Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Matematica

METODI NUMERICI PER SUPERFICI T-SPLINE

Tesi di Laurea Magistrale in Analisi Numerica

Relatore:
Chiar.mo Prof.
Giulio Casciola

Presentata da:
Giacomo Ferrari

II Sessione
Anno Accademico 2012-2013

Alla mia famiglia,
a Carlotta
e a tutti i miei amici!

Introduzione

Gli ultimi decenni sono stati caratterizzati da una crescita esponenziale del settore ingegneristico che ha portato alla realizzazione di oggetti di qualità sempre maggiore in tempi sempre minori. Tale crescita è strettamente legata allo sviluppo dei software di progettazione e analisi di oggetti, i sistemi CAD e CAE.
La sempre maggiore necessità di ottimizzare i tempi e i costi dell'intero processo di progettazione e produzione industriale di un oggetto ha portato alla creazione di nuovi metodi che hanno l'obiettivo di semplificare tale processo. Si è cercato sia di rendere più semplice ed efficiente la fase di progettazione sia di migliorare il passaggio da sistemi CAD a sistemi CAE, che ad oggi risulta la fase di maggiore perdita di tempo.
Entrambi gli ambiti hanno reso necessario lo studio sempre più approfondito delle varie tipologie di superfici che possono essere utilizzate nella progettazione di un oggetto.

Questa tesi ha come obiettivo lo studio delle superfici T-Spline. In particolare sono stati analizzati e realizzati i principali metodi numerici per la visualizzazione e la modellazione di tali superfici.
Nel capitolo introduttivo si spiega l'importanza delle T-Spline nell'ingegneria moderna descrivendo brevemente il metodo dell'*Isogeometric Analysis* e mostrando la relazione presente tra le T-Spline e tale metodo.
Nel capitolo 2 si introducono le superfici NURBS e le T-Spline, cercando di evidenziare le similitudini e le differenze presenti tra le due tipologie di superfici.

Nel capitolo 3 si introducono teoricamente gli algoritmi principali di editing di superfici T-Spline. In particolare si confrontano tali metodi con i corrispondenti algoritmi per superfici NURBS e si cerca di comprendere i vantaggi e gli svantaggi che presentano i vari algoritmi.
Nel capitolo 4 si descrive dettagliatamente come sono stati realizzati gli algoritmi per superfici T-Spline introdotti nel capitolo precedente.
Nel capitolo 5 infine si descrive brevemente il software di visualizzazione e modellazione di superfici creato e si mostrano alcuni esempi di modellazione di superfici T-Spline.

Indice

Capitolo 1

Le T-Spline e le loro applicazioni nell'ingegneria moderna

Le superfici T-Spline sono state introdotte nell'anno 2003 da Thomas W.Sederberg (Brigham Young University, Utah, USA), che le propose come un superamento delle superfici NURBS (*Non Uniform Rational B-Spline Surfaces*) e come un caso particolare delle T-NURCCs (*Non Uniform Rational Catmull-Clark's T-Surfaces*)[1].
Le superfici NURBS sono superfici che derivano direttamente dalle superfici Spline. Localmente infatti ogni *patch* di una NURBS si riconduce ad una superficie Spline con nodi uniformi. Lo svantaggio maggiore di una NURBS è dato dall'eccessiva pesantezza della sua struttura, che non consente di raffinare localmente la superficie e non permette di rappresentare oggetti di genere topologico superiore a 1 con un'unica Spline.
Gli oggetti (varietà differenziabili limitate) di genere maggiore di 1 possono essere rappresentati solamente utilizzando le Trimmed NURBS, che sono superfici ottenute dalla combinazione di più NURBS. Nel processo di "Trimming" si ritagliano parti di una o più superfici. Il problema consiste nel fatto che, avendo una struttura prodotto tensoriale, non è possibile rappresentare

esattamente il bordo della regione ritagliata nella base della superficie. Di conseguenza è possibile avere solamente un'approssimazione di tale bordo, creando così buchi e fessure che rendono la superficie *not watertigh* [4].
Le T-Spline sono superfici NURBS che ammettono nodi straordinari a valenza 2 e *T-junctions*, ovvero particolari nodi a valenza 3. L'esistenza di tali giunzioni a T consente sia di ridurre sensibilmente il numero di control point superflui presenti in una superficie NURBS sia di poter effettuare un raffinamento locale della superficie stessa, che invece le NURBS non consentono.
Le superfici T-NURCCs sono invece una generalizzazione delle superfici di Catmull-Clark. Le T-NURCCs, come le T-Spline ammettono *T-junctions* e nodi straordinari, anche se in questo caso i nodi straordinari possono avere valenza maggiore di 4.
Nell'ambito della progettazione CAD (Computer Aided Design) le T-NURCCs e le T-Spline sono molto utilizzate perché consentono di ottenere modelli geometricamente equivalenti a quelli ottenuti con superfici NURBS, ma che hanno solamente la metà dei punti di controllo [10]. Inoltre, se per raffinare una superficie NURBS è necessario inserire un'intera riga o un'intera colonna di control point, per raffinare invece una T-NURCCs o una T-Spline spesso è necessario un numero molto limitato di control point, che sono contenuti in un intorno del punto inserito.
Il vero grande vantaggio delle T-Spline e delle T-NURCCs rispetto alle NURBS consiste nella possibilità di rappresentare oggetti di genere maggiore di 1 senza aver necessariamente bisogno di "trimmare" più superfici. Per questo motivo in letteratura sono presenti algoritmi che uniscono superfici NURBS convertendole in T-Spline e successivamente le uniscono ottenendo modelli *watertigh*, ovvero privi di buchi [4].
Essendo le T-NURCCs una generalizzazione delle T-Spline, queste ultime per oltre 5 anni sono state messe da parte e non considerate. Viene quindi spontaneo chiedersi come mai negli ultimi anni le T-Spline siano state riprese in considerazione e quale sia il motivo che ha portato i progettisti a preferire tali superfici alle altre presenti in letteratura.

La progettazione e la realizzazione di oggetti in ingegneria è solitamente divisa in due grandi settori: CAD (Computer Aided Design) e CAE (Computer Aided Engineering).
L'industria CAD, come dice il nome stesso, si occupa della progettazione e della creazione della geometria che rappresenta un oggetto. Si stima che il business annuo di questo settore sia di circa 10 bilioni di dollari. L'industria CAE invece ha un business annuo di circa 2 bilionii di dollari e si occupa dell'analisi dell'oggetto, mediante l'utilizzo di metodi agli elementi finiti (che permettono di risolvere numericamente equazioni alle derivate parziali).
Più in dettaglio, nella progettazione di un oggetto si alternano fasi di modifica geometrica e fasi di analisi dell'oggetto stesso. Nella prima parte si crea o si modifica un modello CAD, mentre nella seconda parte si costruisce una mesh approssimante il modello CAD e la si analizza. Se l'analisi dell'oggetto rende necessarie alcune modifiche strutturali si deve modificare il progetto CAD, creare una nuova mesh e ripetere nuovamente l'analisi. Questo fatto comporta l'utilizzo di due geometrie completamente differenti nelle due fasi della progettazione e richiede di passare spesso dall'una all'altra. È stato studiato che circa l'80% del tempo dedicato all'analisi viene utilizzato per generare una mesh da un modello CAD con una notevole perdita di tempo che rallenta la progettazione.
È diventata così chiara a tutti la necessità di trovare un modo per integrare i sistemi CAD e CAE, riducendo i tempi di passaggio da geometria CAD a mesh. Inoltre le mesh approssimano solamente la geometria CAD, per cui è possibile che durante il processo di analisi tali approssimazioni portino ad errori consistenti nei risultati. Di conseguenza non è sufficiente trovare un metodo per passare velocemente da geometria CAD a mesh, ma si deve riformulare l'intero processo che lega design ingegneristico e analisi in modo da ovviare ad entrambi i problemi.

Nel 2004 T.J.R.Hughes (The University of Texas, Austin, USA) ha ideato un innovativo metodo per l'analisi di problemi legati alla risoluzione di equa-

zioni alle derivate parziali: l'*isogeometric analysis* (IGA) [5].
Il metodo di Hughes si propone di eliminare il passaggio da CAD a CAE applicando l'analisi direttamente alla superficie NURBS. In questo modo il dominio computazionale coincide esattamente con il dominio descritto dal modello CAD.
Oltre al notevole risparmio di tempo che si ottiene, le NURBS consentono di effettuare sia *h-refinement* (inserimento di nodi) che *p-refinement* (aumento del grado del polinomio), esattamente come si può fare sulle mesh. Inoltre è possibile effettuare anche un particolare tipo di raffinamento, il *k-refinement* che, sfruttando la non commutabilità dei processi di h-refinement e p-refinement, permette di raffinare la geometria aumentando sia il grado sia la regolarità globale delle funzioni di base.
Le NURBS hanno l'enorme vantaggio di poter rappresentare esattamente le superfici quadriche e inoltre sono a tutt'oggi le superfici utilizzate nell'ambiente CAD.
Come si è detto in precedenza, le NURBS hanno però una struttura molto pesante in cui molti punti di controllo sono superflui e presentano problemi nell'intersezione di due o più superfici. Di conseguenza l'aumento esponenziale o quasi della complessità dei modelli ingegneristici ha reso necessaria, negli anni successivi al 2004, la ricerca di nuove superfici che ovviassero ai problemi delle NURBS.
Nel 2008, in un articolo scritto da vari autori tra cui Hughes e Sederberg, sono elencate le principali superfici che consentono di ovviare ai limiti delle NURBS: le T-NURCCs, le T-Spline e le Subdivision Surfaces.
Le Subdivision Surfaces sono superfici che, come le T-Spline, non presentano problemi di buchi o fessure nel trimming tra due o più superfici e inoltre consentono di rappresentare oggetti a topologia arbitraria. Queste particolari superfici, proposte da alcuni studiosi tra cui E.Catmull, sono tuttora utilizzate per modellare quasi tutti i personaggi delle animazioni Pixar.
Le T-Spline, tra le superfici elencate, sono le uniche compatibili con le NURBS in quanto una NURBS si ottiene da una T-Spline semplicemnte aggiungendo

punti di controllo che non modificano la superficie.
La compatibilità tra i due tipi di superfici presenta un duplice vantaggio. Innanzitutto non vengono resi inutili gli investimenti milionari che vengono fatti per portare avanti lo sviluppo delle NURBS in ambito CAD. Inoltre, cosa ancora più importante, la compatibilità tra i due tipi di superficie garantisce la coesistenza tra NURBS e T-Spline nel periodo di graduale adozione di queste ultime e permette così di fare molteplici test per ottimizzare i metodi implementati.

Gli studi più recenti (2011-2012) stanno cominciando a testare le potenzialità delle T-Spline nell'*isogeometric analysis*. Si è visto che le T-Spline possiedono proprietà di convergenza simili a quelle delle NURBS, però con molti meno gradi di libertà e consentono di trovare soluzioni più accurate e robuste per i metodi agli elementi finiti utilizzati nell'analisi.
È stata poi definita una classe di superfici T-Spline, le *Analysis Suitable T-Spline*, che rispondono alle necessità sia del design che dell'analisi, mantenendo alcune proprietà fondamentali delle NURBS quali l'indipendenza lineare e la partizione dell'unità.

Capitolo 2

Introduzione alle superfici T-Spline

Le superfici T-Spline possono essere considerate un'evoluzione delle superfici NURBS. Per poter comprendere come è definita una T-Spline e quali sono le idee di base che hanno permesso l'evoluzione dalle NURBS alle T-Spline è necessario introdurre per primi gli strumenti che hanno consentito di costruire le superfici NURBS.
Una superficie NURBS (Non Uniform Rational B-Spline) si può vedere come rapporto di due superfici NUBS (Non Uniform B-Spline) che appartengono allo spazio delle funzioni spline polinomiali bivariate. Tale spazio è definito come prodotto tensoriale di due spazi di funzioni spline polinomiali monovariate. Per questo motivo nella prima parte del capitolo si definiscono le funzioni NUBS e NURBS monovariate.
Successivamente viene introdotto il concetto di spazio prodotto tensoriale, che consente di definire le superfici NUBS. Come nel caso monovariato le superfici NURBS e gli spazi di funzioni Spline bivariate razionali vengono definiti di seguito ampliando il concetto di superficie NUBS.
L'ultima parte del capitolo è dedicata all'introduzione delle T-Spline e in particolare si definiscono le T-Mesh e le loro regole fondamentali.

2.1 Curve NUBS e NURBS

Funzioni NUBS

Si consideri $\mathbb{P}_m$ spazio dei polinomi a coefficienti reali di ordine m.

