Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Progressive Iteration Approximation: una classe di metodi per interpolazione e approssimazione spline

Relatore:
Chiar.mo Prof.
GIULIO CASCIOLA

Presentata da:
GIACOMO ORSI

Sessione I
Anno Accademico 2020/2021

Introduzione

La disciplina della *Computer-Aided Geometric Design* (CAGD) si occupa della descrizione della forma e degli aspetti computazionali degli oggetti geometrici. In molte applicazioni della CAGD è necessario interpolare o approssimare un insieme di dati con una curva o una superficie parametrica. La letteratura scientifica ha proposto diversi metodi per risolvere questi problemi in modo efficiente. Lo scopo di questa tesi è quello di introdurre alcuni di questi metodi di interpolazione e di approssimazione e di analizzarne le differenze. In particolare, si studia una famiglia di metodi che sfrutta la proprietà di *progressive iteration approximation* di una curva o di una superficie. Oltre alla definizione dei concetti di base dei tipi di curve utilizzati per interpolare o approssimare i dati e alla breve introduzione dei metodi numerici tradizionali impiegati per risolvere questo tipo di problemi, si presentano diversi algoritmi iterativi che, a differenza dei metodi tradizionali, sfruttano le proprietà geometriche delle curve interpolanti o approssimanti. Inoltre, si presenta una sintesi di alcuni risultati numerici ottenuti in ambiente MATLAB a seguito dello studio, dell'implementazione e della sperimentazione degli algoritmi analizzati.

Indice

Capitolo 1

Curve B-Spline

In questa sezione si introducono le funzioni B-Spline, che verranno utilizzate nei capitoli successivi per costruire curve di interpolazione e di approssimazione. In realtà, gli algoritmi illustrati trovano applicazione anche in altri spazi di funzioni, quando la loro base soddisfa alcune proprietà (la base di Bernstein per lo spazio dei polinomi è una di queste). Tuttavia, visto l'impiego estensivo delle B-Spline nelle applicazioni pratiche, si è scelto di focalizzare l'analisi su queste.

1.1 Funzioni spline polinomiali

Una funzione spline è una funzione costituita da un insieme funzioni polinomiali dello stesso grado raccordate in modo "automatico" tra loro con una certa continuità, al fine di ottenere una forma morbida e senza bruschi cambiamenti nella pendenza.

Definizione 1.1.1 (Spline)**.** Sia $[a, b]$ un intervallo chiuso e limitato, $\mathbf{\Delta} = \{x_i\}_{i=1}^{k}$ un insieme di punti tale che

$$a = x_0 < x_1 < x_2 < \cdots < x_{k+1} = b \ ;$$

sia inoltre assegnato un intero positivo n (grado della spline) ed un vettore di interi non negativi $M = \{\mu_i\}_{i=1}^{k}$, con $0 \leq \mu_i \leq n-1$. Una funzione *spline* $s(x)$

di grado n con continuità μ_i nei nodi x_i con $i = 1, \ldots, k$ è una funzione su $[a, b]$ tale che:

(i) in ogni sotto intervallo $[x_i, x_{i+1})$ con $i = 0, \ldots, k$ la funzione $s(x)$ è un polinomio di grado n

(ii) la funzione $s(x)$ in ogni nodo x_i ha continuità μ_i, ossia

$$s_{i-1}^{(l)}(x_i) = s_i^{(l)}(x_i), \quad l = 0, \ldots, \mu_i, \quad i = 1, \ldots, k$$

Con $S(\mathbb{P}_n, M, \boldsymbol{\Delta})$ si indica l'insieme delle funzioni spline con vettore delle continuità. Si può dimostrare che tale insieme è uno spazio di funzioni di dimensione $n + K + 1$ dove

$$K = \sum_{i=1}^{k} (n - \mu_i)$$

Una spline a massima continuità nei nodi è una spline dove $\mu_i = n - 1,\ i = 1, \ldots, n$.

Come vedremo, la distribuzione dei nodi all'interno della partizione nodale e la continuità fissata in essi influenza notevolmente l'aspetto della spline e i risultati dei metodi geometrici di interpolazione e approssimazione. Con partizione nodale *uniforme* si intende quando i nodi interni della partizione $\boldsymbol{\Delta}$ sono equispaziati, *non uniforme* altrimenti.

1.1.1 B-Spline

Con il termine B-Spline ci si riferisce sia a delle particolari funzioni base dello spazio S, che alle funzioni espresse come combinazione lineare di queste funzioni base. Queste spline hanno diverse proprietà che verranno sfruttate nei metodi presentati nei capitoli successivi.

Ogni elemento dello spazio $S(\mathbb{P}_n, M, \boldsymbol{\Delta})$, cioè ogni funzione spline, può essere rappresentata come combinazione lineare di funzioni base $N_{i,n}$ dello spazio S, ovvero:

$$s(x) = \sum_{i=1}^{n+K+1} c_i N_{i,n}(x), \quad x \in [a, b] \tag{1.1.1}$$

A tale scopo introduciamo il concetto di partizione estesa e su di essa definiamo le funzioni base B-Spline $N_{i,n}(x)$.

Definizione 1.1.2. L'insieme $\mathbf{\Delta}^* = \{t_i\}_{i=1}^{2n+K+2}$, con $K = \sum_{i=1}^{k}(n-\mu_i)$ si chiama partizione estesa associata ad $S(\mathbb{P}_n, M, \mathbf{\Delta})$ se e solo se:

- $t_1 \leq t_2 \leq \cdots \leq t_{2n+K+2}$
- $t_{n+1} = a, \quad t_{n+K+2} = b$
- $t_{n+2} \leq \cdots \leq t_{n+K+1} \equiv (\underbrace{x_1 = \cdots = x_1}_{n-\mu_1 \text{volte}} < \cdots < \underbrace{x_k = \cdots = x_k}_{n-\mu_k \text{volte}})$

Sia $\mathbf{\Delta}^*$ la partizione estesa associata allo spazio $S(\mathbb{P}_n, M, \mathbf{\Delta})$. L'insieme delle funzioni base B-Spline normalizzate $\{N_{i,n}\}_{i=1}^{n+1+K}$ è definito dalla formula ricorrente

$$N_{i,n}(x) = \frac{x - t_i}{t_{i+n} - t_i} N_{i,n-1}(x) + \frac{t_{i+n+1} - x}{t_{i+n+1} - t_{i+1}} N_{i+1,n-1}(x) \tag{1.1.2}$$

dove

$$N_{i,0}(x) = \begin{cases} 1 & t_i \leq x < t_{i+1} \\ 0 & \text{altrimenti} \end{cases}$$

dove le funzioni non definite devono essere considerate nulle e i casi $0/0$ devono essere considerati come 0.

1.1.2 Proprietà delle funzioni B-Spline

Le funzioni B-Spline hanno le seguenti proprietà:

(i) *Supporto locale*: $N_{i,n}(x) = 0, \quad \forall x \notin (t_i, t_{i+n+1})$

(ii) *Non negatività*: $N_{i,n} > 0, \forall i, n, \quad x \in (t_i, t_{i+n+1})$

(iii) *Partizione dell'unità*: $\sum_{i=1}^{n+K+1} N_{i,n}(x) = 1, \forall x \in [a, b]$

Per la proprietà di supporto locale, si ha che per ogni $x \in [t_l, t_{l+1})$, $s(x)$ è data dalla somma di al più $n+1$ funzioni base B-Spline. Di conseguenza, cambiando un

qualsiasi coefficiente c_i, la forma della funzione cambia solo su $n+1$ intervalli nodali consecutivi. Quindi, la (1.1.1) diventa

$$s(x) = \sum_{i=l-n}^{l} c_i N_{i,n}(x), \quad x \in [t_l, t_{l+1}).$$

1.2 Curve B-Spline

Dopo aver definito le funzioni spline possiamo dare la definizione di curva spline che verrà utilizzata nei capitoli successivi per costruire delle curve di interpolazione e approssimazione di insiemi di punti.

Definizione 1.2.1. Dato uno spazio spline $S(\mathbb{P}_n, M, \mathbf{\Delta})$ e i punti $\mathbf{P}_i = (x_i, y_i), i = 1, \ldots, ncp$ in $\mathbb{R}^2$, dove ncp è il numero di punti di controllo, la curva spline di grado n e punti di controllo $\mathbf{P}_i$ è data da:

$$\mathbf{C}(t) = \sum_{i=1}^{ncp} \mathbf{P}_i N_{i,n}(t) = \begin{pmatrix} \sum_{i=1}^{ncp} x_i N_{i,n}(t) \\ \sum_{i=1}^{ncp} y_i N_{i,n}(t) \end{pmatrix} \qquad t \in [a, b] \tag{1.2.1}$$

Se la partizione estesa non presenta nodi interni (e quindi $ncp = n + 1$) e i nodi aggiuntivi sono coincidenti, ovvero $\mathbf{\Delta}^* = \{0, \ldots, 0, 1, \ldots, 1\}$, allora la curva spline è una curva di Bézier. Di conseguenza, le curve di Bézier sono un particolare tipo di spline.

1.2.1 Proprietà

Una curva spline di grado n in $[a, b]$ con nodi $\mathbf{\Delta}$ e nodi aggiuntivi coincidenti gode di diverse proprietà. Di seguito elenchiamo quelle che sfrutteremo successivamente:

(i) La curva interpola il primo e l'ultimo punto di controllo

$$\mathbf{C}(a) = \mathbf{P}_1 \qquad \mathbf{C}(b) = \mathbf{P}_{ncp}$$

(ii) La curva è continua e ha derivate continue C^{μ_i} nei nodi interni x_i

(iii) La curva è contenuta nel *guscio convesso globale* formato dai suoi punti di controllo ed è contenuta nel *guscio convesso locale* di quel tratto.

1.2.2 Valutazione

La valutazione di una curva spline in corrispondenza di un valore del parametro $\bar{t} \in [a, b]$ prevede:

(i) la ricerca dell'intervallo nodale $[t_l, t_{l+1})$ contenente $\bar{t}$

(ii) la valutazione in $\bar{t}$ del tratto polinomiale che definisce $\mathbf{C}(t)$ in $[t_l, t_{l+1})$

Una volta individuato l'intervallo $[t_l, t_{l+1})$ contenente $\bar{t}$, per valutare $\mathbf{C}(\bar{t})$ è sufficiente valutare

$$\mathbf{C}(\bar{t}) = \sum_{i=l-n}^{l} P_i N_{i,n}(\bar{t})$$

utilizzando la formula ricorrente (1.1.2) o l'algoritmo proposto da Carl de Boor. L'algoritmo di de Boor risulta una generalizzazione per curve spline dell'algoritmo di de Casteljau per curve di Bézier e prevede di calcolare in modo ricorrente il valore della curva in $\bar{t}$ senza usare le funzioni base ma utilizzando la seguente formula ricorsiva

$$P_i^j(\bar{t}) = \frac{(\bar{t} - t_i)P_i^{j-1}(\bar{t}) + (t_{i+n-j+1} - \bar{t})P_{i-1}^{j-1}(\bar{t})}{t_{i+n-j+1} - t_i}$$

per $j = 1, \ldots, n$ e $i = l-n+j, \ldots, l$, dove inizialmente $P_i^0 := P_i, \quad i = l-n, \ldots, l$

1.2.3 Knot insertion

Una caratteristica interessante di una curva spline $\mathbf{C}(t)$ in uno spazio S è che è possibile rappresentarla esattamente in uno spazio $\hat{S}$ ottenuto da S inserendo un nodo $\hat{t}$. Dunque, è possibile aggiungere un nuovo nodo e assicurarsi che la curva rimanga invariata. Inserire un nuovo nodo nella partizione nodale comporta l'inserimento di un nuovo punto di controllo e la modifica di quelli esistenti.

