Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea Triennale in Matematica

CURVE SPLINE DI INTERPOLAZIONE LOCALE E ANALISI GRAFICA DELLA LORO QUALITÀ

Tesi di Laurea in Informatica

Relatore:
Chiar.mo Prof.
Giulio Casciola

Presentata da:
Giacomo Ferrari

II Sessione
Anno Accademico 2010-2011

A coloro che
mi hanno sempre sostenuto

Introduzione

L'industria del design automobilistico (e non solo) ha vissuto negli ultimi decenni una notevole evoluzione, che ha portato alla creazione di modelli dalle forme sempre più varie e aerodinamiche.
È stata data un'importanza sempre maggiore al concetto di qualità delle superfici e delle curve.
Non esistono regole matematiche che possano definire formalmente la qualità di una curva, anche se esistono proprietà comuni alle curve (regolarità, curvatura, ecc...) che ci forniscono indicazioni sulla loro qualità.
Sviluppando programmi di progettazione e disegno che rappresentano questi indici è stato possibile creare superfici e oggetti che risultano belli alla vista, partendo da curve cosiddette "di qualità".

Questa tesi ha l'obiettivo di fare un confronto tra differenti curve interpolanti gli stessi punti, arrivando ad alcune considerazioni sulla qualità di ciascuna di queste curve.
Per fare ciò è stato ideato e implementato un programma che permette di rappresentare graficamente le curve e alcuni loro indici di qualità.
Nel primo capitolo si descrivono i metodi di *interpolazione polinomiale* e di *interpolazione polinomiale a tratti* utilizzati per ottenere le curve interpolanti.
Nel secondo capitolo si illustra approfonditamente il progetto realizzato, soffermandosi prima sui procedimenti utilizzati e poi sulle strutture dati e sulle funzioni che costituiscono il programma.
Nel terzo ed ultimo capitolo si mostrano esempi sperimentali di curve inter-

polanti i medesimi punti, ma con proprietà differenti, inoltre vengono espresse considerazioni sulla qualità di queste curve.

Indice

Elenco delle figure

Capitolo 1

Interpolazione di curve

Con il termine *interpolazione* si intende generalmente un procedimento che permette di individuare una funzione che passi attraverso un insieme di punti dati, nell'ipotesi che tale funzione appartenga ad un determinato spazio funzionale.
Esistono vari tipi di interpolazione, tra cui si ricordano l'interpolazione polinomiale e quella trigonometrica. Ci si soffermerà sull'interpolazione polinomiale.
Il procedimento per trovare un polinomio interpolante, partendo da un ristretto insieme di punti di interpolazione, consiste nella risoluzione di un sistema lineare che ha come soluzione i coefficienti del polinomio stesso.
Inizialmente è quindi necessario introdurre una base di rappresentazione dei polinomi, per esempio la *Base di Bernstein*, che permetta di avere una matrice del sistema ben condizionata, e di conseguenza adatta alla risoluzione di un sistema lineare.
Successivamente si vedranno più nello specifico i metodi di *interpolazione polinomiale* e *interpolazione polinomiale a tratti*, introducendo brevemente funzioni di blending, utilizzate nello sviluppo del secondo metodo.
Infine si considerano curve parametriche, che sono alla base dell'esperimento proposto.

1.1 Base di Bernstein

Consideriamo Π_n come lo spazio vettoriale dei polinomi di grado minore o uguale a n.

Definizione 1.1. Dato $t \in [0,1]$, si definiscono *polinomi base di Bernstein* o *polinomi di Bernstein* di grado n, in numero $n+1$:

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad i = 0 \dots n$$

dove

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

Definizione 1.2. Un polinomio $p(x) \in \Pi_n$ espresso nella *base di Bernstein* si scrive nella forma:

$$p(t) = \sum_{i=0}^{n} b_i B_{i,n}(t) \qquad t \in [0,1].$$

dove i b_i sono i coefficienti del polinomio chiamati *Coefficienti di Bernstein.*

Osservazione 1. I *polinomi di Bernstein* di grado n costituiscono una base per Π_n.

Dimostrazione. Abbiamo definito:

$$B_{i,n}(t) \;=\; \binom{n}{i} t^i (1-t)^{n-i} \;=\; \binom{n}{i} t^i \sum_{k=0}^{n-i} \binom{n-i}{k} (1)^k (-t)^{n-i-k}$$

Riscriviamo l'ultima espressione in forma matriciale:

$$\begin{pmatrix} B_{0,n}(t) \\ B_{1,n}(t) \\ \vdots \\ B_{n,n}(t) \end{pmatrix} = \begin{pmatrix} 1 & * & \dots & * \\ & & & \vdots \\ & \binom{n}{i} & \dots & * \\ & & & \vdots \\ & & & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ \vdots \\ t^n \end{pmatrix}$$

dove $*$ indica termini non nulli.
Si vede subito che la matrice è triangolare superiore, così il suo determinante

è dato dalla produttoria degli elementi sulla sua diagonale, che sono tutti non nulli. Di conseguenza questa matrice è non singolare e i $B_{i,n}(t) \quad i = 0 \dots n$ formano una base per Π_n.

□

Osservazione 2. È possibile esprimere le *funzioni base di Bernstein* in forma generalizzata per $x \in [a, b]$ così:

$$B_{i,n}(x) = \binom{n}{i} \frac{(x-a)^i (b-x)^{n-i}}{(b-a)^n} \quad i = 0 \dots n.$$

Applicando un opportuno cambio di variabile

$$f : [a, b] \to [0, 1] \quad \text{tale che} \quad f(x) = t = \frac{x-a}{b-a}$$

e sfruttando la proprietà di invarianza delle funzioni base per cambio di variabile, si può agevolmente passare da una forma all'altra. Verranno utilizzti solamente i polinomi di Bernstein per $t \in [0, 1]$ in quanto sono più facili da calcolare.

I *polinomi di Bernstein* godono di tre importanti proprietà:

1. Sono funzioni non negative:

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

2. Costituiscono una partizione dell'unità:

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

3. Si possono calcolare ricorsivamente:

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

   Si pone per convenzione $B_{i,n}(t) = 0$ se $i < 0$ oppure $i > n$.

Le derivate di un *polinomio di Bernstein* di grado n sono polinomi di grado $n-1$ che si possono scrivere come combinazione lineare di *polinomi di Bernstein*. In particolare si ha:

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

Quindi, considerando un generico polinomio $p(t)$ di grado n espresso nella *base di Bernstein*, la sua derivata prima sarà ancora un polinomio nella base di Bernstein di grado $n-1$ nella forma:

$$p^1(t) = \sum_{i=0}^{n} b_i \frac{d}{dt}B_{i,n}(t) = n\sum_{i=0}^{n-1}(b_{i+1} - b_i)B_{i,n-1}(t)$$

1.2 Interpolazione Polinomiale

All'inizio del capitolo è stato già introdotto brevemente il concetto di interpolazione, ora viene approfondito l'argomento dandone una formulazione più precisa:

Problema 1.1. Sia dato uno spazio funzionale $\mathbb{F}$, costituito da funzioni definite in un intervallo $\Omega \in \mathbb{R}$, e siano assegnati $n+1$ punti (x_i, y_i) con $x_i < x_{i+1} \ \in \Omega \ i = 0 \dots n$. Si vuole trovare una funzione $f \in F$ tale che:

1. $$f(x) = \sum_{j=0}^{n} \alpha_j \varphi_j(x)$$

2. $$f(x_i) = y_i \quad i = 0 \dots n.$$

dove $\varphi_0 \dots \varphi_n$ sono funzioni linearmente indipendenti e $\alpha_0 \dots \alpha_n$ sono i gradi di libertà del problema.

Quando si parla di *interpolazione polinomiale* si considera una base polinomiale $\varphi_i(x)$ $i = 0 \dots n$, quindi il problema si riconduce a quello di trovare un polinomio $p(x)$ di grado al più n tale che $p(x_i) = y_i \quad i = 0 \dots n$.

Per trovare un polinomio interpolante, espresso nella base delle potenze $(\varphi_i(x) = x^i \ i = 0 \dots n)$, partendo dai punti dati (x_i, y_i) è necessario risolvere

il sistema:

$$A \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$$

Dove la matrice A è la *matrice di Vandermonde*, che ha gli elementi $a_{i,j} = x_i^j$, è nota essere una matrice malcondizionata, quindi inadatta per la risoluzione di un sistema lineare.
Per questo motivo conviene esprimere il polinomio interpolante cercato nella *base di Bernstein*. I coefficienti di tale polinomio saranno le soluzioni del sistema lineare:

$$B \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$$

dove B è la matrice di ordine $(n+1)$ contenente i valori assunti dalle funzioni base di Bernstein, calcolate nei punti $x_0 \dots x_n$, mentre $(y_0 \dots y_n)^T$ è il vettore dei termini noti.
L'interpolazione polinomiale ha comunque lo svantaggio che i polinomi di grado alto oscillano fortemente. Questo fenomeno consiste nell'aumento dell'errore in prossimità degli estremi dell'intervallo. Una possibile soluzione consiste nell'utilizzare i nodi di Chebyshev, alternativi ai punti equidistanti, che riducono l'errore massimo all'aumentare del grado del polinomio.
La soluzione migliore però consiste nell'utilizzare polinomi di grado relativamente basso e suddividere l'intervallo di definizione in vari sottointervalli.

1.3 Blending function

Le *blending function* sono funzioni che, come dice il nome stesso, miscelano le funzioni a cui sono applicate in modo da dar luogo a funzioni più complesse.
Un esempio di *blending function* n sono i *polinomi base di Bernstein*, che per-

mettono di costruire le *curve di Beziér* partendo da un set iniziale di punti di controllo. Queste particolari *blending function*, quando sono miscelate con i punti $P_0, \ldots, P_n$, fanno si che la curva di *Bezièr* interpoli il primo ed ultimo punto di controllo e approssimi la forma definita dagli altri.

Definizione 1.3. Si definisce *partizione nodale* t una suddivisione dell'intervallo $[0, 1]$ in N intervalli tale che:

$$t = \{t_i \in [0, 1] : \ 0 = t_1 < t_2 \ldots < t_N = 1\}$$

Definizione 1.4. Si definiscono *blending function* C^k *a supporto compatto di ampiezza* $n + 1$ nella partizione nodale t, le funzioni a tratti tali che:

- $B_{i,n+1}(x) = 0 \ x \notin (t_i, t_{i+n+1})$
- $B_{i,n+1}(x) \geq 0 \ x \in (t_i, t_{i+n+1})$
- $$\sum_i B_{i,n+1}(x) \equiv 1$$

Se le *blending function* che si considerano sono polinomiali o razionali a tratti saranno C^∞ ovunque tranne che nei nodi $t_j \ j = i \ldots i{+}n{+}1$ dove sono almeno C^k .