Definizione 2.1. Sia dato $[a, b]$ intervallo chiuso e limitato e $\Delta = \{x_i\}_{i=1\ldots k}$ insieme di nodi distinti contenuti strettamente in $[a, b]$.
Si pongono $x_0 = a,\ x_{k+1} = b$ e si costruisce la partizione di $[a, b]$ in $k + 1$ sottointervalli data da Δ:

$$\begin{aligned} I_j &= [x_j, x_{j+1}) \quad j = 0 \ldots k-1 \\ I_k &= [x_k, x_{k+1}] \end{aligned}$$

Si definisce spazio dei polinomi a tratti di ordine al più m a coefficienti reali lo spazio:

$$P\mathbb{P}_m = \{f \mid \exists\ p_0 \ldots p_k \in \mathbb{P}_m \ t.c.\ f(x) = p_i(x) \quad \forall x \in I_i, \quad i = 0 \ldots k\}$$

I polinomi a tratti hanno il vantaggio di essere più flessibili rispetto ai polinomi definiti in un intervallo $[a, b]$. La mancanza di condizioni imposte nei nodi appartenenti a Δ può però causare una perdita di regolarità del polinomio. Per mantenere la flessibilità e allo stesso tempo non perdere regolarità si devono introdurre alcune condizioni aggiuntive che portano a definire lo spazio delle Spline a nodi semplici.

Definizione 2.2. Sia dato $[a, b]$ intervallo e Δ insieme di nodi strettamente contenuti in $[a, b]$. Sia m un intero positivo. Si definisce lo spazio delle funzioni spline polinomiali di ordine m a nodi semplici come:

$$S_m(\Delta) = P\mathbb{P}_m \cap C^{m-2}_{[a,b]}$$

In questo modo le funzioni polinomiali a tratti in $S_m(\Delta)$ sono sufficientemente flessibili e regolari.
È possibile poi generalizzare la definizione appena data introducendo lo spazio delle funzioni Spline a nodi multipli:

Definizione 2.3. Sia $[a, b]$ intervallo chiuso e limitato, Δ definita come sopra. Si pongono $x_0 = a$, $x_{k+1} = b$ e si costruisce la partizione di $[a, b]$ in $k + 1$ sottointervalli data da Δ come sopra.
Sia, poi, m un intero positivo e $M = (m_1, m_2, \ldots, m_k)$ un vettore di interi positivi *t.c.* $1 \leq m_i \leq m \ \forall \ i = 1 \ldots k$.
Si definisce l'insieme delle Spline di ordine m con nodi $x_1 \ldots x_k$ di molteplicità $m_1 \ldots m_k$ come:

$$\begin{aligned} S(\mathbb{P}_m, M, \Delta) = \{ & s(x) \mid \exists \ s_0(x) \ldots s_k(x) \ \in \ \mathbb{P}_m \quad t.c. \\ & 1. \ s(x) = s_i(x) \quad \text{per} \quad i \ \in \ I_i \ i = 0 \ldots k \\ & 2. \ s_i^{(l)}(x_i) = s_{i+1}^{(l)}(x_i) \quad l = 0 \ldots m - m_i - 1, \quad i = 1 \ldots k \ \} \end{aligned}$$

Dove $s_i^{(l)}(x_i)$ indica la $l-$esima derivata di s_i valutata in x_i.

Osservazione 1. Lo spazio $S(\mathbb{P}_m, M, \Delta)$ contiene tutti gli spazi definiti in precedenza, infatti:

. Se $M = (1, \ldots, 1)$ si ottiene esattamente lo spazio $S_m(\Delta)$ in quanto si richiede che le funzioni siano polinomiali a tratti con regolarità almeno C^{m-2} nei nodi della partizione Δ.

. Se $M = (m, \ldots, m)$ si ritorna al caso dei polinomi a tratti, visto che nei punti x_i si richiede una regolarità $m - m - 1 \ = \ -1$.

. Valgono poi le seguenti inclusioni:

$$\mathbb{P}_m \ \subseteq \ S_m(\Delta) \ \subseteq \ P\mathbb{P}_m$$

Quindi lo spazio delle funzioni Spline a nodi multipli contiene lo spazio dei polinomi di ordine m.

L'insieme $S(\mathbb{P}_m, M, \Delta)$ è uno spazio di funzioni di dimensione $m + K$ con

$$K = \sum_{i=1}^{k} m_i$$

e ogni elemento $s(x) \in S(\mathbb{P}_m, M, \Delta)$ si può scrivere come:

$$s(x) = \sum_{i=1}^{m+K} c_i \varphi_i$$

dove $\{\varphi_1, \dots \varphi_{m+K}\}$ è una base dello spazio.
Lo spazio delle funzioni Spline a nodi multipli ha una base stabile dal punto di vista computazionale: la base delle B-Spline Normalizzate. Prima di definire tale base è necessario però introdurre il concetto di partizione estesa.

Definizione 2.4. L'insieme $\Delta^* = \{t_i\}_{i=1\dots 2m+K}$ con m, K come sopra si chiama partizione estesa associata a $S(\mathbb{P}_m, M, \Delta)$ se e solo se:

1. $t_1 \leq t_2 \leq \dots \leq t_{2m+K-1} \leq t_{2m+K}$
2. $t_m \equiv a$, $t_{m+K+1} \equiv b$
3. $(t_{m+1}, t_{m+2}, \dots, t_{m+K}) \equiv \underbrace{x_1, \dots, x_1}_{m_1 \ volte}, \dots, \underbrace{x_k, \dots, x_k}_{m_k \ volte}$

Definizione 2.5. Si consideri la partizione estesa Δ^* associata allo spazio $S(\mathbb{P}_m, M, \Delta)$. Si definisce l'insieme delle funzioni B-spline Normalizzate

$$\{N_{i,m}(x)\}_{i=1\dots m+K}$$

mediante la formula ricorrente:

$$N_{i,h}(x) = \begin{cases} \dfrac{x - t_i}{t_{i+h-1} - t_i} N_{i,h-1}(x) + \dfrac{t_{i+h} - x}{t_{i+h} - t_{i+1}} N_{i+1,h-1}(x) & se\ t_i \leq t_{i+h} \\ 0 & altrimenti \end{cases}$$

per $h = 2 \dots m$ dove:

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

e dove eventuali casi di $\dfrac{0}{0}$ si devono considerare come 0.

Definizione 2.6. Si definiscono funzioni NUBS (Non Uniform B-spline) le funzioni dello spazio $S(\mathbb{P}_m, M, \Delta)$ rappresentate nella base delle B-spline Normalizzate, ovvero scritte come:

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

Le funzioni base B-Spline Normalizzate, e di conseguenza le funzioni NUBS godono di alcune fondamentali proprietà:

1. Supporto Locale:

$$N_{i,m}(x) = 0 \; \forall \; x \notin [t_i, t_{i+m}] \; se \; t_i < t_{i+m}$$

2. Non Negatività:

$$N_{i,m}(x) > 0 \; \forall \; x \in (t_i, t_{i+m}) \; se \; t_i < t_{i+m}$$

   L'intervallo $[t_i, t_{i+m})$ è detto supporto della $i-$esima funzione base.

3. Partizione dell'unità:

$$\sum_{i=1}^{m+K} N_{i,m}(x) \equiv 1 \; \forall \; x \in [a, b]$$

4. Sfruttando le proprietà appena elencate, si dimostra che una funzione NUBS $s(x)$ è una combinazione convessa dei coefficienti c_i della sua espressione nella base delle B-Spline Normalizzate. Di conseguenza si ha che:

$$min_{i=1\ldots m+K}(c_i) \leq s(x) \leq max_{i=1\ldots m+K}(c_i) \quad x \in [a, b]$$

   e, per la prima proprietà elencata:

$$min_{i=l-m+1\ldots l}(c_i) \leq s(x) \leq max_{i=l-m+1\ldots l}(c_i) \quad x \in [t_l, t_{l+1})$$

Osservazione 2. Le proprietà 1 e 2 consentono di definire il *vettore nodale* associato alla funzione base B-Spline Normalizzata $N_{i,m}(x)$ come

$$t = (t_i, t_{i+1}, \ldots, t_{i+m})$$

.

Osservazione 3. Osservando la formula delle funzioni NUBS si può comprendere come ad ogni coefficiente c_i sia associata una ben precisa funzione base $N_{i,m}$, la quale è univocamente determinata da un vettore nodale t.
Una funzione NUBS si può rappresentare nel piano $\mathbb{R}^2$ ponendo in ascissa i nodi della partizione nodale e in ordinata i valori che la funzione assume nei punti $x \in [a, b]$. Per ogni funzione base B_i si può poi considerare la coppia (p_i, c_i), dove

$$p_i = \frac{1}{m-1} \sum_{j=i+1}^{i+m-1} t_j$$

è l'ascissa di Greville dell'intervallo nodale di $N_{i,m}$ e c_i il coefficiente associato a $N_{i,m}$. La curva spezzata che si ottiene congiungendo coppie consecutive di punti (p_i, c_i) approssima in forma la curva e si chiama poligonale di controllo della funzione NUBS.

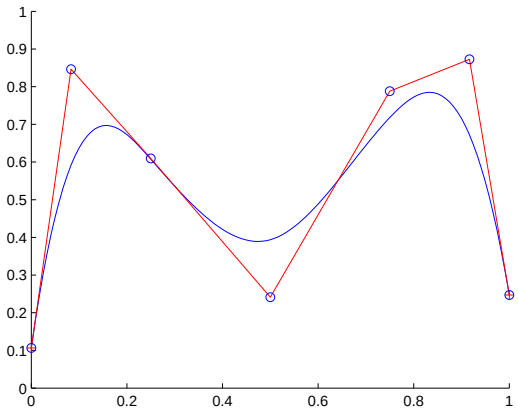

Figura 2.1: Funzione NUBS cubica e poligonale di controllo

Esempio 2.1. Nella figura 2.1 è mostrata in blu una funzione NUBS cubica univariata definita sulla partizione nodale estesa

$$\Delta^* = (0,\ 0,\ 0,\ 0,\ 0.25,\ 0.5,\ 0.75,\ 1,\ 1,\ 1,\ 1)$$

La poligonale di controllo della curva, rappresentata in rosso, è composta da 7 punti di controllo (p_i, c_i) a cui è associata una funzione base $N_{i,4}$ con dominio di influenza dato dal vettore $(t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2})$, dove le t_i sono gli elementi di Δ^*.
Ogni tratto di curva compreso tra due punti di controllo consecutivi è definito da tutte le funzioni base $N_{i,4}$ che hanno dominio di influenza contenente entrambi i punti di controllo considerati.
Come si è osservato in precedenza la poligonale di controllo approssima in forma la curva.

Knot insertion per funzioni NUBS

Si consideri la partizione estesa Δ^* definita sull'intervallo $[a, b]$ e sia $\hat{t}$ un nodo di $[a, b]$ da inserire in Δ^*. Sia l *t.c.* $t_l < \hat{t} \leq t_{l+1}$, si può costruire una nuova partizione estesa $\hat{\Delta}^*$ conentente i $2m + K$ nodi di Δ^* e il nuovo nodo $\hat{t}$ ordinati:

$$\hat{\Delta}^* = \{\hat{t_j}\}_{j=1\ldots 2m+K+1}$$

dove:

$$\hat{t_j} = \begin{cases} t_j & se\ j \leq l \\ \hat{t} & se\ j = l+1 \\ t_{j-1} & se\ j \geq l+1 \end{cases}$$

La nuova partizione estesa $\hat{\Delta}^*$ è associata allo spazio $S(\mathbb{P}_m, \hat{M}, \hat{\Delta})$, di dimensione $m + K + 1$ che contiene lo spazio $S(\mathbb{P}_m, M, \Delta)$ associato a Δ^*.
Esiste un teorema che spiega la relazione che lega le funzioni spline appartenenti a $S(\mathbb{P}_m, M, \Delta)$ alle funzioni di $S(\mathbb{P}_m, \hat{M}, \hat{\Delta})$, ovvero spiega come esprimere una funzione $s(x) \in S(\mathbb{P}_m, M, \Delta)$ come funzione di $S(\mathbb{P}_m, \hat{M}, \hat{\Delta})$.

