Alma Mater Studiorum · Università di Bologna

**SCUOLA DI SCIENZE**
**Corso di Laurea in Informatica**

# Progettazione e Sviluppo di una Web App per curve MD-spline

**Relatore:**
**Chiar.mo Prof.**
**Giulio Casciola**

**Presentata da:**
**Davide Tarquini**

**Sessione I**
**2016/2017**

*Alla mia famiglia*
*e amici...*

# Introduzione

La modellazione geometrica di curve e superfici è una branca molto importante della matematica computazionale, le cui applicazioni spaziano dalla progettazione industriale al semplice disegno artistico.

I sistemi CAD (Computer Aided Design) sono i sistemi software che implementano questa matematica e assolvono ogni aspetto della progettazione di modelli digitali mediante, curve e superfici spline polinomiali in coordinate omogenee, che sono a tutt'oggi lo standard di fatto per la progettazione a "forma libera".

Per realizzare forme complesse si è soliti progettare separatamente le parti di grado basso da quelle di grado alto, cioè il modello che ne risulta non è unico; nel caso di oggetti solidi si dice che il modello non è watertight e questo ha come conseguenza che, nel modificare la forma dell'oggetto, si creeranno degli "strappi". In realtà ogni parte di grado basso si può rappresentare con un grado maggiore e quindi avere un unico modello, tuttavia questo comporta più punti di controllo e una modifica di forma altererà le parti iniziali.

Un approccio migliore è considerare una nuova classe spline che permetta gradi differenti per ogni parte o tratto; queste sono chiamate Multi Degree Spline (MD-Spline), sono in grado di risolvere gli inconvenienti sopra descritti e contengono le spline come caso particolare, ma non sono ancora popolari.

In questo lavoro di tesi ci siamo occupati di questa nuova classe di curve ed in particolare di mettere a disposizione di un utente la possibilità di provare e sperimentare le potenzialità delle MD-spline, mediante una semplice ma potente applicazione web. In particolare ci siamo soffermati sulle curve MD-

Spline per il disegno artistico, un settore applicativo in cui la richiesta di regolarità delle forme progettate è limitato alla continuità visiva, ossia alla continuità $G^1$ o della tangente. Per questo particolare tipo di applicazione è sufficiente utilizzare la classe delle MD-Spline con continuità $C^1$, per le quali sono noti algoritmi di calcolo molto efficienti e specifici rispetto al caso generale.

L'applicazione web, realizzata usando la tecnologia HTML5 e canvas, mette a disposizione un ambiente interattivo in cui l'utente può sperimentare le curve MD-Spline applicando tutti i classici tool di modellazione e apprezzarne le notevoli potenzialità.

# Indice

# Elenco delle figure

# Capitolo 1

# Funzioni MD-Spline

Le componenti $(C_x(t), C_y(t))$ di una curva 2D di Bézier sono polinomi definiti nell'intervallo parametrico $[a, b]$. Tuttavia curve definite da un unico tratto polinomiale sono spesso inadeguate per rappresentare una forma geometrica complessa. I principali svantaggi sono:

- 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 Bézier 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 partizionare l'intervallo $[a, b]$ in sottointervalli e su ciascuno definire un polinomio di grado basso, 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) che estendono le tradizionali funzioni spline. 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. Nel seguito tutte le definizioni e proprietà verranno date nel caso generale delle MD-spline, mentre gli algoritmi di calcolo nel caso particolare di $C^1$ MD-spline.

## 1.1 Spazi polinomiali a tratti

**Definizione 1.1.1.** *[**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* $\Delta$*, 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* $P_{n_i}$ *definito sull'intervallo* $I_i$*. Definiamo lo spazio MD (Multi-Degree) dei polinomi a tratti come:*

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

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*.

## 1.2 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 1.2.1.** *[**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 1.1.1. Sia, poi,* $\mathcal{K} := (k_1, \ldots, k_q)$ *un vettore di interi positivi, detto vettore delle continuità, tale che:*

$$0 \leq k_i \leq \begin{cases} min(n_{i-1}, n_i) & se \quad n_{i-1} \neq n_i \\ n_i - 1 & se \quad n_{i-1} = n_i \end{cases} \qquad per\ ogni\ i = 1, \ldots, q \tag{1.1}$$

*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:*

$$\begin{aligned} S(P_{\mathcal{N}}, \mathcal{K}, \Delta) := \{f \mid & \text{esiste } p_i \in P_{n_i},\ i = 0, \ldots, q, \text{ tale che:} \\ & \text{i) } f(x) = p_i(x) \text{ per } x \in I_i, i = 0, \ldots, q; \\ & \text{ii) } p_{i-1}^{(\ell)}(x_i) = p_i^{(\ell)}(x_i) \text{ per } \ell = 0, \ldots, k_i, \\ & i = 1, \ldots, q\}. \end{aligned} \tag{1.2}$$

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

**Osservazione 1.2.1.** *Lo spazio* $S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$ *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 condizioni sulla continuità 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 1.2.1.** *L'insieme delle spline multi-degree $S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$ 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 $S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$, 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 $S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$.

Lo spazio delle funzioni MD-spline possiede un'opportuna base B-Spline, stabile dal punto di vista computazionale. Le funzioni base B-spline per questo spazio sono definite da una formula ricorrente integrale molto elegante, ma estremamente complessa per il loro calcolo. In [9] si è mostrato che limitando la continuità nei break-point che separano tratti con gradi differenti ad essere al più $C^1$ le funzioni B-spline possono essere definite e calcolate in modo efficiente tramite una formula ricorrente non integrale molto simile a quella nota per le B-spline mono-degree.

Nel seguito di questa trattazione ci limiteremo a questa particolare sottoclasse delle spline Multi-Degree, ossia considereremo MD-spline con continuità $C^{k_i}$ nei break-points con $k_i$ tale che:

$$0 \leq k_i \leq \begin{cases} 1, & se \quad n_{i-1} \neq n_i, \\ n_i - 1, & se \quad n_{i-1} = n_i. \end{cases}$$

Per semplicità chiameremo tale sotto classe $C^1$ MD-spline.

## 1.3 Funzioni B-spline $C^1$ 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 break-point $x_i$, sia ripetuto precisamente $(n_i - k_i)$ volte in $\Delta_t^*$ e $(n_{i-1} - k_i)$ volte in $\Delta_s^*$.

**Definizione 1.3.1.** *[**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:*

*[label=()]*

1. $t_1 \leq t_2 \leq \cdots \leq t_K$
2. $t_{n_0+1} \equiv a$
3. $\{t_{n_0+2}, \ldots, t_{n_0+K_t+1}\} \equiv \{\underbrace{x_1, \ldots, x_1}_{n_1-k_1 \text{ volte}}, \ldots, \underbrace{x_q, \ldots, x_q}_{n_q-k_q \text{ volte}}\}$

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

*[label=()]*

1. $s_1 \leq s_2 \leq \cdots \leq s_K$
2. $s_{K_s+1} \equiv b$
3. $\{s_1, \ldots, s_{K_s}\} \equiv \{\underbrace{x_1, \ldots, x_1}_{n_0-k_1 \text{ volte}}, \ldots, \underbrace{x_q, \ldots, x_q}_{n_{q-1}-k_q \text{ volte}}\}$

Per semplicità e senza perdere in generalità, limiteremo la nostra discussione al caso di partizioni *clamped*, cioè partizioni estese con i due break-point estremi ripetuti $n_0+1$ volte in $^*_t$, e $n_q+1$ volte in $^*_s$, cioè $t_1 = \ldots = t_{n_0+1} = x_0$ e $s_{K-n_q} = \ldots = s_K = x_{q+1}$.

Nel nostro lavoro abbiamo considerato anche il caso peridico, che per MD-spline risulta estremamente più articolato e complesso del caso spline classico.

**Esempio 1.3.1.** *Se* $\Delta = \{1, 2\}$ *partizione in* $[0, 3]$, $\mathcal{N} = (3, 3, 4)$, $\mathcal{K} = (2, 1)$, *allora* $K_t = 4$, $K = n_0 + K_t + 1 = 3 + 4 + 1 = 8$ *e le partizioni estese sono:*

$$\Delta_t^* = \{0, 0, 0, 0, 1, 2, 2, 2\}$$

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

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*).

**Osservazione 1.3.1.** *Le partizioni estese per il caso Multi-Degree sono costruite in modo tale che alcuni break-point presenti in* $\Delta_t^*$ *possono non essere presenti in* $\Delta_s^*$ *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* $\Delta_t^*$ *ed aggiungiamo alla fine gli ultimi* $n_q + 1$ *elementi di* $\Delta_s^*$ $(s_{K_s+1} \equiv b, \ldots, s_K)$ *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 che sia al più $C^1$ fra tratti di gradi differenti.

**Definizione 1.3.2.** *[**Funzioni B-spline** $C^1$ **Multi-Degree**]*

*Consideriamo le partizioni estese* $\Delta_t^*$ *e* $\Delta_s^*$ *associate allo spazio* $S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$ *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.*

*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:*

$$N_{\ell,m-n_j}(x) = \begin{cases} 1 & x_j \le x < x_{j+1} \quad t_\ell \le x_j \le \min(t_{\ell+1}, b) \\ 0 & \textit{altrimenti} \end{cases}$$

$$N_{i,n}(x) = \phi_i^{n-1}(x)N_{i,n-1}(x)+(1-\phi_{i+1}^{n-1}(x))N_{i+1,n-1}(x), \quad n = m-n_j+1,\dots,m, \tag{1.3}$$

*con*

$$\phi_i^{n-1}(x) = \frac{x - t_i}{s_{i+m-n-1} - t_i} \quad n \leq m-1$$

*e*

$$\phi_i^{m-1}(x) = \frac{\tilde{x} - \tilde{t}_i}{\tilde{s}_{i-1} - \tilde{t}_i};$$

*dove* $\tilde{\Delta} = \{\tilde{x}_i\}_{i=1}^{q}$ *è la partizione scalata e traslata ottenuta da* $\Delta$ *nel seguente modo:*

$$\tilde{x}_j = \sum_{i=1}^{j} \frac{x_i - x_{i-1}}{n_{i-1}}, \qquad j = 1,\dots,q+1 \quad con \quad \tilde{x}_0 = 0, \tag{1.4}$$

*e se* $x \in [x_j, x_{j+1})$ *il suo traslato e scalato è dato da:*

$$\tilde{x} = \tilde{x}_j + \frac{x - x_j}{n_j}. \tag{1.5}$$

*Le partizioni estese* $\tilde{\Delta}_s^* = \{\tilde{s}_j\}_{j=1}^{K}$ *e* $\tilde{\Delta}_t^* = \{\tilde{t}_j\}_{j=1}^{K}$ *sono ottenute da* $\tilde{\Delta}$ *allo stesso modo di come* $\Delta_s^* = \{s_j\}_{j=1}^{K}$ *e* $\Delta_t^* = \{t_j\}_{j=1}^{K}$ *sono ottenute da* $\Delta$.

