Saranno, eventualmente, i punti di controllo a variare in numero e posizione per riflettere la diversa rappresentazione sottostante.

4.3.5 Valutazione di una curva MD-Spline

Dati i punti di controllo $\{\mathbf{P}_i\}_{i=1}^{K}$, la valutazione di una curva MD-spline può avvenire in due modi.

Il primo modo non è nient'altro che un'estensione del procedimento descritto in 3.5 dallo spazio delle funzioni allo spazio vettoriale delle curve.

In particolare, per $x \in [x_j, x_{j+1}]$, $t_\ell \leq x_j < t_{\ell+1}$, la (3.12) diventa:

$$\mathbf{C}(x) = \sum_{i=\ell-n_j}^{\ell} \mathbf{P}_i \, N_i(x) =$$

$$= \begin{pmatrix} \sum_{i=\ell-n_j}^{\ell} P_{x_i} N_i(x) \\ \sum_{i=\ell-n_j}^{\ell} P_{y_i} N_i(x) \end{pmatrix} = \begin{pmatrix} \sum_{i=\ell-n_j}^{\ell} P_{x_i} (f_{i,j}(x) - f_{i+1,j}(x)) \\ \sum_{i=\ell-n_j}^{\ell} P_{y_i} (f_{i,j}(x) - f_{i+1,j}(x)) \end{pmatrix}$$

Si noti che le stesse N_i vengono usate nel calcolo della $C_1(x)$ che nel calcolo della $C_2(x)$ e pertanto basta calcolarle una volta sola.

In particolare, riprendendo l'esempio in 3.5.2, i primi tre passi descritti vanno eseguiti identici anche nella valutazione di curve MD-spline. Quello che cambia sarà solamente l'ultimo passo. Questo infatti diventerà:

$$\mathbf{C}(\overline{x}) = \begin{pmatrix} C_1(\overline{x}) \\ C_2(\overline{x}) \end{pmatrix} = \begin{pmatrix} \sum_{i=2}^{4} P_{x_i} \, N_i(\overline{x}) \\ \sum_{i=2}^{4} P_{y_i} \, N_i(\overline{x}) \end{pmatrix} = \begin{pmatrix} \sum_{i=2}^{4} P_{x_i} (f_{i,1}(\overline{x}) - f_{i+1,1}(\overline{x})) \\ \sum_{i=2}^{4} P_{y_i} (f_{i,1}(\overline{x}) - f_{i+1,1}(\overline{x})) \end{pmatrix}$$

Il secondo metodo di valutazione di una curva MD-spline consiste nell'utilizzare la corrispondente rappresentazione di Bézier. Ad esempio, si supponga di voler valutare la curva MD-spline $f(x)$ in un punto $\overline{x}$, tale che $\overline{x} \in [x_j, x_{j+1}]$. Se il grado di tale intervallo è n_j, i punti di controllo nella rappresentazione di Bézier, per la (4.7), sono dati da:

$$\mathbf{Q}_h = \sum_{i=\ell-n_j}^{\ell} \mathbf{P}_i \, (b_{i,h_{i,j}+h} - b_{i+1,h_{i+1,j}+h}), \qquad \begin{array}{l} t_\ell \leq x_j < t_{\ell+1}, \quad h_{i,j} = \sum_{\ell=p^{t_i}}^{j-1} (n_\ell + 1) \\ h = 0, \ldots, n_j \end{array}$$

dove i $b_{i,k}$ sono i coefficienti delle espansioni locali della funzione di transizione $f_{i,k}$ nella base di Bernstein e sono calcolati come mostrato nell'esempio (3.5.2). Pertanto, dalla (2.2), il valore della curva in $\overline{x}$ è dato da:

$$\mathbf{C}(\overline{x}) = \sum_{h=0}^{n_j} \mathbf{Q}_h \, B_{h,n_j}(\overline{x})$$

4.3.6 Conversione da Bézier a MD-spline

L'algoritmo di conversione dalla rappresentazione di Bézier a tratti ad MD-spline è costituito dai seguenti passi:

1. si itera sui tratti della curva di Bézier salvando il grado di ogni tratto e la relativa ampiezza parametrica dell'intervallo di definizione ed imponendo una continuità C^0 nei punti di raccordo. Al termine si saranno quindi costruiti i vettori Δ, $\mathcal{N}$ e $\mathcal{K}$ e sarà possibile definire lo spazio $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ a cui appartiene la curva MD-spline;

2. i punti di controllo della curva MD-spline vengono ottenuti da quelli della curva di Bézier a tratti prendendo una sola volta quelli comuni di raccordo tra due segmenti.

Si osservi che, nel primo passo, viene imposta una continuità C^0 dal momento che una curva di Bézier a tratti non contiene alcuna informazione sulle continuità nei punti di raccordo. Possiamo dire di avere una curva MD-spline non in forma minima, per cui si esegue un passo ulteriore per ottenere la forma minima.

3. si applica alla curva MD-spline ottenuta dai precedenti due passi l'algoritmo di knot-simplify, descritto in 4.3.1.

Al termine di quest'ultimo passo, lo spazio multi-degree $S(\mathbb{P}_{\mathcal{N}}, \mathcal{K}, \Delta)$ associato alla curva MD-spline sarà stato opportunamente modificato in modo che il vettore $\mathcal{K}$ contenga i reali valori di continuità presenti nei punti di raccordo della curva di Bézier.