Teorema 2.1.1. *Siano Δ^* e $\hat{\Delta}^*$ le due partizioni nodali estese introdotte sopra. Vale la seguente relazione:*

$$N_{im}(x) = \begin{cases} \hat{N}_{i,m}(x) & se\ i \leq l-m \\ \dfrac{\hat{t} - \hat{t}_i}{\hat{t}_{i+m} - \hat{t}_i}\hat{N}_{i,m}(x) + \dfrac{\hat{t}_{i+m+1} - \hat{t}}{\hat{t}_{i+m+1} - \hat{t}_{i+1}}\hat{N}_{i+1,m}(x) & se\ l-m+1 \leq i \leq l \\ \hat{N}_{i+1,m}(x) & se\ i \geq l+1 \end{cases}$$

Di conseguenza, utilizzando la formula del teorema, si ha che una funzione NUBS $s(x) = \sum_{i=1}^{m+K} c_i N_{i,m}(x)$ espressa nella base delle B-Spline normalizzate dello spazio $S(\mathbb{P}_m, \hat{M}, \hat{\Delta})$ si può esprimere come

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

nella corrispondente base dello spazio $S(\mathbb{P}_m, \hat{M}, \hat{\Delta})$. I coefficienti c_i e $\hat{c}_i$ sono legati dalla seguente relazione:

$$\hat{c}_i = \begin{cases} c_i & se\ i \leq l - m + 1 \\ \lambda_i c_i + (1 - \lambda_i) c_{i-1} & se\ l - m + 2 \leq i \leq l \\ c_{i-1} & se\ i \geq l + 1 \end{cases} \tag{2.1}$$

dove $\lambda_i = \dfrac{\hat{t} - \hat{t}_i}{\hat{t}_{i+m} - \hat{t}_i} = \dfrac{\hat{t} - t_i}{t_{i+m-1} - t_i}$.
L'algoritmo di knot insertion per funzioni NUBS segue pedissequamente le formule appena date. Vengono calcolati i coefficienti moltiplicativi λ_i e successivamente si riscrivono sia i coefficienti $\hat{c}_i$ che i vettori nodali delle funzioni base associate ai coefficienti modificati. In questo modo la funzione NUBS rimane invariata e viene espressa nella nuova base B-Spline Normalizzata associata al nuovo spazio ottenuto.

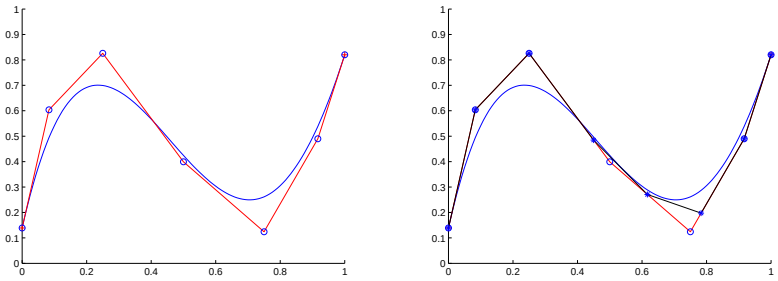

Figura 2.2: Knot insertion su una curva NUBS scalare

Esempio 2.2. Per vedere praticamente il funzionamento dell'algoritmo di knot insertion per funzioni NUBS, si consideri la figura 2.2a in cui è mostrata

una curva NUBS scalare definita sulla partizione nodale estesa

$$\Delta^* \;=\; (0,\ 0,\ 0,\ 0,\ 0.25,\ 0.5,\ 0.75,\ 1,\ 1,\ 1,\ 1)$$

Si inserisce nella partizione il nodo singolo $t = 0.6$. Seguendo l'equazione (2.1) i coefficienti c_i già presenti vengono modificati e viene introdotto un nuovo punto di controllo associato al nodo inserito. In figura 2.2a si vede che la nuova poligonale di controllo approssima meglio in forma la funzione NUBS, che rimane invariata.

Funzioni NURBS

Le funzioni NURBS (Non Uniform Rational B-Spline) sono particolari funzioni Spline razionali. Per questo motivo è necessario innanzitutto definire gli spazi di funzioni Spline razionali per poi vedere gli elementi che caratterizzano le NURBS.

Definizione 2.7. Si consideri un intervallo $[a, b]$ chiuso e limitato.
Sia $\Delta = \{x_i\}_{i=1\ldots k}$ insieme di nodi distinti contenuti strettamente in $[a, b]$.
Si pongono $x_0 = a$, $x_{k+1} = b$ e si costruisce la partizione di $[a, b]$ in $k + 1$ sottointervalli data da Δ:

$$\begin{aligned} I_j &= [x_j, i_{j+1}) \quad i = 0 \ldots k-1 \\ I_k &= [x_k, i_{k+1}] \end{aligned}$$

Si fissa poi $\forall\ I_i$ un polinomio $q_i(x) \in \mathbb{P}_m$ *t.c.* $q_i(x) > 0 \quad \forall\ x \in I_i$.
L'insieme delle funzioni Spline razionali su Δ è definito come:

$$\begin{aligned} \{r(x) \mid \exists\ p_i(x) \ \ in\ \mathbb{P}_n \text{ e valgano} : 1.\ & r(x) = \frac{p_i(x)}{q_i(x)} \quad x \in I_i \\ 2.\ & p_{i-1}^{(l)}(x_i) = p_i^{(l)}(x_i) \\ & q_{i-1}^{(l)}(x_i) = q_i^{(l)}(x_i) \\ & l = 0 \ldots h, \quad i = 1 \ldots k, \\ & h = \min(m, n) - 1\ \} \end{aligned}$$

Osservazione 4. Tale definizione non è unica, ma consente di mantenere il concetto di massima regolarità visto nel caso delle funzioni Spline e permette di dire che una funzione Spline razionale altro non è che il rapporto tra due funzioni Spline non razionali.

Definizione 2.8. Si definisce funzione NURBS (Non Uniform Rational B-Spline) una funzione appartenente alla classe delle funzioni Spline razionali date dal rapporto di due funzioni non razionali, entrambe di ordine m, rappresentate nella base delle B-Spline Normalizzate.
Tali funzioni si esprimono come:

$$r(x) = \frac{\sum_{i=1}^{m+K} d_i N_{i,m}(x)}{\sum_{j=1}^{m+K} w_j N_{j,m}(x)} \qquad x \in [a,b]$$

dove si richiede che i w_i siano positivi per evitare singolarità. Poichè è sempre possibile trovare dei valori $c_i = \frac{d_i}{w_i}$, una funzione NURBS si può scrivere come:

$$r(x) = \frac{\sum_{i=1}^{m+K} c_i w_i N_{i,m}(x)}{\sum_{j=1}^{m+K} w_j N_{j,m}(x)} \qquad x \in [a,b]$$

Definizione 2.9. Lo Spazio delle funzioni NURBS è uno spazio di dimensione $m + K$ e si indica con

$$R(\mathbb{P}_m, M, \Delta, W)$$

dove $\mathbb{P}_m$ spazio dei polinomi di grado m, $M = (m_1, \ldots, m_k)$, Δ partizione nodale di $[a,b]$, $W = (w_1, \ldots, w_k)$ vettore dei pesi positivi e $K = \sum_{i=1}^{k} m_i$.

Anche in questo caso è possibile trovare una base per lo spazio delle funzioni NURBS analoga alla base delle B-Spline Normalizzate.

Definizione 2.10. Dato lo spazio $R(\mathbb{P}_m, M, \Delta, W)$ di dimensione $m+K$ si definiscono funzioni B-Spline Razionali Normalizzate di ordine m le $m+K$ funzioni

$$R_{i,m}(x) = \frac{w_i N_{i,m}(x)}{\sum_{i=1}^{m+K} w_i N_{i,m}(x)}$$

dove $\{N_{i,m}(x)\}$ sono le funzioni B-Spline Normalizzate definite sulla partizione estesa Δ^*.

Osservazione 5. Dalla definizione segue che una funzione NURBS $r(x) \in R(\mathbb{P}_m, M, \Delta, W)$ si può esprimere come

$$r(x) = \sum_{i=1}^{m+K} c_i R_{i,m}(x)$$

Inoltre si vede facilmente che le funzioni B-Spline Razionali Normalizzate e le funzioni NURBS godono delle stesse proprietà citate in precedenza per le funzioni B-Spline Normalizzate e per le funzioni NUBS.

Knot insertion per funzioni NURBS

Sia data una partizione estesa Δ^* in cui si vuole inserire un nodo $\hat{t}$. Esattamente come accade per le NUBS inserendo il nodo $\hat{t}$ si ottiene una nuova partizione estesa $\hat{\Delta}^*$ associata ad uno spazio di funzioni spline razionali $R(\mathbb{P}_m, \hat{M}, \hat{\Delta}, \hat{W})$ che contiene lo spazio $R(\mathbb{P}_m, M, \Delta, W)$.
In questo caso, essendo la funzione NURBS espressa come rapporto di due funzioni NUBS, i nuovi coefficienti $\hat{c}_i$ si esprimeranno come

$$\hat{c}_i = \frac{c_i \hat{w}_i}{\hat{w}_i}$$

dove:

$$\hat{w}_i = \begin{cases} w_i & se\ i \leq l-m+1 \\ \lambda_i w_i + (1-\lambda_i) w_{i-1} & se\ l-m+2 \leq i \leq l \\ w_{i-1} & se\ i \geq l+1 \end{cases}$$

mentre:

$$c_i\hat{w}_i = \begin{cases} c_i & se\ i \leq l-m+1 \\ \lambda_i c_i \dfrac{w_i}{\hat{w}_i} + (1-\lambda_i)c_{i-1}\dfrac{w_{i-1}}{\hat{w}_{i-1}} & se\ l-m+2 \leq i \leq l \\ c_{i-1} & se\ i \geq l+1 \end{cases}$$

con $\lambda_i = \dfrac{\hat{t} - \hat{t}_i}{\hat{t}_{i+m} - \hat{t}_i}$ in entrambi i casi. L'algoritmo di knot insertion per funzioni NURBS ha la stessa struttura di quello per funzioni NUBS e segue precisamente le formule appena date. I coefficienti λ_i vengono calcolati allo stesso modo in entrambi gli algoritmi. Ciò che cambia sono le formule che vengono successivamente utilizzate per ottenere i nuovi coefficienti e i nuovi pesi.

Curve NUBS e NURBS in forma parametrica

Una curva NUBS in $\mathbb{R}^n$ espressa in forma parametrica è una funzione vettoriale le cui n componenti sono le funzioni già introdotte. Più precisamente:

Definizione 2.11. Una curva $C(t) \in \mathbb{R}^n$ si scrive come:

$$C(t) = \begin{pmatrix} x_1 = C_1(t) \\ \vdots \\ x_n = C_n(t) \end{pmatrix}$$

dove $t \in [0,1]$ è un parametro.
$C(t)$ è una funzione vettoriale. In particolare se le $C_i(t)$ sono funzioni spline, ovvero $C_i(t) \in S(\mathbb{P}_m, M, \Delta)\ \ i = 1 \ldots n$, allora si dice che $C(t)$ è una curva NUBS in forma parametrica.

Definizione 2.12. Una curva $C(t) \in \mathbb{R}^n$ le cui componenti sono funzioni spline razionali, ovvero $C_i(t) \in R(\mathbb{P}_m, M, \Delta, W)\ \ i = 1 \ldots n$, è detta curva NURBS in forma parametrica.

L'espressione di tale curva è:

$$C(t) = \sum_{i=1}^{m+K} P_i R_{i,m}(t) = \frac{\sum_{i=1}^{m+K} w_i P_i N_{i,m}(t)}{\sum_{j=1}^{m+K} w_j N_{j,m}(t)}$$

dove i punti $P_i \in \mathbb{R}^n$.

Osservazione 6. Per le curve NUBS o NURBS in forma parametrica valgono le stesse osservazioni fatte in precedenza per le funzioni NUBS. In questo caso i coefficienti della curva sono i punti P_i, detti anche punti di controllo. La poligonale di controllo della curva si rappresenta in $\mathbb{R}^n$ come la spezzata che unisce punti di controllo consecutivi. In generale esiste quindi una stretta relazione tra punti di controllo, funzioni base ad essi associate e intervalli nodali.
Gli algoritmi di knot insertion per curve NUBS/NURBS sono identici a quelli visti per funzioni NUBS/NURBS. Nel caso delle curve vettoriali semplicemente bisogna calcolare i nuovi coefficienti $\hat{c}_i$ per ognuna delle n componenti della curva.