Esiste un algoritmo proposto da Böhm che descrive come inserire un nuovo nodo nella partizione nodale, come determinare il nuovo punto di controllo e come aggiornare quelli esistenti.

Sia $\{t_i\}_{i=1}^{2n+K+2}$ con $K = \sum_{i=1}^{k}(n-\mu_i)$ la partizione estesa in S e $\hat{t} \in [t_l, t_{l+1})$ il nuovo nodo da inserire. La nuova partizione estesa in $\hat{S}$ è data da $\{\hat{t}_i\}_{i=1}^{2n+\hat{K}+2}$ con $\hat{K} = \sum_{i=1}^{\hat{k}} n - \hat{\mu}_i$ dove

$$\hat{t}_i = \begin{cases} t_i & i \leq l \\ \hat{t} & i = l+1 \\ t_{i-1} & i > l+2 \end{cases}$$

Sia $\mathbf{C}(t)$ la curva spline in S

$$\mathbf{C}(t) = \sum_{i=1}^{n+K+1} c_i N_{i,n}(t)$$

allora per knot insertion sarà

$$\mathbf{C}(t) = \sum_{i=1}^{n+\hat{K}+2} \hat{c}_i \hat{N}_{i,n}(t)$$

dove

$$\hat{c}_i = \begin{cases} c_i & i \leq l-n \\ (1-\lambda_i)c_{i-1} + \lambda_i c_i & l-n+1 \leq i \leq l+1 \\ c_{i-1} & i > l+1 \end{cases}$$

con

$$\lambda_i = \frac{\hat{t} - t_i}{t_{i+n+1} - t_i}, \quad \hat{t} \in [t_l, t_{l+1})$$

Capitolo 2

Interpolazione spline

Con il termine *interpolazione* si intende il problema di determinare una funzione o una curva che passi per un insieme di punti fissato. Nei prossimi paragrafi si presenta il problema di interpolazione di spline e quello di curve e si introduce brevemente il metodo risolutivo tradizionale. Nel capitolo 2.3 si presenta l'algoritmo iterativo che permette di giungere alla stessa soluzione del metodo tradizionale ma sfrutta le proprietà geometriche della curva interpolante.

2.1 Interpolazione con funzioni spline

Siano $\{N_{i,n}\}_{i=1}^{n+K+1}$ le funzioni base B-Spline di grado n con nodi $t_1, \ldots, t_{2n+K+2}$ e siano $p_1, \ldots, p_{n+K+1}$ i punti da interpolare con parametro $u_1 < \cdots < u_{n+K+1}$. Vogliamo trovare la spline $s(x) = \sum_{i=1}^{n+K+1} c_i N_{i,n}(x)$ che soddisfa le condizioni di interpolazione

$$s(u_j) = \sum_{i=1}^{n+K+1} c_i N_{i,n}(u_j) = p_j \tag{2.1.1}$$

che equivale a risolvere il seguente sistema lineare

$$\begin{pmatrix} N_{1,n}(u_1) & \ldots & N_{n+K+1,n}(u_1) \\ \vdots & \ddots & \vdots \\ N_{1,n}(u_{n+K+1}) & \ldots & N_{n+K+1,n}(u_{n+K+1}) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_{n+K+1} \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_{n+K+1} \end{pmatrix} \tag{2.1.2}$$

che possiamo abbreviare con $N\mathbf{c} = \mathbf{p}$

dove la matrice N è detta *matrice di collocazione*.

Il teorema di Schoenberg-Whitney fornisce le condizioni di esistenza e unicità dell'interpolante spline, ossia della soluzione del sistema lineare.

Teorema 2.1.1 (Teorema di Schoenberg-Whitney)**.** La matrice N è invertibile se e solo se N ha una diagonale positiva, ovvero

$$N_{i,n}(u_i) > 0, i = 1, \ldots, n + K + 1$$

Questo accade se e solo se

$$u_i \in (t_i, t_{i+n+1}), \ \forall i$$

2.2 Curve di interpolazione

Analizziamo il problema di interpolazione di curve e diamo la definizione di curva interpolante di punti nello spazio bidimensionale. Sebbene siano state introdotte le funzioni B-Spline e i risultati mostrati nelle sezioni seguenti utilizzino questo tipo di funzioni, si introducono le curve di interpolazione per una generica base $\{B_i\}_{i=0}^{n}$, dove $n+1$ è la dimensione dello spazio, poiché gli algoritmi illustrati trovano applicazione anche in tipi di curve differenti dalle B-Spline. È da intendersi in tutta la trattazione che qualora si utilizzino curve B-Spline è da scegliere un'opportuna partizione nodale.

Siano assegnati i punti

$$\mathbf{Q}_i = (qx_i, qy_i)^T, \quad i = 0, \ldots, n$$

si vuole determinare una curva

$$\mathbf{C}(u) = \sum_{i=0}^{n} \mathbf{P}_i B_i(u) \quad u \in [a, b]$$

che soddisfi le condizioni di interpolazione, ovvero tale che

$$\mathbf{C}(u_i) = \mathbf{Q}_i, \quad i = 0, \ldots, n$$

In questo caso, occorre risolvere due sistemi lineari

$$B\mathbf{px} = \mathbf{qx} \qquad B\mathbf{py} = \mathbf{qy} \tag{2.2.1}$$

dove B è la matrice di collocazione della base nei punti $\{u_i\}_{i=0}^{n}$.

Il problema di interpolazione per curve tridimensionali è equivalente a quello nello spazio bidimensionale. È soltanto richiesta la risoluzione di un ulteriore sistema lineare per determinare il coefficiente z dei punti di controllo.

I due sistemi lineari in (2.2.1) forniscono le coordinate x, y dei punti di controllo $\mathbf{P}_i = (px_i, py_i)^T, i = 0, \ldots, n$.

In questo caso, i punti di interpolazione $\mathbf{Q}_i$ sono fissati. Tuttavia, il loro parametro u_i può essere stabilito con varie tecniche:

- *Parametrizzazione uniforme*: si scelgono parametri equispaziati nell'intervallo $[a, b]$:
$$u_i = a + i \cdot h, \quad i = 0, \ldots, n \quad \text{con} \quad h = \frac{b-a}{n}$$
Questa scelta dei parametri non tiene però conto della distanza fra i punti di interpolazione della curva.

- *Parametrizzazione della corda*: questa soluzione tiene conto delle distanze tra i punti di interpolazione. L'obiettivo è fare in modo che la lunghezza ad arco della curva tra $\mathbf{Q}_i$ e $\mathbf{Q}_{i+1}$ sia approssimativamente proporzionale alla distanza tra t_i e t_{i+1}, si richiede pertanto che
$$\frac{u_{i+1} - u_i}{u_{i+2} - u_{i+1}} = \frac{\|\mathbf{Q}_{i+1} - \mathbf{Q}_i\|_2}{\|\mathbf{Q}_{i+2} - \mathbf{Q}_{i+1}\|_2}$$

- *Parametrizzazione centripeta*: è ancora più efficiente della parametrizzazione della corda ed è motivata da un'interpretazione fisica
$$\frac{u_{i+1} - u_i}{u_{i+2} - u_{i+1}} = \frac{\|\mathbf{Q}_{i+1} - \mathbf{Q}_i\|_2^{\frac{1}{2}}}{\|\mathbf{Q}_{i+2} - \mathbf{Q}_{i+1}\|_2^{\frac{1}{2}}}$$

Negli esempi mostrati nelle sezioni successive viene utilizzata la *parametrizzazione centripeta*.

2.2.1 Partizione nodale

Come anticipato precedentemente, è necessario assicurarsi che la matrice di collocazione B sia non singolare. Per quanto riguarda le B-Spline, il teorema di Schoenberg-Whitney fornisce le condizioni affinché questo accada. È quindi necessario individuare una partizione nodale che permetta di rispettare queste condizioni, altrimenti non è possibile risolvere i sistemi lineari e determinare i punti di controllo della curva spline interpolante.

Nella (2.2.2) è mostrato il metodo proposto da de Boor per disporre i nodi t_i della B-Spline di grado g rispetto ai parametri dei punti di interpolazione $\{u_i\}_{i=1}^{n}$ assicurandosi che vengano rispettate le condizioni di Schoenberg-Whitney.

$$t_i = \begin{cases} u_1 & i = 1, \dots, g+1 \\ \frac{\sum_{j=i-g}^{i-1} u_i}{g} & i = g+2, \dots, n \\ u_n & i = n+1, \dots, g+n+1 \end{cases} \tag{2.2.2}$$

Un'alternativa al metodo proposto da de Boor utilizzabile per costruire la partizione nodale di B-Spline cubiche prevede di far coincidere i nodi della partizione nodale con i parametri dei punti di interpolazione, nel modo seguente:

$$\boldsymbol{\Delta}^* = \{u_0, u_0, u_0, u_0, u_1, \dots, u_{n-1}, u_n, u_n, u_n, u_n\} \tag{2.2.3}$$

Qualora venga utilizzata questa partizione, è necessario utilizzare due punti di controllo in più rispetto agli $n+1$ previsti in un problema di interpolazione di $n+1$ punti. In particolare, per fare in modo di avere un numero di punti di controllo uguale alla dimensione dello spazio spline $(n+3)$, è possibile aggiungere un punto di controllo identico al primo e uno identico all'ultimo.

2.3 Progressive Iteration Approximation (PIA)

Come spiegato nei paragrafi precedenti, determinare i coefficienti di una curva interpolante richiede normalmente la risoluzione di un sistema lineare. Questo metodo, tuttavia, presenta alcuni svantaggi. Ad esempio, il metodo di risoluzione di un sistema lineare è globale e la modifica di un solo punto di interpolazione comporta la necessità di ricalcolare tutto il sistema lineare. Inoltre, in alcune situazioni ci si può accontentare di avere delle approssimazioni dei coefficienti della curva interpolante. Questo non è possibile risolvendo il sistema lineare, poiché non fornisce alcun risultato parziale che sia significativo dal punto di vista geometrico.

Per superare gli svantaggi della risoluzione del sistema lineare, sono stati presentati dei metodi che permettono di ottenere i coefficienti della curva interpolante in modo iterativo e incrementale. Il metodo presentato da Lin, Bao e Wang, denominato *Progressive Iteration Approximation* (PIA) è uno di questi [LBW05].

Sia $\{\mathbf{P}_i\}_{i=0}^n$ la sequenza di punti da interpolare. Nelle prossime sezioni, indicheremo con t_i il parametro assegnato all'i-esimo punto. Sia $\{B_i(t) \geq 0 \mid t \in \mathbb{R}, i = 0, 1, \ldots, n\}$ una base non negativa con $\sum_{i=0}^n B_i(t) = 1$. La curva iniziale si genera con

$$\mathbf{C}^0(t) = \sum_{i=0}^{n} \mathbf{P}_i^0 B_i(t)$$

dove $\{\mathbf{P}_i^0 = \mathbf{P}\}_{i=0}^n$. La curva di approssimazione dopo $k+1$ ($k \geq 0$) iterazioni è definita da

$$\mathbf{C}^{k+1}(t) = \sum_{i=0}^{n} \mathbf{P}_i^{k+1} B_i(t_i)$$

dove i punti di controllo sono aggiornati iterativamente

$$\mathbf{P}_i^{k+1} = \mathbf{P}_i^k + \boldsymbol{\Delta}_i^k \qquad \text{con } \boldsymbol{\Delta}_i^k = \mathbf{P}_i - \mathbf{C}^k(t_i) \tag{2.3.1}$$

Si ottiene quindi una sequenza di curve $\{\mathbf{C}^k(t) \mid k = 0, 1, \ldots\}$. Se $\lim_{k\to\infty} \mathbf{C}^k(t_i) = \mathbf{P}_i^0, i = 0, 1, \ldots, n$ la curva gode della proprietà di *progressive iteration approximation* e dunque, per $k \to \infty$ interpola i punti $\{\mathbf{P}_i\}_{i=0}^n$.