Le *blending function* che si utilizzano nel progetto sono note con il nome di *B-Spline* [1], e si applicano alle curve polinomiali interpolanti un determinato numero di punti, permettendo di ottenere un'unica curva interpolante come combinazione delle precendenti.

Osservazione 3. Le *B-spline* utilizzate sono *blending function a supporto compatto*di ampiezza $n + 1$ e C^n.

1.4 Interpolazione polinomiale a tratti

Per *interpolazione polinomiale a tratti* si intende l'interpolazione di un insieme di dati appartenenti ad un certo intervallo tramite più polinomi,

ciascuno dei quali è definito in un sottointervallo dell'intervallo dato.
Matematicamente il problema dell'*interpolazione polinomiale a tratti* viene formulato nel modo seguente.

Problema 1.2. Siano assegnati N punti distinti e ordinati (x_i, y_i)
$i = 1 \dots N$ con x partizione nodale .
Si vuole trovare un polinomio a tratti $pp(x)$ tale che:

$$pp(x) = \begin{cases} S_1(x) & x \in [x_1, x_2] \\ \vdots & \\ S_{N-1}(x) & x \in [x_{N-1}, x_N] \end{cases}$$

e interpoli i punti dati, ovvero $N-1$ polinomi $S_i(x)$ di grado d definiti sugli intervalli $[x_i, x_{i+1}]$ con le seguenti proprietà:

1. $S_{i-1}(x_i) \equiv S_i(x_i) = y_i \quad i = 2 \dots N-1$

2. $S_{i-1}^{(l)}(x_i) \equiv S_i^{(l)}(x_i) \quad l = 1 \dots k, \quad i = 2 \dots N-1, \quad k < d-1$
   e si dirà che l'interpolante a tratti è di classe C^k.

Per questo esperimento si utilizzerà un metodo di interpolazione locale che fa uso delle *B-spline* per miscelare due o più funzioni interpolanti polinomiali.

Siano assegnati N punti (x_i, y_i) $i = 1 \dots N$ con $x = \{x_i\}$ partizione nodale e si consideri il tratto $[x_l, x_{l+1}]$.
Il polinomio a tratti interpolante $pp(x)$ sarà dato da:

$$pp(x) = \sum_{i=r-n}^{N-r} p_{i-s}(x) B_{i,n+1}(x) \quad x \in [x_r, x_{N-r-1}]$$

e in ogni intervallo sono:

$$S_l(x) = \sum_{i=l-n}^{l} p_{i-s}(x) B_{i,n+1}(x) \quad x \in [x_l, x_{l+1}], \quad l = r \dots N-r$$

dove

$$r = \frac{m+n+1}{2} \ , \ s = \frac{m-n-1}{2}$$

con m grado dei polinomi $p_{i-s}(x)$ interpolanti ciascuno $m+1$ punti consecutivi degli $N+1$ assegnati e n numero di *B-Spline* miscelate.
Più precisamente si ha che il *j-esimo* polinomio $p_j(x)$ è l'interpolante dei punti (x_h, y_h) $h = j \dots j+m$ appartenenti all'insieme iniziale.

Osservazione 4. Si osserva che:

- $pp(x)$ interpola i punti (x_i, y_i).

  Dimostrazione. Riprendendo la definizione generale di $pp(x)$ si nota che :

  $$pp(x_l) = S_{l-1}(x_l) \text{ e } pp(x_l) = S_l(x_l)$$

  Si consideri ora $S_l(x)$. In base alla definizione fornita sopra vale che:

  $$S_l(x_l) = \sum_{i=l-n}^{l} p_{i-s}(x_l) B_{i,n+1}(x_l)$$

  Per come è stato calcolato il valore s ognuno dei polinomi p_{i-s} con $i = l-n, \dots, l$ è tale che $p_{i-s}(x_l) = y_l$.
  Questo permette di affermare che:

  $$S_l(x_l) = \sum_{i=l-n}^{l} p_{i-s}(x_l) B_{i,n+1}(x_l) = y_l \sum_{i=l-n}^{l} B_{i,n+1}(x_l) = y_l$$

  perchè le *B-Spline* costituiscono una partizione dell'unità.
  Il medesimo ragionamento vale per S_{l-1}. □

- Si ha che $pp(x)$ è almeno C^k, perchè si esprime come combinazione lineare di polinomi C^∞ e funzioni blending che sono C^k.

La derivata prima del polinomio $pp(x)$ viene calcolata come:

$$pp^1(x) = \sum_{i=l-n}^{l} p^1_{i-s}(x) B_{i,n+1}(x) + p_{i-s}(x) B^1_{i,n+1}(x)$$

mentre la derivata seconda si ottiene derivando la derivata prima e si scrive come:

$$pp^2(x) = \sum_{i=l-n}^{l} p^2_{i-s}(x)B_{i,n+1}(x) + 2p^1_{i-s}(x)B^1_{i,n+1}(x) + p_{i-s}(x)B^2_{i,n+1}(x)$$

1.5 Curve parametriche e parametrizzazione

Sono definite ora le curve parametriche di $\mathbb{R}^2$ facendo alcune considerazioni sull'utilità della forma parametrica per il progetto da creare.

Definizione 1.5. Si definisce *curva parametrica* di classe C^k una curva espressa nella forma:

$$\begin{cases} x(t) = f_x(t) \\ y(t) = f_y(t) \end{cases}$$

$\forall t \in I$.
Sia $f(t) = (f_x(t), f_y(t))$. Se $I = [a, b]$ e $f(a) = f(b)$, allora si dice che la curva è chiusa.

Da questo momento si consideri $I = [0, 1]$.
Prima di formulare il problema dell'interpolazione polinomiale di punti del piano è necessario definire cosa è una *parametrizzazione* e come viene calcolata.
Siano assegnati N punti (x_i, y_i) $\imath = 1 \dots N$.

Definizione 1.6. Si definisce *parametrizzazione* dell'intervallo I il vettore t che è una partizione nodale di I in cui i t_i sono calcolati a seconda dei punti $p_i = (x_i, y_i)$ da interpolare come segue:
Si inizializza $\tau[0] = 0$, poi $\forall i = 1 \dots N$ siano

$$\tau[i] = \tau[i-1] + \|p_i - p_{i-1}\|_2^a \; i = 2 \dots N$$

Infine per avere i valori appartenenti a $[0, 1]$ si pone:

$$t[i] = \frac{\tau[i]}{\tau[n-1]}$$

Il coefficiente a può assumere un valore qualsiasi nell'intervallo $[0, 1]$ e in base alla scelta di tale valore si ottengono parametrizzazioni differenti e di conseguenza curve interpolanti differenti.

Ora è possibile dare una formulazione del problema di interpolazione dei punti del piano:

Problema 1.3. Siano assegnati N punti del piano (x_i, y_i) $i = 1 \dots N$. Si vuole trovare la curva parametrica che interpoli tutti i punti assegnati.

Per ottenere quanto richiesto dal problema si considerano separatamente le componenti x e y, si crea una parametrizzazione t e si applica il metodo di interpolazione che fa uso delle *B-Spline*, applicandolo separatamente alle compontenti (t_i, x_i) e (t_i, y_i).
In questo modo si ottengono due polinomi $pp_x(t)$ e $pp_y(t)$ che sono le componenti x e y della curva parametrica interpolante finale.

Osservazione 5. Nel caso di una curva aperta si interpola nell'intervallo $[t_2, t_{N-r-+1}]$.

Osservazione 6. Nel caso di un insieme di punti $p_i = (x_i, y_i)$ $i = \dots N$ con $P_1 = P_n$, si ottiene una curva chiusa periodica e si deve estendere l'insieme dei punti replicando i primi. Sia $r = \dfrac{m+n+1}{2}$ allora si pone $P_{N+i} = P_{1+i}$ $i = 1 \dots r$.

Capitolo 2

Progettazione di un codice per la visualizzazione di curve

Questo lavoro ha portato alla realizzazione di un codice in linguaggio C, ambiente *Linux* e grafica SDL [3] che permette all'utente di rappresentare graficamente le curve interpolanti precedentemente descritte e di effettuare uno studio sulla qualità di queste curve.
Il capitolo descrive accuratamente la progettazione del programma, spiegando il procedimento seguito per rappresentare ognuno dei grafici presenti.
In tutto il capitolo si intende con n il grado della curva da calcolare e con $g+1$ il numero di funzioni blending utilizzate.

2.1 Progettazione

All'apertura del programma viene visualizzata una finestra grafica con il titolo e un menù di opzioni:

- Impostazioni
- Singola curva
- Curva totale
- Comb

- Hodograph
- Tangenti
- Uscita

È possibile scegliere ognuna di queste opzioni cliccando il tasto indicato a lato dell'opzione stessa.
Selezionando l'opzione *impostazioni* è possibile scegliere il grado della curva interpolante e il numero di B-Spline da miscelare. Sono presenti 7 possibili combinazioni: tre per le curve a supporto 6 e quattro per le curve a supporto 8.
Scegliendo una qualsiasi delle altre opzioni del menù principale (ad eccezione dell'uscita), viene disegnato sulla parte sinistra dello schermo un elenco contenente le istruzioni da seguire per visualizzare quanto richiesto.
L'elenco è simile per tutte le opzioni.
Come prima cosa, se non è già stato inserito in precedenza, viene chiesto all'utente di digitare da shell il nome del file (di tipo .dat) contenente l'insieme dei punti da interpolare.
Una volta inserito il nome del file, suppondendo di trovarsi nella parte relativa alla visualizzazione di un singolo tratto di curva (*singola curva*), viene chiesto prima di inserire il numero relativo al tratto di curva da rappresentare e successivamente di scegliere la parametrizzazione della curva.
Nelle altre opzioni è possibile scegliere solo la parametrizzazione in quanto i grafici rappresentati sono riferiti all'intera curva interpolante.
Sono implementate tre possibili parametrizzazioni, come spiegato nel capitolo 1, dipendenti dalla scelta del valore a che può essere:

- $a = 0$: *parametrizzazione uniforme.*
- $a = 0.5$: *parametrizzazione centripeta.*
- $a = 1$: *parametrizzazione cordale.*

Nell'ultima parte dell'elenco viene spiegato sinteticamente come fare per cambiare file e per zoomare.