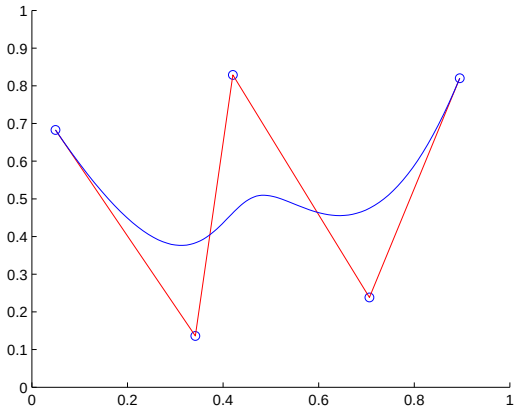

Figura 2.3: Curva NUBS cubica di $\mathbb{R}^2$

Esempio 2.3. Si consideri la figura 2.3 in cui è rappresentata una curva NUBS cubica di $\mathbb{R}^2$ e la sua poligonale di controllo. Le funzioni base $N_{i,4}$ sono 5, quindi essendo $5 = m+K$ e $m = g+1 = 4$, necessariamente $K = 1$. I nodi della partizione estesa sono $2m+K = 9$ e i nodi esterni sono coincidenti, quindi si ha il vettore

$$\Delta^* \;=\; [t_1^* \dots t_9^*] \;=\; [0\ 0\ 0\ 0\ 0.5\ 1\ 1\ 1\ 1]$$

La funzione base $N_{i,4}$ ha vettore nodale dato dagli elementi $(t_i^*, \dots t_{i+4}^*)$. I punti di controllo sono rappresentati con dei cerchi blu, mentre la poligonale di controllo è disegnata in rosso.

2.2 Superfici NUBS e NURBS

L'evoluzione da funzione NUBS monovariata a bivariata avviene nel modo più intuitivo possibile: attraverso il prodotto tensoriale.
Successivamente, il passaggio da funzione NUBS bivariata a funzione NURBS bivariata avviene in modo analogo al caso monovariato.
Il passaggio da NURBS monovariata a bivariata è invece l'unico che non avviene attraverso un prodotto tensoriale, in quanto le funzioni base NURBS bivariate non sono in generale prodotto di due funzioni base NURBS monovariate.

Superfici NUBS

Siano dati $S_x = (\mathbb{P}_n, N, \Delta_x)$ e $S_y = (\mathbb{P}_m, M, \Delta_y)$ due spazi di funzioni spline monovariate, definite rispettivamente in $[a, b]$ e $[c, d]$ di ordine m e n, con partizioni nodali $\Delta_x = \{x_i\}_{i=1\dots h}$ e $\Delta_y = \{y_i\}_{i=1\dots k}$, con vettori delle molteplicità $N = (n_1, \dots, n_h)$ e $M = (m_1, \dots, m_k)$ associati a Δ_x e Δ_y e di dimensione rispettivamente $n + H$ e $m + K$, dove $H = \sum_{i=1}^{h} n_i$ e $K = \sum_{i=1}^{k} m_i$. Lo spazio prodotto tensoriale di S_x e S_y è lo spazio $S_x \otimes S_y$ di dimensione $(n + H) * (m + K)$ che si può scrivere come

$$S_x \otimes S_y = (\mathbb{P}_{n,m}, N, M, \Delta_x \times \Delta_y)$$

Definizione 2.13. Si definisce funzione NUBS prodotto tensoriale associata alla partizione nodale $\Delta_x \times \Delta_y$

$$s(x,y) = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} c_{ij} N_{i,n}(x) N_{j,m}(y)$$

dove le $N_{i,n}(x)$ e le $N_{j,m}(y)$ sono le funzioni B-Spline Normalizzate degli spazi S_x e S_y, associate alle partizioni nodali estese $\Delta_u^* = \{u_i\}_{i=1\ldots 2n+H}$ e $\Delta_v^* = \{v_j\}_{j=1\ldots 2m+K}$ rispettivamente.

Le funzioni NUBS prodotto tensioriale sono gli elementi dello spazio di funzioni $S_x \otimes S_y$. Una base dello spazio prodotto tensoriale è data dalle funzioni bivariate

$$N_{i,n,j,m}(x,y) = N_{i,n}(x) N_{j,m}(y)$$

che godono di proprietà analoghe a quelle delle funzioni B-Spline Normalizzate viste nel caso monovariato:

1. Supporto Locale:

$$N_{i,n,j,m}(x) = 0$$

$$\forall\ (x,y) \notin [u_i, u_{i+n}] \times [v_j, v_{j+m}]\ se\ u_i < u_{i+n}\ se\ v_j < v_{j+m}$$

2. Non Negatività:

$$N_{i,n,j,m}(x,y) > 0$$

$$\forall\ (x,y) \in (u_i, u_{i+n}) \times (v_j, v_{j+m})\ se\ u_i < u_{i+n}\ se\ v_j < v_{j+m}$$

3. Partizione dell'unità:

$$\sum_{i=1}^{n+H} \sum_{j=1}^{m+K} N_{i,n}(x) N_{j,m}(y) = 1$$

$$\forall\ (x,y)\ \in\ [a,b] \times [c,d]$$

4. Si dimostra poi che $s(x,y)$ è una combinazione convessa dei coefficienti $c_{i,j}$ della sua espressione nella base delle B-Spline Normalizzate bivariate.

Osservazione 7. I *vettori nodali* in u e in v associati alla funzione base $N_{i,n,j,m}(x)$ si determinano in modo analogo al caso univariato. Tali vettori definiscono una regione del piano (u, v) chiamata dominio di influenza della funzione base, che corrisponde alla parte di piano in cui la funzione base è non nulla.

Superfici NURBS

Esattamente come nel caso monovariato, una funzione NURBS bivariata si può vedere come rapporto di due funzioni NUBS bivariate non razionali.

Definizione 2.14. Si definisce funzione NURBS bivariata

$$r(x,y) = \frac{\sum_{i=1}^{n+H}\sum_{j=1}^{m+K} c_{i,j} w_{i,j} N_{i,n}(x) N_{j,m}(y)}{\sum_{i=1}^{n+H}\sum_{j=1}^{m+K} w_{i,j} N_{i,n}(x) N_{j,m}(y)}$$

dove $w_{i,j} > 0$.

Tale funzione si può scrivere anche come:

$$r(x,y) = \sum_{i=1}^{n+H}\sum_{j=1}^{m+K} c_{i,j} R_{i,n,j,m}(x,y)$$

dove le $R_{i,n,j,m}(x,y)$ sono le funzioni base B-Spline Razionali Bivariate definite da:

$$R_{i,n,j,m}(x,y) = \frac{w_{i,j} N_{i,n}(x) N_{j,m}(y)}{\sum_{i=1}^{n+H}\sum_{j=1}^{m+K} w_{i,j} N_{i,n}(x) N_{j,m}(y)}$$

Osservazione 8. Il modo in cui sono definite le funzioni base $R_{i,n,j,m}(x,y)$ consente di vedere che lo spazio

$$R(\mathbb{P}_{n,m}, N, M, \Delta_x \times \Delta_y, W)$$

da esse formato non è uno spazio prodotto tensoriale, in quanto le funzioni base non sono prodotto delle corrispondenti funzioni B-Spline Bivariate.

Osservazione 9. Le superfici NUBS e NURBS possono essere rappresentate utilizzando una griglia di punti 3D (ξ_i, η_j, c_{ij}) con $i = 1, \ldots, n + H$ e $j = 1, \ldots, m + K$ dove ξ_i e η_j sono le ascisse di Greville dei vettori nodali in u e v. Tale griglia è chiamata griglia di controllo della funzione ed è una mesh regolare che contiene solamente facce con 4 lati e ha tutti i punti di controllo interni con valenza 4.
Se invece si rappresenta nel piano la superficie (ξ_i, η_j) con $i = 1, \ldots, n + H$ e $j = 1, \ldots, m + K$ si ha la preimmagine della griglia di controllo.
La griglia di controllo estende il concetto di poligonale di controllo visto per curve NUBS e NURBS. Le relazioni tra coefficienti, funzioni base e intervalli nodali sono le stesse. Nel caso bivariato ad un coefficiente è associata la sua funzione base, definita su un dominio di influenza dato dai suoi vettori nodali.

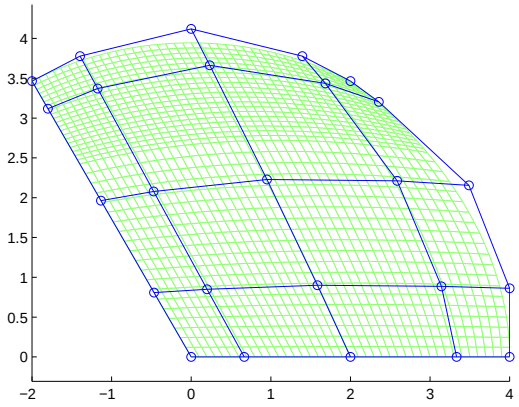

Figura 2.4: Griglia di una superficie NURBS

Esempio 2.4. Per vedere meglio quanto appena detto si consideri ad esempio la figura 2.4 in cui è rappresentata una superficie NURBS cubica. La griglia 5×5 mostra i punti di controllo della superficie, che hanno tutti come coefficiente associato 0.
Si osserva che le funzioni base che definiscono la superficie sono 25, tante

quante i punti di controllo. Inoltre si può vedere come la griglia di controllo dia una approssimazione della forma della superficie.

Knot Insertion per superfici NUBS e NURBS

L'algoritmo di knot insertion per superfici NUBS/NURBS scalari consente di inserire un nodo in una delle due partizioni nodali estese (Δ_u^* o Δ_v^*). I nuovi coefficienti $\hat{c}_{ij}$ vengono calcolati a partire dai vecchi seguendo le stesse regole viste per il caso univariato. Poichè il nodo viene inserito in una delle due partizioni nodali la modifica dei coefficienti avviene su ogni riga (o colonna) della matrice dei coefficienti.
Esattamente come nel caso univariato l'inserimento di nuovi nodi su una delle due partizioni nodali comporta l'inserimento di nuovi punti di controllo, che saranno aggiunti su un'intera riga (o colonna) della griglia di controllo. Di conseguenza non è possibile modificare localmente la superficie, ma solo globalmente.

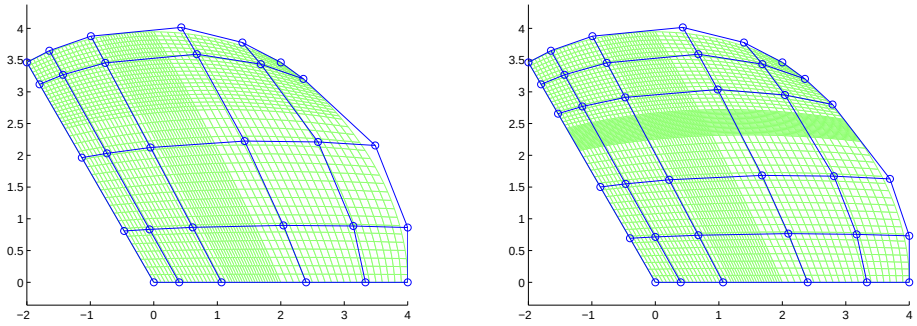

Figura 2.5: Raffinamento di una superficie NURBS: Fig a: knot insertion in u, Fig b: knot insertion in v

Esempio 2.5. Per comprendere quanto appena affermato si consideri la superficie NURBS in figura 2.4. Se si vuole raffinare la superficie inserendo un nodo in u si ottiene una griglia di controllo con una colonna di nodi aggiuntivi.

In figura 1.5a è rappresentata la stessa superficie della figura 2.4 raffinata aggiungendo il nodo di coordinate parametriche 0.3 in u. Esattamente come ci si aspettava la griglia di controllo è diventata 5×6.
Successivamente è stato inserito in v il nodo 0.6. La superficie resta invariata, ma alla griglia di controllo viene aggiunta una nuova riga, come si vede in figura 1.5b.
Si può osservare come la griglia di controllo raffinata approssimi meglio la forma della superficie rispetto alla griglia originale.

Superfici NUBS e NURBS in forma parametrica

Definizione 2.15. Una superficie $s(u, v) \in \mathbb{R}^3$ si scrive come:

$$s(u,v) = \begin{pmatrix} s_1(u,v) \\ s_2(u,v) \\ s_3(u,v) \end{pmatrix}$$

dove $u, v \in [0, 1]$ sono due parametri.
Se le $s_i(u, v)$ sono funzioni NUBS scalari appartenenti ad uno stesso spazio prodotto tensoriale, allora si dice che $s(u, v)$ è una superficie NUBS in forma parametrica (o funzione vettoriale NUBS).
Una superficie NUBS in forma parametrica si scrive come:

$$s(u,v) = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} p_{i,j} N_{i,n}(u) N_{j,m}(v)$$

dove i $p_{ij} = (x_{ij}, y_{ij}, z_{ij})$ sono punti di $\mathbb{R}^3$ che formano la griglia di controllo della superficie, detta anche poliedro di controllo.

Definizione 2.16. Una superficie NURBS in forma parametrica, o funzione NURBS vettoriale è una superficie $s(u, v) \in \mathbb{R}^3$ in cui le componenti $s_i(u, v)$ sono funzioni NURBS scalari appartenenti allo stesso spazio. L'espressione di tale superficie è:

$$s(u,v) = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} p_{i,j} R_{i,n,j,m}(u,v)$$

dove $p_{ij} = (x_{ij}, y_{ij}, z_{ij}) \in \mathbb{R}^3$.