Possiamo dimostrare la convergenza del procedimento iterativo (2.3.1) scrivendolo in forma matriciale

$$\begin{aligned}\boldsymbol{\Delta}_j^{k+1} &= \mathbf{P}_j - \mathbf{C}^{k+1}(t_j) = \mathbf{P}_j - \sum_{i=0}^{n}(\mathbf{P}_i^k + \boldsymbol{\Delta}_i^k)B_i(t_j) \\ &= (\mathbf{P}_j - \mathbf{C}^k(t_j)) - \sum_{i=0}^{n}\boldsymbol{\Delta}_i^k B_i(t_j) \\ &= -\sum_{i=0}^{j-1}\boldsymbol{\Delta}_i^k B_i(t_j) + (1 - B_j(t_j))\boldsymbol{\Delta}_j^k - \sum_{i=j+1}^{n}\boldsymbol{\Delta}_i^k B_i(t_j), \quad (j = 0, 1, \dots, n; k = 0, 1, \dots)\end{aligned} \tag{2.3.2}$$

e dunque ottenere

$$[\boldsymbol{\Delta}_0^{k+1}, \boldsymbol{\Delta}_1^{k+1}, \dots, \boldsymbol{\Delta}_n^{k+1}]^T = D[\boldsymbol{\Delta}_0^k, \boldsymbol{\Delta}_1^k, \dots, \boldsymbol{\Delta}_n^k]^T, \quad D = I - B; \quad k = 0, 1, \dots \tag{2.3.3}$$

dove I è la matrice identità di rango $n+1$, e B è la matrice di collocazione per la base $\{B_i \geq 0 \mid i = 0, 1, \dots, n\}$ nei punti $\{t_0, t_1, \dots, t_n\}$, ovvero

$$B = \begin{pmatrix} B_0(t_0) & B_1(t_0) & \dots & B_n(t_0) \\ B_0(t_1) & B_1(t_1) & \dots & B_n(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ B_0(t_n) & B_1(t_n) & \dots & B_n(t_n) \end{pmatrix} \tag{2.3.4}$$

Ricordando che la condizione necessaria e sufficiente affinché un metodo iterativo del tipo $\mathbf{x} = A\mathbf{x} + \mathbf{q}$ converga alla soluzione $\mathbf{x}^*$ è data da $\rho(A) < 1$, è possibile enunciare il seguente teorema.

Teorema 2.3.1. Una curva ha la proprietà di *progressive iteration approximation* se la base è totalmente positiva e la sua matrice di collocazione B nei punti $\{t_0, \dots, t_n\}$ è non singolare.

Nota Una base è detta totalmente positiva se tutte le sue matrici di collocazione sono totalmente positive, ovvero tutti i loro minori sono non negativi.

Dimostrazione. Dal momento che la base $\{B_i \geq 0 \mid i = 0, 1, \dots, n\}$ è totalmente positiva, la sua matrice di collocazione B ha $n+1$ autovalori non negativi $\lambda_i(B), i =$

$0, 1, \ldots, n$. Dal momento che la matrice di collocazione B è non singolare, i suoi $n+1$ autovalori sono tutti positivi. Ricordando che la base B gode della proprietà $\sum_{i=0}^{n} B_i = 1$, allora $\|B\|_\infty = 1$. Di conseguenza, $0 < \lambda_i(B) \leq 1, i = 0, \ldots, n$ e $0 \leq \lambda_i(D) = 1 - \lambda_i(B) < 1$. Questo implica che $\rho(D) < 1$, quindi il procedimento iterativo (2.3.3) converge al vettore nullo. Quindi, $\lim_{k\to\infty} \mathbf{C}^k(t_i) = \mathbf{P}_i^0, i = 0, \ldots, n$ □

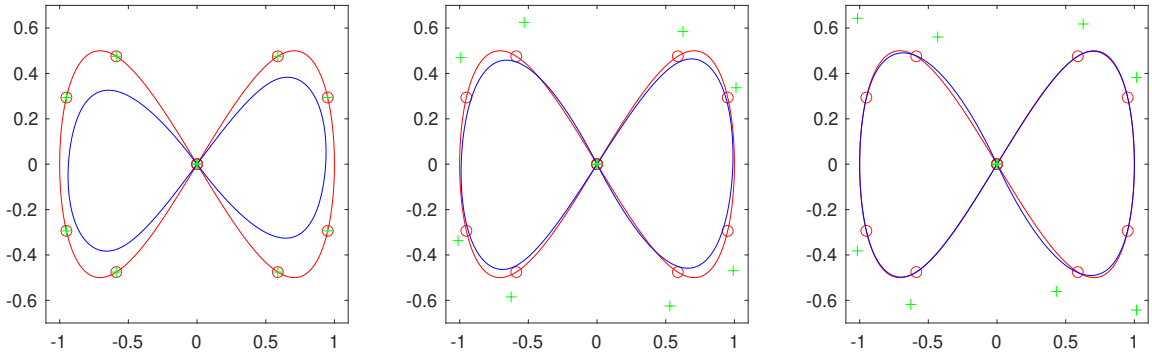
Le funzioni base B-Spline sono totalmente positive e se la partizione nodale soddisfa le condizioni di Schoenberg-Whitney la matrice B è non singolare. In tal caso, le B-Spline possiedono la proprietà di *progressive iteration approximation*.

Nella figura 2.1 è mostrato un esempio di funzionamento dell'algoritmo PIA nel quale una B-Spline cubica interpola la Lemniscata di Gerono in 11 punti. La partizione nodale è stata fissata con il metodo di de Boor (2.2.2). La Lemniscata di Gerono è definita da

$$(x(t), y(t)) = (\cos t, \sin t \cos t), \quad t \in [0, 2\pi] \tag{2.3.5}$$

e la sequenza di 11 punti di interpolazione $\{\mathbf{P}\}_{i=0}^{10}$ è campionata dalla curva parametrica nel modo seguente

$$\mathbf{P}_i = (x(t_i), y(t_i)), \quad t_i = -\frac{\pi}{2} + i \cdot \frac{2\pi}{10}, i = 0, \ldots, 10$$



(a) $k = 0$ (b) $k = 1$ (c) $k = 10$

Figura 2.1: Interpolazione della Lemniscata di Gerono

In rosso la curva interpolata e i punti di interpolazione, in blu la B-Spline ($g = 3$) interpolante e in verde i punti di controllo

Le tre figure permettono di capire il funzionamento di PIA: al passo $k = 0$, i punti di controllo $\{\mathbf{P}_i^0\}_{i=0}^{10}$ della B-Spline vengono inizializzati al valore dei punti di interpolazione $\{\mathbf{P}_i\}_{i=0}^{10}$. Dopo un solo passo (figura (b)), i punti di controllo $\{\mathbf{P}_i^1\}_{i=0}^{10}$ vengono ricalcolati aggiungendo a $\{\mathbf{P}_i^0\}_{i=0}^{10}$ il vettore delle distanze $\mathbf{\Delta}_i^0$. Nella figura (c) è possibile notare che dopo soltanto 10 iterazioni la B-Spline interpola $\{\mathbf{P}_i\}_{i=0}^{10}$ con una buona precisione.

Nella tabella 2.1, per ogni passo k, è mostrato il massimo errore di interpolazione nei punti $\{\mathbf{P}_i\}_{i=0}^{10}$ e l'errore di ricostruzione. Quest'ultimo è dato dalla distanza massima tra la curva interpolante e la Lemniscata di Gerono.

k	**Errore di interpolazione**	**Errore di ricostruzione**
0	1.80322e-01	1.81539e-01
1	7.60106e-02	5.31285e-02
2	3.74101e-02	3.14272e-02
3	2.24277e-02	3.51620e-02
4	1.72547e-02	3.35371e-02
5	1.26188e-02	3.10092-02
10	2.30600e-03	2.44457e-02
20	7.78212e-05	2.29331e-02
30	2.64605e-06	2.28838e-02
40	8.99859e-08	2.28821e-02
50	3.06023e-09	2.28820e-02
60	1.04072e-10	2.28820e-02
70	3.53930e-12	2.28820e-02
80	1.20389e-13	2.28820e-02
90	4.12616e-15	2.28820e-02
100	1.66533e-16	2.28820e-02

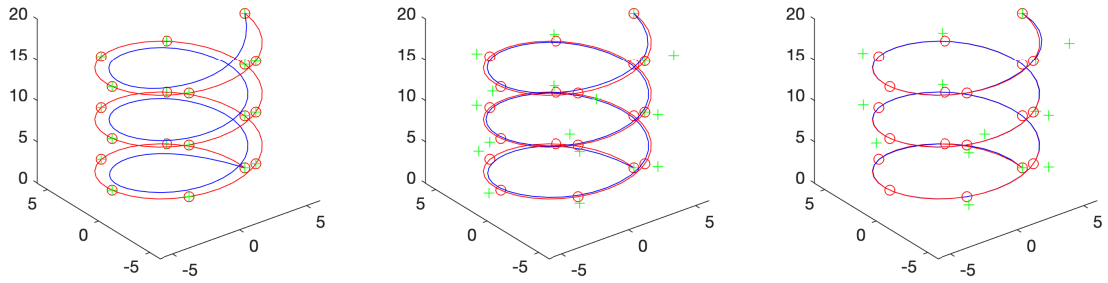
Tabella 2.1: Errore di interpolazione e di ricostruzione nella Lemniscata di Gerono

È interessante mostrare anche un esempio di interpolazione con B-Spline di una curva parametrica tridimensionale. Nella figura 2.2 è mostrata l'interpolazione con una B-Spline cubica di 19 punti campionati da un'elica di raggio 5 la cui forma analitica è data da

$$(x(t), y(t), z(t)) = (5\cos t, 5\sin t, t), \quad t \in [0, 6\pi].$$

La sequenza di punti $\{\mathbf{P}_i\}_{i=0}^{18}$ è ottenuta nel modo seguente:

$$\mathbf{P}_i = (x(t_i), y(t_i), z(t_i)), \qquad t_i = i \cdot \frac{\pi}{3}, \quad i = 0, \ldots, 18$$



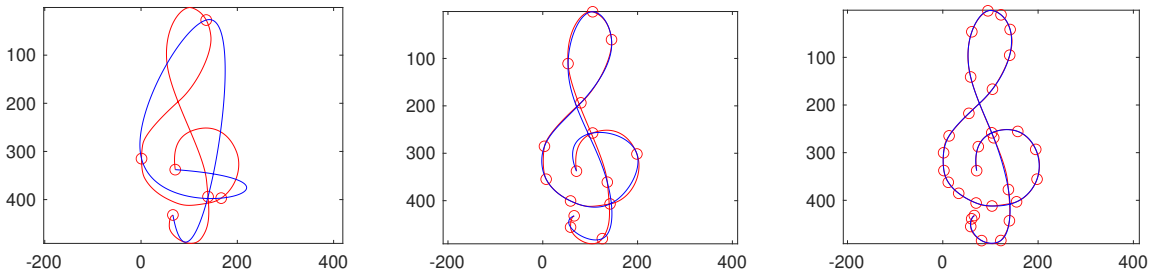
(a) $k = 0$ (b) $k = 1$ (c) $k = 10$

Figura 2.2: Interpolazione di un'elica di raggio 5
In rosso la curva interpolata e i punti di interpolazione, in blu la B-Spline ($g = 3$) interpolante e in verde i punti di controllo

Nella tabella 2.2 sono mostrati gli errori di interpolazione e di approssimazione.

k	Errore di interpolazione	Errore di ricostruzione
0	1.77667e+00	1.28129e+00
1	7.30589e-01	3.37145e-01
2	3.67395e-01	1.73370e-01
3	2.19803e-01	1.87291e-01
4	1.43719e-01	1.88688e-01
5	9.76607e-02	1.77197e-01
10	1.66185e-02	1.42075e-01
20	5.70404e-04	1.32759e-01
30	2.03391e-05	1.32464e-01
40	7.30134e-07	1.32453e-01
50	2.62516e-08	1.32453e-01
60	9.44239e-10	1.32453e-01
70	3.39669e-11	1.32453e-01
80	1.22169e-12	1.32453e-01
90	4.37760e-14	1.32453e-01
100	1.83103e-15	1.32453e-01

Tabella 2.2: Errore di interpolazione e di ricostruzione nell'elica



(a) $n = 6$ (b) $n = 15$ (c) $n = 30$

Figura 2.3: Punti di interpolazione

Analizzando le tabelle 2.1 e 2.2 relative agli esempi citati, è possibile notare come l'errore di ricostruzione si assesti già dalle prime iterazioni mentre l'errore di interpolazione converga alla precisione di macchina. In un problema di interpolazione, l'errore di ricostruzione può essere ridotto modificando o aumentando i punti di interpolazione o agendo sul tipo di curva interpolante (ad esempio modificando il grado della B-Spline o la partizione nodale). Nella figura 2.3 è mostrato chiaramente come la posizione e il numero di punti di interpolazione influenzi la capacità della curva interpolante di ricostruire la forma della curva interpolata. L'aumento dei punti di interpolazione porta ovviamente ad una migliore ricostruzione della curva.