Una volta visualizzato un grafico è possibile scegliere una qualsiasi opzione del menù principale (che resta visibile sulla destra della schermata) e, dopo aver selezionato la parametrizzazione, visualizzarne il grafico corrispondente. Per uscire dal programma è necessario cliccare "ESC".
Vengono ora descritti più in dettaglio i grafici che è possibile rappresentare.

2.1.1 Singolo tratto di curva e Curva totale

Dato un file di punti, si vuole ottenere una curva che interpoli tutti i punti del file utilizzando il metodo di *interpolazione polinomiale a tratti* introdotto nel capitolo 1. La curva finale è ottenuta come unione di singoli tratti di curva che hanno come estremi due punti successivi del file e tali che la loro unione abbia una continuità almeno C^{g+1}. Quando viene letto il file di dati viene immediatamente effettuata una chiamata alla funzione *dividing* che determina il tipo di curva che vogliamo rappresentare (aperta o chiusa) e quanti punti è necessario aggiungere in testa o in coda al file.
I punti vengono suddivisi in gruppi di $n+1$ punti successivi tali che il primo elemento di ogni gruppo sia il secondo elemento del gruppo precendente.
È presente una struttura di tipo *CURVE* (vedi appendice A) che permette di raccogliere e organizzare i vari gruppi di punti in modo tale che ogni elemento della struttura contenga tutte le informazioni necessarie per calcolare i coefficenti di una curva interpolante. Tale suddivisione resta fino a quando non viene letto un nuovo file.
Ogni singolo tratto della curva finale si ottiene miscelando $g+1$ curve interpolanti di grado n che hanno in comune un intervallo compreso tra due punti della parametrizzazione totale.

Esempio 2.1. .
Si consideri il caso $g+1=3$ e $n=3$ e si supponga di voler rappresentare il primo tratto di curva (trovandosi nella condizione di avere già sistemato i punti).
Si hanno tre curve di grado 3 che interpolano ciascuna 4 punti:

- la curva 0 interpola i punti $\{0, 1, 2, 3\}$ del file.
- la curva 1 interpola i punti $\{1, 2, 3, 4\}$ del file.
- la curva 2 interpola i punti $\{2, 3, 4, 5\}$ del file.

L'intervallo $[2, 3]$ è interpolato da tutte le curve e è quindi l'intervallo in cui sono definite le tre blending functions (b_i) e in cui si costruisce la curva interpolante finale utilizzando la formula:

$$pp(i) = \sum_{j=0}^{g} b_j(i) p_j(i)$$

dove i appartiene all'intervallo $[p(2), p(3)]$, ovvero al terzo intervallo della parametrizzazione totale dei punti del file.
In questo modo si ottiene il primo tratto di curva.

Nella funzione *interp_blending_curve* (vedi appendice C) viene tradotto in linguaggio C il procedimento appena visto. Si vede che l'intervallo in cui si sovrappongono le $g + 1$ curve corrisponde all'intervallo $[i + s, i + s + 1]$ dove i è l'*i-esima* curva e $s = ((n + g + 1)/2) - 1)$. Viene creato un vettore Y che partiziona l'intervallo $[0, 1]$ creando il vettore v della parametrizzazione da considerare e viene eseguito un ciclo che per ogni punto del vettore Y calcola le *B-spline* e i valori delle curve interpolanti, miscelandoli come spiegato in precedenza per ottenere la valutazione del tratto di curva in quel determinato punto.
Questo procedimento può essere ripetuto per tutti i singoli pezzi di curva presenti, in questo modo l'*i-esimo* elemento dei vettori di disegno *pointx* e *pointy* (che è una struttura di tipo *DRAWING* (vedi appendice A)) contiene le informazioni necessarie per disegnare l'*i-esimo* tratto di curva e le $g + 1$ curve curve interpolanti utilizzate nell'*interpolazione polinomiale a tratti*.
Se si sceglie di visualizzare un singolo tratto di curva viene riempito solo l'*i-esimo* elemento dei vettori di disegno, mentre se si sceglie di rappresentare l'intera curva vengono calcolati tutti i singoli tratti.

2.1.2 Comb

Il grafico del *Comb* (o *Pettine*) rappresenta la curva e i valori delle curvature calcolate in alcuni dei punti della curva stessa e rappresentati lungo la normale alla curva in quei punti. Lavorando in $\mathbb{R}^2$, a seconda del verso di percorrenza della curva varia il versore normale e si ha un *Pettine* rivolto verso l'interno o verso l'esterno. Vista l'impossibilità di stabilire a priori il verso di percorrenza della curva è possibile visualizzare sia il *Comb* interno che quello esterno. Si supponga di avere un file di dati e di avere già organizzato la struttura *curv* di tipo *CURVE* come nel caso del disegno di un singolo tratto di curva. Si supponga inoltre di avere già calcolato tutti i coefficenti di ognuna delle curve interpolanti presenti.
Si consideri il procedimento per un singolo tratto. Conoscendo l'intervallo con estremi due punti del vettore nodale v da prendere in considerazione, si crea un vettore Y di punti (in questo caso 25) appartenenti a tale intervallo e si valutano le $g+1$ curve interpolanti quei punti.
Lo stesso procedimento viene effettuato per le derivate prima e seconda, calcolando i coefficenti come spiegato nel capitolo 1 e valutando le curve in ogni punto di Y.
Si trova così la curva e le sue derivate. Lo stesso ragionamento viene applicato alle *blending functions*, calcolando i coefficienti delle derivate prima e seconda e valutandole su tutto Y.
Infine sono calcolati i vettori relativi al tratto di curva interpolante definitiva e i vettori dei coefficenti delle derivate prima e seconda ottenuti miscelando blending e curve e seguendo le consuete regole di derivazione.

Esempio 2.2. La derivata prima della curva risultante è calcolata utilizzando la formula introdotta nel capitolo 1, e viene valutata nell'*i-esimo* punto di Y così:

$$dpbx[i] \;=\; \sum_{j=0}^{g} bp_j(i)p_j(i) + b_j(i)dp_j(i)$$

dove:

- $b_j(i)$ è la *j-esima B-Spline* valutata in $Y(i)$.

- $bp_j(i)$ è la derivata della *j-esima B-Spline* valutata in $Y(i)$.
- $p_j(i)$ è la *j-esima* curva valutata in $Y(i)$.
- $dp_j(i)$ è la derivata della *j-esima* curva valutata in $Y(i)$.

Una volta ricavati i valori per la curva risultante viene calcolata la curvatura utilizzando la formula:

$$co[i] = \frac{Dx[i]D2y[i] - D2x[i]Dy[i]}{(Dx[i]^2 + Dy[i]^2)^{\frac{3}{2}}}$$

dove Dx e Dy sono i vettori della derivata prima in x e y e $D2x$ e $D2y$ sono i vettori della derivata seconda. Successivamente si calcolano le compontenti coX e coY della curvatura:

$$coX[i] = \frac{Dy[i]}{\parallel D \parallel}$$

$$coY[i] = \frac{-Dx[i]}{\parallel D \parallel}$$

dove D è il vettore (Dx, Dy).
Nel campo *.comb* delle strutture *pointx* e *pointy* si inseriscono i valori:

$$pointx[l].comb[i] = pointx[l].comb[i] + pointx[l].d\_points[0][i] + coX[i]co[i]$$

Il procedimento viene ripetuto per ogni tratto di curva e poi vengono disegnate tutte assieme con colori differenti per poter osservare attentamente sia la il grafico relativo alla curva totale che quello riferito ad un singolo tratto di curva.

2.1.3 Hodograph

Il grafico dell'*hodograph* rappresenta l'andamento delle derivate prime senza rappresentare la curva. Le derivate sono rappresentate attraverso segmenti che partono dall'origine e hanno come modulo il modulo del vettore della derivata prima (indicato in precedenza con $D = (Dx, Dy)$.
Come nel caso della rappresentazione del *Comb*, stabilito il tratto di curva da

disegnare e il corrispondente intervallo parametrico, si crea un vettore di punti da valutare Y e vi si valutano le g+1 curve interpolanti. Allo stesso modo si valutano la derivata prima e le blending e le si miscela seguendo le regole delle derivate. I valori ottenuti sono inseriti nel campo *.hodo* delle strutture *pointx* e *pointy* (vedi appendice A). Si ripete il procedimento per ogni singolo tratto di curva e si rappresenta in un unico grafico con un differente colore per ogni tratto di curva.

2.1.4 Grafico delle tangenti

Il grafico delle tangenti permette di visualizzare la curva finale e la retta tangente alla curva in un determinato punto. Il procedimento per calcolare la tangente della curva è simile a quello utilizzato nel caso del grafico dell'*hodograph*, con la differenza che il valore effettivo che viene inserito nel campo *.tang* delle strutture di disegno è:

$$pointx[l].tang[i] = pointx[l].hodo[i] + pointx[l].d\_points[0][i]$$

dove l è il tratto di curva considerato e i l'*i-esimo* punto di $Y(i)$ In questo modo si rappresenta il grafico delle tangenti disegnando la curva e, per ogni punto di essa un segmento tangente che ha come estremi il punto sulla curva e il corrisponde punto di cordinate $(pointx[l].tang[i], pointy[l].tang[i])$ (vedi appendice A).

2.2 Implementazione del codice

Il programma è costituito da un main principale, *progetto.c*, che fa riferimento alle librerie *G34Math.c* e *G34Lib.c*. Oltre a queste librerie, create appositamente per questo progetto, si utilizza la libreria grafica *GCGraLib.c* che contiene funzioni base per il disegno 2d.
Si faccia riferimento all'Appendice per la descrizione dettagliata delle strutture di dati utilizzate e delle librerie *G34Math.c* e *G34Lib.c*.