Osservazione 10. Esattamente come accade nel passaggio da funzioni NUBS/NURBS a curve NUBS/NURBS in forma parametrica, le osservazioni fatte per le funzioni bivariate sono valide per le superfici in forma parametrica.
La griglia di controllo di una superficie NUBS/NURBS vettoriale si ottiene estendendo in due dimensioni il procedimento utilizzato per costruire la poligonale di controllo delle curve vettoriali. In particolare se si ha una superficie in $\mathbb{R}^3$ la griglia di controllo forma una mesh regolare nello spazio che approssima in forma la superficie.
L'algoritmo di knot insertion per superfici NUBS/NURBS in forma parametrica è identico a quello visto per le funzioni NUBS/NURBS bivariate. Come si è visto nel caso univariato, l'algoritmo visto per superfici scalari deve essere ripetuto per ognuna delle componenti della superficie vettoriale.

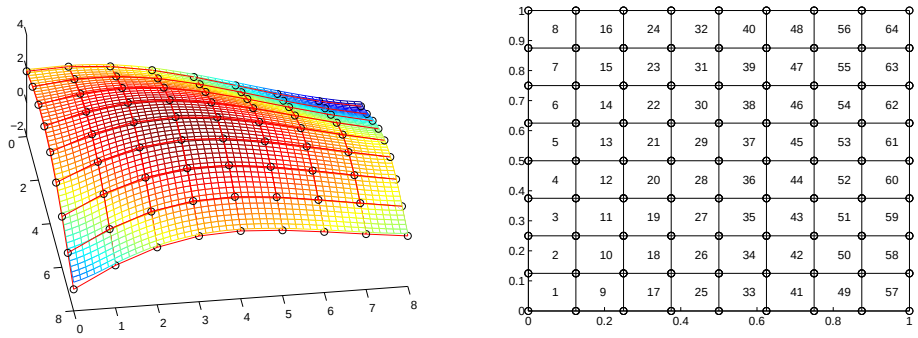

Figura 2.6: Fig a: Superficie NURBS vettoriale, Fig b: Preimmagine della griglia di controllo

Esempio 2.6. In figura 2.6a è mostrata una semplice superficie NURBS cubica vettoriale mentre in 2.6b è rappresentata la preimmagine della sua griglia di controllo sul piano parametrico $[0,1] \times [0,1]$. Come si può vedere semplicemente esiste una biezione tra i punti di controllo della superficie e i nodi della preimmagine della superficie.
Osservando l'immagine 2.6a si può notare come la griglia approssimi in forma la superficie, esattamente come accade per le superfici scalari e per le curve.

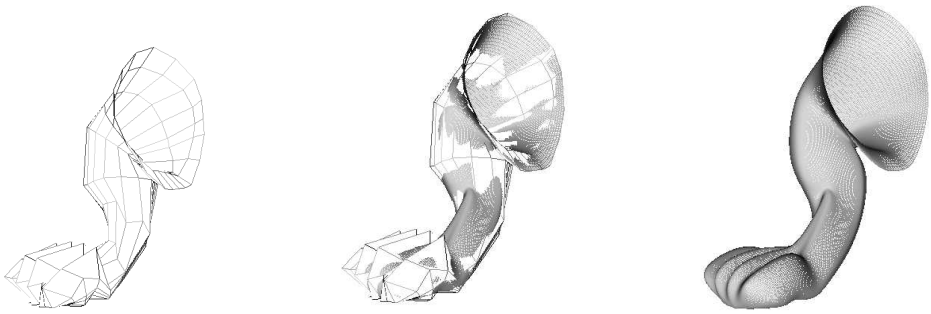

Figura 2.7: Superficie NURBS vettoriale,
Fig a: Griglia, Fig b: Griglia e Superficie, Fig c: Superficie

Esempio 2.7. La figura 2.7 mostra una particolare superficie NURBS cubica che raffigura una zampa di coniglio. L'immagine 2.7a rappresenta la griglia di controllo della superficie, l'immagine 2.7c mostra la superficie che si ottiene mentre l'immagine 2.7b in mezzo raffigura sia la griglia che la superficie e permette di vedere come la superficie approssima in forma la griglia di controllo, senza necessariamente passare per tutti i punti di controllo.

2.3 Superfici T-Spline

Le funzioni T-Spline bivariate sono superfici che si possono definire brevemente come:

"Superfici B-Spline Razionali Non Uniformi associate ad una mesh regolare che ammette giunzioni a T"

La sostanziale differenza tra una T-Spline e una NURBS risiede quindi nella griglia di controllo, in cui i punti di controllo possono avere valenza minore di 4 (e necessariamente maggiore di 1).
Le funzioni base utilizzate per rappresentare una superficie T-Spline sono

strettamente legate alle funzioni base B-Spline Razionali Cubiche. Per questo motivo d'ora in poi tutte le funzioni base B-Spline di cui si parlerà, anche se non specificato esplicitamente, saranno di grado 3, con un vettore nodale composto da 5 elementi.
Per poter dare una definizione più precisa di una superficie T-Spline è necessario introdurre i concetti di *intervallo nodale* e *T-Mesh*.

Knot Interval (Intervallo Nodale)

Definizione 2.17. Un intervallo nodale è un numero non negativo assegnato ad ogni lato della griglia di controllo di una T-Spline.

La nozione di intervallo nodale fu introdotta da T.W.Sederberg nel 1998 per poter ricavare maggiori informazioni da una generica griglia di controllo. Più precisamente gli intervalli nodali consentono di creare in ogni situazione una parametrizzazione locale della superficie, che consente poi di valutare la superficie stessa.
Se la superficie considerata ha genere ≤ 1 gli intervalli nodali permettono di creare una parametrizzazione globale, mentre invece se la superficie ha genere > 1 è possibile solamente creare una partizione locale di una regione della superficie di genere ≤ 1.

Esempio 2.8. Si consideri la regione di superficie NURBS rappresentata in Figura 2.8. Come si vede dall'immagine ad ogni lato della griglia è stato assegnato un intervallo nodale d_i $i = 0, \ldots, 4$ se il lato è orizzontale, e_j $j = 0, \ldots, 4$ se il lato è verticale.
Una parametrizzazione locale della superficie si crea scegliendo un generico punto di controllo P_{i^*,j^*} a cui si assegnano le coordinate parametriche $(0, 0)$. Le coordinate di tutti gli altri punti P_{ij} della mesh sono poi determinate partendo da P_{i^*,j^*} e sommando a $(0, 0)$ gli intervalli nodali d_i e e_j necessari per arrivare da P_{i^*,j^*} a P_{ij}.
Una volta creata la parametrizzazione è possibile calcolare i vettori nodali associati ad ogni punto di controllo e valutare localmente la superficie. Ad

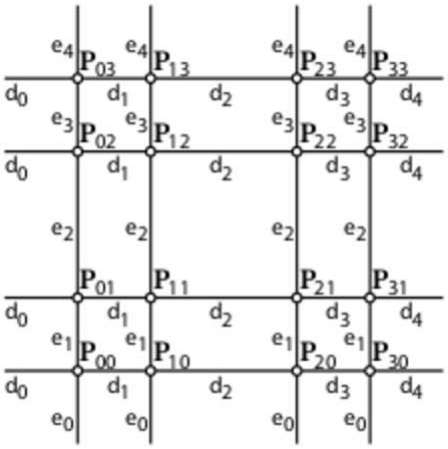

Figura 2.8: Griglia di una superficie NURBS

esempio nell'immagine, una volta assegnati i vettori nodali di tutti i punti di controllo, si può valutare correttamente la faccia centrale.

T-Mesh

La T-Mesh è la griglia di controllo di una superficie T-Spline e si distingue dalle griglie di controllo delle superfici NURBS (o NUBS) per la presenza di giunzioni a T. Nel caso in cui una T-Mesh sia rettangolare e non presenti nodi a T la superficie T-Spline ad essa associata si riconduce banalmente ad una NURBS.
La T-Mesh risulta fondamentale nella determinazione dei vettori nodali associati ad ogni funzione base della T-Spline. Infatti, contrariamente a quanto accade per le superfici B-Spline che hanno una griglia di controllo rettangolare, nelle T-Spline determinare il vettore nodale di un generico punto di controllo non è sempre così immediato.

Esempio 2.9. Si consideri ad esempio la Figura 2.9 che rappresenta la preimmagine di una T-Mesh nello spazio parametrico (s, t). Si nota subito che il punto di controllo P_2 è una giunzione a T e la mancanza di un lato (con corrispondente intervallo nodale) nella T-Mesh non consente di determinare immediatamente il vettore nodale in t del punto stesso.

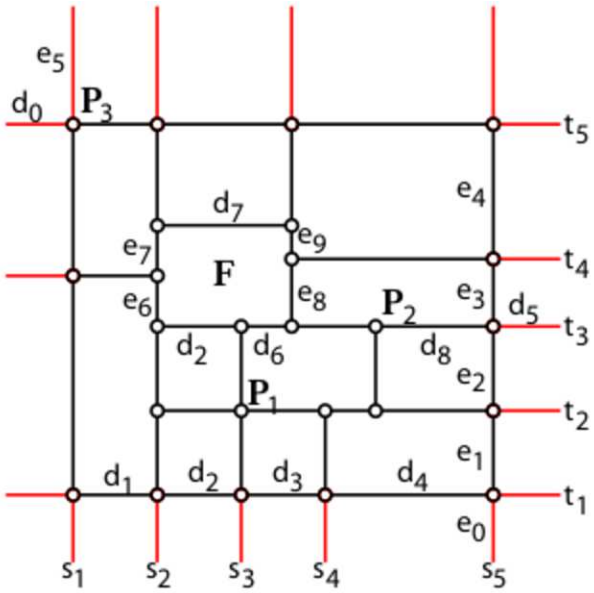

Figura 2.9: Preimmagine di una T-Mesh

La preimmagine di una T-Mesh è composta solamente da punti di controllo, *s-edge* e *t-edge*. Gli *s*-edge sono lati che hanno coordinata s costante, mentre i *t*-edge sono lati con coordinata t costante.
Una giunzione a T (o T-junction) nella preimmagine di una T-Mesh è un vertice di valenza 3 da cui partono o due *s*-edge e un *t*-edge o due *t*-edge e un *s*-edge.
Nella preimmagine di una T-Mesh è possibile associare ad ogni lato un corrispondente intervallo nodale seguendo due regole fondamentali:

1. La somma degli intervalli nodali dei lati opposti di ogni faccia deve essere uguale.

2. Se una giunzione a T su un lato di una faccia può essere connessa con una giunzione a T su un lato opposto della faccia, senza violare la regola 1 allora si deve aggiungere alla T-Mesh il lato che unisce le due giunzioni a T.

Successivamente è possibile determinare per ogni punto di controllo sulla preimmagine della T-Mesh una coordinata nodale (s_i, t_j) come si è visto nell'esempio 2.8.
I vettori nodali $s_i = (s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4})$ e $t_i = (t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4})$ associati ad un punto di controllo P_i di coordinate parametriche (s_{i2}, t_{i2}) si determinano nel modo seguente:

. Gli elementi centrali di s_i e di t_i sono le coordinate parametriche di P_i in s e t rispettivamente.

. I due elementi successivi a s_{i2} in s_i sono le coordinate dei primi due s-edges o punti di controllo intersecati dal raggio $R(\alpha) = (s_{i2} + \alpha, t_{i2})$ nello spazio parametrico (con $\alpha > 0$). Analogamente si determinano i due elementi successivi a t_{i2} in t_i.

. I due elementi mancanti nei vettori s_i e t_i si calcolano con un procedimento identico al caso precedente, con l'unica differenza che in questo caso $\alpha < 0$.

Superfici T-Spline scalari

A questo punto è possibile fornire una definizione più dettagliata di superficie T-Spline.

Definizione 2.18. Si definisce T-Spline la superficie di equazione

$$P(s,t) = \frac{\sum_{i=1}^{n} P_i w_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}$$

dove P_i sono i punti di controllo, w_i sono pesi e $B_i(s,t)$ le funzioni base, che si ottengono come

$$B_i(s,t) = N_{i,4}(s) N_{i,4}(t).$$

Come si è detto in precedenza le funzioni base monovariate $N_{i,4}(s)$ e $N_{i,4}(t)$ sono B-Spline Cubiche Normalizzate associate ai vettori nodali

$$s_i = (s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}) \quad \text{e} \quad t_i = (t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4})$$

Osservazione 11. Dalla definizione 2.18 si nota subito la somiglianza tra l'equazione di una T-Spline e quella di una superficie NURBS cubica (2.14). Questo conferma quanto affermato in precedenza: se una T-Mesh è rettangolare e non presenta giunzioni a T, allora i vettori nodali associati alle funzioni base $N_{i,4}(s)$ e $N_{i,4}(t)$ coincidono coi vettori nodali delle funzioni base di una superficie NURBS e quindi la superficie T-Spline si riconduce ad una NURBS.