È importante evidenziare che in PIA i parametri dei punti di interpolazione e dei punti interpolati dalla curva sono gli stessi. Di conseguenza, le iterazioni in PIA dipendono dalla *distanza parametrica*. Altri metodi, come *geometric interpolation* (GI), si basano sulla *distanza geometrica*. In questi ultimi, quindi, ad ogni iterazione il vettore delle distanze è calcolato considerando la distanza tra un punto di interpolazione e il punto ad esso più vicino nella curva interpolante.

2.3.1 Confronto con risoluzione del sistema lineare

Nella figura 2.4 è mostrato un confronto nel tempo richiesto per la risoluzione del problema di interpolazione utilizzando PIA o risolvendo il sistema lineare.

Il vantaggio di utilizzare un algoritmo iterativo-geometrico come PIA è che consente di ottenere una buona approssimazione della curva interpolante in pochissime

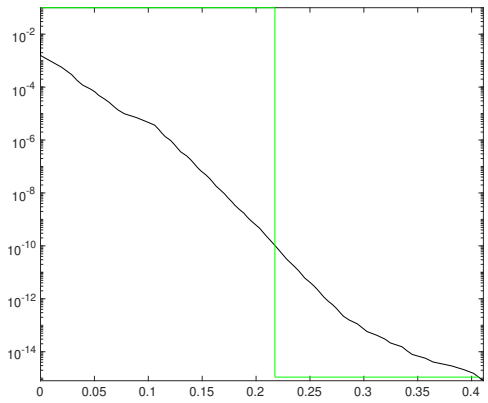


Figura 2.4: Confronto con la risoluzione di un sistema lineare
Interpolazione di 2000 punti campionati da un'epitrocoide (3.1.5). In verde è mostrato il tempo richiesto per la risoluzione del sistema lineare, in nero quello richiesto da PIA

iterazioni, con un costo computazionale molto ridotto. Tuttavia, se si desidera ottenere la precisione massima, la risoluzione di un sistema lineare richiede meno tempo rispetto a PIA. Inoltre, l'implementazione di PIA è molto semplice, mentre integrare un algoritmo numerico di risoluzione di un sistema lineare è una pratica complessa.

Come illustrato nelle sezioni successive, un altro fattore da considerare è che un metodo iterativo-geometrico permette di compiere delle azioni che non sarebbero perseguibili in un metodo di risoluzione di un sistema lineare, come ad esempio la possibilità di proseguire le iterazioni solo su alcuni punti di controllo o quella di aggiungere dei punti di controllo in modo progressivo.

2.4 Weighted PIA

In questa sezione si introduce una variante del metodo PIA che consente di ottenere una velocità di convergenza superiore rispetto al metodo tradizionale. Weighted PIA (WPIA) [Lu10] prevede di aggiornare i punti di controllo ad ogni iterazione nel modo seguente

$$\mathbf{P}_i^{k+1} = \mathbf{P}_i^k + \omega \mathbf{\Delta}_i^k$$

Similmente a (2.3.3), è possibile scrivere il nuovo procedimento iterativo in forma matriciale

$$[\mathbf{\Delta}_0^k, \mathbf{\Delta}_1^k, \dots, \mathbf{\Delta}_n^k]^T = (I - \omega B)[\mathbf{\Delta}_0^{k-1}, \mathbf{\Delta}_1^{k-1}, \dots, \mathbf{\Delta}_n^{k-1}]^T = (I - \omega B)^k [\mathbf{\Delta}_0^0, \mathbf{\Delta}_1^0, \dots, \mathbf{\Delta}_n^0]^T \tag{2.4.1}$$

Il procedimento iterativo (2.4.1) converge quando $\rho(I - \omega B) < 1$. Più $\rho(I - \omega B)$ è piccolo, più il procedimento converge velocemente. Chiaramente, con $\omega = 1$ si ottiene la versione di PIA tradizionale.

Di seguito è enunciato il teorema dimostrato dagli autori di WPIA che permette di calcolare il peso ω che offre la maggiore velocità di convergenza.

Teorema 2.4.1. Consideriamo una base totalmente positiva e la matrice di collocazione B nei punti $\{t_0, \dots, t_n\}$ non singolare. L'algoritmo WPIA ha velocità di convergenza massima quando

$$\omega = \frac{2}{1 + \lambda_n(B)}$$

dove $\lambda_n(B)$ è il più piccolo autovalore di B. In tal caso,

$$\rho(I - \omega B) = \frac{1 - \lambda_n(B)}{1 + \lambda_n(B)}$$

k	Errore PIA	Errore WPIA
0	1.80322e-01	1.80322e-01
1	7.60106e-02	6.58704e-02
2	3.74101e-02	2.09819e-02
3	2.24277e-02	1.30520e-02
4	1.72547e-02	6.56173e-03
5	1.26188e-02	3.70432e-03
10	2.30600e-03	1.76206e-04
20	7.78212e-05	4.56456e-07
30	2.64605e-06	1.19103e-09
40	8.99859e-08	3.10596e-12
50	3.06023e-09	8.02918e-15
60	1.04072e-10	1.24127e-16
70	3.53930e-12	1.24127e-16
80	1.20389e-13	1.24127e-16
90	4.12616e-15	1.24127e-16
100	1.66533e-16	1.24127e-16

Lemniscata di Gerono

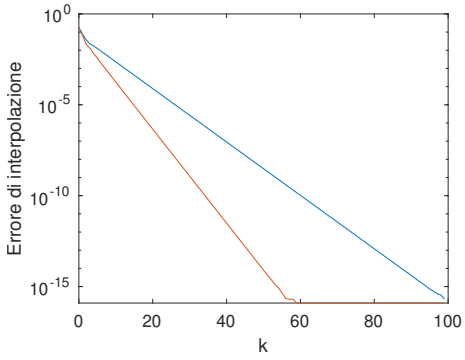
k	Errore PIA	Errore WPIA
0	1.77667e+00	1.77667e+00
1	7.30589e-01	6.32259e-01
2	3.67395e-01	1.82570e-01
3	2.19803e-01	1.05195e-01
4	1.43719e-01	4.42275e-02
5	9.76607e-02	2.57692e-02
10	1.66185e-02	1.33753e-03
20	5.70404e-04	3.88687e-06
30	2.03391e-05	1.15399e-08
40	7.30134e-07	3.43296e-11
50	2.62516e-08	1.02604e-13
60	9.44239e-10	2.03507e-15
70	3.39669e-11	2.03507e-15
80	1.22169e-12	2.03507e-15
90	4.37760e-14	2.03507e-15
100	1.83103e-15	2.03507e-15

Elica

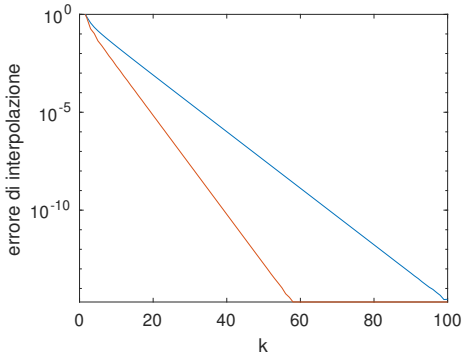
Tabella 2.3: Confronto tra PIA e WPIA

Nella tabella 2.3 è mostrato un confronto tra l'errore di interpolazione ottenuto con l'algoritmo PIA e quello ottenuto con WPIA per gli esempi della Lemniscata di Gerono e dell'elica mostrati nel capitolo 2.3. Nella figura 2.5 sono mostrati gli stessi risultati graficamente.

Come si può notare, a fronte di un maggiore costo computazionale dovuto al calcolo iniziale di ω, è possibile ridurre notevolmente il numero di iterazioni per ottenere un errore di interpolazione pari alla precisione di calcolo.



(a) Lemniscata di Gerono

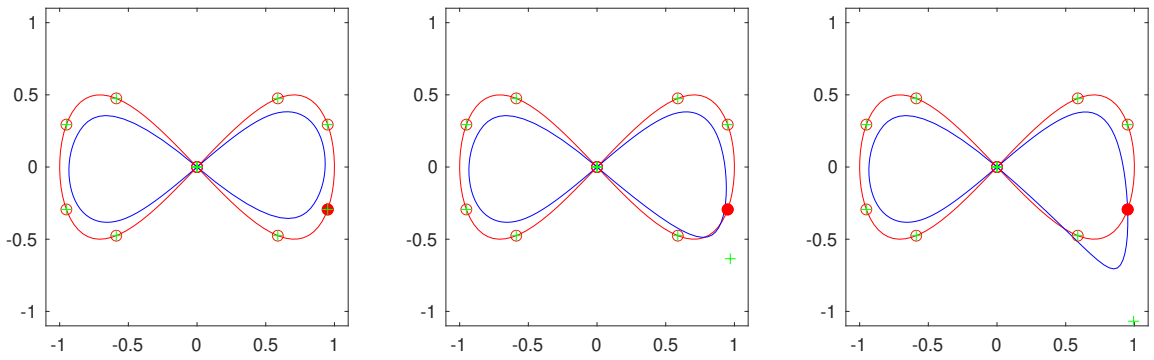


(b) Elica

Figura 2.5: Confronto tra PIA e WPIA
In rosso l'andamento degli errori di interpolazione con WPIA, in blu con PIA

2.5 Adaptive PIA

Nella figura 2.6 è mostrata la Lemniscata di Gerono dove si fissano tutti i punti di controllo tranne quello evidenziato in rosso. Come si può vedere, è possibile iterare solo su alcuni punti di controllo e fissarne altri, sfruttando le proprietà di località delle B-Spline. In questo modo, quando la curva interpolante approssima con una buona precisione alcuni punti di interpolazione non è più necessario proseguire con altre iterazioni su questi punti e quindi è opportuno procedere con ulteriori iterazioni soltanto sugli altri. In questa maniera, è possibile ridurre notevolmente il costo computazionale dell'interpolazione.



(a) $k = 0$ (b) $k = 1$ (c) $k = 5$

Figura 2.6: Esecuzione passi PIA solo su un punto di controllo

Un variante dell'algoritmo PIA, denominata *Adaptive PIA* [Lin12], prevede di classificare i punti di controlli in due classi ad ogni iterazione, i punti *attivi* e quelli *fissati*. Nelle iterazioni successive alla classificazione, vengono aggiornati soltanto i punti attivi, mentre quelli fissati rimangono invariati. Il numero di punti attivi decresce fino ad arrivare a zero, mentre quello dei punti fissati cresce finché tutti i punti attivi non sono fissati.