Capitolo 3

Sperimentazioni su curve spline

Il progetto descritto nel capitolo 2 è stato utilizzato per testare alcuni file di punti ed è stato così possibile formulare una congettura sulla qualità delle curve al variare delle loro proprietà.
Dato un generico file di dati, il progetto permette di trovare curve interpolanti i punti ottenute in 7 differenti modi, che variano a seconda del grado n delle curve interpolanti e del numero $g + 1$ di funzioni blending utilizzate.
Le curve ottenute si possono suddividere in due gruppi: curve a supporto 6 e curve a supporto 8.
Si verifica che il supporto di una curva è il numero di intervalli che vengono modificati al variare di un punto del file di riferimento: se per esempio si considera una curva a supporto 6, dove n e $g + 1$ sono tali che $n + g + 1 = 6$, se si modifica un punto, cambiano i tratti di curva che interpolano i 5 intervalli precedenti e i 5 successivi, mentre il resto non viene modificato.
Inoltre il supporto della curva indica il minimo numero di punti necessari per poter ottenere un singolo tratto di curva.
Maggiore è il supporto, maggiori sono le informazioni che si possono ottenere dall'osservazione di un file e migliori sono le curve che si ottengono. Purtroppo però non si può aumentare a dismisura il supporto di una curva perchè diventano eccessive le operazioni da eseguire per calcolare i tratti dell'interpolante finale.

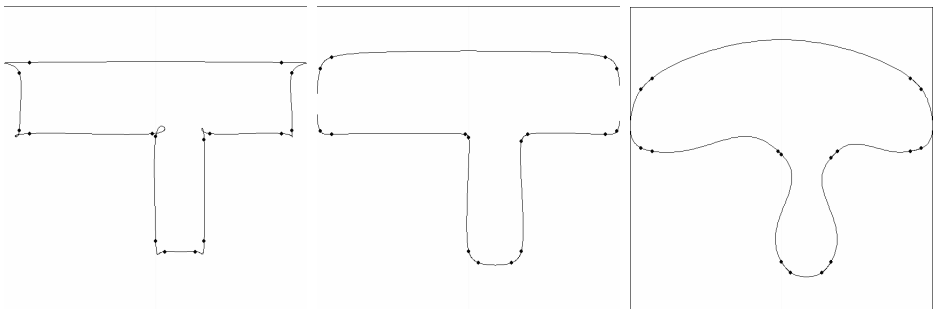

Figura 3.1: sez.dat: parametrizzazioni uniforme, centripeta e cordale

In aggiunta a ciò è possibile calcolare ogni curva con tre parametrizzazioni differenti: *uniforme, centripeta, cordale*, ognuna delle quali permette di ottenere una curva con caratteristiche differenti dalle altre.
Nella prima parte del capitolo si fanno brevi considerazioni sulle tre possibili parametrizzazioni, successivamente invece si tiene fissa la parametrizzazione e si confrontano curve con proprietà differenti.

3.1 Considerazioni sulla parametrizzazione

Si consideri il file *sez.dat* contenente punti che rappresentano la sezione di un braccio meccanico.
Sono fissati il grado della curva e il numero di *B-Spline* utilizzate.
Se si sceglie la parametrizzazione uniforme (figura 3.1 immagine a sinistra) la curva non è necessariamente nè semplice (senza autointersezioni) nè differenziabile (in particolare potrebbe presentare cuspidi).

Implementando la parametrizzazione centripeta (figura 3.1 immagine al centro) si ottiene una curva priva di cuspidi e autointersezioni che però presenta tratti in cui la curvatura è molto elevata misti a tratti con curvatura quasi nulla.
Scegliendo la parametrizzazione cordale (figura 3.1 immagine a destra) si ha una curva di forma più rotondeggiante, con un intervallo di variazione della

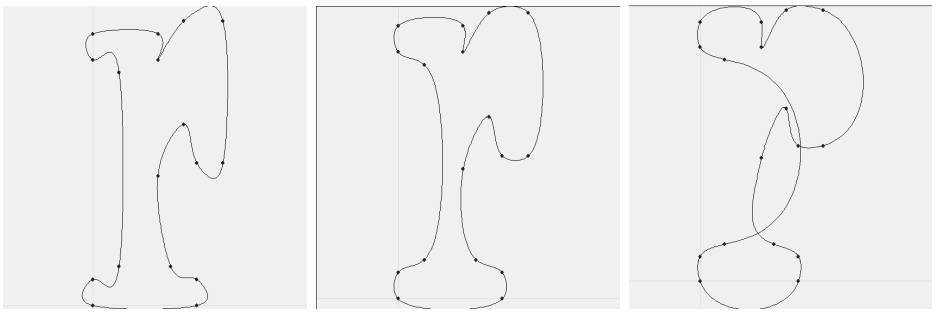

Figura 3.2: r.dat: parametrizzazioni uniforme, centripeta e cordale

curvatura di ampiezza minore rispetto ai casi precedenti.
In questo caso la *parametrizzazione centripeta* è la più adatta per rappresentare questa sezione di braccio meccanico, ma non si può certo affermare che questa parametrizzazione sia migliore delle altre in ogni campo.
Si consideri ad esempio il file *r.dat* che rappresenta la lettera r minuscola . Dal confronto delle curve ottenute (figura 3.2) si vede che la curva che più rappresenta la lettera r è quella ottenuta con la parametrizzazione uniforme. Non esiste quindi una parametrizzazione migliore delle altre, ma la scelta è relativa al tipo di oggetto o forma che si vuole rappresentare (per approfondimenti [4]).

3.2 Analisi sperimentale di file di dati

Questa analisi ha come obiettivo quello di arrivare a formulare una congettura sulla variazione della qualità delle curve al variare dei valori di n e $g+1$.
Si rappresentano curve ottenute utilizzando la *parametrizzazione centripeta*.
Per quanto riguarda le curve a supporto 6 sono implementati tre possibili casi che producono curve di grado 5 con regolarità C e riproducibilità polinomiale P che variano a seconda delle scelte di n e $g+1$ (tabella 3.1).

Per quanto riguarda invece le curve a supporto 8, si studiano quattro

Caso	n	g+1	Proprietà curva risultante
1	3	3	deg=5, C=2, P=3
2	4	2	deg=5, C=1, P=4
3	5	1	deg=5, C=0, P=5

Tabella 3.1: Curve a supporto 6: valori di n e g+1 e proprietà della curva risultante

Caso	n	g+1	Proprietà curva risultante
1	4	4	deg=7, C=3, P=4
2	5	3	deg=7, C=2, P=5
3	6	2	deg=7, C=1, P=6
4	7	1	deg=7, C=0, P=7

Tabella 3.2: Curve a supporto 8: valori di n e g+1 e proprietà della curva risultante

casi che forniscono come risultato curve di grado 7 con rispettive regolarità e riproducibilità (tabella 3.2).
Si consideri inizialmente il file *square.dat* (figura 3.3) che contiene i vertici di un quadrato (i punti sono 5: il primo e l'ultimo sono uguali per indicare che si tratta di una curva chiusa).
Osservando le tre curve finali a supporto 6, esse non presentano differenze visibili e possono indurre a pensare che si tratti della medesima curva. Le differenze diventano però evidenti quando si osservano i grafici del *Comb* e delle tangenti: quando la riproducibilità è minima (colonna 1) si ha un grafico della curvatura C^0 che presenta varie oscillazioni.

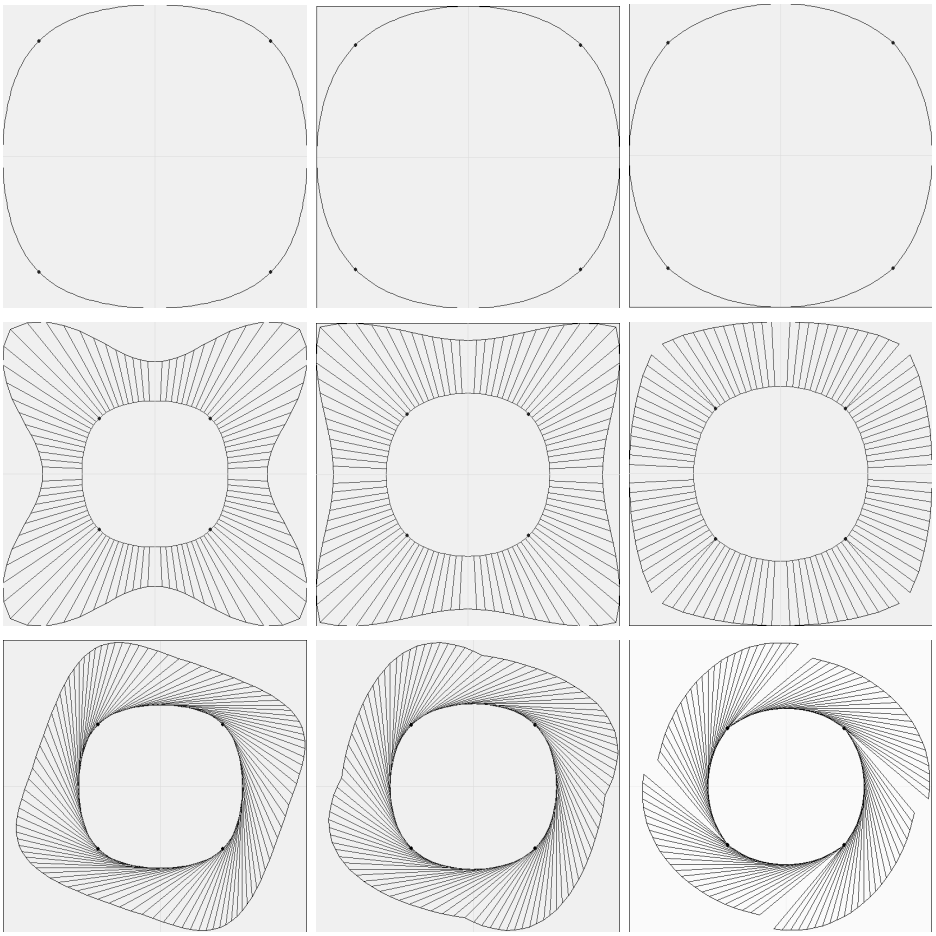

Figura 3.3: *square.dat supp.6: Nelle righe vengono presentati rispettivamente la curva finale, il grafico del Comb e il grafico delle tangenti. Nelle colonne variano i valori di n e g+1 come illustrato nella tabella sottostante*

.	Colonna 1	Colonna 2	Colonna 3
n	3	4	5
g+1	3	2	1

Tabella 3.3: *Proprietà delle curve illustrate nell'immagine sovrastante*

Se si considera separatamente ogni tratto di curva si nota che la curvatura è maggiore ai lati e minore al centro. La variazione del valore della curvatura è ampia e produce quindi una curva con una notevole oscillazione nella parte centrale.
Se si aumenta la riproducibilità (colonna 2) si ottiene un *pettine* in cui le oscillazioni diminuiscono e di conseguenza diminuisce pure la variazione di curvatura nei singoli tratti di curva. Nel caso di riproducibilità massima (colonna 3), in cui la curva interpolante è C^0, si perde la continuità del grafico del *Comb*. Si può osservare che i singoli tratti di curva hanno maggiore qualità rispetto agli altri casi, perchè la variazione di curvatura è minima: questo si vede osservando che il pettine segue l'andamento della curva interpolante.
È possibile osservare un comportamento simile anche nei grafici delle tangenti, nei quali si nota una progressiva diminuzione della variazione delle lunghezze dei vettori tangenti all'aumentare della riproducibilità polinomiale della curva. Considerando le quattro curve ottenute a supporto 8 (figura 3.4), si nota lo stesso comportamento evidenziato per le curve a supporto 6. La curva di massima riproducibilità (ottenuta con $n = 7$ e $g + 1 = 1$) presenta un grafico del *Comb* nel quale la curvatura di ogni singolo tratto ha una variazione minima e inoltre, rispetto alle curve a supporto 6, diminuiscono gli "spazi" tra due tratti di *pettine* consecutivi. Ciò indica come la curva si avvicini maggiormente ad una curva C^1, anche se resta C^0.
Diminuendo la riproducibilità aumentano le variazioni di curvatura e si ha una perdita di qualità della curva, anche se aumenta la regolarità.