Capitolo 5

Modellazione di curve MD-spline

In seguito allo studio teorico delle curve MD-spline, il lavoro di tesi si è focalizzato sul lato implementativo delle stesse. In particolare, questo ha comportato la modifica di un software di modellazione esistente, chiamato *Mini-System*, che inizialmente permetteva di gestire solo curve di Bézier a tratti. A questo è stata aggiunta la possibilità di creare curve MD-spline e di modellarle, in maniera interattiva, utilizzando gli algoritmi di editing visti nei capitoli precedenti.

È importante sottolineare che, in seguito alle modifiche effettuate, il Mini-System è il primo software a consentire la modellazione da zero di curve MD-Spline, possibilità non fornita nemmeno dai più noti software CAD commerciali.

In questo capitolo si vanno a descrivere le funzionalità che sono state aggiunte al Mini-System per renderlo a tutti gli effetti un software di modellazione di curve MD-spline. L'ultima sezione del capitolo illustra un esempio di modellazione, eseguita mediante il suddetto software, che permette di apprezzare i vantaggi offerti dalla rappresentazione spline multi-degree.

5.1 Introduzione al Mini-System

Il Mini-System è un software didattico, nato inizialmente come strumento di supporto e di esercitazione per gli studenti frequentanti i corsi di Calcolo Numerico e di Grafica tenuti all'Università di Bologna.

Il codice è scritto in linguaggio C e per la parte grafica fa uso delle librerie SDL 2.0, sia per le funzioni di disegno che per la GUI. Quest'ultima, mostrata in figura 5.1, è composta da quattro componenti:

- una barra di menu a tendina, posizionata nella parte alta della finestra;
- una barra di due colonne di bottoni, nella parte sinistra della finestra;
- un'area informativa, posizionata nella parte bassa della finestra;
- un'area di disegno, in cui si creano e modellano le curve.

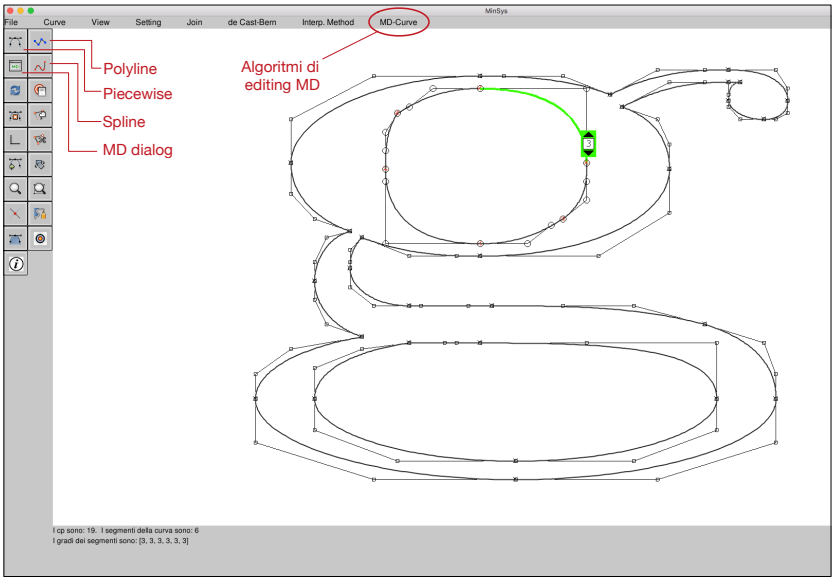


Figura 5.1: L'interfaccia del Mini-System

Il Mini-System, nella sua versione originaria, permetteva di sperimentare unicamente con curve di Bézier a tratti ed algoritmi su polinomi e polinomi a tratti espressi nella base di Bernstein.

Il motivo per cui tale sistema è stato ritenuto, fin da subito, il candidato ideale per l'implementazione delle curve MD-spline è che questo rende disponibile un motore di calcolo efficiente che opera sulla rappresentazione locale della curva, ovvero su polinomi di Bernstein nei singoli intervalli. Le relative funzioni di calcolo possono quindi essere sfruttate anche con curve MD-spline il cui spazio sia localmente generato proprio dalla base di Bernstein.

La generalizzazione del Mini-System a sistema di modellazione di curve MD-spline è avvenuta mantenendo la netta distinzione tra funzioni di calcolo e interfaccia di modellazione. Mentre le prime, tra cui ad esempio gli algoritmi che determinano i punti di intersezione tra curve o quelli che ne permettono di fare lo *split*, sfruttano la rappresentazione locale nella base di Bernstein per garantire l'efficienza computazionale e sono state in gran parte ereditate dalla versione originaria del Mini-System, la seconda è completamente nuova e permette di agire sui punti di controllo, gradi, continuità, nodi ed ampiezze parametriche della curva MD-spline allo scopo di garantire all'utente la migliore esperienza possibile di modellazione. In pratica, ogni curva MD-spline presente nel sistema ha sempre associata una corrispondente rappresentazione di Bézier a tratti che può essere passata alle funzioni di calcolo. In tal modo è possibile combinare i vantaggi offerti dagli efficienti algoritmi che operano su polinomi di Bernstein con i vantaggi offerti dalla modellazione con MD-spline.