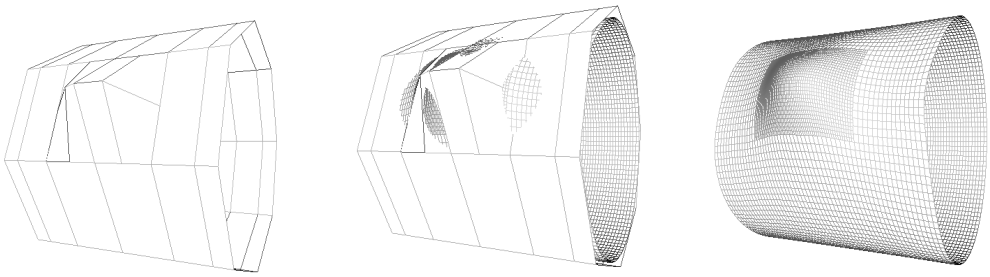

Figura 2.10: Superficie T-Spline vettoriale,
Fig a: Griglia, Fig b: Griglia e Superficie, Fig c: Superficie

Superfici T-Spline in forma parametrica

Una superficie T-Spline in forma parametrica, detta anche funzione T-Spline vettoriale si esprime come:

$$P(s,t) = \frac{\sum_{i=1}^{n} p_i w_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}$$

dove $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$.
Esattamente come accade per le superfici NURBS in forma parametrica, ogni compontente di p_i definisce una superficie T-Spline scalare. I punti p_i definiscono poi la griglia di controllo in $\mathbb{R}^3$ della superficie.
Un esempio di superficie T-Spline vettoriale è rappresentato in figura 2.10.

Genere di una Superficie T-Spline

Definizione 2.19. Una mesh di controllo associata a una superficie NURBS chiusa si dice regolare se è costituita da facce di 4 lati e vertici a valenza 4. Analogamente una T-Mesh di una superficie T-Spline si dice regolare se è formata solamente da vertici di valenza 2,3 o 4.

Definizione 2.20. Il genere di una superficie chiusa si definisce come il numero più grande di curve semplici chiuse disgiunte che possono essere disegnate sulla superficie senza separarla in due componenti connesse distinte. Alternativamente il genere di una superficie chiusa si può definire come il numero di manici della superficie stessa.

Il genere di una superficie chiusa è strettamente legato alla *Caratteristica di Eulero* χ. In particolare per superfici chiuse orientabili vale che:

$$\chi = 2 - 2g$$

La caratteristica di Eulero è un'invariante topologica che può essere calcolata facilmente creando una tassellazione della superficie. Vale

$$\chi = V + F - E$$

dove V,E,F sono rispettivamente i vertici, i lati e le facce della tassellazione della superficie. Si ha quindi la formula di Eulero

$$2 - 2g = V + F - E$$

Osservazione 12. Si osserva che nel caso di superfici definite da una mesh di controllo la caratteristica di Eulero di una tassellazione della superficie è uguale alla caratteristica di Eulero della mesh associata alla superficie stessa.

Teorema 2.3.1. *Ogni NURBS chiusa con mesh di controllo regolare ha genere 0 o 1. In particolare ogni NURBS con queste proprietà ottenuta collassando vertici ha genere 0.*

Dimostrazione. Sia M una generica griglia di controllo di una NURBS chiusa costituita da n facce di 4 lati e 4 vertici. Ogni vertice ha valenza 4, mentre ogni lato appartiene a 2 facce. La somma dei vertici è n, mentre la somma dei lati è $2n$, quindi

$$2 - 2g = 0 \ \rightarrow \ g = 1$$

Sia M una generica mesh di controllo di una NURBS chiusa con k vertici collassati e composta da $n * m$ facce. Necessariamente $k = 2$ o $k = 1$, altrimenti M non potrebbe essere chiusa.
Se M ha 2 vertici collassati si ha che:

$$V = m * (n - 1) + 2 \ \text{ e } \ E = m * n + m * (n - 1)$$

Di conseguenza

$$2 - 2g = m * n + m * n - m + 2 - m * n - m * n + m = 2 \ \rightarrow \ g = 0$$

Se M ha 1 vertice collassato si ha

$$V = (m - 1) * (n - 1) + 1 \ \text{ e } \ E = m * (n - 1) + n * (m - 1)$$

Quindi

$$2 - 2g = m * n + m * n - m - n + 2 - m * n + m - m * n + n = 2 \ \rightarrow \ g = 0$$

□

Esempio 2.10. Si consideri la figura 2.14a in cui è rappresentata la preimmagine di una mesh di controllo associata a una superficie NURBS. Si chiude la superficie unendo i lati in rosso e si collassano i punti in verde e i punti in blu, ottenendo 2 soli punti di controllo. La mesh ottenuta ha 16 facce, 14 vertici (12 colorati in rosso, 1 che rappresenta i vertici in verde e 1 che rappresenta quelli in blu) e 28 lati. Si vede quindi che la superficie ha genere 0 e il teorema è rispettato.

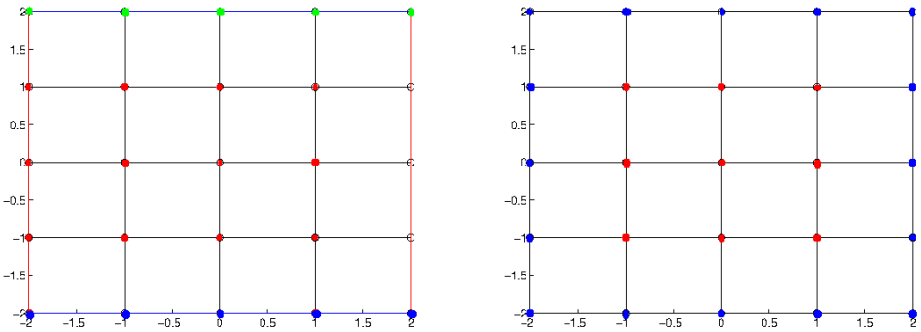

Figura 2.11: fig a: Mesh chiusa con 2 punti collassati. fig b: Mesh chiusa con 1 punto collassato

La figura 2.14b mostra la preimmagine della mesh di controllo di una NURBS. Si collassano tutti i vertici colorati in blu in un unico punto, chiudendo così la superficie. La mesh che si ottiene ha 16 facce, 10 vertici (9 colorati in rosso e 1 che rappresenta i vertici in blu) e 22 lati. Anche in questo la superficie ha genere 0 e il teorema è rispettato.

Proposizione 2.3.2. *Le superfici T-Spline a cui è associata una T-Mesh regolare possono essere solamente di genere 0 o 1.*

Dimostrazione. Vale il seguente fatto:

> In letteratura sono noti algoritmi di raffinamento che consentono di ottenere una mesh regolare da una generica T-Mesh senza modificare la superficie.

Questo fatto e il teorema 2.3.1 consentono di affermare che provare la proposizione è equivalente a dimostrare che l'inserimento di nuovi nodi nella T-Mesh mantiene il genere della superficie.
In una T-Mesh è possibile inserire nodi solamente quando si hanno punti di controllo a valenza minore di 4, ovvero quando si hanno T-junction, nodi a L o nodi a I.

Si consideri la formula di Eulero

$$2 - 2g = V + F - E$$

Si deve provare che ogni inserimento lascia invariato $V + F - E$.

. Sia p_j una T-junction e f_i la faccia per cui p_j è una T. Per regolarizzare p_j è necessario aggiungere un vertice p^* e un lato l^* che unisce p_j e p^*. il lato l_k opposto a p_j e la faccia f_i sono divisi a metà. Si aggiungono un vertice, una faccia e 2 lati, di conseguenza $V+F-E$ resta invariato.

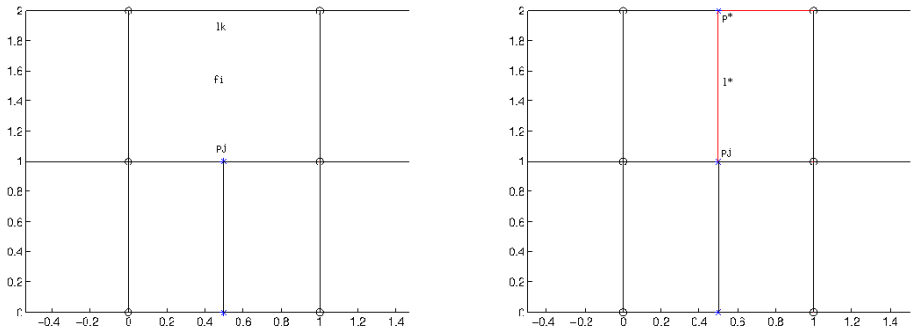

Figura 2.12: fig a: T-Junction. fig b: Mesh raffinata

. Sia p_j una nodo a L e f_i la faccia per cui p_j è una L. Per regolarizzare p_j è necessario aggiungere due vertici p^* e $\bar{p}$, e i corrispondenti lati l^* e $\bar{l}$ che uniscono p_j a p^* e $\bar{p}$ rispettivamente. I lati l_k e l_m opposti a p_j sono divisi a metà. La faccia f_i è divisa in 3 facce. Si aggiungono due vertici, due facce e 4 lati, di conseguenza $V + F - E$ resta invariato.

. Sia p_j una nodo a I e f_i e f_k le facce per cui p_j è una I. Per regolarizzare p_j è necessario aggiungere due vertici p^* e $\bar{p}$, e i corrispondenti lati l^* e $\bar{l}$ che uniscono p_j a p^* e $\bar{p}$ rispettivamente. I lati l_m e l_n opposti a p_j sono divisi a metà così come le facce f_i e f_k. Si aggiungono due vertici, due facce e 4 lati, di conseguenza $V + F - E$ resta invariato.

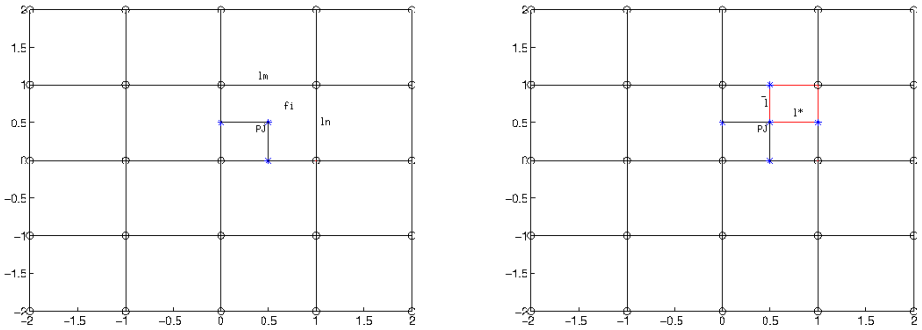

Figura 2.13: fig a: Nodo a L. fig b: Mesh raffinata

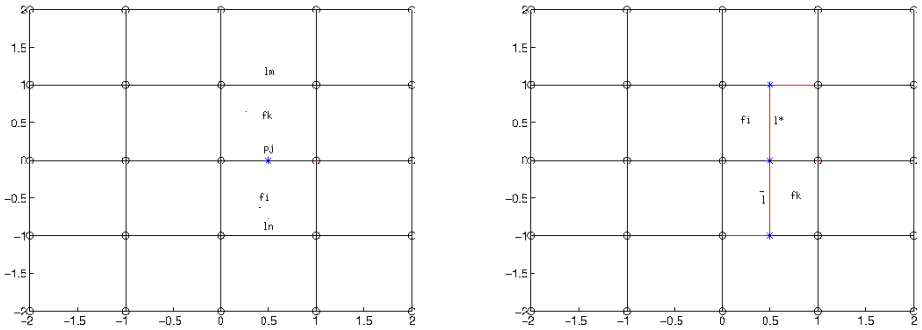

Figura 2.14: fig a: Nodo a I. fig b: Mesh raffinata

Ogni raffinamento fatto non varia il genere della T-Mesh, quindi la proposizione è provata. □

Osservazione 13. La proposizione consente di affermare che se si vuole rappresentare con le T-Spline una superficie di genere maggiore di 1 si deve avere una T-Mesh con punti di controllo straordinari, che hanno valenza maggiore di 4, esattamente come accade per le T-NURCC's [9].
In questo lavoro si considereranno solamente superfici di genere ≤ 1 ma si cercherà, quando possibile, di descrivere e realizzare algoritmi validi anche per superfici di genere maggiore di 1.