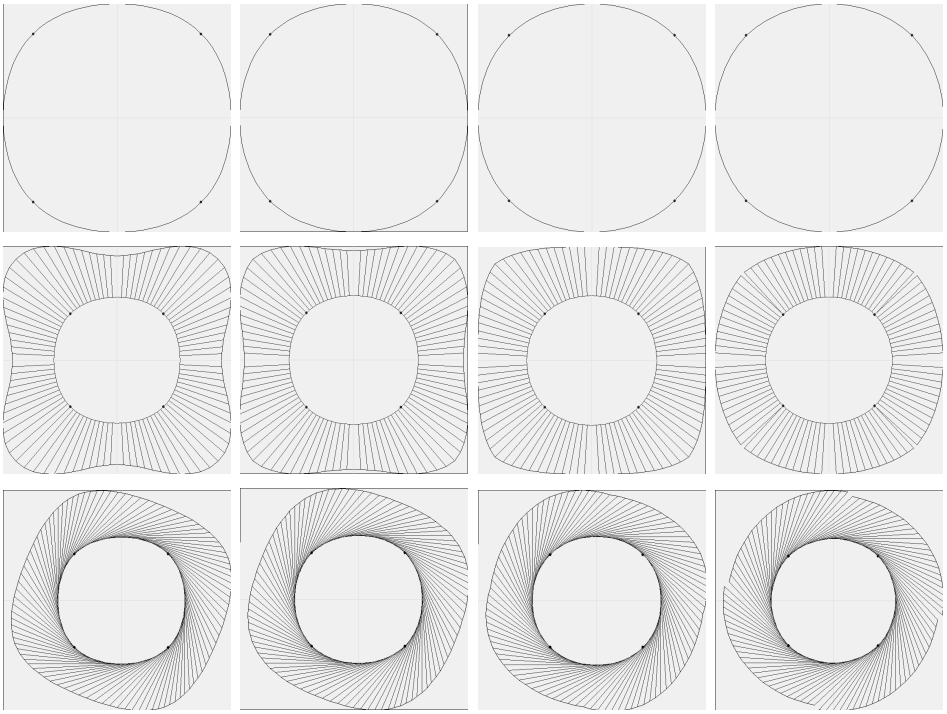

Figura 3.4: *square.dat supp.8: Nelle righe vengono presentati rispettivamente la curva finale, il grafico del Comb e il grafico delle tangenti. Nelle colonne variano i valori di n e g+1 come illustrato nella tabella sottostante*

.	Colonna 1	Colonna 2	Colonna 3	Colonna 4
n	4	5	6	7
g+1	4	3	2	1

Tabella 3.4: *Proprietà delle curve illustrate nell'immagine sovrastante*

Si analizza ora il file *crux.dat* (figura 3.5) che rappresenta una croce greca diritta.
Sono considerate le curve a supporto 8 che forniscono maggiori informazioni. Osservando le curve finali non si notano differenze nei casi 1,2,3 (vedi tabella 3.2), mentre l'ultima curva si differenzia dalle altre perchè presenta tratti più rettilinei ed è C^0 negli spigoli della croce.
Quando si osservano i grafici relativi alle curvature si cominciano a notare le differenze tra le varie curve. La curva di minore riproducibilità (colonna 1) ha un *pettina* almeno C^2 (ovvero la curva che unisce gli estremi dei vettori delle curvature è di classe C^2). I singoli tratti di curva presentano una grande variazione delle curvature, che si traduce in ondulazioni evidenti. Aumentando la riproducibilità e diminuendo di conseguenza la regolarità la qualità della curva aumenta anche se in modo poco evidente (colonna 2).
Il grafico del Comb della curva di regolarità C^1 e riproducibilità 6 (colonna 3) presenta piccole discontinuità in alcuni punti, ma si comincia a vedere un'effettiva diminuzione della variazione delle curvature nei singoli tratti.
L'ultima curva ha un grafico molto significativo (colonna 4), nel quale i singoli tratti di curva hanno una curvatura che varia molto meno rispetto agli altri casi e sono presenti quattro tratti (quelli che indicano le punte della croce) nei quali la variazione di curvatura è quasi nulla. Osservando i grafici delle tangenti si nota lo stesso andamento: all'aumentare della riproducibilità polinomiale della curva si ha una diminuzione della variazione del valore dei vettori delle tangenti e un conseguente aumento della qualità, d'altra parte si ha una perdita di regolarità.

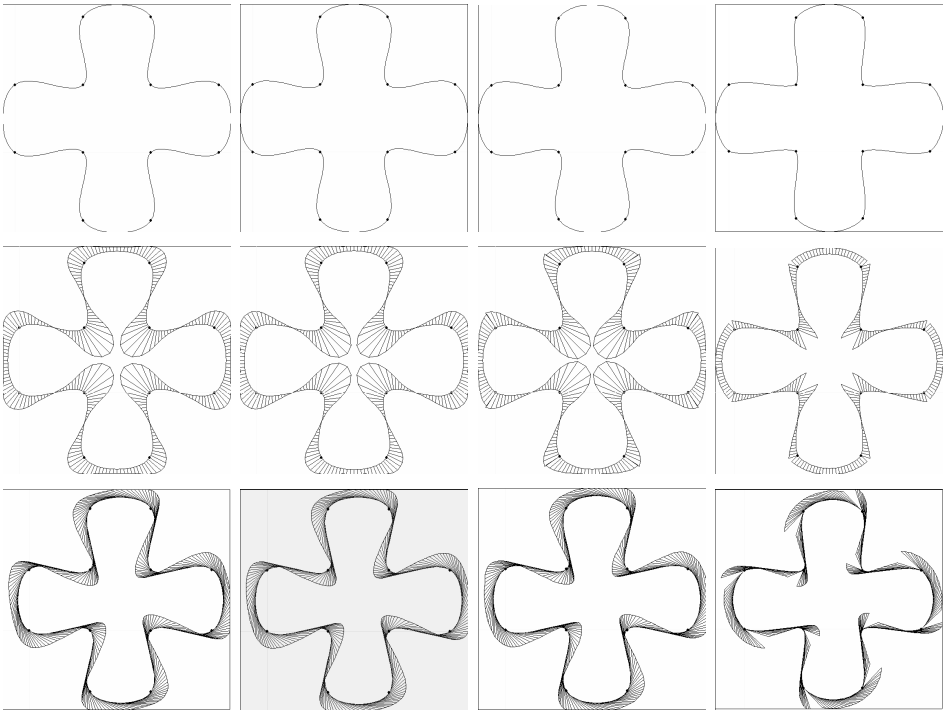

Figura 3.5: *crux.dat supp.8: Nelle righe vengono presentati rispettivamente la curva finale, il grafico del Comb e il grafico delle tangenti. Nelle colonne variano i valori di n e g+1 come illustrato nella tabella sottostante*

.	Colonna 1	Colonna 2	Colonna 3	Colonna 4
n	4	5	6	7
g+1	4	3	2	1

Tabella 3.5: *Proprietà delle curve illustrate nell'immagine sovrastante*

Da ultimo si considera ora il file *x.dat* (figura 3.6), composto da 21 punti e 20 tratti che rappresenta la lettera X dell'alfabeto.
Osservando le curve finali a supporto 6 e 8 non si notano sostanziali differenze nei casi in cui laregolarità è almeno C^1. Nei casi di riproducibilità massima e quindi di regolarità C^0 invece si hanno alcuni tratti in cui la singola curva presenta meno oscillazioni e si avvicina di più ad una retta.
I grafici del *Comb* relativi alle curve con riproducibilità minima (casi 1 per le curve a supporto 6 e 8) presentano variazioni maggiori rispetto agli altri casi.
Ingrandendo una parte della curva (figura 3.8) si può notare che, considerando separatamente i singoli tratti di curva, sono presenti oscillazioni del grafico del *Comb* nella parte centrale della curva stessa, ovvero al centro la curvatura è minore che ai lati.
Aumentando la riproducibilità anche in questo caso diminuiscono le oscillazioni. Nei tratti in cui la curvatura è maggiore permangono lievi oscillazioni nella parte centrale delle curve. Tali oscillazioni vengono eliminate solo nell'ultimo caso sia per le curve a supporto 6 che per quelle a supporto 8. Sono i casi in cui la curva finale ha riproducibilità polinomiale rispettivamente 5 e 7.
Per quanto riguarda i grafici delle tangenti, si ha una diminuzione della variazione del valore dei vettori tangenti nei singoli tratti di curva all'aumentare della riproducibilità. Tale diminuzione risulta più evidente nel passaggio dalla curva C^1 a quella C^0, ovvero le due curve di riproducibilità massima.