Lo sviluppo del software ha comportato l'implementazione delle nuove funzionalità, sia grafiche che di calcolo, relative alle curve MD-spline, nonché l'estensione di numerose funzioni preesistenti affinché potessero gestire il caso multi-degree. In particolare, le principali aggiunte sono le seguenti:

- sono state implementate le funzioni relative alla definizione dello spazio multi-degree, alla valutazione di una curva MD-spline (calcolo delle funzioni di transizione e delle funzioni base Multi-Degree), gli algoritmi di conversione tra le rappresentazioni di Bézier ed MD-spline e tutti gli algoritmi di editing discussi nei precedenti capitoli;

- tutte le nuove funzionalità correlate alla creazione e modellazione di

curve MD-spline sono state rese accessibili attraverso la pre-esistente GUI del Mini-System;

- l'interfaccia è stata arricchita di varie funzionalità grafiche che permettono all'utente di modellare le curve MD-spline in maniera interattiva.

Osservazione 5.1. Dal momento che nel Mini-System sono stati implementati gli algoritmi di conversione da MD-spline a Beziér e viceversa, descritti in 4.3.4 e 4.3.6, e siccome una curva MD-spline può ridursi ad una curva B-spline, il Mini-System è in grado di gestire tutti e tre i tipi di curve.

In figura 5.2 è schematizzata l'architettura del Mini-System, comprensiva del nuovo modulo MD-spline. Il CONTROLLER traduce i comandi di input (GUI) in chiamate a funzioni di CALCOLO. Il motore di calcolo è formato da due sottocomponenti che interagiscono tra loro: la prima contiene le funzioni di calcolo che operano sulla rappresentazione locale della curva, mentre la seconda contiene gli algoritmi che permettono di definire e modellare le curve MD-spline. Queste interagiscono tra loro grazie alla possibilità di passare dalla rappresentazione MD-spline a quella di Bézier a tratti e viceversa. Il motore grafico (DISEGNO) si occupa di aggiornare la finestra della GUI tenendo conto delle modifiche eseguite dalle funzioni di calcolo.

Ad esclusione del motore di calcolo, le componenti del sistema utilizzano le librerie SDL 2.0 per le funzionalità grafiche e di gestione dell'input.

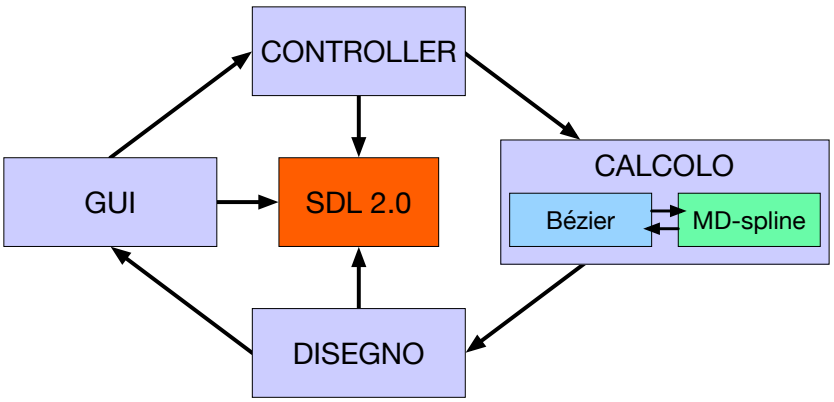


Figura 5.2: L'architettura del Mini-System

Nei prossimi due paragrafi andiamo a descrivere, brevemente, le nuove funzionalità associate alle curve Multi-Degree.

5.2 Creazione delle curve MD-spline

Il Mini-System permette di creare una curva MD-spline attraverso una delle seguenti opzioni. Queste sono accessibili dal menu *Curve* oppure mediante gli appositi bottoni mostrati in figura 5.1.

- *Piecewise*: l'utente disegna una curva di Bezier a tratti C^0, C^1 o C^2 posizionando nell'area di disegno, mediante clic del mouse, i punti di controllo e scegliendo quando terminare un tratto ed iniziare il successivo. Il grado di ogni tratto sarà determinato dal numero di punti di controllo inseriti per quel tratto. Appena l'operazione si conclude, il software genera la curva MD-spline associata mediante l'algoritmo di conversione descritto in 4.3.4.

- *Polyline*: l'utente disegna una poligonale che definisce una curva MD-spline composta da tratti di grado 1, raccordati con continuità C^0.

- *Spline*: l'utente posiziona nell'area di disegno un certo numero di punti di controllo. Il software usa la relazione $numNodi = numCPs - ordine$ per determinare un'opportuna curva (MD-)spline di grado 3 con continuità C^2 nei punti di raccordo.

- *MD dialog*: finestra di creazione di una MD-spline mediante input numerico. In particolare l'utente può specificare numero di segmenti, *knot intervals*, gradi, continuità e periodicità della curva. Il programma poi si occuperà automaticamente della creazione dello spazio Multi-Degree associato e chiederà all'utente di inserire nell'area di disegno i K (dimensione dello spazio MD) punti di controllo della curva. In figura 5.3 viene mostrata tale finestra con i campi compilati in modo da dare origine allo spazio MD dell'esempio 3.1.