H. Lin ha introdotto il seguente teorema il cui corollario viene utilizzato dall'algoritmo PIA adattivo per classificare i punti attivi.

Teorema 2.5.1. Sia $I_k = \{i_0, i_1, \ldots, i_{h_k}\}$ l'insieme degli indici dei punti attivi alla k-esima iterazione, denotiamo con

$$D_k = [\boldsymbol{\Delta}_{i_0}^k, \boldsymbol{\Delta}_{i_1}^k, \ldots, \boldsymbol{\Delta}_{i_{h_k}}^k]^T$$

e con

$$\|D_k\|_M = \max\{\left\|\boldsymbol{\Delta}_i^k\right\|_2, i \in I_k\}.$$

Supponendo che alla k-esima iterazione venga fissato il j-esimo punto di controllo $\mathbf{P}_j^k$ e che l'insieme degli indici dei punti attivi sia $I_k = \{i_0, i_1, \ldots, i_{h_k}\}$, se $\|I - B_k\|_\infty < 1$, il limite del vettore delle differenze corrispondente al j-esimo punto soddisfa la seguente disuguaglianza:

$$\left\|\boldsymbol{\Delta}_j^\infty\right\|_2 \leq \left\|\boldsymbol{\Delta}_j^k\right\|_2 + \frac{1}{1 - \|I - B_k\|_\infty} \left\|L_j^k\right\|_\infty \|D_k\|_M$$

dove I è la matrice identità, $L_j^k = [-B_{i_0}(t_j), -B_{i_1}(t_j), \ldots, -B_{i_{h_k}}(t_j)]$ e

$$B_k = \begin{pmatrix} B_{i_0}(t_{i_0}) & B_{i_1}(t_{i_0}) & \ldots & B_{i_{h_k}}(t_{i_0}) \\ B_{i_0}(t_{i_1}) & B_{i_1}(t_{i_1}) & \ldots & B_{i_{h_k}}(t_{i_1}) \\ \vdots & \vdots & \ddots & \vdots \\ B_{i_0}(t_{i_{h_k}}) & B_{i_1}(t_{i_{h_k}}) & \ldots & B_{i_{h_k}}(t_{i_{h_k}}) \end{pmatrix}$$

Dal teorema segue il corollario che determina la disuguaglianza che permette di fissare i punti attivi, con la garanzia che l'errore di interpolazione nelle iterazioni successive nei punti fissati non potrà essere superiore alla soglia di precisione ε_0.

Corollario 2.5.1. Sia ε_0 la soglia di precisione da raggiungere. Supponiamo che il j-esimo punto di controllo sia attivo nelle iterazioni $l = 0, 1, \ldots, k-1$ e che alla k-esima iterazione l'insieme degli indici dei punti di controllo attivi sia

$I_k = \{i_0, i_1, \ldots, i_{h_k}\}, j \notin I_k$. Se il j-esimo vettore delle differenze $\boldsymbol{\Delta}_j^k$ soddisfa

$$\left\|\boldsymbol{\Delta}_j^k\right\|_2 \leq \varepsilon_0 - \frac{1}{1 - \|I - B_k\|_\infty} \left\|L_j^k\right\|_\infty \|D_k\|_M \tag{2.5.1}$$

allora il j-esimo punto di controllo può essere fissato nelle iterazioni $l = k+1, k+2, \ldots$ e la norma del vettore delle differenze soddisfa

$$\left\|\boldsymbol{\Delta}_j^l\right\|_2 \leq \varepsilon_0, l = k+1, k+2, \ldots$$

Di seguito viene presentato l'algoritmo PIA adattivo, che fa uso del corollario (2.5.1) per classificare i punti di controllo.

Algorithm 1: PIA Adattivo

Input: i punti da interpolare $\{\mathbf{P}_i\}_{i=0}^n$ e la soglia ε_0
Output: la curva $\mathbf{P}^k(t)$ che interpola $\{\mathbf{P}_i\}_{i=0}^n$ con precisione ε_0

Calcolo della matrice di collocazione $B = [B_j(t_i)], i = 0, \ldots, n,\ j = 0, \ldots, n$;
Costruzione della curva iniziale $\mathbf{P}^0(t)$ e il vettore delle differenze $\{\boldsymbol{\Delta}_i^0\}_{i=0}^n$;
`init_active_points` $= \{0, \ldots, n\}$;
$k = 0$;
`satisfied_points` $= \emptyset$;
`unsatisfied_points` $= \emptyset$;
while *init_active_points non è vuoto* **do**
    `// procedura iterativa`
    `ser_active_points` = `init_active_points`;
    while *ser_active_points non è vuoto* **do**
        $\mathbf{P}_i^{k+1} = \mathbf{P}_i^k + \boldsymbol{\Delta}_i^k, i \in$ `ser_active_points` ;
        $D_{k+1} = \{\boldsymbol{\Delta}_i^{k+1}, i \in$ `ser_active_points` $\}$;
        Prendo i vettori $D_{k+1}^g = \{\Delta_i^k, i \in G_k\}$ con norme maggiori di ε_0 da D_{k+1};
        `ser_active_points` $= G_k$;
        $k = k + 1$;
    end
    `// procedura di classificazione`
    $D_k = \{\boldsymbol{\Delta}_i^k, i \in$ `init_active_points` $\cup$ `satisfied_points` $\cup$ `unsatisfied_points` $\}$;
    Inserisco in D_k^g tutti gli elementi di D_k con norma maggiore di ε_0;
    Inserisco in D_k^l tutti gli elementi di D_k con norma minore di ε_0;
    `init_active_points` = indici dei $\boldsymbol{\Delta}_i$ in D_k^g ;
    Inserisco in S_p tutti gli elementi di D_k^l che soddisfano (2.5.1);
    Inserisco in U_p tutti gli elementi di D_k^l che non soddisfano (2.5.1);
    if *U_p non è vuoto* **then**
        `satisfied_points` = indici dei $\boldsymbol{\Delta}_i$ in S_p ;
        `unsatisfied_points` = indici dei $\boldsymbol{\Delta}_i$ in U_p ;
end

Il metodo adattivo prevede due procedure: quella iterativa e quella di classificazione. La procedura iterativa calcola i vettori delle differenze $\{\boldsymbol{\Delta}_i\}_{i=0}^{n}$. Se il vettore i-esimo delle differenze soddisfa $\|\boldsymbol{\Delta}_i\| \geq \varepsilon_0$ allora i viene inserito in `ser_active_points`, altrimenti il punto di controllo corrispondente viene fissato. Ad ogni iterazione, i punti di controllo che vengono aggiornati sono quelli i cui indici sono presenti in `ser_active_points`. La procedura iterativa termina quando non sono più presenti punti di controllo attivi.

Quando la procedura iterativa termina, viene invocata la procedura di classificazione che consiste nel stabilire quali punti possano essere fissati definitivamente perché soddisfano la condizione imposta dal corollario (2.5.1).

Il metodo adattivo termina quando non sono presenti punti di controllo attivi per la successiva procedura iterativa, ovvero quando `init_active_points` $= \emptyset$.

k	**PIA tradizionale**	**PIA Adattivo**	
	Errore di interpolazione	Punti attivi	Errore di interpolazione
0	4.46889e-02	21	4.46889e-02
1	1.32064e-02	17	1.32064e-02
2	5.82997e-03	16	5.82997e-03
3	2.96023e-03	8	2.96023e-03
4	1.59431e-03	8	1.59431e-03
5	8.88542e-04	8	8.88542e-04
6	5.07026e-04	8	5.07026e-04
7	3.04153e-04	8	3.04067e-04
8	1.85913e-04	8	1.85671e-04
9	1.14075e-04	8	1.22475e-04
10	7.02673e-05	8	1.30017e-04
11	4.34443e-05	8	1.34965e-04
12	2.69548e-05	8	1.38159e-04
13	1.67785e-05	6	1.40198e-04
14	1.04757e-05	2	1.40198e-04
15	6.55880e-06	4	1.40198e-04
16		4	4.84053e-05
17		4	3.62609e-05
18		0	3.96921e-05

Tabella 2.4: Confronto tra PIA e PIA adattivo

Nella tabella 2.4 è mostrato un confronto tra PIA e PIA adattivo nell'interpolazione con una B-Spline cubica di 21 punti campionati dalla Lemniscata di Gerono

(2.3.5). La soglia di precisione ε_0 è fissata a 10^{-5}. La versione tradizionale di PIA raggiunge un errore di interpolazione minore di ε_0 in 15 passi. PIA adattivo, invece, impiega 18 iterazioni. Quest'ultimo, però, ad ogni iterazione aggiorna soltanto i punti attivi, mentre PIA aggiorna tutti i punti di controllo. Di conseguenza, il numero di aggiornamenti dei punti di controllo in PIA è $15 \cdot 21 = 315$ mentre quello di PIA adattivo è dato dalla somma del numero di punti attivi ad ogni iterazione $21 + 17 + 16 + 8 + \cdots = 150$.

Non è possibile stabilire con esattezza il vantaggio di utilizzare PIA adattivo rispetto ad usare PIA. Entrambi gli algoritmi prevedono di valutare la curva interpolante in tutti i punti di controllo per poter calcolare i vettori delle distanze $\{\boldsymbol{\Delta}_i\}_{i=0}^{n}$. La valutazione della curva interpolante ha un costo computazionale rilevante (si tratta di risolvere il prodotto $B\mathbf{P}^k$, dove B è la matrice di collocazione e $\mathbf{P}^k$ è il vettore dei punti di controllo al passo k). Di conseguenza, PIA adattivo potrebbe richiedere un costo complessivo superiore, a causa del maggior numero di iterazioni richieste per raggiungere la convergenza. Viceversa, su alcuni tipi di curve, continuare le iterazioni su tutti i punti di controllo come previsto da PIA risulta inutile e la versione adattiva permette un buon risparmio di risorse.

Capitolo 3

Approssimazione spline ai minimi quadrati

Data una base B e un insieme di punti $\{\mathbf{Q}_i\}_{i=0}^n$, L'interpolazione permette di trovare i punti di controllo $\{\mathbf{P}_i\}_{i=0}^n$ tali che

$$\mathbf{C}(t_i) = \sum_{j=0}^{n} \mathbf{P}_j B_j^n(t_i) = \mathbf{Q}_i, \quad i = 0, \ldots, n$$

Se i punti di interpolazione sono campionati da una curva, la soluzione del problema di interpolazione consente di trovare la curva che passa esattamente per i punti assegnati ma non garantisce in alcun modo che la curva interpolante ricostruisca la forma della curva interpolata. Inoltre, per curve più complesse, la ricostruzione della curva tramite interpolazione richiederebbe un numero di punti di interpolazione eccessivamente alto che renderebbe il calcolo dei punti di controllo $\{\mathbf{P}_i\}_{i=0}^n$ troppo oneroso.

Analizziamo, dunque, un'altra soluzione che prevede la ricostruzione di una curva per *approssimazione*. In questo caso, si vuole costruire una curva

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P}_j B_i(t)$$

che approssimi un insieme di punti $\{\mathbf{Q}_j\}_{j=0}^m$ dove $m > n$ (se $m = n$ il problema di approssimazione degenera in un problema di interpolazione).

Formalmente, considerando di voler approssimare una curva nello spazio $\mathbb{R}^2$, e considerando $\mathbf{Q}_j = (x_j, y_j)$ con $j = 0, \ldots, m$ i punti di approssimare in $\mathbb{R}^2$, l'obiettivo è costruire la curva $\mathbf{C}$ sotto la condizione che

$$\mathbf{C}(t_j) - \mathbf{Q}_j = \mathbf{r}_j, \quad j = 0, \ldots, m$$

sia minimo.