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

**Osservazione 1.3.2.** *La definizione appena data è valida esclusivamente per* $C^1$ *MD-spline; se lo spazio MD-spline di cui si vuole calcolare la base B-spline richiedesse una continuità maggiore di* $C^1$ *fra tratti di gradi differenti, lo schema ricorrente non le definirebbe in modo corretto.*

**Osservazione 1.3.3.** *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 1.3.4.** *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 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 1.1 vengono mostrate le funzioni base B-spline $C^1$ multi-degree $N_i$ associate allo spazio $S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$ dell'esempio 1.3.1 (rifare FIGURA).

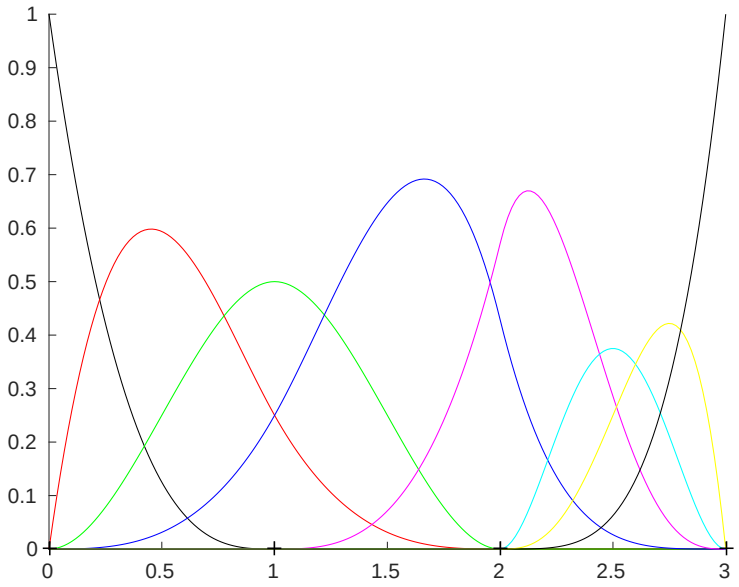



Figura 1.1: Funzioni base B-spline Multi-Degree relative allo spazio dell'esempio 1.3.1

### 1.3.1 Proprietà delle B-spline Multi-Degree

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

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 $N_i$.

2. (*Non Negatività*)
$$N_i(x) > 0 \quad \forall\, x \in (t_i, s_i)$$

3. (*Partizione dell'unità*)

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

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:

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

   dove $p^{t_i}$ è l'indice del break-point associato al nodo $t_i$ e analogamente $p^{s_i}$ è l'indice del break-point associato al nodo $s_i$.

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 $N_{\ell-n_j}, \ldots, N_\ell$.

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 1.3.5.** *Dalla proprietà 1, ogni nodo di $\Delta_t^*$ e $\Delta_s^*$ è, rispettivamente, l'estremo iniziale e finale di una funzione base B-spline.*

**Osservazione 1.3.6.** *La proprietà 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:*

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

*dove $\mu_i^t = \max\{j \geq 0 \,|\, t_i = t_{i+j}\} + 1$ e $\mu_i^s = \max\{j \geq 0 \,|\, s_{i-j} = s_i\} + 1$ e $p^{t_i}$ e $p^{s_i}$ come detto nella proprietà 4.*

## 1.4 Funzioni $C^1$ MD-Spline

In questo paragrafo vengono definite le funzioni $C^1$ MD-spline, che costituiscono le funzioni coordinate delle curve $C^1$ MD-spline.

**Definizione 1.4.1.** *[**Funzioni MD-Spline**] Chiamiamo $C^1$ MD-Spline le funzioni dello spazio $S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$ rappresentate nella base delle B-spline $C^1$ Multi-Degree, cioè:*

$$f(x) = \sum_{i=1}^{K} c_i \, N_i(x), \quad x \in [a, b] \tag{1.6}$$

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

**Osservazione 1.4.1.** *Come si può vedere, ad ogni coefficiente $c_i$ è associata una ben precisa funzione base $N_i$.*

Le proprietà delle funzioni $C^1$ MD-spline sono una diretta conseguenza delle proprietà delle B-spline $C^1$ Multi-Degree. In particolare:

1. Per la proprietà 5, risulta che:

$$f(x) = \sum_{i=\ell-n_j}^{\ell} c_i \, N_i(x), \quad x \in [x_j, x_{j+1}], \quad t_\ell \leq x_j < \min(t_{\ell+1}, b); \tag{1.7}$$

2. Per le proprietà 2 e 3, una funzione $C^1$ 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 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 1 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 dell'intervallo $[t_i, s_i]$.

## 1.5 Valutazione delle funzioni $C^1$ 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:

$$f(\overline{x}) = \sum_{i=\ell-n_j}^{\ell} c_i \, N_i(\overline{x}) \tag{1.8}$$

Per valutare quest'ultima espressione, è possibile usare una variante dell'algoritmo di *de Boor*, ben noto per la valutazione delle spline. In alternativa si può considerare di calcolare le $N_i(x)$ attraverso la formulazione ricorrente della definizione 1.3.2 e poi valutarne la combinazione lineare con i $c_i$.

### 1.5.1 Esempio numerico di valutazione

Riprendiamo l'esempio 1.3.1. Sia, cioè, $\Delta = \{1, 2\}$ partizione in $[0, 3]$, $\mathcal{N} = (3, 3, 4)$, $\mathcal{K} = (2, 1)$ e $K = 8$. Le partizioni estese associate sono $\Delta_t^* = \{0, 0, 0, 0, 1, 2, 2, 2\}$ e $\Delta_s^* = \{1, 2, 2, 3, 3, 3, 3, 3\}$.

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 $\ell$ tale che $t_\ell \leq x_j < t_{\ell+1}$. Nel nostro caso $\ell$ vale 5.

Utilizzando la formula ricorrente sulle B-spline determiniamo i valori delle $\{N_{i,4}(\overline{x})\}_{i=2}^5$ mediante il seguente schema triangolare; determinati $j = 1$, $\ell = 5$, $n_j = 3$ ed $m = 4$, resta individuata la $N_{\ell,m-n_j}(x)$ nella $N_{5,1}(x)$ che varrà 1 in $[x_j, x_{j+1}] = [x_1, x_2]$ e 0 negli altri intervalli. Ora, per $n = m - n_j, \ldots, m = 2, 3, 4$ si determinano, usando la (**??**), le restanti $N_{i,m}$ dello schema in $\overline{x}$.

$$\begin{array}{cccccc} 0 & 0 & 0 & 0 & N_{5,1}(\overline{x}) & 0 \\ 0 & 0 & 0 & N_{4,2}(\overline{x}) & N_{5,2}(\overline{x}) & 0 \\ 0 & 0 & N_{3,3}(\overline{x}) & N_{4,3}(\overline{x}) & N_{5,3}(\overline{x}) & 0 \\ 0 & N_{2,4}(\overline{x}) & N_{3,5}(\overline{x}) & N_{4,4}(\overline{x}) & N_{5,4}(\overline{x}) & 0 \end{array}$$

Dopo aver determinato tutti i valori $\{N_{i,4}(\overline{x})\}_{i=2}^5$ si può calcolare la

$$f(\overline{x}) = \sum_{i=2}^{5} c_i \, N_i(\overline{x}).$$

## 1.6 Algoritmi geometrici per MD-Spline

In questo paragrafo vengono definiti formalmente gli algoritmi di knot-insertion/removal e degree elevation/reduction applicati a funzioni $C^1$ MD-spline. Questi, estendono gli omonimi algoritmi su spline, ampiamente discussi in letteratura, prendendo in considerazione la possibilità di avere tratti polinomiali di gradi differenti con continuità al più $C^1$.

### 1.6.1 Knot insertion

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

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

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

Il termine *knot insertion* si riferisce al procedimento usato per determinare la rappresentazione di una funzione MD-spline $f(x) \in S(P_{\mathcal{N}}, \mathcal{K}, \Delta)$, 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 corrisponde ad un cambio della base di rappresentazione; la funzione MD-spline non viene modificata né geometricamente né parametricamente.

**Teorema 1.6.1.** *(Knot-insertion) Siano $\Delta_t^*$, $\Delta_s^*$ 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:*

$$N_i(x) = \alpha_i \hat{N}_i(x) + (1 - \alpha_{i+1})\hat{N}_{i+1}(x), \qquad \begin{array}{l} i = 1, \ldots, K \\ x \in \mathbb{R}, \end{array} \tag{1.9}$$

*con i coefficienti $\alpha_i$ dati da:*

$$\alpha_i = \begin{cases} 1, & i \leq \ell - d_j \quad con \quad x_j \leq \hat{t} < x_{j+1} \\ \dfrac{\hat{s} - \tilde{s}_i}{\tilde{t}_i - \tilde{s}_i}, & \ell - d_j + 1 \leq i \leq \ell - r + 1 \\ 0, & i \geq \ell - r + 2 \end{cases} \tag{1.10}$$

*con* $1 \leq r < n_j$ *dove* $n_j - r$ *rappresenta l'ordine di continuità richiesto nel break-point associato a* $\hat{t}$ *nel nuovo spazio* $S(\mathcal{P}_{\hat{\mathcal{N}}}, \hat{\mathcal{K}}, \hat{\Delta})$*, mentre* $\{\tilde{s}_j\}$ *e* $\{\tilde{t}_j\}$ *nella* (1.10) *si riferiscono alle partizioni sinistra e destra scalate e traslate costruite a partire da* $\hat{\Delta}_s^*$ *e* $\hat{\Delta}_t^*$*.*

**Osservazione 1.6.1.** *Dalle espressioni* (1.9) *e* (1.10)*, risulta che:*

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

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

$$f(x) = \sum_{i=1}^{K+1} \hat{c}_i \, \hat{N}_i(x) \tag{1.11}$$

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

**Corollario 1.6.1.** *Siano* $\Delta_t^*$*,* $\Delta_s^*$ *e* $\hat{\Delta}_t^*$*,* $\hat{\Delta}_s^*$ *le partizioni nodali già definite, allora:*

$$\hat{c}_i = \begin{cases} c_i & i \leq \ell - n_j \\ \alpha_i c_i + (1 - \alpha_i) c_{i-1} & \ell - n_j + 1 \leq i \leq \ell - r + 1 \\ c_{i-1} & i \geq \ell - r + 2 \end{cases} \tag{1.12}$$

*dove gli* $\alpha_i$ *sono stati definiti in* (1.10)*.*

Infine, è importante ripetere che, se il nodo da inserire è coincidente con uno dei break-point, 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 minore della precedente continuità $k_j$.

### 1.6.2 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 ossia in uno spazio di dimensione inferiore ed in generale di continuità maggiore. Tale rappresentazione può essere esatta oppure approssimata, a seconda che

la rimozione del nodo, ossia lo spazio di dimensione inferiore, contenga la funzione o meno.

Se si applica un knot-removal di un nodo precedentemente inserito mediante knot-insertion, chiaramente sarà possibile rimuoverlo in modo esatto, ossia senza alterare la funzione.