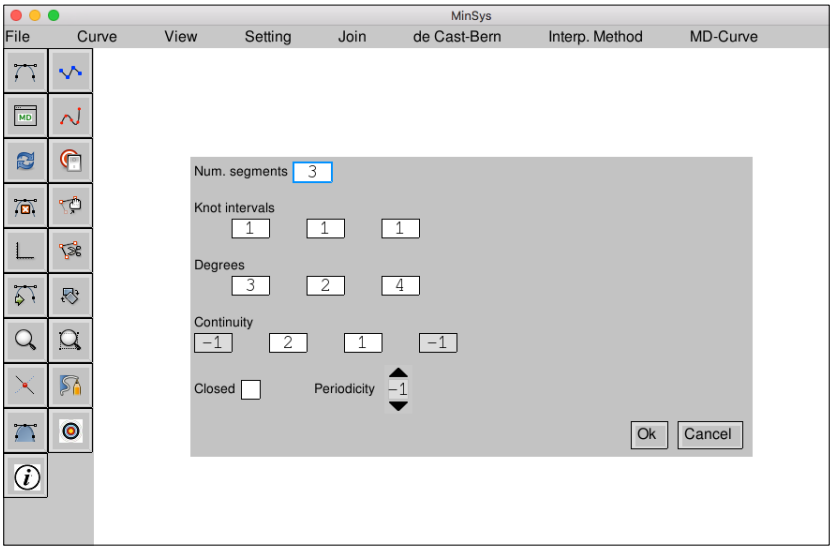


Figura 5.3: La finestra di creazione curve MD

5.3 Editing di curve MD-spline

In questa sezione vengono descritte le funzionalità che permettono di modellare geometricamente le curve MD-spline. Le funzionalità di editing sono accessibili tramite il menu *MD-Curve* oppure mediante l'interazione del mouse con la curva attiva.

5.3.1 Menu MD-Curve

Il menu MD-Curve, indicato in fig. 5.1, è il punto da cui si avviano tutti gli algoritmi di modellazione geometrica che possono essere impiegati su una curva Multi-Degree. Di seguito si elencano le varie voci che lo compongono.

- *Knot Insertion*: apre una finestra in cui è possibile digitare il valore di uno o più nodi che andranno inseriti, mediante l'algoritmo di knot-insertion, nelle partizioni nodali dello spazio Multi-Degree associato alla curva MD-spline correntemente attiva. L'inserimento di un nodo già presente nelle partizioni comporterà la riduzione della continuità nel breakpoint associato.

- *Interactive Knot Insertion*: questa voce dà all'utente la possibilità di eseguire un knot-insertion in maniera interattiva, ovvero mediante il clic del mouse direttamente sulla curva in corrispondenza del punto in cui si vorrebbe aggiungere un nuovo break-point.

- *Knot Refinement*: esegue l'algoritmo di knot-refinement visto in 4.3.1 sulla curva correntemente attiva. Ad ogni chiamata di tale algoritmo la poligonale di controllo della curva si avvicinerà sempre di più alla forma della curva stessa.

- *Knot Simplify*: esegue l'algoritmo di knot-simplify descritto in 4.3.1 sulla curva correntemente attiva. Tutti i nodi superflui ai fini della definizione della forma verranno rimossi.

- *Knot Removal*: apre una finestra in cui è possibile digitare il valore di uno o più nodi da passare all'algoritmo di knot-removal. Se la rimozione di un nodo non è possibile in modo esatto, allora verrà eseguita in modo approssimato. Il knot-removal di un nodo esistente comporta l'aumento della continuità nel breakpoint associato, per cui la possibilità di eseguire il knot removal di un nodo è vincolato al rispetto della proprietà P.4.6.

- *Interactive Knot Removal*: questa voce dà all'utente la possibilità di eseguire la rimozione di un breakpoint in maniera interattiva, ovvero mediante il clic del mouse direttamente sul breakpoint da rimuovere. Se il breakpoint non può essere rimosso in modo esatto verrà rimosso in modo approssimato. In ogni caso, al termine dell'operazione, il breakpoint non farà più parte dello spazio multi-degree associato alla curva. Si noti la differenza con l'opzione di menu Knot Removal che permette la rimozione singola dei nodi associati ai breakpoint.

- *Degree Elevation*: apre una finestra in cui è possibile digitare il numero di uno o più segmenti da elevare di grado. In particolare per specificare il segmento $[x_j, x_{j+1}]$ andrà digitato il valore j.

- *Degree Reduction*: apre una finestra in cui è possibile digitare il numero di uno o più segmenti da ridurre di grado. Viene impedito di avere segmenti con grado minore di 1.

- *Degree Simplify*: esegue l'algoritmo di degree-simplify descritto in 4.3.2 sulla curva attiva. Ogni segmento verrà ridotto di grado finché possibile senza alterare la forma della curva, oppure fino al raggiungimento del grado uno.

5.3.2 Editing interattivo

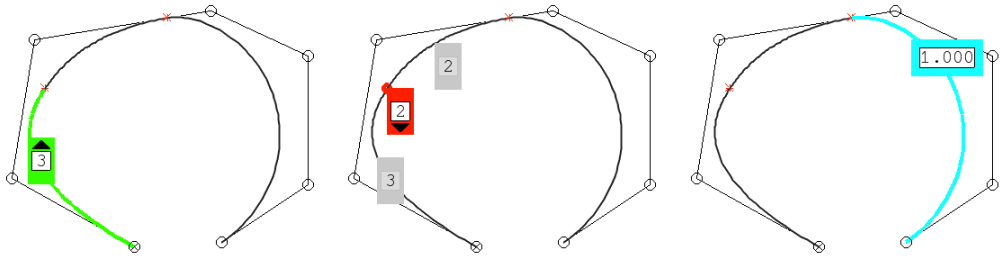
Il Mini-System offre all'utente la possibilità di modificare gradi, ordini di continuità ed ampiezze parametriche degli intervalli di una curva MD-spline in maniera semplice ed interattiva.