.	Colonna 1	Colonna 2	Colonna 3
n	3	4	5
g+1	3	2	1

Tabella 3.6: *Proprietà delle curve illustrate nell'immagine sottostante*

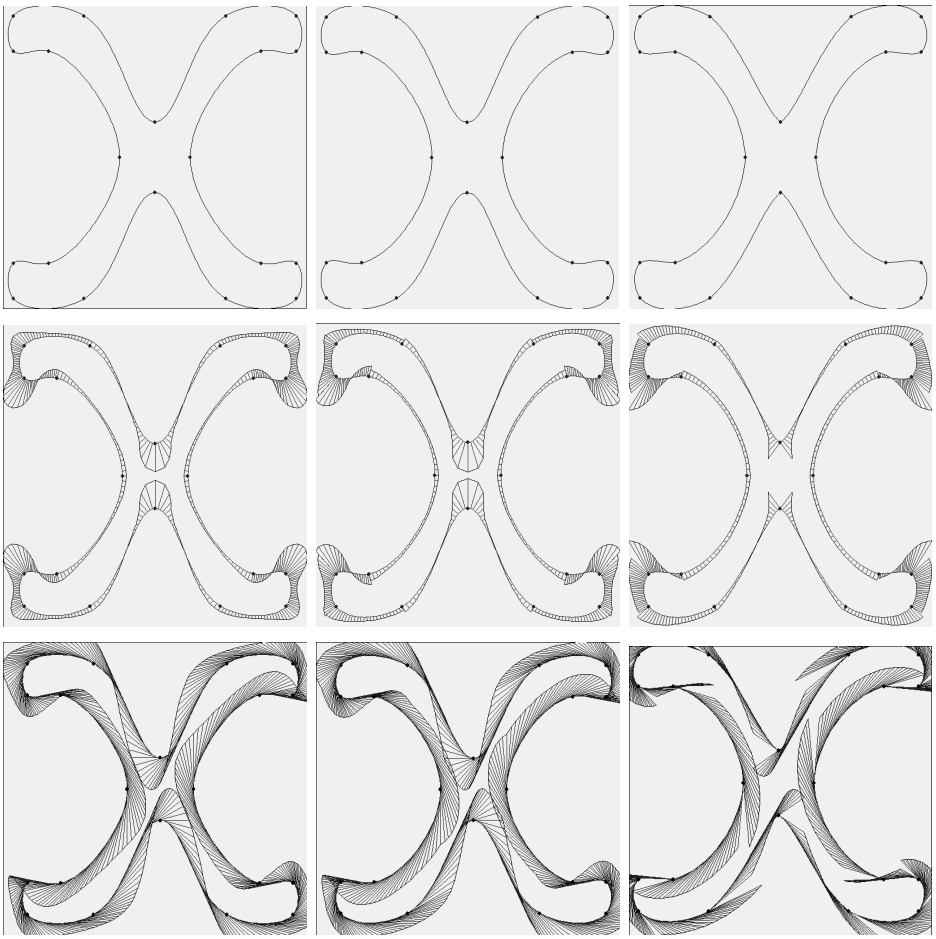

Figura 3.6: *x.dat supp.6: Nelle righe vengono presentati rispettivamente la curva finale, il grafico del Comb e il grafico delle tangenti. Nelle colonne variano i valori di n e g+1 come illustrato nella tabella sovrastante*

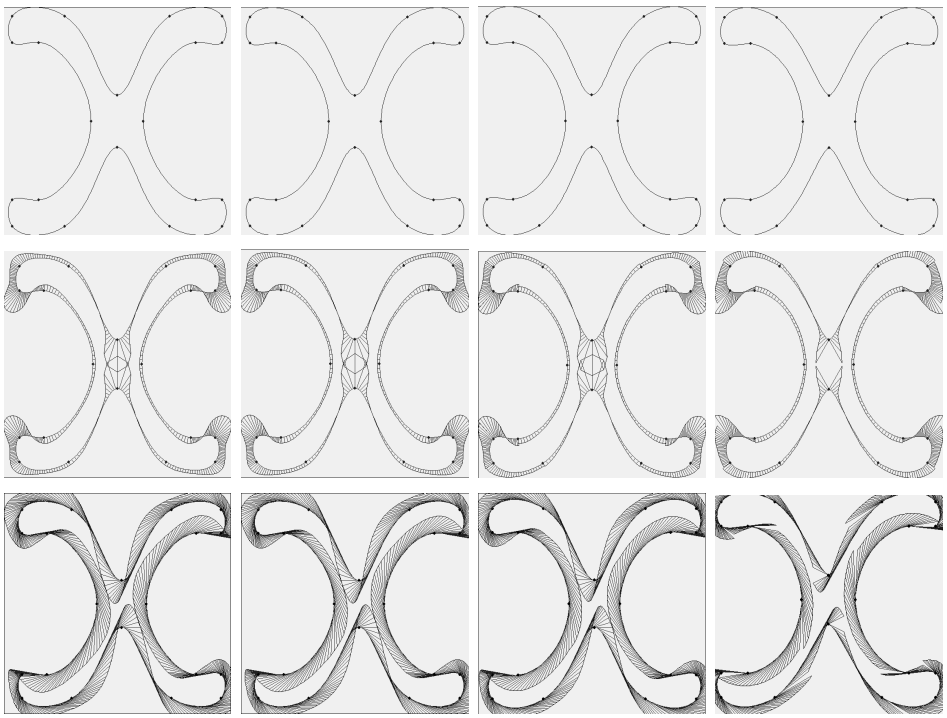

Figura 3.7: *x.dat supp.8: Nelle righe vengono presentati rispettivamente la curva finale, il grafico del Comb e il grafico delle tangenti. Nelle colonne variano i valori di n e g+1 come illustrato nella tabella sottostante*

.	Colonna 1	Colonna 2	Colonna 3	Colonna 4
n	4	5	6	7
g+1	4	3	2	1

Tabella 3.7: *Proprietà delle curve illustrate nell'immagine sovrastante*

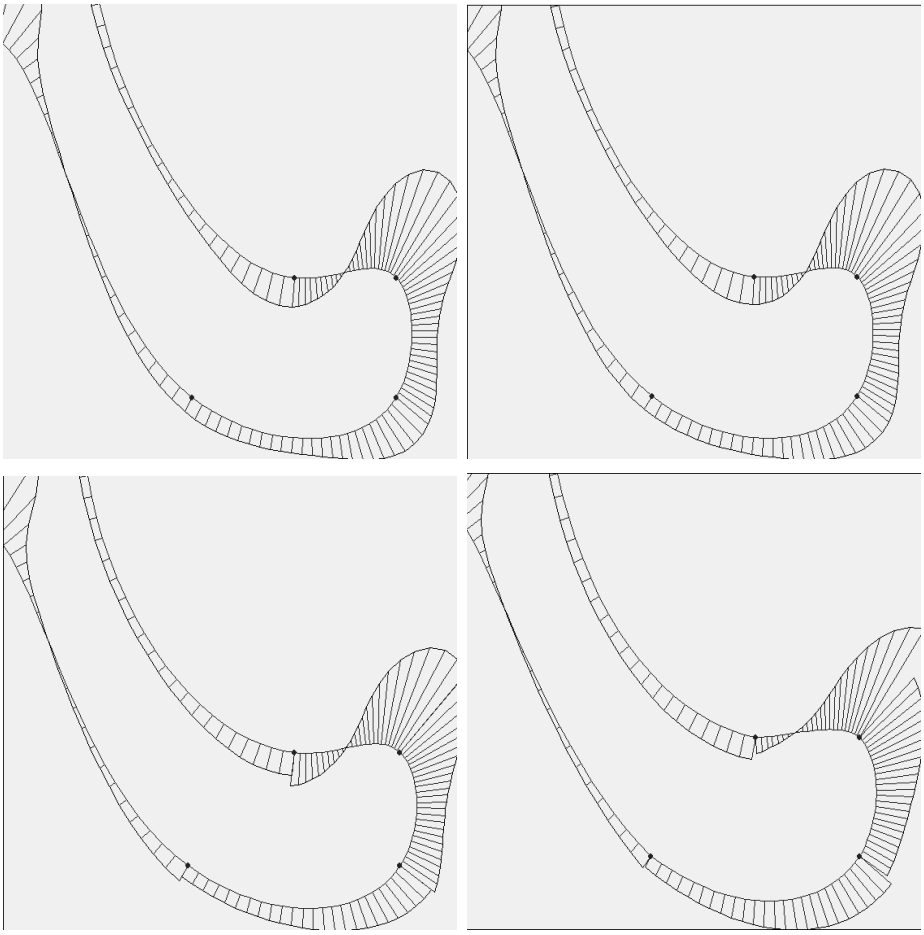

Figura 3.8: x.dat supp.8: zoom grafico Comb

.	Colonna 1	Colonna 2	Colonna 3	Colonna 4
n	4	5	6	7
g+1	4	3	2	1

Tabella 3.8: *proprietè delle Curve illustrate nell'immagine sovrastante*

L'osservazione di questi file induce a formulare la seguente congettura:

Congettura 1. *Si consideri una curva spline di interpolazione a supporto fisso s, di regolarità C e riproducibilità polinomiale P con* $C + P = s$. *Aumentando la riproducibilità della curva aumenta la qualità dei singoli tratti di curva. Allo stesso tempo però la regolarità della spline diminuisce e quindi si hanno grafici del Comb e delle tangenti che finiscono per essere discontinui, risultando indicativi solo per i singoli tratti e non per l'intera curva.*

Tale congettura suggerisce che le curve migliori, tra tutte quelle considerate, sono quelle con regolarità C^1 e riproducibilità 6 per le curve a supporto 8, riproducibilità 4 per quelle a supporto 6. Queste sono un giusto compromesso tra qualità e regolarità.
Se i singoli tratti di curva sono in numero ridotto (ad esempio quattro come nel file *square.dat*), si può anche scegliere una curva con riproducibilità massima, in quanto se si utilizza un supporto abbastanza elevato, le curve che si ottengono sono C^0, ma molto simili a curve C^1.
Se si hanno invece molti tratti di curva conviene utilizzare le curve di regolarità C^1 o al massimo C^2 se il supporto della curva è elevato (pari o superiore a 8).

Capitolo 4

Conclusioni

In questa tesi è stata effettuata un'analisi grafica della qualità di *curve spline interpolanti*.
È stato introdotto un particolare metodo di interpolazione polinomiale a tratti che fa uso di funzioni blending. Applicando tale metodo sono state ottenute curve interpolanti alcuni punti del piano, aventi caratteristiche differenti a seconda dei parametri fissati inizialmente.
È stato poi progettato e implementato un programma in linguaggio C che calcola curve interpolanti con il metodo citato sopra, le rappresenta e permette di visualizzare grafici utili per lo studio della qualità delle curve stesse.
Successivamente il progetto è stato testato su alcuni file: sono state espresse considerazioni sulle parametrizzazioni implementate ed è stato fatto uno studio sulla qualità delle curve, facendo variare le condizioni iniziali quali supporto della curva, grado delle interpolanti e numero di blending utilizzate.
Questo studio ha portato alla formulazione di una congettura sulla qualità delle differenti curve e ha permesso di fare alcune considerazioni su quali fossero le curve migliori tra quelle viste.
È possibile sviluppare ulteriormente questo studio introducendo nuovi metodi di interpolazione che permettano di ottenere curve con proprietà superiori rispetto a quelle viste fino ad ora, pur mantenendo basso il supporto della curva stessa.