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, mediante una classica approssimazione ai minimi quadrati.

Si tratta in ogni caso di un problema di approssimazione di una funzione di uno spazio con una funzione di uno spazio di dimensione inferiore. In questo lavoro per approssimare tale funzione si è fatto ricorso ad una approssimazione ai minimi quadrati realizzata mediante la base duale dello spazio in cui si cerca l'approssimazione; per il calcolo della base duale si è utilizzato il recente risultato presentato in [8].

### 1.6.3 Degree elevation

L'algoritmo di elevamento di grado classico su spline permette di incrementare di uno il grado di tutti i tratti polinomiali della funzione. Nel caso Multi-Degree, invece, è possibile 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.

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

$$f(x) = \sum_{i=1}^{K+1} \hat{c}_i \, \hat{N}_{i,\hat{m}}(x) \tag{1.13}$$

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

**Corollario 1.6.2.** *Siano $\Delta_t^*$, $\Delta_s^*$ e $\hat{\Delta}_t^*$, $\hat{\Delta}_s^*$ le partizioni nodali già definite, allora:*

$$\hat{c}_i = \begin{cases} c_i & i \leq \ell - n_j \\ \alpha_i c_i + (1-\alpha_i) c_{i-1} & \ell - n_j + 1 \leq i \leq \ell \\ c_{i-1} & i \geq \ell + 1 \end{cases} \tag{1.14}$$

Si osservi che elevando di un grado un tratto, si potrà incorrere nella situazione di avere intervalli adiacenti di gradi differenti ed in questo caso la continuità dovrà essere al max $C^1$; prima quindi di procedere a tale operazione, si dovrà verificare se si è in questa situazione e nel caso procedere ad un opportuno knot insertion per ridurre la continuità a $C^1$.

Si possono progettare differenti algoritmi di degree elevation; avendo un algoritmo efficiente di knot-insertion seguiremo l'approccio standard proposto da Piegl e Tiller che consiste nel fare un knot insertion multiplo negli estremi dell'intervallo interessato, al fine di ridurre la base B-spline sul tratto considerato ad essere una base polinomiale di Bernstein, quindi nell'applicare la formula esplicita per il degree elevation di un polinomio di grado $n_j$, ed infine rimuovere i nodi precedentemente inseriti fino a ritornare alla continuità iniziale $C^1$.

### 1.6.4 Degree reduction

l *Degree reduction* è l'operazione inversa del degree elevation. Se si applica un degree reduction su un intervallo precedentemente elevato di grado sarà possibile riottenere la funzione iniziale; in caso contrario si tratterà di una approssimazione determinabile mediante un algoritmo di minimi quadrati esattamente come detto nella sezione sul knot-removal. Anche in questo caso faremo uso della base duale per il cui calcolo usiamo l'algoritmo proposto in [8].

# Capitolo 2

# HTML5 e Canvas

## 2.1 Introduzione

HTML5 è un linguaggio di markup progettato per la strutturazione di pagine web, pubblicato dall'W3C nel 2014, ormai diventato uno standard per tutti i browser.

Ciò comporta un enorme vantaggio, ovvero permette di rendere uniforme la visualizzazione di una pagina web attraverso diversi browser (chrome, firefox, edge). In particolare andremo a parlare di un elemento specifico di questo standard, le canvas; queste permettono ad uno sviluppatore di "ritagliare" un'area del browser e attraverso comandi specifici disegnare punti, linee, rettangoli e font.

## 2.2 Canvas

La canvas è un elemento che fu inizialmente introdotto da Apple e permette il rendering dinamico di immagini "bitmap", gestibili attraverso un linguaggio di scripting tipo JavaScript.

Data una regione del browser, si definisce una "pixmap", dov'è possibile creare dinamicamente elementi 2D.

Per poter utilizzare la canvas è necessario dichiararla con il tag $< canvas >< /canvas >$, definendone le dimensioni in pixel. Ciò comporta dei problemi di visualizzazione quando si va a modificare la dimensione della pagina o nel caso la si vada a visualizzare su dispositivi che hanno uno schermo ridotto, come gli smartphone e i tablet.

### 2.2.1 Come lavora

Quando, durante il parsing della pagina, HTML riscontra un elemento canvas (di una determinata larghezza e altezza) alloca una "surface" per coprire quella specifica area.

L'allocazione di questa "pixmap" può essere eseguita in 2 modi:

- manualmente
- attraverso API del SO nativo per creare una surface su cui disegnare.

L'API nativa potrebbe essere Windows, Gtk, Kde, Qt o qualsiasi altra libreria di disegno che l'implementatore del browser ha scelto, questa operazione è fortemente dipendente dal sistema operativo.

**Esempio:** Internet Explorer chiama la libreria nativa di Windows (DirectX).

Creata la surface di disegno, viene resa accessibile all'interno dell'interprete JavaScript, tramite una variabile che ne indica la rappresentazione.

Quindi, durante l'esecuzione di una funzione JavaScript, si chiama uno dei metodi della canvas e questo si trasforma in una chiamata al comando nativo al SO appropriato.

Le chiamate effettuate dalla canvas, sono di tipo software, ovvero il disegno avviene pixel per pixel. Questo, però, non è sempre vero, se il browser è di ultima generazione, è presente una GPU e i driver della scheda video sono adeguati, il disegno sfrutta l'accelerazione hardware.

### 2.2.2 Gestione Eventi

Poichè la canvas è un elemento del linguaggio di markup HTML, intercetta gli eventi come un qualunque altro elemento della pagina. Si possono definire "risposte", ad un determinato evento scatenato dall'utente, attraverso degli "handler" (o "listener").

Possiamo definire un handler come una funzione che viene eseguita in automatico, quando l'evento a cui è associato viene scatenato, come la pressione di un tasto sulla tastiera o il click di un bottone del mouse.

I meccanismi per associare gli handler ad un evento sono:

- via codice, sfruttando la funzione addEventListener();
- sfruttando speciali attributi dei tag HTML;

#### 2.2.2.1 addEventListener()

La funzione addEventListener() è un metodo esposto dagli elementi del "DOM" (Document Object Model) e rappresenta la più comune tra le modalità usate per associare un evento al rispettivo handler. La sua sintassi è molto semplice:

element.addEventListener("nome evento", "funzione da eseguire");

**Esempio:**

canvas.addEventListener("click", alert());

Con questo comando stiamo indicando che ogni volta che si scatenera l'evento "click", sull'elemento canvas verrà eseguita la funzione alert().

#### 2.2.2.2 All'interno dei tag HTML

Possiamo anche inserire la chiamata alla nostra funzione handler direttamente, nella definizione di un tag HTML, utilizzando alcuni attributi speciali tipo "onclick".

**Esempio:**

<canvas onclick="alert()"></canvas>

Questo metodo, però, è sempre meno utilizzato in quanto favorisce troppo l'unione tra HTML (visualizzazione) e JavaScript (logica).

### 2.2.3 Definizione canvas e surface

Come abbiamo detto nella sezione 2.2.1, il tag "canvas" e la rispettiva surface di disegno non sono la stessa cosa, perchè la prima definisce dove, nella pagina HTML, andare ad inserire l'area di disegno e la seconda invece ricopre l'area indicata dalla prima, rendendola disegnabile.

La distinzione diventa evidente quando nel codice JavaScript, che si occupa di gestire la pagina HTML, troviamo queste due istruzioni:

- var canvas = document.getElementById("canvas");
- var ctx = canvas.getContext("2d").

#### 2.2.3.1 document.getElementById("canvas")

Questa funzione appartiene al linguaggio JavaScript nativo ed è la funzione che permette, assegnato un id ad uno specifico tag HTML, di recuperarne l'elemento. La prima parte indica dove viene effettuata questa ricerca, in questo caso è l'intero "document". Una volta recuperato si potranno utilizzare le funzioni specifiche di quel determinato elemento.

#### 2.2.3.2 canvas.getContext("2d")

Questa istruzione si occupa di recuperare il contesto della canvas, a differenza dell'istruzione precedente, che può essere applicata ad ogni tag HTML presente nella pagina, questa funzione è propria dell'elemento canvas ed è l'interfaccia che mette a disposizione le primitive di disegno. In altre parole, qui recuperiamo l'oggetto di cui andremo a chiamare le funzioni di disegno come la "stroke".

### 2.2.4 Area di lavoro

La canvas è l'area di disegno, come detto nella sezione precendente non sono la stessa cosa; volendo si potrebbero definire due dimensioni: una per la canvas e una per l'area di disegno come mostrato nella fig. 2.1.

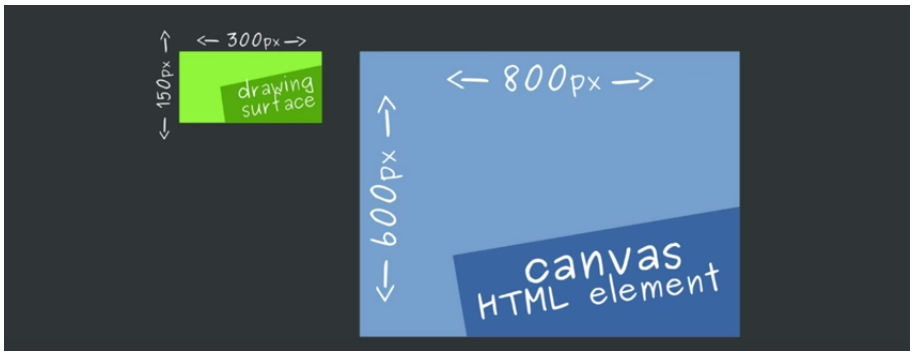



Figura 2.1: Surface Vs Tag Html

Tale operazione però è sconsigliata perche al momento della visualizzazione l'area di disegno (surface o ctx) viene adattata alla grandezza dell'elemento HTML, causando così una distorsione dell'immagine, come possiamo vedere nella fig. 2.2.

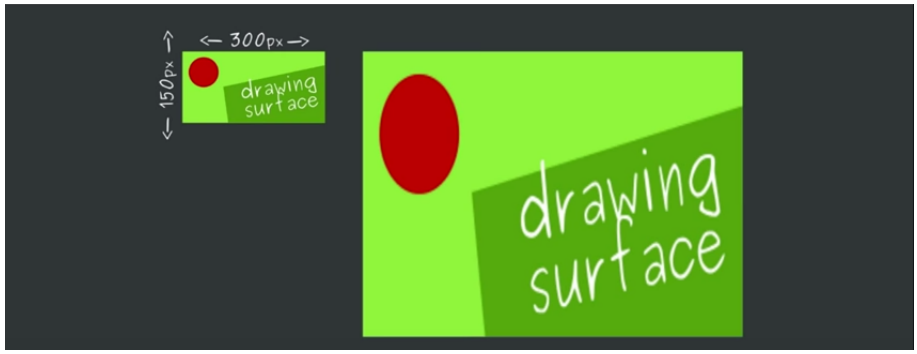



Figura 2.2: Surface

Un'altra cosa che va tenuta in considerazione, quando si utilizzano le canvas, sono gli assi di riferimento: l'asse y è invertito e il punto 0,0 non

si trova al centro dello schermo come siamo abituati ad immaginare, ma partono dall'angolo in alto a sinistra.