In forma matriciale, significa minimizzare

$$\begin{pmatrix} B_0(t_0) & \ldots & B_n(t_0) \\ \vdots & \ddots & \vdots \\ B_0(t_m) & \ldots & B_n(t_m) \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \vdots \\ \mathbf{P}_n \end{pmatrix} - \begin{pmatrix} \mathbf{Q}_0 \\ \vdots \\ \mathbf{Q}_m \end{pmatrix} = \begin{pmatrix} \mathbf{r}_0 \\ \vdots \\ \mathbf{r}_m \end{pmatrix}$$

ovvero

$$\begin{pmatrix} B_0(t_0) & \ldots & B_n(t_0) \\ \vdots & \ddots & \vdots \\ B_0(t_m) & \ldots & B_n(t_m) \end{pmatrix} \begin{pmatrix} px_0 & py_0 \\ \vdots & \vdots \\ px_n & py_n \end{pmatrix} - \begin{pmatrix} x_0 & y_0 \\ \vdots & \vdots \\ x_m & y_m \end{pmatrix} = \begin{pmatrix} rx_0 & ry_0 \\ \vdots & \vdots \\ rx_m & ry_m \end{pmatrix}$$

o, in modo più conciso,

$$BP - Q = R \tag{3.0.1}$$

Comunemente, si usa minimizzare la somma dei quadrati degli errori, ovvero minimizzare

$$\sum_{i=0}^{m} \mathbf{r}_i^2 = \sum_{i,j} r_{ij}^2 = \sum_{i=0}^{m} (rx_i^2 + ry_i^2)$$

Quindi, il problema diventa quello di determinare i punti di controllo $\{\mathbf{P}_i\}_{i=0}^n$ tali che

$$\|BP - Q\|_2$$

sia minimo.

Questo problema viene detto *problema dei minimi quadrati*.

Teorema 3.0.1. Se B ha rango massimo, la soluzione del problema dei minimi quadrati esiste, è unica e coincide con la soluzione del sistema lineare

$$B^T BP = B^T Q$$

Tale sistema viene detto sistema delle *equazioni normali*.

Il sistema delle equazioni normali può essere risolto numericamente tramite fattorizzazione QR di B, la cui complessità computazionale è $O(n^3)$, usando l'algoritmo di Householder.

Nella prossima sezione viene illustrata una soluzione geometrico-iterativa per la risoluzione di questo problema, basata sull'algoritmo PIA.

3.1 Least Square Progressive Iterative Approximation (LSPIA)

Nella versione classica di PIA, presentata nel capitolo 2.3, il numero di punti di controllo è uguale al numero di punti di interpolazione. Chiaramente, questa soluzione non è perseguibile quando il numero di punti da approssimare è molto alto. La proposta di Deng C. e Lin H. prevede di costruire iterativamente una serie di curve che approssimano i punti di valutazione con il metodo dei minimi quadrati [DL14].

Sia $\{\mathbf{Q}_j\}_{j=0}^m$ l'insieme di punti da approssimare e $\{0 = t_0 < t_1 < \cdots < t_m = 1\}$ i parametri di $\{\mathbf{Q}_j\}_{j=0}^m$ scelti con una delle tecniche illustrate nella sezione 2.2. Alla prima iterazione, scegliamo i punti $\{\mathbf{P}_i\}_{i=0}^n$ da $\{\mathbf{Q}_j\}$ $(n < m)$ come punti di controllo iniziali e costruiamo la prima curva $\mathbf{P}^0(t)$

$$\mathbf{P}^0(t) = \sum_{i=0}^{n} B_i(t)\mathbf{P}_i^0, \quad t \in [t_0, t_m]$$

dove B è una base non negativa $\{B_i(t) \geq 0 \mid t \in \mathbb{R}, i = 0, 1, \ldots, n\}$ con $\sum_{i=0}^n B_i(t) = 1$.

Ad ogni passo costruiamo la k-esima curva $\mathbf{P}^k(t)$

$$\delta_j^k = \mathbf{Q}_j - \mathbf{P}^k(t_j), \quad j = 0, 1, \ldots, m$$

$$\mathbf{\Delta}_i^k = \mu \sum_{j=0}^{m} B_i(t_j)\delta_j^k, \quad i = 0, 1, \ldots, n \tag{3.1.1}$$

$$\mathbf{P}_i^{k+1} = \mathbf{P}_i^k + \mathbf{\Delta}_i^k, \quad i = 0, 1, \ldots, n \tag{3.1.2}$$

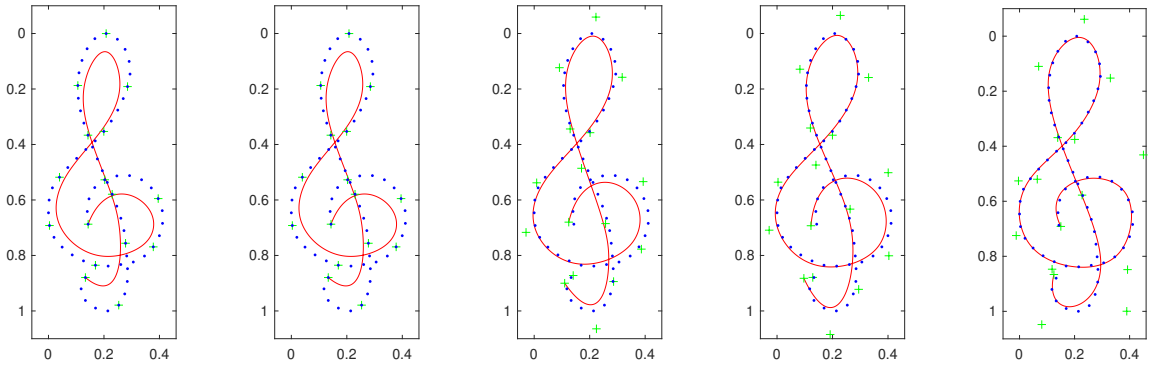
dove μ è una costante che soddisfa la condizione $0 < \mu < \frac{2}{\lambda_0}$ e λ_0 è il più grande autovalore di $A^T A$.

La $(k+1)$-esima curva è data data

$$\mathbf{P}^{k+1}(t) = \sum_{i=0}^{n} B_i(t)\mathbf{P}_i^{k+1}$$

Nel caso di uno spazio spline la partizione nodale va scelta in modo opportuno in modo che negli $m+1$ punti da approssimare siano presenti $n+K+1$ punti che insieme ai nodi soddisfano le condizioni di non singolarità della matrice.

Nella figura 3.1 è mostrato un esempio di approssimazione con una B-Spline di 70 punti equispaziati campionati da un'immagine vettoriale di una chiave di violino. La partizione nodale è calcolata con il metodo proposto da de Boor (2.2.2). Nella tabella 3.1 sono mostrati gli errori di approssimazione dati da $\max_j \left\|\delta_j^k\right\|_2$. L'algoritmo LSPIA termina quando l'errore di approssimazione raggiunge una soglia fissata o quando l'avanzamento (determinato dalla differenza degli errori di approssimazione al passo k e al passo $k-1$) è minore di una certa tolleranza. Nell'esempio mostrato, le iterazioni si fermano al passo $k=76$ poiché l'avanzamento raggiunge la tolleranza fissata a 10^{-5}.



(a) $k=0$ (b) $k=1$ (c) $k=3$ (d) $k=5$ (e) $k=76$

Figura 3.1: Esempio funzionamento LSPIA

In rosso l'approssimazione con una B-Spline di grado 3 con 15 punti di controllo di 70 punti equispaziati campionati da un'immagine vettoriale di una chiave di violino.

k	Errore di approssimazione	Avanzamento
0	1.48836e-01	$\emptyset$
1	6.72781e-02	8.15579e-02
3	5.46144e-02	5.46144e-02
5	4.77135e-02	4.77135e-02
76	2.81295e-02	9.84385e-06

Tabella 3.1: Errore di approssimazione relativo alla figura 3.1

Teorema 3.1.1. Il metodo LSPIA è convergente e la curva limite è la curva che approssima ai minimi quadrati i punti iniziali $\{\mathbf{Q}_j\}_{j=0}^m$

Dimostrazione. Consideriamo la sequenza di curve generate $\{\mathbf{P}^k(t), k = 0, 1, \dots\}$ e siano

$$\mathbf{P}^k = \{\mathbf{P}_0^k, \mathbf{P}_1^k, \dots \mathbf{P}_n^k\}^T$$
$$\mathbf{Q} = \{\mathbf{Q}_0, \mathbf{Q}_1, \dots \mathbf{Q}_m\}^T$$

Da (3.1.2) otteniamo

$$\begin{aligned}\mathbf{P}_i^{k+1} &= \mathbf{P}_i^k + \mu \sum_{j=0}^{m} B_i(t_j)(\mathbf{Q}_j - \mathbf{P}^k(t_j)) \\ &= \mathbf{P}_i^k + \mu \sum_{j=0}^{m} B_i(t_j)\Big[\mathbf{Q}_j - \sum_{l=0}^{n} B_l(t_j)\mathbf{P}_l^k\Big]\end{aligned}$$

e quindi

$$\mathbf{P}^{k+1} = \mathbf{P}^k + \mu A^T(Q - A\mathbf{P}^k) \tag{3.1.3}$$

dove A è la matrice di collocazione della base B nei punti $t_0, \dots, t_m$.

Sia I la matrice identità di rango $n+1$ e $D = I - \mu A^T A$, da (3.1.3) otteniamo

$$\begin{aligned}\mathbf{P}^{k+1} - (A^T A)^{-1}A^T\mathbf{Q} &= (I - \mu A^T A)[\mathbf{P}^k - (A^T A)^{-1}A^T Q] \\ &= (I - \mu A^T A)^2[\mathbf{P}^{k-1} - (A^T A)^{-1}A^T Q] \\ &= \dots \\ &= D^{k+1}[\mathbf{P}^0 - (A^T A)^{-1}A^T Q]\end{aligned} \tag{3.1.4}$$