In particolare, come mostrato in figura 5.4a, avvicinandosi con il mouse ad un segmento della curva attiva questo si illumina di verde e, cliccandoci sopra con il tasto sinistro del mouse, compare un riquadro che permette di aumentarne o diminuirne il grado.

Se, invece, dopo che il segmento si è illuminato di verde, si fa clic su questo utilizzando il tasto destro del mouse, allora il segmento diventa azzurro, come mostrato in figura 5.4c, e si apre il riquadro che permette di modificarne l'ampiezza parametrica, detta anche *knot interval*.

Infine, avvicinandosi con il mouse ad un breakpoint della curva attiva, questo si trasforma in un pallino rosso e cliccandovi sopra appaiono tre riquadri: uno rosso in corrispondenza del breakpoint, che permette di aumentare o diminuirne la continuità, e due di colore grigio, posizionati al centro dei tratti precedente e successivo, che ne mostrano i relativi gradi. Questi ultimi due sono solo di tipo informativo, ovvero non danno alcuna possibilità di interazione, ma sono lì al solo scopo di rendere l'utente consapevole della continuità massima ottenibile nel breakpoint considerato. Questo tipo di interazione è mostrato in figura 5.4b.

Osservazione 5.2. Si noti che, mentre la modifica interattiva dei gradi richiama gli algoritmi di degree-elevation e degree-reduction, la modifica interattiva delle continuità richiama gli algoritmi di knot-insertion (per diminuire la continuità) e knot-removal (per aumentare la continuità).



(a) grado *(b)* continuità *(c)* knot interval

Figura 5.4: Editing interattivo

Va sottolineato che le modifiche ai gradi ed alle continuità sono vincolate al rispetto della proprietà P.4.6 ed i riquadri interattivi sono stati implementati in modo tale da impedire all'utente di infrangere tali limitazioni. Per cui se, ad esempio, un breakpoint x_j di continuità 2 è compreso tra due segmenti che hanno, il primo grado 2 ed il secondo grado 3, allora non sarà possibile eseguire il knot-removal del nodo $\hat{t} \equiv x_j$, ovvero la continuità del breakpoint non potrà essere incrementata. Questo caso è proprio quello mostrato in figura 5.4b; come si vede il riquadro di modifica della continuità manca della freccia superiore e quindi non permette di incrementare la continuità nel breakpoint. Analogamente, la figura 5.4a mostra come non sia possibile diminuire il grado del primo segmento; questo perché tale modifica non consentirebbe al break-point di figura 5.4b di mantenere l'ordine di continuità C^2. Infine, il limite inferiore per la continuità in un breakpoint è C^0 per impedire che la curva venga divisa in due parti mentre, come detto in precedenza, il grado di un segmento non può essere inferiore a uno.

5.4 Esempio di modellazione

In questo paragrafo viene mostrato un esempio di modellazione di una curva MD-spline all'interno del Mini-System. In particolare, dal momento che una possibile applicazione delle MD-spline è la descrizione dei font, vengono descritti i passi che portano a disegnare il contorno della lettera U del font *Times New Roman Bold*, riportata in figura 5.5. Lo scopo è dimostrare come l'utilizzo di curve MD-spline permetta, a partire da zero, di modellare una determinata forma, in modo semplice e veloce. Di seguito vengono de-

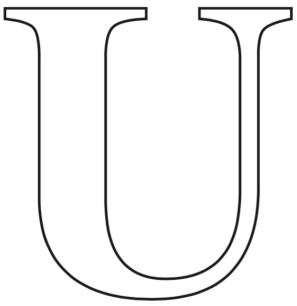


Figura 5.5: Contorno della U del font Times New Roman Bold

scritti i passi di modellazione. Nelle figure i punti di raccordo tra i segmenti della curva vengono identificati mediante un asterisco di colore rosso mentre i punti di controllo con un cerchietto di contorno nero.

1. (*Creazione della poligonale*). A partire dall'area di disegno vuota si crea la curva MD-spline mediante poligonale (segmenti di grado 1 e continuità C^0) tracciando, in maniera grossolana, il contorno della lettera U, come mostrato in figura 5.6. Come si può vedere lo spazio spline definito dalla poligonale genera breakpoint coincidenti con i punti di controllo.

2. (*Diamo forma all'asta curva della U*). Eleviamo, attraverso l'apposito riquadro interattivo, i gradi dei due segmenti orizzontali in basso nella poligonale fino a portarli, entrambi, a grado 3, come mostrato in

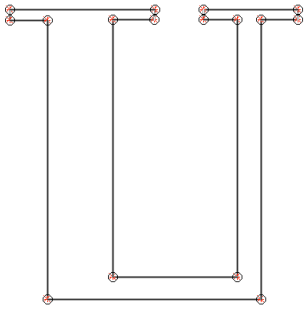
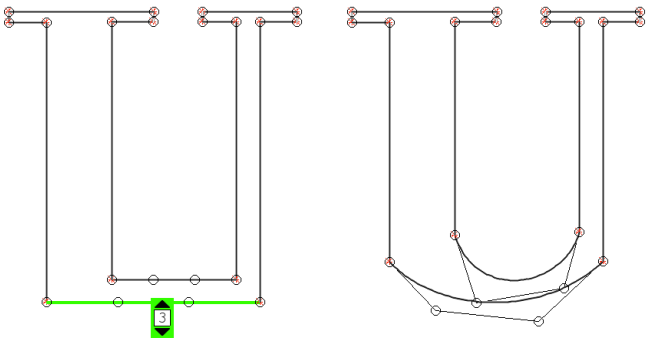


Figura 5.6: Passo 1. Creazione della poligonale

figura 5.7a. Dopodiché, con l'apposita voce *Modify CP* del menu *Curve*, spostiamo i nuovi punti di controllo fino a raggiungere il risultato mostrato in figura 5.7b.