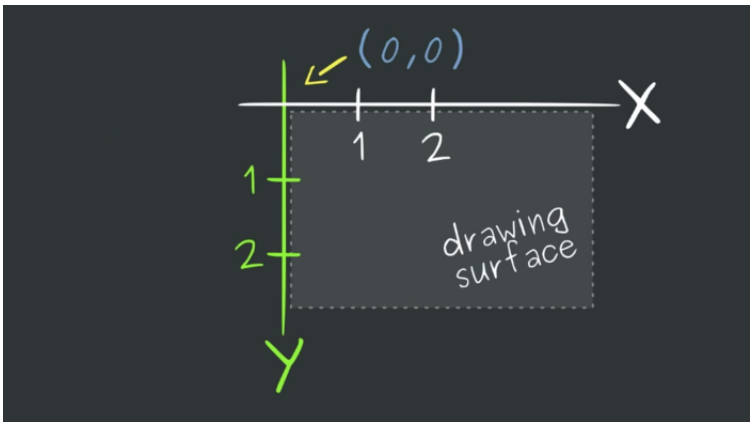



Figura 2.3: Assi canvas

### 2.2.5 Metodi della canvas

Una volta definita la canvas e la sua surface, si possono richiamare, attraverso il codice JavaScript, diversi metodi tra cui:

- arc(): che permette di creare elementi rotondi quindi definendo una posizione e un raggio;
- clearRect(): che permette di cancellare il contenuto della canvas o di una porzione di essa;
- lineTo(): che definisce una serie di punti tra due coordinate x, y;
- fill(): che permette di riempire con un colore ad esempio, un area precedentemente definita;
- stroke(): che permette di andare a disegnare effettivamente l'elemento o il path definito in precedenza.

Vengono esposti molti altri metodi utili, per maggiori informazioni, si visiti il sito:

https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D

### 2.2.6 Scelta della canvas

Uno dei principali motivi per cui si è preferito utilizzare l'elemento canvas è la sua versatilità, ma soprattutto la compatibilità con ogni browser, facendo esso parte dello standard HTML5.

Inizialmente si era pensato di utilizzare la libreria grafica WebGL insieme a Three.js, non tanto per sfruttare la grafica 3D, visto che il nostro progetto è solo 2D, ma per sfruttare l'accelerazione hardware; dopo una breve analisi si è arrivati alla conclusione che la libreria era troppo pesante per quello che si voleva realizzare, inoltre attraverso una rapida ricerca on line si è riscontrato (come illustrato in fig. 2.4 e 2.5) che ad oggi non sono ancora molti i browser che la supportano.

| Feature | Chrome | Edge | Firefox (Gecko) | Internet Explorer | Opera | Safari |
|---|---|---|---|---|---|---|
| Basic support (2d context) | 4 | (Yes) | 3.6 (1.9.2) | 9 | 9 | 3.1 |
| `webgl` context | 9[1] 33 | (Yes) | 3.6 (1.9.2)[1] 24 (24) | 11.0[2] | 9.0[3] | 5.1[2] |
| `webgl2` context | 56 | ? | 25 (25)[4] | No support | No support | No support |
| 2d `alpha` context attribute | 32 | ? | 30 (30) | No support | (Yes) | No support |
| `failIfMajorPerformanceCaveat` attribute | (Yes) | ? | 41 (41) | (Yes) | (Yes) | ? |
| bitmaprenderer context | No support | ? | 46 (46) | No support | No support | No support |

Figura 2.4: Desktop

| Feature | Android | Chrome for Android | Edge | Firefox Mobile (Gecko) | IE Mobile | Opera Mobile | Safari Mobile |
|---|---|---|---|---|---|---|---|
| Basic support (2d context) | (Yes) | (Yes) | (Yes) | 1.0 (1.9.2) | (Yes) | (Yes) | (Yes) |
| `webgl` context | ? | ? | (Yes) | (Yes)[2] | ? | ? | ? |
| `webgl2` context | No support | No support | ? | No support | No support | No support | No support |
| 2d `alpha` context attribute | No support | No support | ? | 30.0 (30) | No support | No support | No support |
| `failIfMajorPerformanceCaveat` attribute | ? | ? | ? | 41.0 (41) | ? | ? | ? |
| bitmaprenderer context | No support | No support | ? | 46.0 (46) | No support | No support | No support |

Figura 2.5: Mobile

# Capitolo 3

# Architettura della Web App

In questo capitolo viene trattata l'architettura generale dell'applicazione.

Brevemente, l'obiettivo era progettare una WebApp che permettesse di sperimentare le potenzialità di modellazione di forma delle curve MD-Spline.

Come detto inizialmente esse sono una generalizzazione delle curve spline standard.

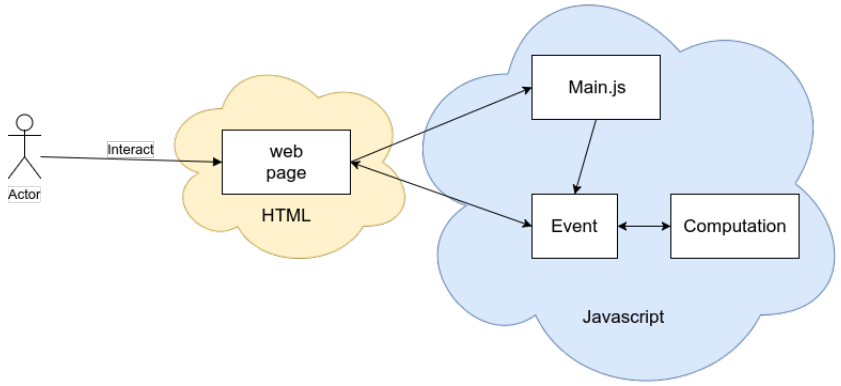



Figura 3.1: Architettura

## 3.1 Web Page

La web page è l'interfaccia che l'utente può utilizzare per accedere alle funzionalità dell'applicazione: da qui si possono attivare tutti i tool di modellazione e di visualizzazione delle informazioni, ed è presente l'elemento canvas, che mostra la curva insieme alle poligonale di controllo, e le relative informazioni.

La pagina utilizza Bootstrap: che rende il sito "responsive", ovvero adattabile ad ogni tipo di schermo (l'unico inconveniente è dato dalla canvas stessa che come abbiamo detto ha dimensioni fisse).

## 3.2 Main.js

Questo script si occupa di gestire tutte le interazioni con l'area di disegno, cioè il recupero dell'elemento HTML della canvas, la sua definizione e i collegamenti ad essa associati; inoltre è presente la gestione del click destro e sinistro del mouse che in base allo stato ("iniziale" o di "modellazione") avranno funzionalità diverse.

### 3.2.1 Binding

- canvas.addEventListener("mouseup", mouseUpFunction);
- canvas.addEventListener("mousedown",mouseDownFunction);
- canvas.addEventListener("mousemove", mouseMoveFunction).

## 3.3 Event

Questa è la sezione che si occupa di intercettare gli eventi, che per lo più si trovano nella funzione appendOption, come: degree elevation, knot insertion, knot removal e degree reduction; non intercetta invece l'evento che riguarda la creazione iniziale della curva.

Alla creazione della curva si attiverà un piccolo pannello di controllo, che in base al click dell'utente si occuperà di eseguire le diverse funzioni presenti negli script di computation: queste servono per generare le informazioni da mostrare all'utente, oltre che per chiamare le funzioni di disegno, una volta recuperati i dati dalla sezione di computation.

## 3.4 Computation

Questa sezione è dedicata a tutti gli algoritmi di calcolo necessari a determinare le coordinate dei punti della curva.

Gli algoritmi usati sono una rielaborazione di quelli sviluppati in un codice matlab e poi in C nel pacchetto MiniSystem, sviluppati presso l'Università di Bologna, per la modellazione di curve polinomiali a tratti; quest'ultimo è stato recentemente ripreseo ed esteso in [10] per gestire curve MD-Spline.

## 3.5 Draw

Questa sezione è formata da due file:

- createpoint.js;
- redraw.js.

Questi sono i file che si occupano di interagire con la canvas attraverso le sue primitive e far apparire le immagini nell'area di disegno.

Le primitive utilizzate sono:

- ctx.arc(): per la creazione di punti;
- ctx.lineTo(): definisce una linea tra due elementi;
- ctx.stroke(): effettiva funzione di disegno.

## 3.6 Progettazione

La Web App si presenta come una normalissima pagina web.
Dalla root del progetto sono presenti le seguenti sottodirectory:

- Js: contenente tutti i file .js che verranno utilizzati nell'applicazione. Essa si divide in:
  - Lib: contenente le librerie di supporto che definiscono funzioni di utilità;
    - JQuery: libreria che permette di interagire in modo semplice con gli elementi di una pagina HTML;
    - Bootstrap: libreria che mette a disposizione diverse classi per definire bottoni, label, grandezza dei div, ecc;
    - Loadash: libreria che mette a disposizine funzioni di manipolazione degli array;
    - Math.js: libreria che mette a disposizione una gamma piu specifica di funzioni relative al calcolo numerico e alla manipolazione di matrici;
  - Utils: directory che contiene tutti i file che andremo ad utilizzare nella nostra Web App ed è suddivisa per aree d'interesse:
    - Computation: contenente tutti i file relativi alla logica dell'applicazione, e allo sviluppo dei calcoli necessari per l'individuazione dei punti da disegnare;
    - Draw: cartella contenente i file che si occupano del disegno e del ridisegno sulla canvas;
    - Event: cartella contenente i file che si occupano della gestione degli eventi relativi a tutta la pagina. Per esempio: cosa succede al click di un bottone o al cambio di valore in una determinata input-box.

Al di fuori di quest'ultima cartella ci sono dei file js che non hanno una funzione strettamente inerente alle prime, ma sono funzioni che agiscono sulla pagina aggiungendo informazioni o tool che poi l'utente potrà utilizzare.

- main.js: è il file che si occupa della gestione di tutti gli elementi presenti nella pagina. In essa è settata l'inizializzazione della canvas e le strutture globali che poi verranno utilizzate nelle altre funzioni;
- Css: contenente i file di stile di bootstrap;
- index.html: è l'HTML della nostra pagina web, dove poi andremo ad interagire.