Inoltre si può provare ad implementare nuovi indici che permettano di giungere ad ulteriori considerazioni sulla qualità delle curve (per approfondimenti si veda [2]).
Seguendo quest'ultima direzione si entra però in un campo in cui le considerazioni soggettive sono preponderanti rispetto alle formule matematiche. Infatti, come si è detto nell'introduzione a questo lavoro, non è possibile definire matematicamente la qualità di un oggetto, si può solo cercare di fornire indicazioni su ciò che si considera bello o di qualità.

Appendice A

Strutture di dati

Tutte le informazioni necessarie per calcolare e rappresentare le curve e i grafici del programma sono organizzate in due strutture: una struttura contenente informazioni di tipo geometrico chiamata *CURVE* e una struttura contenente tutti i dati necessari per la rappresentazione grafica chiamata *DRAWING*.
La struttura CURVE è composta da 8 campi:

- g: area di memoria contenente il valore intero che indica il grado della curva da disegnare.
- *supp*: contiene l'intero relativo al supporto delle blending functions.
- *nip*: spazio contenente il numero dei punti da interpolare.
- tpf: area di memoria in cui viene salvato il numero totale di punti presenti nel file.
- cpx, cpy: variabili di tipo *vector* (array di tipo *double*) che contengono i valori dei coefficenti delle curve interpolanti (espresse in forma polinomiale).
- ipx, ipy: campi di tipo *vector* contenenti le coordinate dei punti che la curva deve interpolare.

La struttura *DRAWING* è composta da 6 campi:

- p: area di memoria dedicata al numero di punti da disegnare.
- nl: variabile di tipo intero contenente il numero di linee della matrice d_points.
- d_points: variabile di tipo *matrix* (matrice a valori *double*) contenente in ogni riga le valutazioni della curva in un array di punti (ottenuto raffinando uno o più intervalli della parametrizzazione).
- $comb, hodo, tang$: variabili di tipo *vector* contenenti rispettivamente le valutazioni del *comb*, dell'*hodograph* e della tangente in un array di punti ottenuto come in precedenza.

Poiché si ha la necessità di rappresentare più tratti di una stessa curva, considerandoli ognuno come una singola curva, è stato necessario definire per entrambe le strutture una nuova variabile (*POLI* per la struttura *CURVE* e *DRAW* per la *DRAWING*) che è un vettore che ha come elementi le strutture di quel tipo.

Appendice B

G34Math.c

La libreria *G34Math.c* contiene tutte le funzioni create per svolgere operazioni di tipo matematico.
Le funzioni principali contenute in questo file sono:

- $calcola\_curve(curv, a, p, l)$: questa funzione prende in input una struttura dati di tipo *POLI*, chiamata *curv*, il valore a della parametrizzazione, un vettore p contenente la parametrizzazione di tutti i punti del file e un intero l che indica in quale elemento del vettore *curv* devono essere inseriti i valori calcolati.
  La struttura *curv* viene passata alla funzione con i campi $curv[].ipx$, $curv[].ipyecurv[].nip$ (grado della curva +1) già impostati. La funzione calcola i coefficienti della $l-esima$ curva interpolante i punti dei vettori $curv[l].ipxecurv[l].ipy$ e li inserisce nei vettori $curv[l].ipxecurv[l].cpy$. Per fare ciò utilizza due funzioni create per risolvere i sistemi lineari.

- $createx(x, n, x0, xn)$: riceve in input un intero n, un punto iniziale $x0$ e un punto finale xn (di tipo *double*). Crea un vettore x (passato alla funzione inizialmente vuoto) di n elementi equispaziati con punto iniziale $x0$ e finale xn.

- $decomp(n, A, p, *sing)$: vengono passati una matrice A, un intero n che ne indica l'ordine e un vettore p, inizialmente vuoto, che conterrà le

permutazioni tra le righe della matrice. La funzione restituisce il vettore p modificato e un intero *sing* (passato alla funzione per indirizzo) che assume valore 0 se la matrice è singolare, 1 altrimenti.
La funzione calcola le permutazioni necessarie per ridurre poi la matrice A ad una triangolare superiore. Per scambiare tra loro gli elementi della matrice si effettua una chiamata alla funzione *swap* che esegue un semplice scambio utilizzando una variabile di appoggio.

- $solve(n, a, p, x)$: questa funzione, in modo analogo alla decomp prende in input una matrice A con il suo ordine n e un vettore p di permutazioni, questa volta già riempito dalla *decomp*. Viene passato poi un vettore x dei termini noti. La funzione risolve il sistema lineare con matrice dei coefficienti A (triangolare superiore) e vettore dei termini noti x, restituendo il vettore x modificato e contenente le soluzioni del sistema.

- $bernst(n, m, u, B)$: la funzione prende in input un vettore u, due interi n e m che indicano rispettivamente il grado dei *polinomi di base di Bernstein* aumentato di uno e la lunghezza del vettore u. B è una matrice (inizialmente vuota) che nella *i-esima* colonna conterrà gli n valori dei polinomi di Bernstein di grado $n-1$ calcolati nell'*i-esimo* valore del vettore u.

- $Calcolacoeff(n, x, p, A, *sing)$: vengono passati un vettore x a cui è associato l'intero n che ne indica il numero di elementi (che è anche il grado +1 della curva da calcolare), la matrice dei coefficienti A, un vettore p che conterrà le permutazioni di A e l'intero *sing*, passato per indirizzo, che è lo stesso che sarà passato alla *decomp*.
Questa funzione calcola inizialmente la matrice dei polinomi di Bernstein di grado $n-1$ valutati in x. Successivamente la trasposta di tale matrice viene messa in A e si effettua una chiamata alla *decomp*. In questo modo la funzione restituisce la matrice A triangolare superiore,

il vettore p delle permutazioni e l'intero *sing* che indica la singolarità della matrice A.

- $Calcolapunti(n, m, x, px, bc)$: la funzione prende in input n, grado $+1$ della curva interpolante, m, numero di punti in cui viene valutata la curva, x vettore dei punti da valutare, bc vettore dei coefficienti della curva e px, vettore inizialmente vuoto nel quale verranno messi i valori dei punti valutati. Crea una matrice di Bernstein di ordine $n \times m$ e poi valuta il polinomio espresso nella *base di Bernstein* in ogni punto di x.

- $param(n, v, a, x, y)$: riceve in input un intero n che indica il numero di punti della parametrizzazione, un *double* a che è il coefficente che permette di stabilire il tipo di parametrizzazione da utilizzare e due vettori x e y che contengono i punti da interpolare. Viene poi passato un vettore (inizialmente vuoto) v in cui verranno messi i valori della parametrizzazione.
  La parametrizzazione è calcolata seguendo esattamente il procedimento spiegato nel capitolo 1.

- $Calcolapuntiderivate(n, m, X, x1, y1, cdx, cdy, cd2x, cd2y, Dx, Dy,$.
  $D2x, D2y)$: la funzione prende in input gli interi n e m (che sono gli stessi passati alla funzione calcolapunti), il vettore X dei punti da valutare e i vettori $x1$ e $y1$ contenenti i coefficienti delle curve interpolanti. Inoltre vengono passati inizialmente vuoti alcuni vettori, che, quando ritorneranno alla funzione chiamante conterranno i valori dei coefficenti delle derivate prima e seconda $(cdx, cdy, cd2x, cd2y)$. Gli altri vettori $(Dx, Dy, D2x, D2y)$ conterranno i valori effettivi dei polinomi di Bernstein derivati, valutati nei punti di X.
  La funzione crea inizialmente i coefficienti dei polinomi derivate (seguendo quanto detto nel capitolo 1 sezione dedicata ai polinomi espressi nella *base di Bernstein*), poi viene chiamata la funzione *Calcolapunti* e vengono creati i vettori $Dx, Dy, D2x, D2y$.

- $Curvatura(n, m, Dx, Dy, D2x, D2y, co)$: in maniera analoga alla funzione precendente vengono passati n e m e i vettori $Dx, Dy, D2x, D2y$ che però ora contengono i valori calcolati in *Calcolapuntiderivate*. Infine viene passato un vettore vuoto co, che viene riempito con i valori ottenuti dal calcolo della formula della curvatura.

- $b\_spl(t, u, g, *d, *h, bs)$: la funzione riceve in input t vettore nodale, u punto in cui viene calcolata la *B-Spline* e g grado-1 della *B-Spline*. Nel vettore bs vengono inseriti i valori delle *B-Spline* calcolate in u. Vengono poi passati per indirizzo d e h che sono interi che conterranno rispettivamente l'indice del primo e dell'ultimo valore della *B-Spline* non nullo.

- $bp\_spl(t, u, g, *d, *h, bs, bs1)$ e $bps\_spl(t, u, g, *d, *h, bs, bs1, bs2)$: funzioni simili alla precedente, che calcolano rispettivamente derivata prima e derivate prima e seconda delle *B-Spline* in u e le mettono nei vettori $bs1$ e $bs2$.

- $dicotomica(g, nt, t, u)$: la funzione prende in input g grado delle *B-Spline*, t vettore nodale, nt numero di punti di t e u valore da cercare. Calcola tra quali due valori di t è compreso u e restituisce alla funzione chiamante l'indice del sottointervallo a cui appartiene.

Appendice C

G34Lib.c

La libreria *G34Lib.c* è il file principale del progetto. In essa sono contenute tutte le funzioni di pulizia, costruzione menù, lettura file e rappresentazione grafica.

Funzioni di pulizia

- $Clear(screen, larghezza, altezza, u, colore\_sfondo)$: vengono passate in input le variabili $larghezza$ e $altezza$ che permettono di determinare un rettangolo, la variabile $colore\_sfondo$, la variabile relativa alla schermata e l'unità di misura u. La funzione "pulisce" lo schermo disegnando un rettangolo del colore dello sfondo.