(a) prima *(b)* dopo

Figura 5.7: Passo 2. Diamo forma all'asta curva

3. (*Creo i tratti lineari delle aste verticali*). Come si vede confrontando la situazione attuale (fig. 5.7b) e il contorno che si vuole ottenere (fig. 5.5), è evidente che i segmenti lineari che costituiscono le aste verticali di quest'ultimo hanno una minore estensione. Andiamo quindi ad attivare, attraverso il menu *MD-Curve*, la funzionalità di inserimento interattivo di nodi (*Interact. Knot Ins.*) ed aggiungiamo, sulle aste verticali della U, gli otto breakpoint che vanno ad identificare le nuove posizioni iniziale e finale dei segmenti retti. Il risultato dell'inserimento

è mostrato in figura 5.8, in cui i nuovi breakpoint sono identificati da un cerchio blu e raccordano i segmenti adiacenti, di grado 1, con continuità C^0. Si noti che l'inserimento dei nuovi nodi non ha modificato la forma della curva. Per ora abbiamo solo fissato i limiti di tali segmenti retti, nel prossimo passo andremo a ridurli all'estensione fissata.

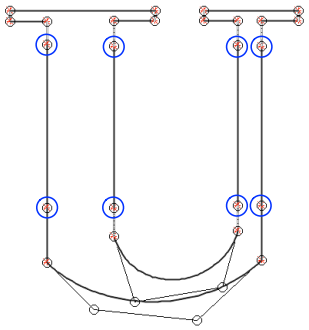
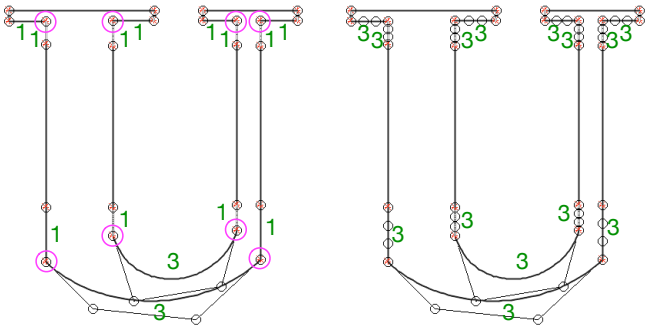


Figura 5.8: Passo 3. Creo i tratti lineari delle aste verticali

4. (*Addolcisco la forma della curva*) Come si vede dal confronto con la figura 5.5, la forma della U, in corrispondenza degli 8 punti di raccordo cerchiati di viola in figura 5.9a, risulta troppo spigolosa e va addolcita.



(a) Prima *(b)* Dopo

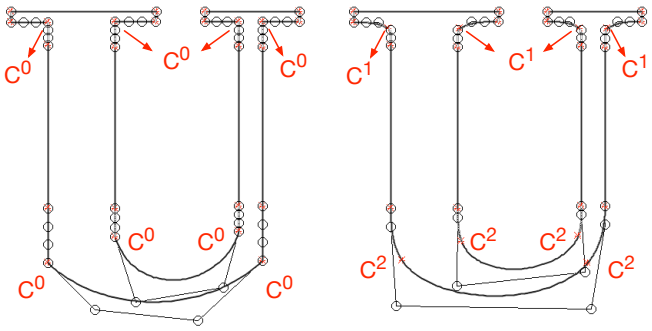
Figura 5.9: Preparo la curva per essere ammorbidita

Questo può essere fatto aumentando l'ordine di continuità dei corrispondenti breakpoint che, al momento, è C^0. In particolare vogliamo

avere, in corrispondenza dei 4 breakpoint in alto, continuità C^1 e, in corrispondenza dei 4 in basso, continuità C^2.

Come mostrato in figura 5.9a, i quattro breakpoint superiori sono tra due segmenti di grado 1 ed i quattro inferiori sono tra un segmento di grado 3 ed un segmento di grado 1. Questo significa che, per la proprietà P.4.6 (continuità di una curva MD-spline nei punti di raccordo), si ha che il massimo ordine di continuità ottenibile nei primi è C^0 e nei secondi è C^1. Pertanto, per poter impostare le continuità desiderate, è necessario, prima di tutto, aumentare il grado di tali segmenti.

In particolare, elevando i segmenti di grado uno, indicati in figura 5.9a, fino a grado tre, si ottiene il risultato mostrato in figura 5.9b. Si noti l'incremento nel numero di punti di controllo in corrispondenza dei tratti che sono stati elevati di grado. I breakpoint cerchiati in viola in figura 5.9a, sono ora circondati da segmenti di grado 3 e pertanto, in tali punti di raccordo, l'ordine di continuità potrà arrivare a C^2.