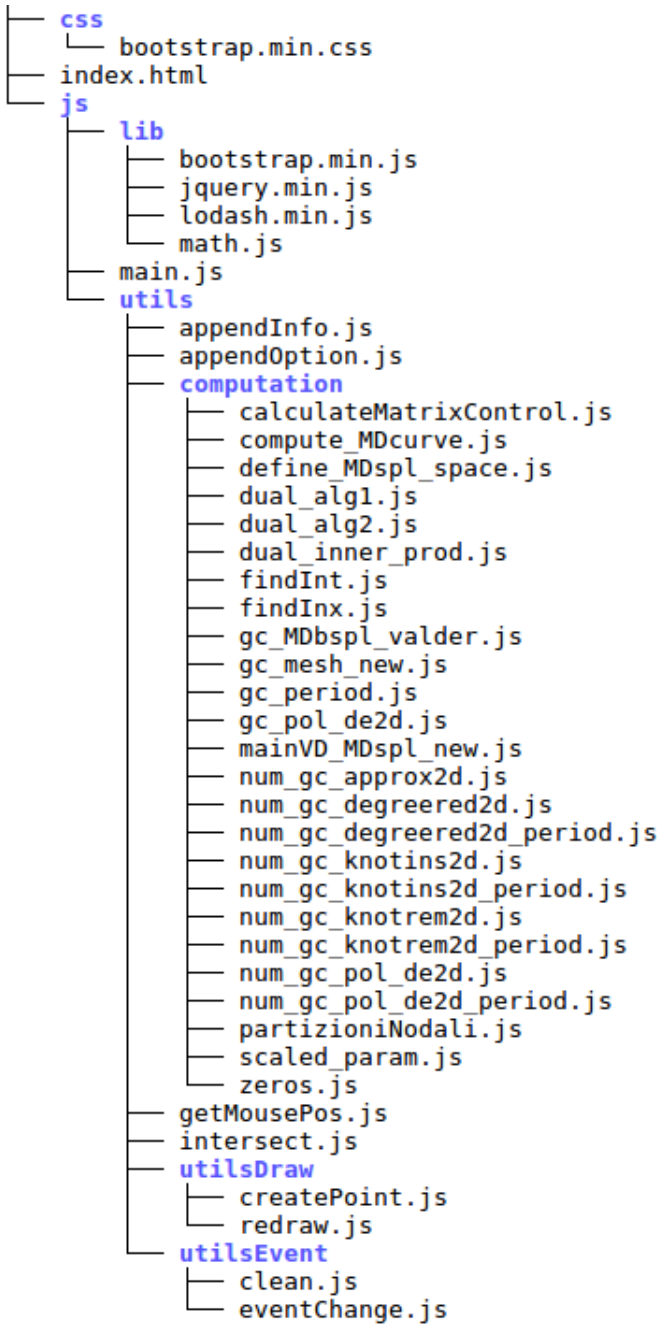



Figura 3.2: Struttura directory

# Capitolo 4

# Work Flow Web App

In questo capitolo viene trattato il flusso di esecuzione della pagina in base a come l'utente interagisce con la Web App, spiegando cosa accade ad ogni possibile click.

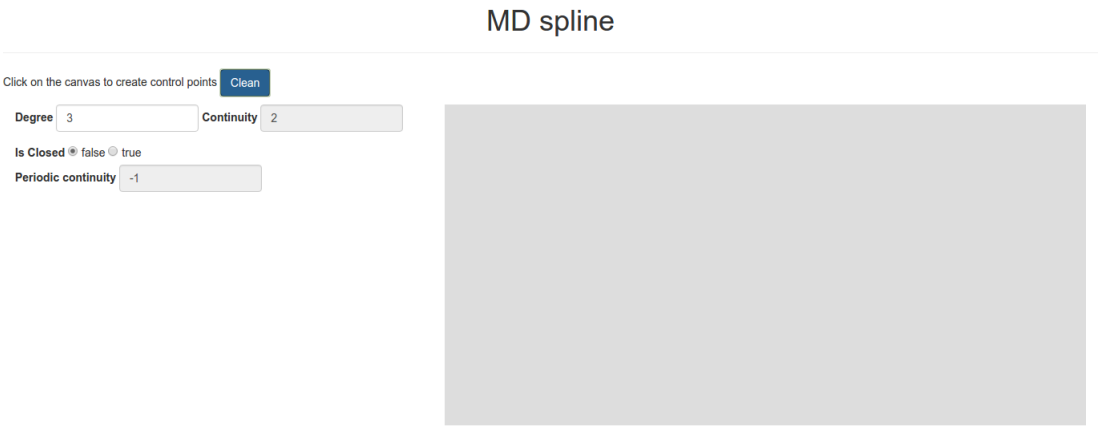



Figura 4.1: Stato iniziale

La schermata iniziale riporta a destra l'area della canvas, a sinistra i tool per creare la curva "base", col quale l'utente potrà interagire.

- Degree: indica il grado che inizialmente avrà la curva su ogni tratto;
- Continuity: indica la continuità nel caso in cui l'utente decida di creare una curva con più tratti (il valore rappresenta l'ordine di conitnuità nel punto di raccordo dei due tratti);
- Is Closed: è il campo dove viene definito se la curva creata sarà aperta o chiusa;
- Periodic Continuity: indica l'ordine continuità sugli estremi nel caso la curva sia chiusa.

## 4.1 Stato iniziale


Figura 4.2: Creazione control point

In questa fase l'utente può definire quanti controlpoint desidera, semplicemente usando il click sinistro del mouse. Una volta conclusa la fase d'inserimento dei controlpoint, con il click destro del mouse viene chiamata

la funzione “mainVD_MDspl_new”, che si occupa di effettuare i calcoli necessari per la definizione dei punti della curva e per la sua visualizzazione iniziale.

## 4.2 Modellazione Curva

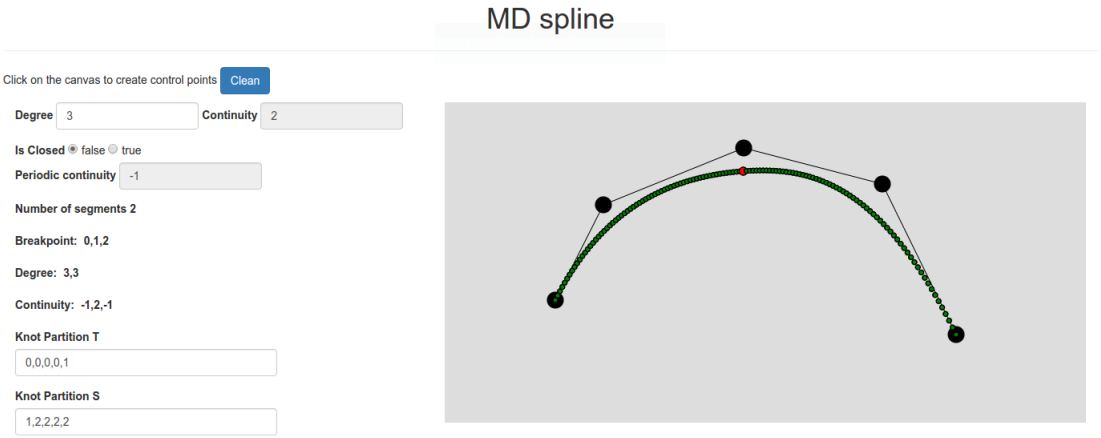



Figura 4.3: Creazione curva

Una volta visualizzata la curva sulla canvas, appariranno una serie di label dove verranno fornite tutte le informazioni relative alla curva stessa, tra cui:

- Number of segment: ovvero di quanti segmenti è composta la curva disegnata. Ovviamente questo valore dipende strettamente dal grado impostato nella fase preliminare e dal numero di control point inseriti dall’utente (non è possibile creare una curva per esempio di grado 3 dando meno di 4 controlpoint);
- Breakpoint: Breakpoint: indica i breakpoint presenti nella curva (vedi definizione 1.1.1) insieme agli estremi dell’intervallo [a, b];

- Degree: indica il grado che la curva ha per ogni tratto;
- Continuity: indica l'ordine di continuità nei punti di contatto di due tratti della curva compreso di estremi dell'intervallo;
- Knot Partition T e S: indica la partizione nodale T e S in cui sono stati mappati i breakpoint, come spiegato nella parte teorica (vedi 1.3). Sono fondamentali per sapere il numero delle funzioni base della curva e il loro supporto.

Nella canvas, la curva viene presentata per punti colorati, ogni colore ha un significato preciso:

**Verde:** indica i punti della curva;

**Rosso:** indica i breakpoint della curva;

**Blu:** sta ad indicare un punto specifico della curva: questa funzionalità è legata al movimento del mouse; spostando il cursore lungo la curva anche il pallino blu si muoverà di conseguenza;

**Nero:** control point.

### 4.2.1 mainVD_MDspl_new

Questa è una funzione di calcolo presente nella directory "computation" e si occupa del calcolo dei punti della curva iniziale (come spiegato nella sezione 1.5). Essa prende in input diversi parametri, tra cui:

**Param:** struttura dati contentente tutte le informazioni relative alla curva come grado, continuità, numero breakpoint, ecc. Ovviamente in questa fase la struttura è ancora vuota;

**ControlPoint:** struttura dati preposta al contenimento delle coordinate x, y di ogni control point creato dall'utente;

**Period:** variabile che contiene il valore di continuità sugli estremi, i valori possibili sono: -1 indica che la curva è aperta, mentre tutti i valori maggiori stanno ad indicare che si tratta di una curva chiusa, ed il valore stesso indica il grado di continuità che si vuole nel punto di chiusura;

**Degree:** variabile contenente il grado della curva al momento della creazione (valore recuperato dalla omonima input-box);

**Continuity:** variabile contente il valore della continuità di ogni punto di raccordo (inizialmente impostato automaticamente al grado-1);

**NUMBER_POINT:** Costante che indica in quanti punti valutare la curva (impostato a 64).

#### 4.2.1.1 define_MDspl_space

Questa funzione si occupa di generare lo spazio MD attraverso le informzioni iniziali date dall'utente in fase di creazione della curva. Essa prende come parametri tutte le variabili della "mainVD_MDspl_new" tranne "NUMBER_POINT" e restituisce la struttura param adeguatamente riempita.

In essa vengono anche settate tutte le informazioni che appariranno all'utente come detto in precedenza (knot partition , breakpoint,...); da questo momento in poi tutti gli altri tool di modellazione prenderanno e modificheranno la struttura param restituita da questa funzione.

#### 4.2.1.2 gc_mesh_new

Questa funzione si occupa di generare i punti di valutazione della curva, in base alla struttura param e al valore di "NUMBER_POINT". Essa infatti crea "NUMBER_POINT" punti di valutazione per ogni intervallo nodale.

#### 4.2.1.3 gc_MDbspl_valder

Questa è una delle funzioni più importanti del progetto: essa si occupa di creare la matrice "bs", che insieme ai controlpoint, saranno poi utilizzati per il calcolo di ogni punto della curva. La matrice "bs" contiene tante righe, quanti sono i punti di valutazione, e tante colonne, quanti sono i breakpoint della curva (le funzioni base). La funzione non solo restituisce la matrice "bs", ma anche un altro array chiamato "fl" contenente gli indici che indicano i valori non nulli della matrice; Le righe di quest'ultima sono infatti vettori con elementi nulli, tranne per alcuni valori consecutivi che corrispondono alle funzioni base non nulle nel punto di valutazione, che è associato a quella riga.