Capitolo 3

Algoritmi di valutazione e raffinamento di superfici T-Spline

Le superfici T-Spline hanno una struttura molto meno rigida rispetto alle superfici NURBS, fatto che comporta da un lato notevoli vantaggi mentre dall'altro richiede la creazione di metodi numerici per l'elaborazione della superficie più complessi di quelli noti per NURBS [8].
Il primo grande vantaggio che le superfici T-Spline hanno rispetto alle NURBS risiede nella minore pesantezza della griglia di controllo. Nella T-Mesh, contrariamente a quanto accade nella griglia di una NURBS, ogni punto di controllo è strettamente necessario. Questo fatto consente solitamente di definire una generica superficie T-Spline utilizzando meno della metà dei punti di controllo necessari per definire una analoga superficie NURBS.
Il secondo grande vantaggio delle T-Spline consiste nella possibilità di raffinare localmente la superficie, avendo così una maggiore libertà nella costruzione e nella modellazione della superficie stessa. Al contrario le superfici NURBS richiedono l'inserimento di punti di controllo superflui che appesantiscono solamente la struttura della superficie.
In questo capitolo si introducono alcuni algoritmi ideati per superfici T-

Spline. In particolare si vedrà un algoritmo di valutazione, due algoritmi di knot insertion e i due corrispondenti algoritmi di raffinamento locale della superficie. Gli algoritmi proposti verranno confrontati con i corrispondenti algoritmi per superfici NURBS, osservando le similitudini e le differenze presenti.

3.1 Algoritmo di Valutazione

Le T-Spline e le NURBS sono entrambe definite a partire da una griglia di controllo. La valutazione su un punto (s,t) consiste nel determinare il valore che la superficie assume in quel punto, ovvero trovare

$$P(s,t) = \frac{\sum_{i=1}^{n} P_i w_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)} \tag{3.1}$$

La valutazione di tutta la superficie consiste poi nella valutazione su ognuno dei punti (s,t) appartenenti ad una discretizzazione del dominio parametrico. L'obiettivo di chi progetta un algoritmo di valutazione è quello di riuscire ad effettuare questi calcoli utilizzando il minore numero possibile di operazioni. Per capire meglio quanto detto, si consideri un primo semplicissimo esempio di valutazione di una superficie NURBS.

Esempio 3.1. In figura 3.1 si ha la preimmagine, o dominio parametrico, di una griglia di controllo di una superficie Spline cubica. Si vuole valutare la superficie nel punto p di coordinate parametriche $(s,t) = (4.5, 4.5)$ centro della faccia 37.
Trattandosi di una superficie NURBS cubica, le funzioni base in s e in t associate ad un punto di controllo della griglia sono definite da un vettore nodale composto di 5 elementi. Essendo la griglia regolare il vettore nodale in s di un punto di controllo si determina andando a guardare sulla preimmagine della griglia di controllo le coordinate parametriche dei due punti di controllo

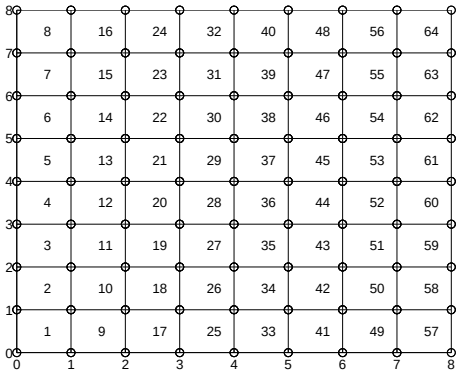

Figura 3.1: Superficie NURBS

a sinistra e dei due a destra del punto considerato. In t il ragionamento è lo stesso, con l'unica differenza che si scelgono i due punti sotto e i due punti sopra quello considerato. Ad esempio in figura 3.1 il punto di controllo che ha coordinate parametriche $(4, 5)$ avrà come vettori nodali $s = [2, 3, 4, 5, 6]$ e $t = [3, 4, 5, 6, 7]$.
Si vede subito quindi che nel caso di B-Spline cubiche per la valutazione del punto p non è necessario calcolare il valore di tutte le funzioni base associate ai punti di controllo. Nella sommatoria infatti gli unici termini non nulli sono quelli relativi ai punti di controllo che hanno una funzione base con dominio di influenza che contiene p. In particolare nelle NURBS gli unici punti di controllo che hanno dominio di influenza che contiene p sono i vertici delle facce f_i che hanno almeno un lato in comune con F. Nell'esempio $F = 37$ e le facce f_i sono $28, 29, 30, 36, 38, 44, 45, 46$.

Dall'esempio si capisce che la valutazione di un punto di una superficie dipende da un numero limitato di punti di controllo. Questo fatto vale anche per le superfici T-Spline, con la sostanziale differenza che spesso non è così semplice determinare i punti coinvolti nella valutazione su un punto del dominio parametrico.
Si introducono ora due semplici algoritmi di valutazione per superfici NURBS

e T-Spline, che riprendono quanto visto nell'esempio proposto.

- Si valuta in ogni punto di coordinate (s,t) applicando semplicemente l'equazione (3.1).
- Per ogni punto di coordinate (s,t) si calcola il valore che assume l'equazione:

$$P(s,t) = \frac{\sum_{i=j_1}^{j_m} P_i w_i B_i(s,t)}{\sum_{i=j_1}^{j_m} w_i B_i(s,t)} \tag{3.2}$$

  dove gli indici $j_1, \ldots, j_m$ sono associati ai punti di controllo che hanno dominio di influenza contenente (s,t).

Il primo algoritmo è molto costoso e inefficiente sia per le NURBS che per le T-Spline in quanto effettua quasi sempre calcoli inutili. Come si è visto nell'esempio precedente non è necessario calcolare i valori delle funzioni base che hanno un dominio di influenza che non contiene il punto che si vuole valutare.
Il secondo algoritmo risulta più efficiente rispetto al precedente perché compie un numero minore di calcoli, ma sono comunque presenti eccessivi controlli. Infatti per ogni punto (s,t) della griglia di valutazione si controllano i domini di influenza di tutti i punti di controllo della T-Mesh.
Per le superfici NURBS il secondo algoritmo può essere ulteriormente migliorato eliminando i controlli e calcolando solamente i valori delle funzioni base associate ai punti di controllo appartenenti alle facce adiacenti la faccia che contiene il punto (s,t).
Per le superfici T-Spline l'algoritmo effettua comunque un numero eccessivo di controlli. La perdita di regolarità della griglia di controllo che si ha passando da una NURBS ad una T-Spline comporta l'impossibilità di determinare a priori un intorno locale di punti controllo che sono coinvolti nella valutazione di un generico punto della superficie, di conseguenza è possibile solamente studiare un modo più efficiente di effettuare i controlli necessari,

ma non è possibile evitarli del tutto.
Un primo modo per ridurre notevolmente i controlli eccessivi effettuati dal precedente algoritmo è valutare separatamente le singole facce della superficie. Si consideri il seguente algoritmo:

- Per ogni faccia della superficie si determina l'insieme J dei punti di controllo coinvolti nella valutazione della faccia. Successivamente per ogni (s,t) interno alla faccia si applica l'equazione (3.2) dove i j_i sono gli elementi di J.

Per le superfici NURBS l'algoritmo proposto risulta sicuramente ottimale, infatti la struttura regolare della griglia di controllo consente di affermare che le valutazioni di tutti gli (s,t) di una faccia F sono ottenute sommando valori relativi agli stessi punti di controllo, di conseguenza non si effettuano calcoli inutili. Inoltre se si vuole avere una griglia di valutazione globale regolare, che permetta di studiare le varie proprità della superficie NURBS, è sufficiente impostare una stessa discretizzazione per ogni faccia.
Per le superfici T-Spline invece l'algoritmo proposto non è ottimale perché in alcuni casi si svolgono ancora delle operazioni inutili. Infatti si osserva che, contrariamente a quanto accade per le NURBS, non è necessariamente vero che tutti gli (s,t) di una stessa faccia F si scrivono come combinazione lineare delle funzioni base associate agli stessi punti di controllo. Esistono casi in cui due punti (s_1,t_1) e (s_2,t_2) appartenenti alla stessa faccia F si scrivono come combinazione di due insiemi distinti di punti di controllo.

Esempio 3.2. Nella figura 3.1a i punti $P1$ e $P2$ appartengono entrambi alla faccia 12. Se si osserva bene però il punto $P1$ appartiene al dominio di influenza del punto di controllo 7, che è colorato in rosso nella regione verde (figura 3.1c), ma non al dominio del punto di controllo 29, colorato in rosso nella regione azzurra (figura 3.1b). Viceversa $P2$ non appartiene al dominio di influenza di 7, ma è contenuto in quello di 29.

L'ultimo algoritmo proposto è sicuramente efficiente, e permette di valutare correttamente una superficie T-Spline. Non si può però affermare con

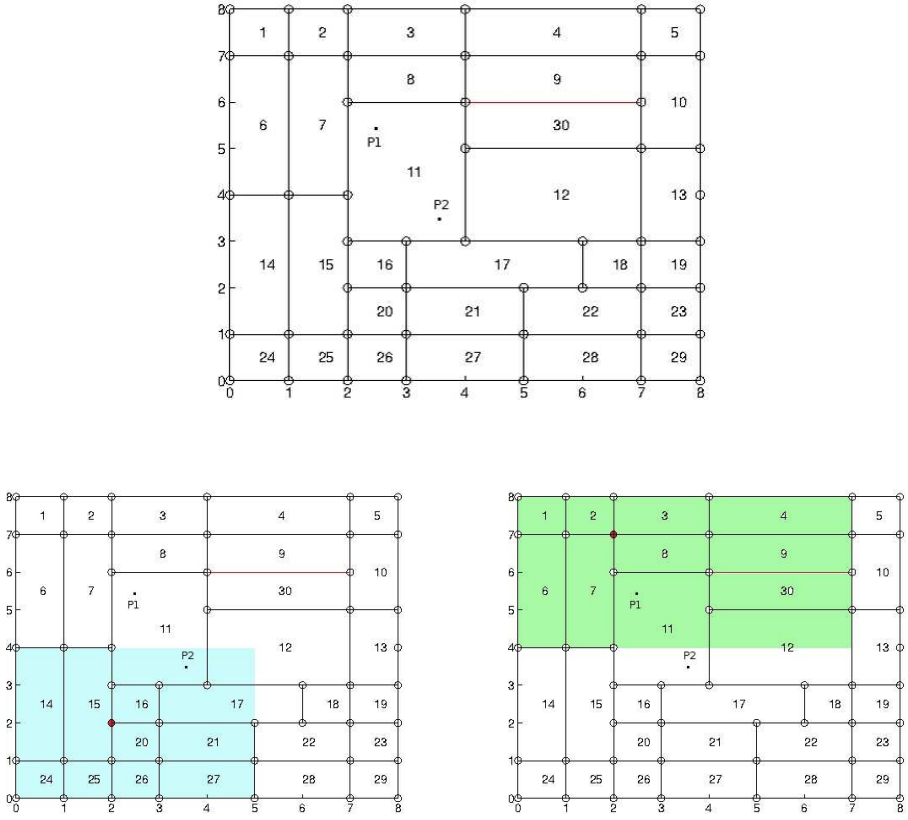

Figura 3.2: Superficie T-Spline

certezza che l'algoritmo sia il migliore possibile per valutare una T-Spline perchè si è visto che in alcuni casi vengono comunque effettuati dei calcoli inutili. Inoltre tale algoritmo non consente di ottenere in ogni caso una valutazione globale coerente, ovvero una valutazione in cui è possibile raccordare tutti i punti delle griglie di valutazione adiacenti.

Determinazione dei nodi coinvolti nella valutazione

La parte più complessa e costosa degli ultimi algoritmi proposti è sicuramente la determinazione dell'insieme J dei punti di controllo necessari per valutare un generico punto della discretizzazione del dominio parametrico.
Non è possibile determinare l'insieme J associato ad una faccia F parten-

do da F stessa e considerando i nodi delle facce adiacenti, poichè esistono casi come ad esempio quello nella figura 3.3 in cui il punto di controllo $P1$ di coordinate $(-1, -1)$ è coinvolto nella valutazione di tutti i punti di una qualsiasi discretizzazione del dominio parametrico, indipendentemente dalla distanza dello stesso $P1$ dalla faccia considerata.