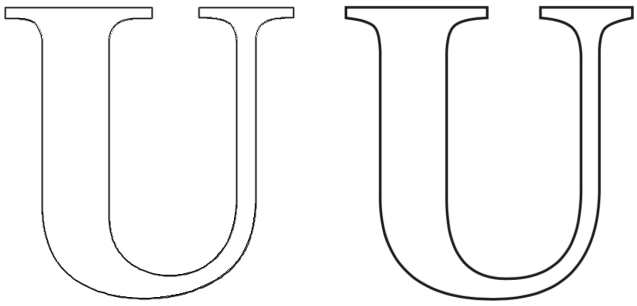
(a) prima *(b)* dopo

Figura 5.10: Passo 4. Addolcisco la forma della curva

A questo punto, per ognuno degli otto breakpoint considerati è possibile, tramite il riquadro che appare cliccando sul breakpoint stesso, incrementarne l'ordine di continuità fino a quello desiderato (al massimo C^2). In figura 5.10a viene mostrata la situazione delle continuità nei breakpoint prima di eseguire l'incremento ed in figura 5.10b viene

mostrata quella successiva. In entrambe le figure, in prossimità di ogni breakpoint coinvolto nella modifica, è riportata la corrispondente continuità. Si noti la conseguente riduzione del numero dei punti di controllo dovuta al fatto che l'incremento della continuità di un breakpoint altro non è che un knot-removal di uno dei suoi nodi associati.

Dopo alcuni aggiustamenti alle posizioni dei punti di controllo il risultato finale ottenuto è mostrato in figura 5.11 in cui viene riportato, per favorire il confronto, anche il contorno della lettera U che si voleva replicare.

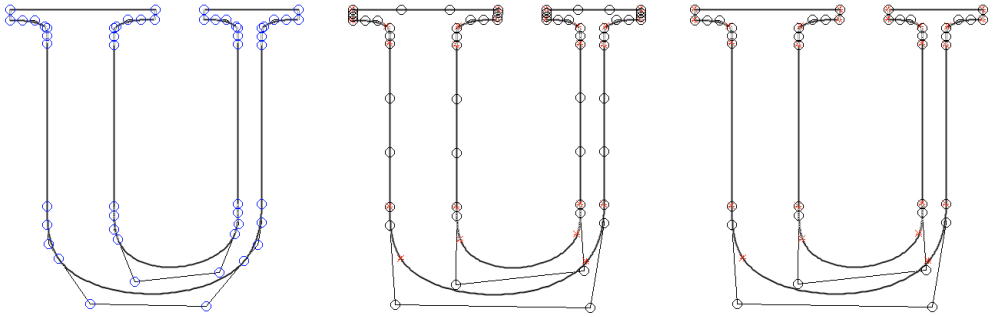


(a) Mini-System *(b)* Font outline

Figura 5.11: Risultato dei 4 passi di modellazione

In figura 5.12, viene fatto un confronto della stessa curva espressa come curva di Beziér a tratti, come curva B-Spline di grado 3 e come curva MD-spline. Come già detto, tutte e tre tre le rappresentazioni sono gestite all'interno del Mini-System. In particolare la curva di Bézier mostrata in figura è stata ricavata dalla curva MD-spline attraverso l'algoritmo di conversione discusso in 4.3.4.

Dalla figura è evidente che la rappresentazione MD-spline permette di descrivere la stessa forma con un minor numero di informazioni, per la precisione 40 punti di controllo e 24 segmenti. La curva di Bézier, infatti, cerca di sopperire all'assenza di vincoli sulla regolarità della curva mediante l'inserimento di punti di controllo aggiuntivi laddove i breakpoint hanno una continuità maggiore di C^0, per un totale di 56 punti di controllo. La curva B-spline, invece, utilizza, per ogni tratto, lo stesso numero di punti di

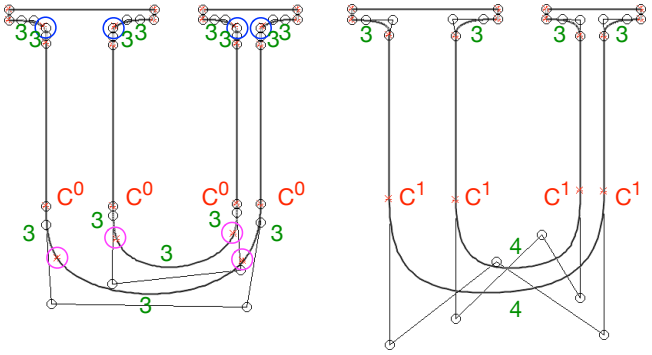


(a) Bézier *(b)* B-spline *(c)* MD-spline

Figura 5.12: Confronto tra rappresentazioni di curva

controllo anche laddove, come nei tratti lineari, questo introduce un numero superfluo di informazioni. Quest'ultima è composta da 60 punti di controllo e 24 segmenti.