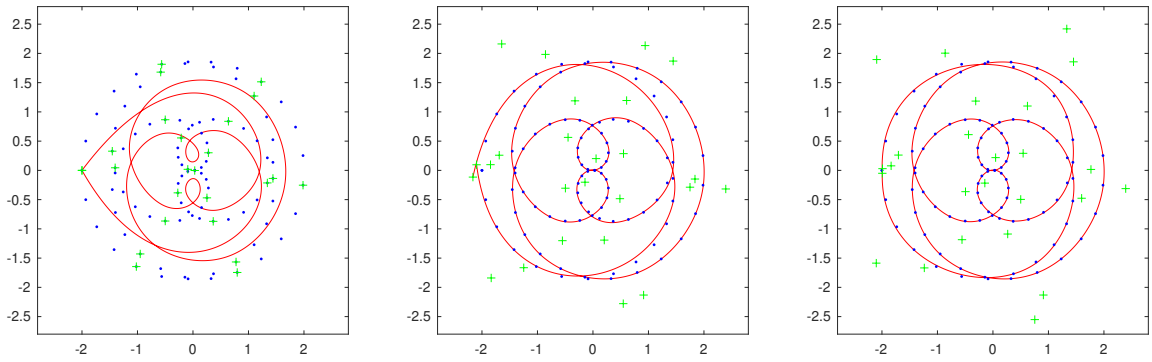
Siano $\{\lambda_i(D)\}, i = 0, \dots, n$ gli autovalori di D ordinati in ordine non decrescente, otteniamo $\lambda_i(D) = 1 - \mu\lambda_i$, dove $\lambda_i, i = 0, \dots, n$ sono gli autovalori di $A^T A$ in ordine non decrescente. Dal momento che A è non singolare, $A^T A$ è una matrice definita positiva. Sia $0 < \mu < \frac{2}{\lambda_0}$, abbiamo $0 < \mu\lambda_i < 2$ e $-1 < \{\lambda_i(D)\} < 1, i = 0, \dots, n$. Questo porta a $0 < \rho(D) < 1$, dove $\rho(D)$ è il raggio spettrale di D. Di conseguenza,

$$\lim_{k\to\infty} D^k = \mathbf{0}$$

Dalla (3.1.4) segue

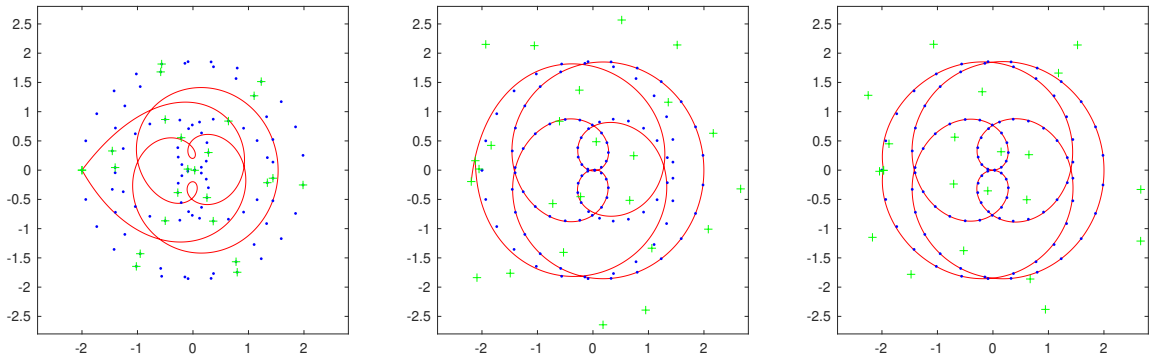
$$\begin{aligned}\mathbf{P}^{\infty} &= (A^T A)^{-1} A^T Q + D^{\infty}[\mathbf{P}^0 - (A^T A)^{-1} A^T Q] \\ &= (A^T A)^{-1} A^T Q\end{aligned}$$

che è equivalente a $(A^T A)\mathbf{P}^{\infty} = A^T Q$. Questo significa che il metodo LSPIA è convergente e la curva limite è l'approssimazione ai minimi quadrati dei dati iniziali. □



$k = 0$ $k = 5$ $k = 100$

(a) B-Spline di grado 3



$k = 0$ $k = 5$ $k = 100$

(b) B-Spline di grado 5

Figura 3.2: Esempio funzionamento LSPIA

In rosso l'approssimazione con B-Spline di terzo e quinto grado con 25 punti di controllo di 100 punti campionati da un'epitrocoide.

Nella figura 3.2 è mostrata l'approssimazione di 100 punti campionati da un'epitrocoide con una B-Spline di terzo e di quinto grado con 25 punti di controllo. La partizione nodale è stata fissata con il metodo di de Boor (2.2.2). L'epitrocoide è definita da

$$\begin{aligned} x(t) &= (a-b)\cos(t) - c\cos((\frac{a}{b}+1)t) \\ y(t) &= (a-b)\sin(t) - c\sin((\frac{a}{b}+1)t) \end{aligned} \tag{3.1.5}$$

dove $a = 2, b = 3, c = 1$. I 100 punti di approssimazione sono stati campionati prendendo i loro parametri t_i equispaziati in $[0, 6\pi]$.

Sebbene l'epitrocoide non sia stata approssimata con una curva spline chiusa e periodica è possibile notare come dopo un certo numero di iterazioni LSPIA ricostruisca correttamente anche la chiusura. Nella tabella 3.2 sono mostrati gli errori di ricostruzione all'aumentare del numero di iterazioni. Gli errori di ricostruzione sono dati dalla distanza massima tra la curva spline approssimante e l'epitrocoide approssimata.

k	B-Spline di grado 3	B-Spline di grado 5
0	3.05192e-01	3.90952e-01
1	2.08171e-01	3.27597e-01
3	1.28322e-01	2.12834e-01
5	9.76380e-02	1.62661e-01
10	6.08464e-02	1.03729e-01
20	4.68688e-02	7.63306e-02
30	4.65104e-02	6.60335e-02
50	4.62346e-02	5.21464e-02
75	4.62011e-02	4.10464e-02
100	4.61968e-02	3.32490e-02
200	4.61962e-02	1.79538e-02

Tabella 3.2: Errori di ricostruzione relativi all'approssimazione in figura 3.2

Nonostante non sia stato oggetto di studio in questo documento, la classe di algoritmi PIA può essere utilizzata anche su superfici. Per approssimare una superficie complessa è necessario utilizzare molti punti di approssimazione. In questi casi, LSPIA permette di ottenere ottimi risultati.

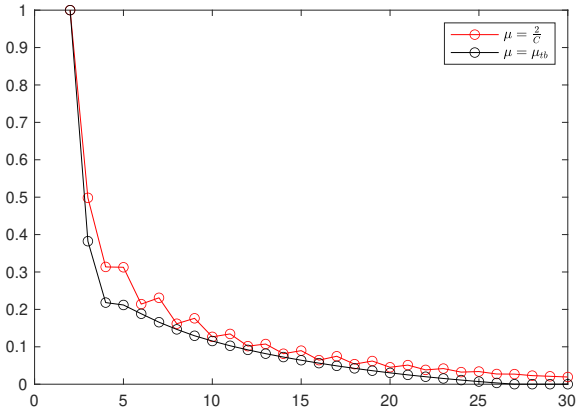


Figura 3.3: Confronto del peso LSPIA

3.1.1 Scelta del peso

Come abbiamo visto per WPIA nel capitolo 2.4, anche per LSPIA è possibile scegliere un peso μ che permetta di velocizzare la convergenza alla curva che approssima ai minimi quadrati.

Deng C. e Lin H. hanno dimostrato che il peso μ che massimizza la velocità di convergenza è dato da

$$\mu = \frac{2}{\lambda_0 + \lambda_n} := \mu_{tb}$$

dove λ_0 e λ_n sono, rispettivamente, il più piccolo e il più grande autovalore di $A^T A$.

Utilizzare μ_{tb} come peso richiede un grande sforzo computazionale per calcolare il più piccolo e il più grande autovalore di $A^T A$. Per evitare il calcolo degli autovalori, gli autori hanno proposto un metodo alternativo per il calcolo del peso, che approssima con una buona precisione μ_{tb} e ha un ridotto costo computazionale.

Sia A la matrice di collocazione della base B nei punti $t_0, \dots, t_m$ e sia $A^T A = \{a_{i,j}\}, i = 0, \dots, m; j = 0, \dots n$ dove $a_{i,j} = \sum_{k=0}^{m} B_i(t_k) B_j(t_k)$. Ricordando che $\sum_{i=0}^{n} B_i(t_k) = 1$ abbiamo che

$$\sum_{j=0}^{n} a_{i,j} = \sum_{j=0}^{n} \Big[\sum_{k=0}^{m} B_i(t_k) B_j(t_k) \Big] = \sum_{k=0}^{m} B_i(t_k) \Big[\sum_{j=0}^{n} B_j(t_k) \Big] = \sum_{k=0}^{m} B_i(t_k) := c_i$$

Dunque, c_i è la somma dell'i-esima riga degli elementi di $A^T A$. Di conseguenza, $\lambda_0 \leq \max_i\{c_i\} := C, i = 0, \ldots, n$ e $\frac{2}{C} < \frac{2}{\lambda_0}$. Di conseguenza, è possibile definire μ come

$$\mu = \frac{2}{C}$$

Nella figura 3.3 è mostrato l'andamento dell'errore di approssimazione di 500 punti con una B-Spline cubica con 50 punti di controllo campionati dalla chiave di violino già mostrata in figura 3.1. È possibile notare che l'ordine di grandezza degli errori è lo stesso e la differenza fra l'uso di μ_{tb} e μ è minimale. Risultati simili sono stati riscontrati in tutte le prove effettuate prima di redigere questo documento.

3.2 LSPIA progressivo

In questa sezione analizziamo una versione di LSPIA denominata *LSPIA progressive*, proposta dagli autori di LSPIA [DL14].

In questo documento di tesi sono stati analizzati diversi algoritmi. Tra questi, LSPIA progressivo è quello più complesso ma è anche quello più interessante perché permette di sfruttare a pieno i vantaggi di un metodo iterativo-geometrico, rispetto alla risoluzione tradizionale dei sistemi di equazioni normali per determinare i punti di controllo della curva di approssimazione.

Normalmente, in un problema di interpolazione o di approssimazione, se non si riesce a ricostruire con una buona tolleranza la forma della curva da interpolare o approssimare è necessario aumentare i punti di controllo. Il vantaggio di utilizzare un algoritmo iterativo come LSPIA è che se la curva di approssimazione non ricostruisce con una buona tolleranza i punti di approssimazione, è possibile fare in modo di aumentare incrementalmente i punti di controllo, senza dover ricominciare le iterazioni. Un metodo tradizionale di risoluzione del problema di approssimazione ai minimi quadrati richiederebbe invece di ricominciare il calcolo di tutti i punti di controllo.

In modo più specifico, qualora dopo un certo numero di iterazioni l'errore di approssimazione sia maggiore di una certa soglia fissata, l'algoritmo incrementale aggiunge un nodo nell'intervallo nodale che presenta il più grande errore di approssimazione.

Consideriamo $\{\bar{t}_1, \ldots, \bar{t}_n\}$ i nodi della partizione nodale della spline di approssimazione. Possiamo calcolare l'errore di approssimazione per ogni intervallo nodale nel modo seguente

$$d_i = \sum_{t_j \in [\bar{t}_i, \bar{t}_{i+1}]} \left\| \mathbf{Q}_j - \mathbf{P}^k(t_j) \right\|$$

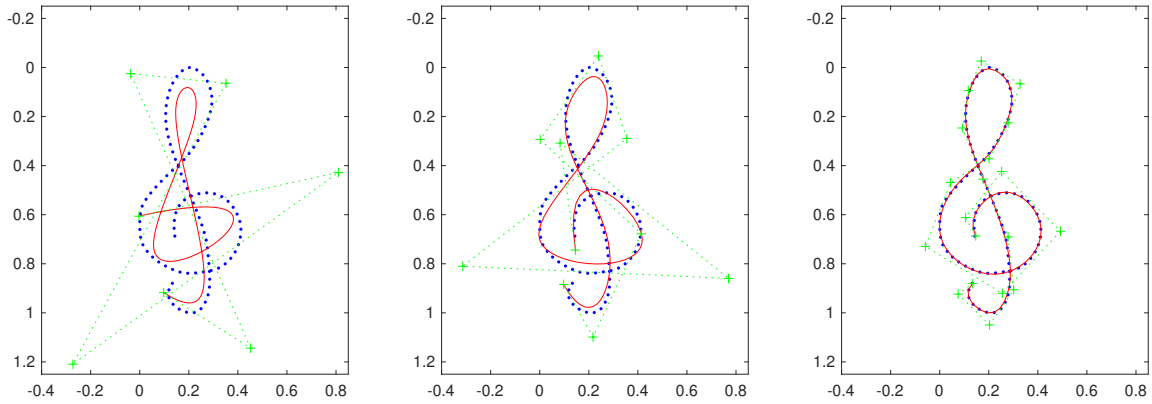
dove $\mathbf{P}^k(t_j)$ è il valore che assume la curva di approssimazione alla k-esima iterazione nel punto t_j.