#### 4.2.1.4 calculateMatrixControl

Questa funzione si occupa di effettuare il calcolo tra la matrice "bs" e i controlpoint inseriti dall'utente: infatti i punti della curva non sono altro che il prodotto tra la matrice bs e il vettore dei controlpoint, ovvero la loro combinazione lineare (vedi 1.8).

## 4.3 Interazioni con la curva

In questa fase l'utente ha la possibilità, tenendo premuto il click sinistro su un controlpoint, di spostarlo e quindi di modificare la curva originale come illustrato nella figura 4.4.

Se si vuole creare una nuova curva è sufficiente clickare il pulsante "Clean" e questo farà tornare alla situazione iniziale pulendo la canvas, le informazioni e tutte le strutture dati associate.

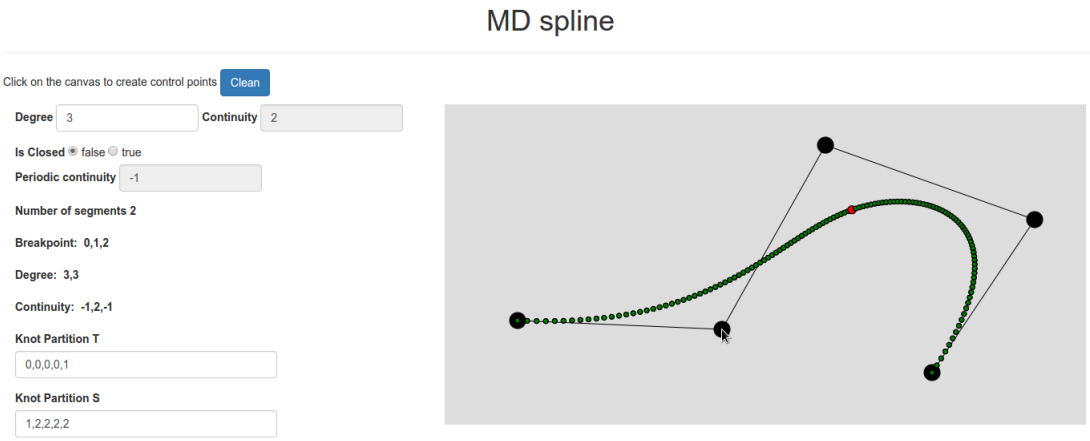



Figura 4.4: Movimento Control Point

## 4.4 Operazioni sulla curva

Tutte le operazioni sono bloccanti: l'utente per poter tornare ad interagire con la curva deve prima effettuare un'operazione o usare il tasto cancel.

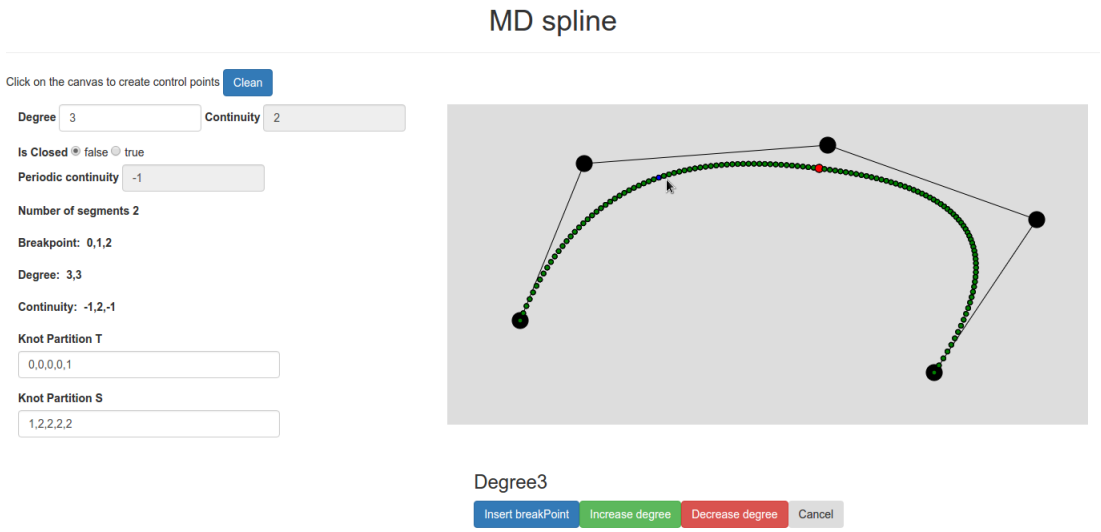



Figura 4.5: Operazioni al click destro su pallino verde


Figura 4.6: Operazioni al click destro su breakpoint

### 4.4.1 Insert break point

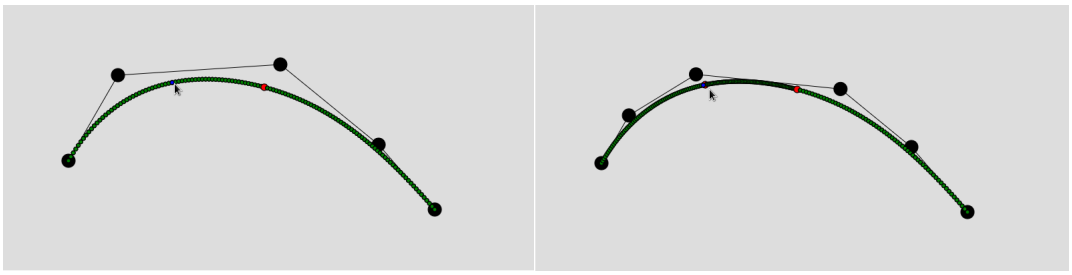


Figura 4.7: Insert Breakpoint

Questa funzionalità permette all'utente di inserire un breakpoint (fig. 4.7), in base a come detto nella sezione 1.6.1. Costruita la curva iniziale e selezionato un punto qualsiasi della curva che non sia un breackpoint, apparirà una piccola sezione dove si potrà scegliere l'operazione da compiere.

Il click sul bottone "insert breakPoint" scatenerà l'evento per cui verrà inserito un nuovo breakpoint nel punto indicato dal pallino blu.

La funzione di calcolo che se ne occupa viene invocata dall'handler associato a quel bottone.

#### 4.4.1.1 InsertBreakPoint

Questo è l'handler preposto all'inserimento di un nuovo breakpoint nella curva. Esso aggiornerà le seguenti strutture:

- bs
- fl
- controlPoint
- param
- pointShape
- pointToEvalutate

in modo che si possa eseguire il calcolo per definire le nuove coordinate x,y dei punti della curva.

Ciò viene delegato alla funzione "num_gc_knotins2d" ("num_gc_knotins2d_period" nel caso periodico) che ha il compito di fornire i nuovi valori. In seguito verrà chiamata la "compute_MDcurve", che a sua volta richiamerà nuovamente la "gc_mesh_new" (per avere i nuovi punti di valutazione essendo stato aggiunto un nuovo breakpoint), "gc_MDbspl_valder" (per avere la nuova matrice "bs" e l'array "fl", in quanto ci sarà un controlpoint in più) e la "calculateMatrixControl" (che eseguirà di nuovo la combinazione lineare tra la matrice e i controlpoint per avere i nuovi punti della curva).

Una volta terminata la funzione, essa restituirà tutti questi valori aggiornati, che poi verranno assegnati alle analoghe strutture globali, in modo che le prossime funzioni utilizzino quelle informazioni. La curva comunque non cambierà nella forma perchè l'inserimento di un nuovo breakpoint è un'operazione esatta.

#### 4.4.1.2 num_gc_knotins2d, num_gc_knotins2d_period

Queste sono le funzioni che si occupano di aggiornare la struttura param e i controlpoint della nostra curva, in modo che ci sia il nuovo breakpoint. La funzione prende in input i seguenti parametri:

- param: la struttura contenente tutte le infomazioni della curva in quel momento, a prescidere dal fatto che si sia nello stato iniziale oppure sono state eseguite altre operazioni di modellazione;

- tc: il punto selezionato dall'utente per l'inserimento del nuovo breakpoint;

- controlpoint: i controlpoint della curva.

### 4.4.2 Increase continuity

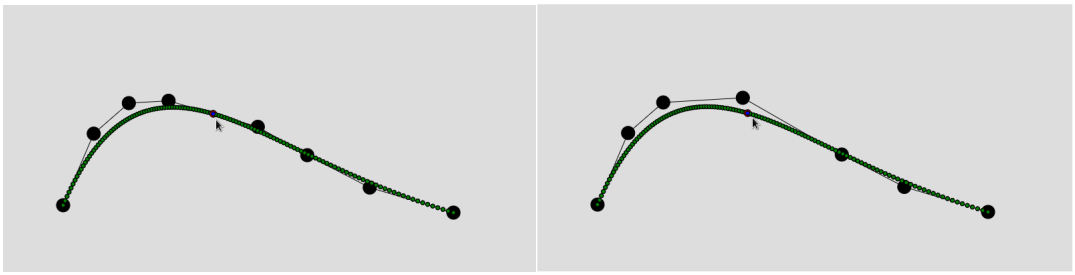


Figura 4.8: Continuity increse

Come mostrato nella fig. 4.6, l'utente cliccando, con il tasto destro del mouse, su un breakpoint può decidere di aumentarne la continuità, sempre rispettando il vincolo che vuole che la massima continuità possibile in quel punto sia pari a 1 se i gradi dei due tratti adiacenti sono diversi, e pari a grado-1 se i due tratti adiacenti sono uguali.

#### 4.4.2.1 increaseContinuity

Questo è l'handler che si occupa di aumentare la continuità in un punto e di restituire le strutture aggiornate. La funzione richiama la num_gc_knotrem2d (num_gc_knotrem2d_period nel caso periodico). I valori aggiornati sono i seguenti:

- bs
- fl
- controlpoint
- param
- pointShape
- pointGcMesh (i nuovi punti di valutazioni)

Con queste nuove informazioni si possono ridisegnare i nuovi punti sulla curva, attraverso le funzioni di calcolo viste nella sezione 4.2.1. A differenza della precedente, la funzione di riduzione della continuità non è un'operazione esatta.