Per concludere, va sottolineato che i passi descritti in quest'esempio sono stati resi volutamente più "complessi" per poter dare un assaggio delle funzionalità del sistema e dunque non costituiscono il modo più efficiente di modellare la curva. La modellazione può essere infatti resa più rapida, ad esempio, utilizzando segmenti polinomiali di grado più alto e/o rimuovendo i breakpoint che danno un contributo superfluo alla definizione della forma finale. A dimostrazione di questo, si veda la figura 5.13, in cui viene mostrato come la curva MD-spline ottenuta all'ultimo passo di modellazione possa essere ulteriormente ottimizzata. Infatti elevando di grado, da 3 a 4, i segmenti centrali dell'asta curva della U è possibile modellare la stessa forma anche in assenza dei 4 punti di raccordo cerchiati di viola, che possono pertanto essere rimossi con lo strumento *Interactive Knot Removal*. Inoltre, se si rimuovono anche i punti di controllo cerchiati in blu e si incrementa la continuità, da C^0 a C^1, nei breakpoint alla base delle aste verticali della U, allora è possibile ridurre ulteriormente il numero di informazioni necessarie a descrivere la curva. Tali ottimizzazioni ci permettono infatti di rappresentare la stessa curva con 26 punti di controllo e 16 segmenti, invece che con i 40 punti di controllo e 24 segmenti della rappresentazione non ottimizzata.



(a) prima *(b)* dopo

Figura 5.13: Ottimizzo la rappresentazione della curva

In questo paragrafo abbiamo quindi potuto apprezzare come le curve MD-spline permettano di modellare forme in maniera semplice e rapida, fornendo al progettista un'arma in più, ovvero la possibilità di agire all'occorrenza sui gradi dei singoli segmenti.

Conclusioni

In questa tesi sono stati descritti aspetti teorici, vantaggi ed algoritmi di modellazione delle curve MD-spline ed è stata sviluppata una loro implementazione software interattiva. Le curve MD-spline sono curve spline formate da segmenti polinomiali di gradi differenti. Estendono le tradizionali curve B-spline eriditandone le importanti proprietà di modellazione tra cui controllo locale, guscio convesso, continuità nei punti di raccordo e invarianza per trasformazioni affini. A queste aggiungono la possibilità di impostare i gradi dei singoli segmenti della curva in maniera indipendente l'uno dall'altro.

I due più importanti tipi di curve usate in modellazione geometrica, ovvero le curve di Bézier e le curve B-spline, costituiscono un caso particolare delle curve MD-spline e pertanto possono essere gestite da tale rappresentazione.

L'utilizzo delle curve MD-spline dà al progettista CAD la possibilità di modellare forme in maniera più semplice e rapida rispetto alle curve B-spline e ne riduce il numero di informazioni necessarie a definirle.

Il lavoro di tesi ha permesso di ottenere un importante risultato: è stato implementato il primo software di modellazione interattiva di curve MD-spline.

Si noti che l'intera trattazione, così come l'implementazione all'interno del Mini-System, è stata limitata a curve MD-spline piane, ma che tutto si può facilmente estendere a curve nello spazio tre-dimensionale. In modo ugualmente semplice si possono considerare superfici MD-spline prodotto tensoriale ottenendo un'estensione delle classiche superfici spline che presentano delle interessanti e aggiuntive proprietà di modellazione.

Bibliografia

[1] Serena Morigi. Dispensa del corso "Analisi Numerica e Modellazione Geometrica". CdS Design del prodotto industriale, A.A. 2014/2015.

[2] Brian A. Barsky e Tony D. DeRose. Geometric continuity of parametric curves: Three equivalent characterizations. *IEEE Comput. Graph. Appl.*, 9(6):60–68, Novembre 1989.

[3] B. Buchwald e G. Mühlbach. Construction of B-splines for generalized spline spaces generated from local ECT-systems. *Journal of Computational and Applied Mathematics*, 159(2):249 – 267, 2003.

[4] Wanqiang Shen e Guozhao Wang. A basis of multi-degree splines. *Comput. Aided Geom. Des.*, 27(1):23–35, Gennaio 2010.

[5] Wanqiang Shen e Guozhao Wang. Changeable degree spline basis functions. *J. Comput. Appl. Math.*, 234(8):2516–2529, Agosto 2010.

[6] Wanqiang Shen, Guozhao Wang, e Ping Yin. Explicit representations of changeable degree spline basis functions. *J. Comput. Appl. Math.*, 238:39–50, Gennaio 2013.

[7] Imre Juhász e Ágoston Róth. A class of generalized B-spline curves. *Comput. Aided Geom. Des.*, 30(1):85–115, Gennaio 2013.

[8] C. V. Beccari, G. Casciola, e S. Morigi. *On multi-degree splines.* sottomesso per la pubblicazione, 2016.

[9] Xin Li, Zhang-Jin Huang, e Zhao Liu. A geometric approach for multi-degree spline. *Journal of Computer Science and Technology*, 27(4):841–850, 2012.

[10] Wanqiang Shen, Ping Yin, e Chengjie Tan. Degree elevation of changeable degree spline. *J. Comput. Appl. Math.*, 300(C):56–67, Luglio 2016.

[11] William J. Gordon e Richard F. Riesenfeld. B-spline curves and surfaces. *Computer aided geometric design*, 167:95, 1974.

[12] Les Piegl e Wayne Tiller. *The NURBS Book (2Nd Ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[13] Thomas W. Sederberg, Jianmin Zheng, e Xiaowen Song. Knot intervals and multi-degree splines. *Comput. Aided Geom. Des.*, 20(7):455–468, Ottobre 2003.

[14] PawełWony. Construction of dual B-spline functions. *J. Comput. Appl. Math.*, 260:301–311, Aprile 2014.