Una volta scelto l'intervallo nodale che presenta il più grande errore di approssimazione d_i, calcoliamo il valore di un nuovo nodo e lo inseriamo nella partizione

nodale tramite *knot insertion* (descritto nel capitolo 1.2.3). Se l'intervallo nodale scelto contiene soltanto due punti di approssimazione $\mathbf{Q}_j, \mathbf{Q}_{j+1}$, con parametri t_j, t_{j+1}, il nuovo nodo $\bar{t}$ si calcola prendendo il valore a metà tra i due parametri, ovvero $\bar{t} = \frac{1}{2}(t_j + t_{j+1})$. Altrimenti, qualora l'intervallo nodale scelto $[\bar{t}_i, \bar{t}_{i+1}]$ presentasse più punti di approssimazione $\mathbf{Q}_j, \ldots, \mathbf{Q}_{j+a}$ con parametri $t_j, \ldots, t_{j+a} \in [\bar{t}_i, \bar{t}_{i+1})$, il nuovo nodo $\bar{t}$ verrebbe calcolato come $\bar{t} = \frac{1}{2}(t_l + t_{l+1})$, dove $l \in [j, j + a - 1]$ e

$$\sum_{t_h \in [\bar{t}_i, t_{j+l}]} \left\| \mathbf{Q}_h - \mathbf{P}^k(t_h) \right\| \geq \frac{d_i}{2} \qquad \text{e} \qquad \sum_{t_h \in [t_{j+l}, \bar{t}_{i+1}]} \left\| \mathbf{Q}_h - \mathbf{P}^k(t_h) \right\| \geq \frac{d_i}{2}$$

La versione progressiva di LSPIA prevede di combinare a delle normali iterazioni LSPIA l'inserimento di nuovi nodi nella partizione nodale quando si verificano certe condizioni. All'inizio, viene fissato il numero di punti di controllo iniziali e vengono effettuate delle iterazioni LSPIA per approssimare l'insieme di punti. L'inserimento di nuovi nodi entra in gioco quando viene effettuato un numero fissato di iterazioni o quando l'avanzamento, ovvero la differenza tra l'errore al passo k e quello al passo $k - 1$, è inferiore ad una soglia stabilita. Una volta inserito uno o più nodi negli intervalli nodali nei quali si presenta l'errore di approssimazione maggiore, vengono effettuate nuovamente diverse iterazioni LSPIA. Questa procedura viene ripetuta finché il numero di punti di controllo aggiunti è uguale ad un numero massimo fissato o l'errore di approssimazione raggiunge la tolleranza richiesta.



(a) $k = 30,\ n = 7$ (b) $k = 120,\ n = 10$ (c) $k = 390,\ n = 19$

Figura 3.4: Approssimazione di una chiave di violino con LSPIA progressive

Nella figura 3.4 è mostrato un esempio di approssimazione con LSPIA progressivo. La B-Spline cubica mostrata in rosso approssima un insieme di 100 punti campionati da una chiave di violino. La partizione nodale è fissata con il metodo proposto da de Boor (2.2.2). In questo esempio, viene inserito un nuovo nodo nella partizione nodale ogni 30 iterazioni, finché l'errore di approssimazione non risulta inferiore alla tolleranza 10^{-2}. Si è scelto di eseguire 30 iterazioni dopo l'inserimento di ogni nodo perché si è sperimentato che dopo circa 30 iterazioni l'avanzamento risulta molto ridotto ed è quindi necessario aumentare i punti di controllo per ridurre l'errore di approssimazione. Alternativamente, sarebbe stato possibile permettere alla procedura di stabilire dinamicamente quando è opportuno inserire un nuovo nodo, ovvero quando l'avanzamento risulta più basso di una tolleranza fissata.

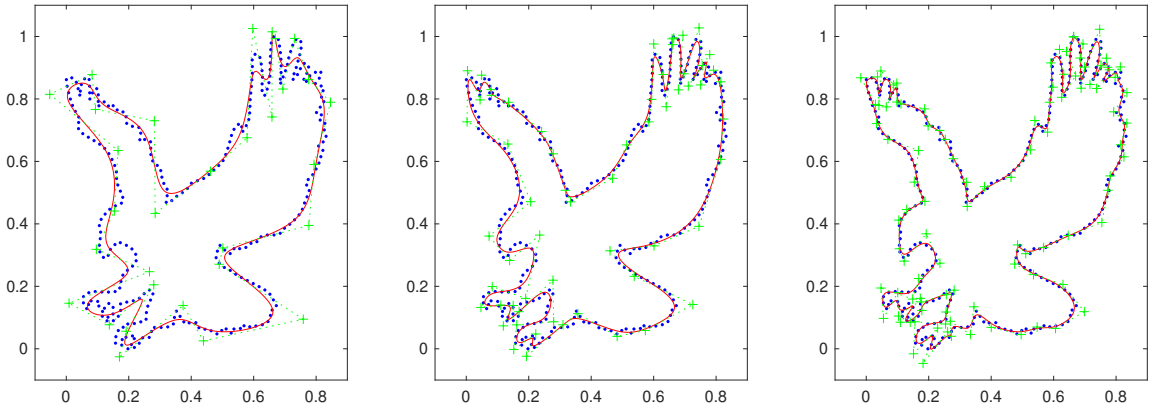
Inizialmente vengono utilizzati solo $n = 7$ punti di controllo. La figura (a) mostra l'approssimazione dopo $k = 30$ iterazioni. Chiaramente, per poter approssimare l'insieme di punti con una buona precisione è necessario aggiungere diversi punti di controllo. Dunque, terminate le prime 30 iterazioni, l'algoritmo aggiunge un nuovo nodo nell'intervallo nodale che presenta il maggior errore di approssimazione e itera nuovamente per 30 volte. La procedura descritta viene ripetuta finché l'errore di approssimazione non risulta inferiore della tolleranza fissata. Nella figura (b) è mostrata la curva approssimante con 10 punti di controllo ($k = 120$). Infine, nella figura (c) è mostrata l'approssimazione con 19 punti di controllo ($k = 390$). Nella tabel-

la 3.3 è mostrato l'andamento dell'errore di approssimazione, dato da $\max_j \left\|\delta_j^k\right\|_2$, all'aumentare del numero di iterazioni k e del numero di punti di controllo.

k	Numero punti di controllo	Errore di approssimazione
0	7	2.53642e-01
10	7	1.61958e-01
20	7	1.59657e-01
30	7	1.68367e-01
40	8	8.41386e-02
50	8	8.63726e-02
60	8	8.72622e-02
90	9	8.90815e-02
120	10	5.69735e-02
150	11	4.58025e-02
180	12	3.50845e-02
210	13	2.38962e-02
240	14	2.44678e-02
270	15	1.65751e-02
300	16	1.45302e-02
330	17	1.27768e-02
360	18	1.26711e-02
390	19	6.54436e-03

Tabella 3.3: Errore di approssimazione relativo all'approssimazione in figura 3.4

Nella figura 3.5 viene approssimato con una B-Spline cubica un insieme di 300 punti estratti da un'immagine vettoriale. Anche in questo caso, i nodi sono disposti secondo il metodo di de Boor (2.2.2) e viene inserito un nuovo nodo ogni 30 iterazioni. È interessante notare come vengano inseriti un maggior numero di punti di controllo nei tratti nei quali la forma della curva da ricostruire è più complessa. L'algoritmo LSPIA progressive è infatti particolarmente adatto nelle situazioni nelle quali è necessario approssimare insiemi molto grandi di punti senza stabilire a priori il numero di punti di controllo. Inoltre, se la forma da approssimare è irregolare e presenta punti di discontinuità, tramite l'inserimento di nodi multipli nella partizione nodale è possibile ridurre la continuità della spline approssimante ed ottenere risultati ancora più apprezzabili.



(a) $k = 30,\ n = 30$ (b) $k = 960,\ n = 61$ (c) $k = 2100,\ n = 99$

Figura 3.5: Approssimazione di un'immagine raffigurante un rapace con LSPIA progressive

k	Numero punti di controllo	Errore di approssimazione
0	30	1.14481e-01
10	30	7.49800e-02
20	30	7.34449e-02
30	30	7.27792e-02
60	31	6.15275e-02
90	32	5.21121e-02
120	33	5.12451e-02
210	36	4.87364e-02
360	41	4.36063e-02
480	45	4.25298e-02
600	49	2.95720e-02
720	53	2.84890e-02
960	61	2.60334e-02
1200	69	2.46944e-02
1890	92	1.77008e-02
2100	99	1.75258e-02
2520	113	1.35793e-02
3000	129	9.78224e-03

Tabella 3.4: Errore di approssimazione relativo all'approssimazione in figura 3.5

Conclusioni

In questa tesi sono stati analizzati alcuni algoritmi geometrici di interpolazione e approssimazione per curve spline della classe dei metodi denominati *Progressive Iteration Appoximation*. Questi metodi permettono di determinare i punti di controllo delle curve interpolanti o approssimanti in modo iterativo con il vantaggio di gestire insiemi di dati molto grandi in modo efficiente. Inoltre, come illustrato nell'ultima sezione, consentono di aumentare incrementalmente i punti di controllo, senza dover ricominciare le iterazioni.

Questo documento, gli algoritmi implementati in MATLAB e i risultati ottenuti possono essere utilizzati come punto di partenza per approfondire i benefici dei metodi iterativo-geometrici e possono essere integrati con numerose ottimizzazioni proposte negli ultimi anni dalla comunità scientifica. Inoltre, i metodi illustrati possono essere applicati anche in situazioni che non sono state oggetto di studio in questa tesi, come le superfici o gli spazi spline *multi-degree*.

Bibliografia

[BCM19] Carolina Vittoria Beccari, Giulio Casciola, and Serena Morigi. *Analisi Numerica e Modellazione Geometrica*. A.A. 2018/19.

[Cas20] Giulio Casciola. *Dispensa di Metodi Numerici per il Calcolo*. A.A. 2019/20.

[DB01] Carl De Boor. *A practical guide to splines*. Springer, 2001.

[DL14] Chongyang Deng and Hongwei Lin. Progressive and iterative approximation for least squares b-spline curve and surface fitting. *Computer-Aided Design*, 47:32–44, 2014.

[LBW05] Hong-Wei Lin, Hu-Jun Bao, and Guo-Jin Wang. Totally positive bases and progressive iteration approximation. *Computers & Mathematics with Applications*, 50(3):575–586, 2005.

[Lin12] Hongwei Lin. Adaptive data fitting by the progressive-iterative approximation. *Computer Aided Geometric Design*, 29(7):463–473, 2012. Geometric Modeling and Processing 2012.

[LMD18] Hongwei Lin, Takashi Maekawa, and Chongyang Deng. Survey on geometric iterative methods and their applications. *Computer-Aided Design*, 95:40–51, 2018.

[Lu10] Lizheng Lu. Weighted progressive iteration approximation and convergence analysis. *Computer Aided Geometric Design*, 27(2):129–137, 2010.

[LWD04] Hongwei Lin, Guojin Wang, and Chenshi Dong. Constructing iterative non-uniform b-spline curve and surface to fit data points. *Science in China Series F: Information Sciences*, 47:315–331, 06 2004.

[PPB02] Hartmut Prautzsch, Marco Paluszny, and Wolfgang Boehm. *Bezier and b-spline techniques.* Springer, 2002.

Elenco delle figure

Elenco delle tabelle

Ringraziamenti

Desidero ringraziare il professore Giulio Casciola per essere stato una guida paziente e disponibile durante tutto lo sviluppo della tesi. A lui va tutta la mia stima.

Esprimo inoltre profonda gratitudine alla mia famiglia e ai miei amici; il loro sostegno è stato per me fondamentale. Un ringraziamento particolare va a mio fratello Matteo che mi ha sempre motivato, supportato e sopportato in questi anni ed è da sempre per me un grande motivo di orgoglio.