#### 4.4.2.2 num_gc_approx2d

E' il fulcro della "num_gc_knotrem2d" e della "num_gc_knotrem2d_period", essa infatti si occupa della modifica delle informazioni passate in input alla funzione. Questa crea un'approssimazione della curva utilizzando "l'approsimazione ai minimi quadrati", come spiegato nella sezione 1.6, per poter trovare i nuovi punti della curva. Questa funzione verrà utilizzata anche successivamente, parlando della riduzione di grado di un tratto. I valori di ritorno sono:

- param
- controlpoint

### 4.4.3 Continuity reduction

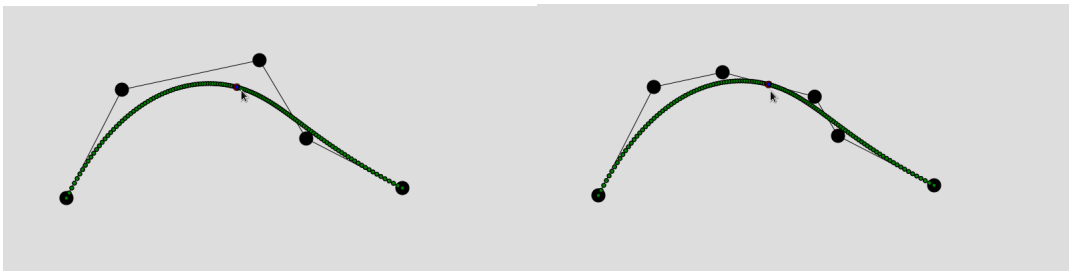


Figura 4.9: Continuity reduction

Un'altra opzione, data all'utente, è quella di diminuire la continuità di un dato breakpoint. Il limite, in questo caso, è che non si può avere una continuità minore di 0. Questa funzionalità è implementata dalla funzione "num_gc_knotins2d" ("num_gc_knotins2d_period" nel caso periodico) di cui

abbiamo parlato in precedenza. L'operazione di continuity reduction, infatti, non è altro che l'applicazione dell'algoritmo di knot insertion su un breakpoint già esistente.

### 4.4.4 Degree elevation

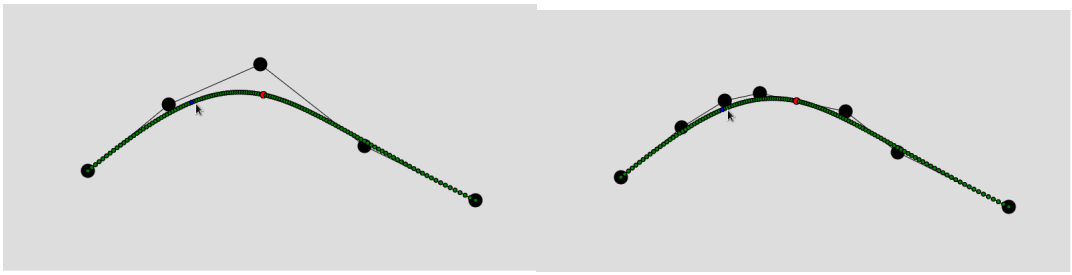


Figura 4.10: Degree elevation

Questa funzionalità permette all'utente che ha selezionato un tratto di curva di elevarne il grado, come descritto nella sezione 1.6.3, e quindi di avere gradi diversi sui diversi tratti. A differenza della "insert break point", essa necessita di un ulteriore controllo: su tratti aventi gradi diversi, la continuità che possiamo avere è al massimo $C^1$, quindi prima di effettuare la degree elevation c'è la necessità di abbassare la continuità del breakpoint iniziale e finale del tratto. E' assolutamente necessario effettuare questa operazione prima della degree elevation: in caso contrario, in un dato momento potremmo avere gradi della curva diversi e continuità nel punto di congiunzione non $C^1$, andando fuori dal nostro spazio $C^1$ MD-spline. Si esegue quindi prima la funzione "num_gc_knotins2d" o la "num_gc_knotins2d_period" (definite nelle precedenti sezioni) sul breakpoint in modo da portare la sua continuità a 1 e poi si passa alla funzione "num_gc_pol_de2d" ("num_gc_pol_de2d_period" nel caso periodico) che restituirà la nuova struttura param e i nuovi controlpoint.

Dal risultato di quest'ultima si rieseguirà la funzione "compute_MDcurve" . Anche in questo caso siamo in presenza di una operazione "esatta" (non cambia la curva).

#### 4.4.4.1 increaseDegree

Questo è l'handler che si occupa di eseguire i controlli sulla continuità del tratto considerato ed eventualmente di lanciare la funzione "num_gc_knotins2d" ("num_gc_knotins2d_period" nel caso periodico) per abbassare la continuità dei breakpoint iniziale e finale del tratto. Inoltre chiama la funzione principale, ovvero la "num_gc_pol_de2d" o la "num_gc_pol_de2d_period" (per il caso periodico) e con il risultato si occupa di calcolare i nuovi punti e restituirli in modo che le strutture globali vengano aggiornate e successivamente disegnate. I valori aggiornati sono:

- bs
- fl
- controlpoint
- pointShape
- pointGcMesh (i nuovi punti di valutazioni)

#### 4.4.4.2 num_gc_pol_de2d, num_gc_pol_de2d_period

Sono le funzioni di calcolo che definiscono il nuovo controlpoint e modificano la struttura param passata, attraverso l'ausilio delle partizioni scalate della knotPartition T ed S.

### 4.4.5 Degree reduction

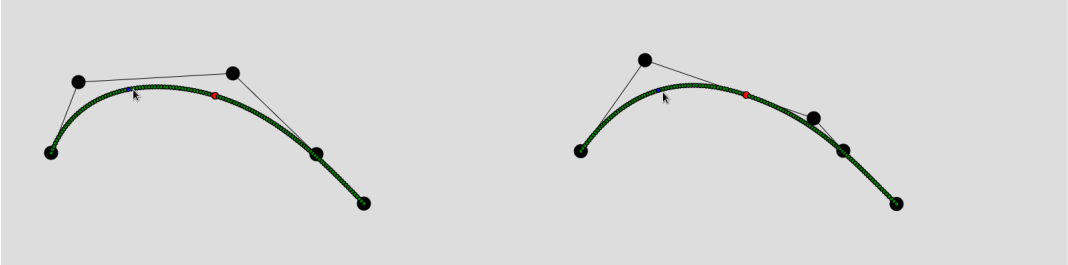


Figura 4.11: Degree reduction

Funzionalità che permette all'utente di abbassare il grado di un tratto di curva selezionato. Anche qui, come nella Degree elevation, bisogna anzitutto controllare se per effettuare l'operazione c'è bisogno di ridurre la continuità del breakpoint iniziale e finale del tratto. Infatti, come prima operazione, viene effettuato il controllo sulla continuità nei breakpoint e se è possibile eseguire l'operazione (quasi mai perchè teoricamente se ha un grado diverso già la continuità sarà settata ad 1), senza cambiare la continuità, si passa subito alla funzione di calcolo del nuovo segmento di curva; in caso contrario viene eseguita la funzione "num_gc_knotins2d" o la "num_gc_knotins2d_period", descritta in precedenza per portare la continuità ad 1. Infine si esegue la funzione di riduzione del grado.

#### 4.4.5.1 decreaseDegree

E' l'handler incaricato di abbassare il grado di un tratto di curva e, anche lui come nel caso del degree elevation, parte con il controllo della continuità e, attraverso il valore del array delle continuità presente nella struttura param, decide se chiamare l'algoritmo che abbasserà la continuità in quel punto. Eseguita questa operazione viene lanciata la funzione che, effettivamente, ridurrà il grado del segmento della curva ovvero la "num_gc_degreered2d" o la "num_gc_degreered2d_period", dove una volta ricevuto il risultato provvederà al ricalcolo dei nuovi punti e al aggiornamento delle strutture.

#### 4.4.5.2 num_gc_degreered2d, num_gc_degreered2d_period

Sono le vere funzioni che effettuano il calcolo spiegato nella sezione 1.6.2. Non riducono, in genere, il grado in modo esatto; determinano un nuovo insieme di punti di controllo che verranno usati per il ricalcolo della curva. Si occupano principalmente dell'impostazione della struttura param, chiamando in seguito la "num_gc_approx2d" definita in precedenza.

# Capitolo 5

# Guida all'utilizzo

## 5.1 Introduzione

In questo capitolo vedremo come utilizzare la Web App e le sue funzionalità.

Si possono distinguere, come visto nei precedenti capitoli due momenti:

- Inserimento di una curva;
- Editing.

E'importante ricordare che, ogni volta che si effettua un'operazione di editing, le altre azioni vengono bloccate finchè l'operazione non è terminata.

## 5.2 Inserimento della curva

In questa sezione spiegheremo com'è possibile da parte dell'utente poter creare una curva "base" su cui poi poter andare ad operare. Sarà possibile tornare in questa fase solo se si decide di utilizzare il tasto "Clean".

### 5.2.1 Settaggio dei parametri

Il primo passo per poter utilizzare il sistema è quello di settare nel pannello a sinistra le informazioni iniziali.

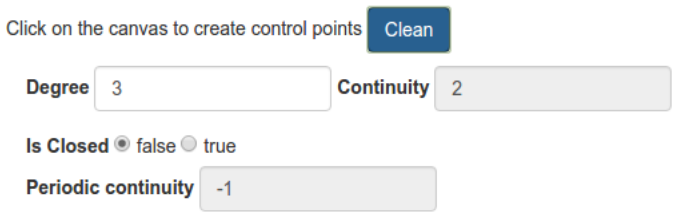



Figura 5.1: Inizializzazione della curva

Attraverso queste impostazioni è possibile definire diversi parametri:

- Degree: grado iniziale che avrà la curva per ogni suo tratto;
- is Closed: campo che determina se la curva debba essere aperta o chiusa.

Come mostrato in figura 5.1 il campo continuity è immutabile, infatti si adatterà in base al grado scelto dall'utente, anche il campo Periodic continuity è immutabile e verrà reso disponibile solo se si sceglie di costruire una curva chiusa.

### 5.2.2 Generazione Control Point

Attraverso il click sinistro del mouse, l'utente potrà inserire i control point nell'area di disegno contraddistinta da uno sfondo grigio. Si può inserire un numero arbitrario di controlpoint e quindi creare tutti i tratti che si vogliono, a patto che il loro numero sia coerente con le informazioni settate in precedenza (almeno ci devono essere un numero di contropoint pari al grado+1).

### 5.2.3 Generazione Curva

Inseriti i controlpoint, attraverso il click destro del mouse, verrà mostrata la curva sulla canvas, con il procedimento descritto in 4.1, e in più apparirà una "info window" che conterrà tutte le informazioni utili sulla curva.

Dopo aver generato la curva, le informazioni presenti nelle aree di settaggio iniziale non saranno più utilizzate e il click destro del mouse sarà utilizzato per sviluppare le operazioni di editing di cui ci occuperemo nella prossima sezione.

## 5.3 Editing

In questa sezione tratteremo tutte le funzionalità di editing che possiamo eseguire sulla curva appena creata:

- Modifica Control Point
- Interazione con un tratto
  - inserimento break point
  - aumento di grado
  - diminuzione di grado
- Interazione con un breakpoint
  - aumento continuità
  - diminuzione continuità

### 5.3.1 Modifica control point

Attraverso il click sinistro su un determinato controlpoint (identificato con il pallino nero) si ha la possibilità di spostarlo a proprio piacimento. Questo tipo di editing non comporta un cambiamento della natura della curva, come ad esempio: aumento di grado, inserimento di un nuovo breakpoint, ecc.. ma ne modifica solo la forma.

E' sempre possibile muovere un controlpoint anche dopo aver effettuato altre operazioni di editing sulla curva.

### 5.3.2 Interazione con un tratto

In questa sezione indicheremo tutte le operazioni di editing che vengono eseguite quando c'è un click destro su un tratto di curva (identificato con il pallino verde), queste operazioni interesseranno solo il segmento selezionato e non si ripercuoteranno su tutta la curva; sono accessibili attraverso una casella di controllo (fig. 5.2).