- $Clear1(screen, larghezza, altezza, u, colore\_sfondo, v)$: molto simile alla funzione precedente. L'unica differenza consiste nel fatto che viene pulito tutto lo schermo con l'eccezione di un rettangolo v.

Funzioni di disegno e lettura file

- $wind\_view(px, py, *ix, *iy, view, win)$: la funzione riceve in input i valori *double* delle coordinate px e py di un punto, la struttura di tipo *VIEWPORT view* che rappresenta il rettangolo di disegno e la struttura di tipo *WINDOW win* che rappresenta il minimo rettangolo contenente

tutti i punti da disegnare. Restituisce i valori interi ix e iy, passati per indirizzo.
Viene effettuata una conversione da *window* a *viewport*, ovvero vengono calcolate le coordinate che il punto (px, py) dovrebbe avere nella viewport per poter ottenere una rappresentazione realistica della curva.

- $set\_view(n, *rect, x, y)$: vengono passati i vettori di punti x e y e il numero intero n che ne indica la lunghezza. Inoltre viene passata per indirizzo la struttura di tipo *WINDOW rect*.
  La funzione imposta le coordinate del minimo rettangolo contenente tutti i punti da visualizzare. In questo modo quando viene effettuata una chiamata alla *wind_view* è possibile adattare il disegno ad un'area grafica ristretta come è la *viewport*.

- $min\_square(x, y, xs, ys)$: vengono passati due vettori x e y contenenti gli estremi di un rettangolo (o quello della *window*, oppure un rettangolo selezionato dall'utente nel caso dello zoom).
  Viene calcolato il minimo quadrato contenente il rettangolo dato in input e gli estremi di tale quadrato vengono inseriti nei vettori xs e ys.

- $leggi\_data(*n, x, y, myfile)$: la funzione riceve in input la stringa *myfile* contenente il nome del file da aprire in modalità lettura. Una volta aperto il file viene inserito in n il numero di punti presenti nel file e in x e y le coordinate di ognuno di questi punti.

- $draw\_data\_line(*s, c, n, j, view, win, x, y)$: viene passato il puntatore alla superficie SDL s, un colore (variabile intera) c, il numero n di punti da disegnare, i rettangoli $view$ e win, le matrici dei punti x e y, con anche il valore intero j che indica quale riga delle matrici bisogna considerare.
  Si disegna una curva come unione di tante piccole linee che congiungono punti poco distanti tra loro. Considerato il primo punto, si chiama la funzione *wind_view* che effettua un adattamento alla *viewport*, poi si

entra in un ciclo for in cui si prende il secondo punto,lo si adatta alla *viewport* e si disegna la linea che congiunge i due punti. Lo stesso ragionamento viene ripetuto poi per il secondo e il terzo punto, e per ogni coppia di punti successivi, fino alla fine dei vettori x e y.

- $draw\_data\_comb(*s, c, d, n, j, view, win, x, y, cx, cy)$: la funzione è simile alla precedente con la differenza che viene passata una variabile d relativa al secondo colore e i vettori cx e cy. Questa funzione permette di rappresentare una curva e il suo *Comb*.

- $draw\_data\_hodo(*s, c, d, n, j, view, win, x, y)$: funzione simile alla *draw_data_line*. Disegna il grafico dell'*Hodograph* di una curva.

- $draw\_data\_tang(*s, c, d, n, j, view, win, x, y, cx, cy)$: Disegna la curva e le tangenti alla curva.

- $dividing(m, x, y, curv, degree, g)$: vengono passati in input il numero di punti presenti nel file, i vettori x e y contenenti tali punti. Si passano inoltre la struttura $curv$ e le variabili $degree$ e g che indicano rispettivamente grado e supporto -1 delle *blending function*.
  Vengono distribuiti all'interno della struttura $curv$ i vari punti contenuti nel file, in modo tale da rappresentare una curva chiusa se il primo e l'ultimo punto del file coincidono, una curva aperta altrimenti.
  Questa suddivisione non è scontata se si vuole rappresentare curve chiuse, poiché a seconda del grado della curva e del supporto delle *blending* è necessario aggiungere punti in testa o in coda ai vettori x e y. Vengono poi aggiornati x e y aggiungendo tali punti e riempiti i campi $curv[].nip, curv[].supp, curv[].tpf$.

- $file(*screen, colore\_menu, ok, myfile, wh, choice, sign, curve, degree, g, x, y)$: la funzione gestisce l'inserimento del nome e la successiva lettura del file di dati necessario per disegnare una curva. In aggiunta a ciò vengono disegnate sulla schermata le istruzioni che permettono all'utente di: scegliere una parametrizzazione, disegnare un particolare

tratto di curva e fare lo zoom.
Alla fine viene chiamata la funzione *dividing*, così che, una volta chiamata la funzione dal main, l'utente abbia già la possibilità di procedere col disegno delle curve.

Funzioni menù e scelta

- *menù_principale, impostazioni, menù_ scelta_param, menù_scelta_curva*: sono funzioni che hanno la stessa struttura: viene creata una matrice contenente le coordinate dei vari punti del menù e successivamente vengono disegnati i cerchi e le scritte così che l'utente possa scegliere graficamente quale azione intraprendere.

- $scelta\_a, scelta\_l, scelta\_nm$: funzioni costituite principalmente da un ciclo di gestione degli eventi che si interrompe quando viene scelta un'opzione del menù (nel caso di *scelta_a* e *scelta_l*) o quando si decide di uscire (*scelta_nm*).

Funzioni principali

- $interp\_blending\_curve(degree, g, a, l, curv, x, y, pt, pointx, pointy)$: vengono presi in input gli interi relativi a grado (*degree*) e supporto (g), il valore a relativo alla parametrizzazione, la struttura *curv*, i vettori x, y e *pt* (che indica la parametrizzazione) e le strutture *DRAW pointx* e *pointy* con relativo intero l che indica su quale elemento dei vettori di tipo *DRAW* lavorare.
  La funzione calcola i valori della funzione polinomiale che interpola una coppia di punti successivi utilizzando il metodo dell'*interpolazione polinomiale a tratti* esposto nel capitolo 1. I punti ottenuti vengono messi nei vettori $pointx[l].d\_points$ e $pointy[l].d\_points$. Infine vengono inizializzati i campi $pointx.p$ e $pointx.nl$.

- *interp_blending_comb*, *interp_blending_hodo*, *interp_blending_tang*: sono simili come struttura alla funzione *interp-blending_curve*, e permettono di calcolare *Comb*, *Hodograph* e *tangenti* delle varie curve.

- $curve(*screen, v, sub\_v, fun\_view, color, sfondo, wh, myfile, curv,.$ $a, l, pointx, pointy)$: funzione che gestisce il disegno di un singolo tratto di curva. Prende in input le strutture *curv*, *pointx* e *pointy* contenenti i valori da disegnare. Questi valori vengono copiati in due vettori CBX e CBY che vengono utilizzati per settare i parametri della window. Successivamente vengono disegnati i punti e le curve da rappresentare. La funzione permette di zoomare un'area della *viewport* e visualizzarla oppure di scegliere un'altra opzione del menù principale.

- $final\_curve(*screen, v, sub\_v, fun\_view, color, sfondo, wh, myfile,.$ $curv, a, l, pointx, pointy)$: le variabili passate sono le stesse della funzione *curve*. La funzione permette di visualizzare l'intera curva interpolante. I singoli tratti di curva vengono rappresentati con colori differenti in modo da rendere evidente che si tratta di un'unione di curve.

- $comb(*screen, v, sub\_v, Cv, fun\_view, comb\_view, color, sfondo, wh,$ $myfile, *sign, curv, a, l, pointx, pointy)$: la funzione gestisce il disegno del grafico del *Comb* di una intera curva interpolante, quindi permette di disegnare ogni singolo tratto di curva con il suo *Comb*.

- $hodograph(*screen, v, sub\_v, Cv, fun\_view, hodo\_view, color, sfondo,$ $wh, myfile, *sign, curv, a, l, pointx, pointy)$: disegna il grafico dell'hodograph dell'intera curva.

- $tang(*screen, v, sub\_v, Cv, fun\_view, tang\_view, color, sfondo, wh,$ $myfile, *sign, curv, a, l, pointx, pointy)$: permette di rappresentare la curva con le rette tangenti.

- $zoom(*screen, sub\_v, fun\_view, fun\_win, color, wh, cx, cy, m, xz, yz)$: vengono presi in input i vettori *cx* e *cy* (di tipo *maxvettore* ovvero vet-

tori con massimo 1500 elementi) contenenti tutti i punti da disegnare, l'intero m che rappresenta la lunghezza di questi vettori, le variabili relative alla schermata, alla *viewport* e alla *window*, il vettore dei colori e l'unità di misura. La funzione ritorna due vettori (passati per indirizzo) xz e yz contenenti le coordinate del quadrato da zoomare.
Inizialmente la funzione gestisce la scelta da parte dell'utente di un'area da zoomare, ottenuta cliccando due volte sulla *viewport*. Si ottengono così 4 valori della *viewport* (2 ascisse e 2 ordinate). Successivamente si cercano all'interno dei vettori cx e cy i punti che (adattati alla *viewport*) hanno minima distanza da almeno uno dei 4 valori ottenuti e si salvano in xz e yz. Infine viene calcolato il minimo quadrato contenente il rettangolo trovato chiamando la *min_square* che ritornerà alla funzione chiamante.

Bibliografia

[1] C. DeBoor; *A practical Guide to Splines*, 1978 Springer-Verlag. pp. 113-114

[2] G. Farin; *Class A Bézier curves*, 2006 Computer Aided Geometric Design 23 pp. 573-581.

[3] *Libreria Grafica SDL* http://www.libsdl.org .

[4] C. Yuksel, S. Schaefer, J. Keyser; *On the Parameterization of Catmull-Rom Curves*, 2009 SIAM/ACM Joint Conference on Geometric and Physical Modelling.

Ringraziamenti

Vorrei ringraziare infinitamente il professor Casciola, che mi ha seguito con grande pazienza durante tutto questo percorso, correggendomi e consigliandomi sempre la strada giusta da seguire.
Vorrei poi ringraziare tutta la mia famiglia, per il sostegno che mi ha sempre dato, e soprattutto per aver creato quell'angolo di paradiso in cui è stata scritta quasi tutta la tesi.
Infine ringrazio tutti i miei amici, che mi sono sempre vicini e ogni giorno rendono piacevole questo percorso.