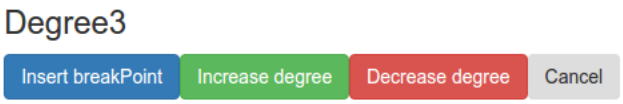



Figura 5.2: Casella di controllo

#### 5.3.2.1 Inserimento break point

Questa funzionalità, come descritto in precedenza, permette all'utente d'inserire un nuovo breakpoint in un punto preciso della curva (identificato dal pallino blu).

L'operazione cambierà la natura della curva in quanto ci sarà un tratto in più.

Il cambiamento sarà registrato anche nella info window nei campi 'Numero segmenti', che adesso sarà incrementato di uno, e nel capo 'Brekpoint' dove verrà mostrato il nuovo valore.

Il breakpoint appena inserito avrà la continuità massima permessa.

#### 5.3.2.2 Aumento di grado

Questà funzionalità permette di incrementare il grado solo di un tratto, questo è possibile perchè stiamo lavorando con curve MD-spline; nelle canoniche curve spline si sarebbe dovuto incrementare il grado di tutti i tratti della curva.

Nel nostro caso, come detto nella sezione 4.4.4, questo comporta la riduzione della continuità ad 1 dei breakpoint iniziali e finali del tratto. La curva non cambia ma cambia la sua rappresentazione.

#### 5.3.2.3 Diminuzione di grado

Funzionalità che permette di diminuire il grado di un tratto scelto dall'utente, descritta nel dettaglio nella sezione 4.4.5. Si accede a questa funzionalità attraverso la casella di controllo che appare ogni volta che viene selezionato un tratto di una curva, che non sia un breakpoint, anche qui si noti che con l'abbassamento di grado c'è una riduzione della continuità nei breakpoint iniziale e finale del tratto.

### 5.3.3 Interazione con un break point

In questa sezione verranno indicate le funzioni che si possono eseguire, quando viene effettuato un click (attraverso il pulsante destro del mouse) su un breakpoint.

Come nel "interazione con un tratto", anche qui apparirà una casella di controllo che permetterà di scegliere l'operazione da eseguire, come mostrato in fig. 5.3

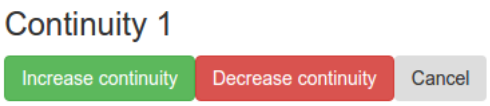



Figura 5.3: Casella di controllo continuity

#### 5.3.3.1 Aumento continuità

Uno dei tool a disposizione della Web App è l'aumento della continuità. Questa operazione descritta nel dettaglio nella sezione 4.4.2 è una funzione di editing che cambia la natura della curva, ma non sempre è possibile eseguirla, infatti, nel caso in cui i due tratti interessati da quel breakpoint abbiano

gradi diversi questa operazione non è possibile (è sempre possibile passare da continuità 0 a continuità 1).

#### 5.3.3.2 Diminuzione continuità

E' l'operazione inversa a quella descritta nella sezione precedente, infatti, permette di abbassare la continuità di un breakpoint, qui l'unico vincolo che abbiamo è dato dal fatto che non possiamo avere una continuità negativa.

## 5.4 Esempio

In questa sezione esporrò un esempio di modellazione creando e modellando la lettera 'T'.

### 5.4.1 Passo 1

Partiamo settando il grado di ogni tratto ad 1, la continuità automaticamente sarà zero. Essendo ogni tratto di grado 1, i punti della curva seguiranno la linea dei controlpoint come possiamo ossevare nella figura 5.4.

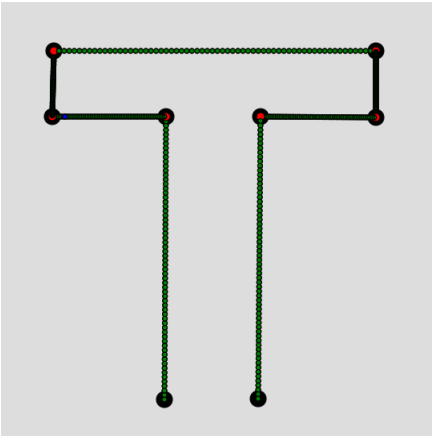


Figura 5.4: Lettera T iniziale

### 5.4.2 Passo 2

Ora vogliamo smussare gli angoli che raccordano l'asta verticale della T con l'asta orizzontale, per farlo devo aumentare la continuità nei breakpoint di raccordo. Prima di procedere, dobbiamo aumentare i gradi nei due tratti adiacenti, questo perchè non possiamo avere continuità maggiore o uguale al grado.

Quindi dobbiamo eseguire una degree elevation sui tratti contrassegnati con D1, come mostrato nella figura 5.5.

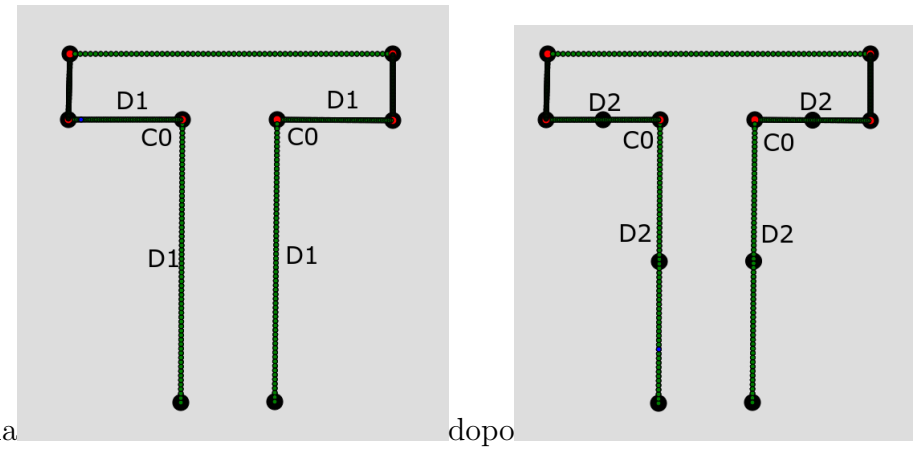



Figura 5.5: Degree Increase Esempio

Una volta eseguita quest'operazione, possiamo andare ad aumentare la continuità dei due breakpoint indicati con $C^0$ e passare da $C^0$ a $C^1$, ottenendo così, attravero la 'increase continuity', il risultato mostrato in figura 5.6.

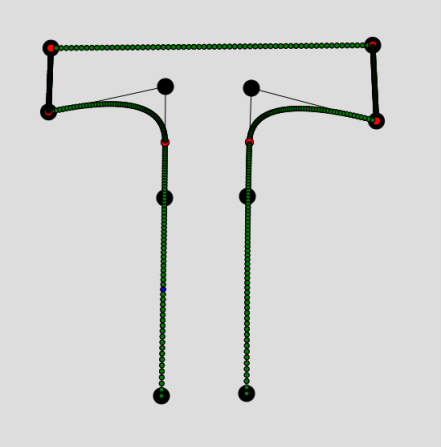


Figura 5.6: Lettera T Conitnuity

### 5.4.3 Passo 3

Da qui, come ultimo passo, inseriremo due nuovi break point alla base, per avere la parte inferiore della lettera piu ampia dell'asta centrale e quindi andiamo ad eseguire un knot insertion alla base della nostra lettera, come mostrato in figura 5.7. Segue una modifica interattiva dei due punti di controllo estremi, per allargare la base della lettera .

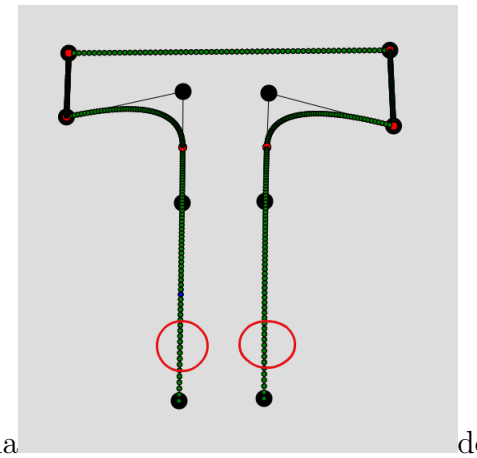
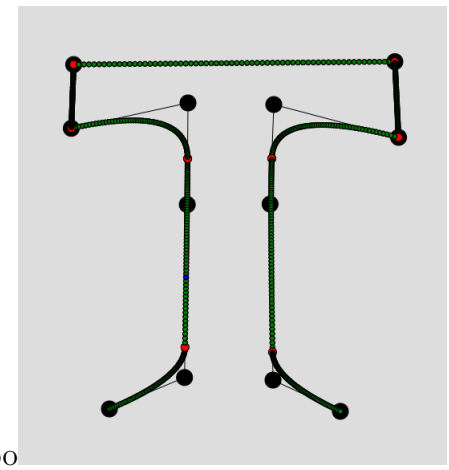
prima  dopo 

Figura 5.7: Lettera T Knot insertion

Con questo semplice esempio abbiamo potuto apprezzare le proprietà delle MD-spline dove ogni tratto è indipendente dall'altro.

# Bibliografia


[1] Sederberg, T.W., Zheng, J., Song, X. (2003) *Knot Intervals and Multi-Degree Splines* Computer Aided Geometric Design, 20(2003) 455-468.

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

[3] Wang, G., Deng C., (2007) *On the degree elevation of B-spline curves and corner cutting* Computer Aided Geometric Design, 24(2007) 90-98.

[4] Shen, W., Wang, G. (2010) *Changeable degree spline basis functions* Journal of Computational and Applied Mathematics, 234(2010) 2516-2529.

[5] Shen, W., Wang, G. (2010) *A basis of multi-degree splines* Computer Aided Geometric Design, 27(2010) 23-35.

[6] Li, X., Huang, Z.J., Liu, Z. (2012) *A Geometric Approach for Multi-Degree Spline* Journal of Computer Science and Technology, 27(4) 841-850, 2012.

[7] Shen, W., Wang, G., Yin P., (2013) *Explicit representations of changeable degree spline basis functions* Journal of Computational and Applied Mathematics, 238(2013) 39-50.

[8] Wozny, P. (2014) *Construction of dual B-spline functions* Journal of Computational and Applied Mathematics, 260(2014) 301-311.

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

[10] Andrea Benetti, Modellazione Geometrica con MD-Spline, tesi di laurea Magistrale in Informatica, A.A.2015/16 (Sessione II), Università di Bologna.

# Ringraziamenti

Ai miei genitori che hanno sempre creduto in me.

Alla social non solo una casa ma una famiglia.

Martina la mia “ragazza magica”.

Morena la migliore amica che si possa avere e con cui ho fatto piu km che con chiunque altra.

Ai miei amici che mi sono sempre stati vicini anche quando mancavo per mesi e mesi.

Al professor Giulio Casciola per la guida e sopratutto per l’inifinita pazienza.