Un algoritmo corretto, ma poco efficiente per determinare tutti i punti

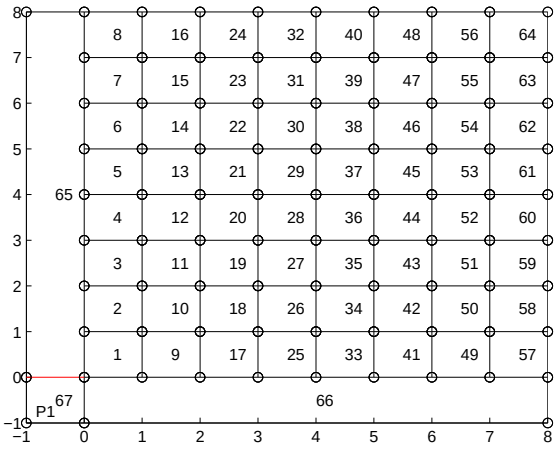

Figura 3.3: Superficie T-Spline

coinvolti nella valutazione di una faccia è il seguente:

. Si costruisce la parametrizzazione di tutta la superficie partendo dalla faccia che si sta valutando e si determina per ogni punto di controllo se il suo dominio di influenza interseca l'interno della faccia.

Lo svantaggio di questo algoritmo consiste nel fatto che richiede in ogni situazione una parametrizzazione globale, di conseguenza si perde la possibilità di valutare localmente la superficie. Inoltre se si volesse valutare una superficie di genere maggiore di 1 l'algoritmo può determinare solamente la preimmagine di una regione della T-Mesh.
Un algoritmo che evita di parametrizzare globalmente la superficie è il seguente:

. Si crea un vettore v_i per ogni punto di controllo. Si considerano tutti i punti di controllo p_j assegnandogli ogni volta coordinate parametriche $(s,t) = (0,0)$ e determinando il loro dominio di influenza.
Successivamente si costruisce una parametrizzazione locale contenente i punti del primo ring di adiacenza. Si determinano i punti p_i che hanno entrambe le coordinate parametriche strettamente contenute nel dominio di influenza di p_j e per ognuno di questi punti si aggiorna il vettore v_i corrispondente inserendo l'indice j. Se esiste almeno un punto contenuto strettamente nel dominio di influenza di p_j si ripete il procedimento allargando la parametrizzazione ai punti del secondo ring. Il procedimento viene ripetuto fino a quando non si arriva ad un passo in cui nessuno dei nuovi punti selezionati è contenuto nell'interno del dominio di influenza di p_j.

Questo algoritmo evita la parametrizzazione globale della superficie nei casi in cui non si ha una situazione analoga a quella proposta nella figura 3.3.
Solitamente le superfici T-Spline che si utilizzano sono superfici NURBS raffinate localmente. In questi casi quindi si può utilizzare il seguente algoritmo:

. Si determinano i 2 o 3 ring di facce attorno alla faccia che si vuole valutare e si considerano come punti interessati dalla valutazione tutti i punti appartenenti a tali facce.

Tale algoritmo non si può definire corretto in quanto esistono casi in cui il suo funzionamento è errato, ma si può utilizzare in un numero elevato di situazioni.
Nel progetto creato per elaborare e visualizzare superifici T-Spline è implementato l'ultimo algoritmo proposto e i nodi dell'insieme J sono determinati prendendo 2 ring di facce attorno alla faccia considerata.

3.2 Algoritmo di Knot Insertion 1

Il principale algoritmo di editing della superficie è l'algoritmo di knot insertion, che viene utilizzato con estrema frequenza dai progettisti per raf-

finare una superficie aggiungendo nuovi nodi e di conseguenza anche nuovi punti di controllo. Esistono in letteratura due tipologie di algoritmi di Knot Insertion:

- Algoritmi che non modificano la superficie, ovvero se si valuta la superficie prima e dopo l'inserimento di un nodo si ottiene lo stesso risultato.
- Algoritmi che approssimano la superficie di partenza. In questo caso l'inserimento di uno o più nodi crea una superficie simile rispetto a quella di partenza.

In questo lavoro di tesi si tratteranno solamente gli algoritmi che lasciano invariata la superficie, che sono più complessi e consentono di avere notevoli vantaggi rispetto agli algoritmi che semplicemente restituiscono una superficie approssimante quella iniziale.
Come si è visto nel capitolo precedente, le superfici NURBS non possono essere raffinate localmente. L'algoritmo di knot insertion per superfici NURBS non modifica la superficie ma appesantisce notevolmente la griglia di controllo inserendo molti punti di controllo geometricamente inutili.
Le superfici T-Spline hanno invece una T-Mesh con minori vincoli di regolarità che permette di raffinare localmente la superficie, inserendo un numero limitato di punti di controllo.
Il primo algoritmo di knot insertion per superfici T-Spline, detto anche Control Point Insertion, fu proposto da T.W.Sederberg nell'articolo introduttivo alle T-Spline [1] e rappresenta un superamento dell'algoritmo di knot insertion visto per le superfici NURBS. Ogni nodo e il suo corrispondente punto di controllo vengono aggiunti seguendo le regole dell'algoritmo di knot insertion per funzioni NURBS univariate, visto nella sezione 2.1 (paragrafo "Knot insertion per funzioni NURBS").
L'algoritmo si basa su una semplice regola:

- Un generico punto P_3 appartenente ad un t-edge (s-edge) di una superficie può essere inserito se e solo se i vettori nodali t_1, t_2, t_4, t_5 (s_1, s_2, s_4, s_5)

associati ai quattro punti P_1, P_2, P_4, P_5 precedenti e successivi a P_3 nella direzione s (t) sono tra loro uguali.

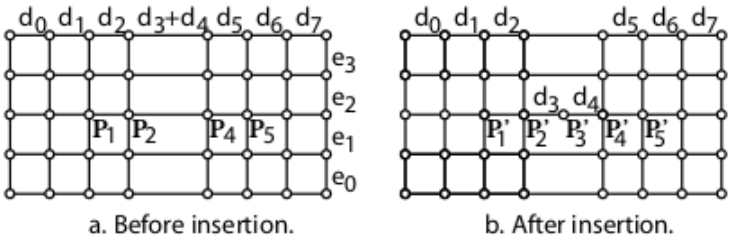

Figura 3.4: Knot Insertion in una T-Mesh

Esempio 3.3. Per comprendere meglio la regola introdotta si consideri la figura 3.4. Il punto P_3 può essere inserito perché i punti P_1, P_2, P_4, P_5 hanno tutti lo stesso vettore nodale in t che è

$$t_1 = t_2 = t_4 = t_5 = (-e_0 - e_1, -e_1, 0, e_2, e_2 + e_3)$$

Nel caso in cui invece la regola non sia rispettata, come ad esempio in figura

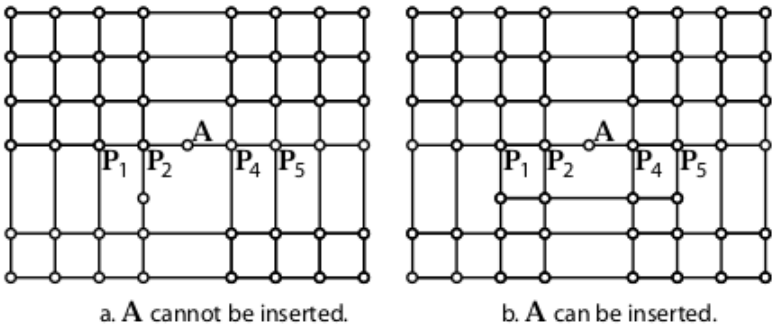

Figura 3.5: Knot Insertion in una T-Mesh

3.5a, l'algoritmo richiede di inserire prima del punto A tutti gli altri punti di controllo necessari per rendere possibile l'inserimento di A.
Nell'esempio proposto in figura si vede che per inserire A sono stati inseriti 3 punti di controllo che hanno reso i vettori nodali t_1, t_2, t_4, t_5 uguali.

Osservazione 14. Si osserva che nell'algoritmo il punto di controllo che si vuole aggiungere viene effettivamente inserito solo alla fine dell'algoritmo. Inoltre, la regola caratteristica dell'algoritmo deve essere applicata a tutti i punti di controllo che si inseriscono, compresi quelli che devono essere aggiunti per rendere verificata la regola per un punto da inserire. Di conseguenza prima di inserire un punto deve essere sempre fatto un controllo per stabilire se può essere inserito oppure no.

Questo algoritmo di Knot Insertion è una evoluzione dell'algoritmo di knot insertion proposto per le superfici NURBS che sfrutta le proprietà delle T-Spline. Per comprendere meglio quanto appena detto si consideri il seguente esempio:

Esempio 3.4. Si consideri la figura 3.4b. Come si è detto in precedenza l'algoritmo permette di inserire P_3 senza dover aggiungere altri punti di controllo.
L'inserimento del punto di controllo avviene esattamente come nell'algoritmo per le superfici NURBS: il valore del punto di controllo P_3 viene calcolato applicando l'algoritmo di knot insertion nel caso univariato. Ovviamente per poter considerare solo il caso univariato è necessario che i vettori nodali nell'altra coordinata, in questo caso t perchè il lato è orizzontale, siano uguali. In sostanza si richiede di poter considerare localmente la superficie come se fosse simile ad una NURBS.

Nota 1. Con simile si intende dire che i punti di controllo interessati nell'inserimento di P_3 possono essere visti come appartenenti ad una superficie NURBS. Non si può richiedere che la superficie localmente sia una NURBS poichè non si richiedono condizioni sui lati orizzontali sopra e sotto il lato in cui si inserisce P_3. Infatti se in 3.4a, nel lato orizzontale subito sotto a quello che ha come estremi P_2 e P_4 ci fosse un ulteriore punto di controllo, la superficie localmente non potrebbe comunque essere una NURBS.

L'algoritmo per le NURBS richiederebbe poi di applicare lo stesso ragionamento a tutte le righe della griglia di controllo, per mantenere la struttura

regolare, ma poichè le T-Spline non hanno tali esigenze di regolarità della griglia di controllo l'algoritmo non inserisce altri punti.

Osservazione 15. L'algoritmo può essere applicato solo per l'inserimento di un nodo su un lato esistente. Qualora si volesse inserire un nodo all'interno di una faccia sarebbe necessario inserire prima due nodi alla stessa altezza su due lati opposti della faccia, aggiungere alla T-Mesh il lato tra i due nodi aggiunti e successivamente inserire il nodo su tale lato.

Osservazione 16. L'algoritmo effettua un raffinamento locale della superficie, ma non è possibile determinare a priori un intorno del punto che si vuole inserire oltre al quale si ha la certezza che non vengano più inseriti punti.

Esempio 3.5. Si consideri ad esempio la superficie in figura 3.6. Si vuole inserire il nodo $(3, 1.5)$ appartenente al lato 17. L'algoritmo di knot insertion fornisce come risultato la superficie in figura 3.7.
Come si può notare la presenza dei nodi $(3, 4.5)$ e $(0.5, 5)$ ha reso necessario l'inserimento di 8 punti di controllo, oltre al punto iniziale. Questi punti non sono poi contenuti in un intorno limitato del punto inserito.
Si potrebbe ipotizzare che in questo caso la superficie considerata sia troppo ristretta per determinare effettivamente un intorno valido per ogni superficie e per ogni punto che si può inserire. Tale ipotesi non è però corretta, perchè è possibile costruire in modo semplice una superficie per cui l'inserimento di un nodo modifica globalmente la superficie.

Osservazione 17. L'algoritmo di Knot Insertion proposto non funziona in tutte le situazioni. Esistono dei casi in cui l'algoritmo non termina in un numero finito di passi, ma entra in un ciclo infinito. Questo accade ad esempio quando uno dei nodi necessari per inserire un nodo A richiede, per essere inserito, l'inserimento del nodo A.

Esempio 3.6. Si prenda in considerazione la figura 3.8 e si supponga di voler inserire il nodo $(4, 4)$. Dalla figura si vede immediatamente che, per esempio, il vettore nodale in t del nodo in $(4, 6)$ richiede l'inserimento del nodo $(3, 6)$,

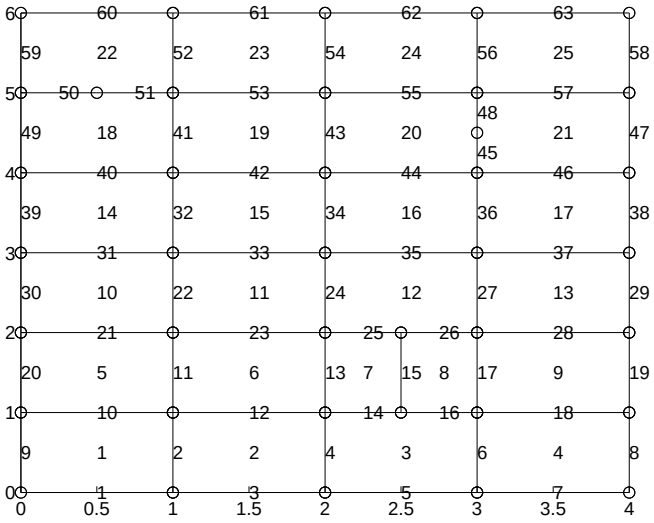

Figura 3.6: T-Mesh iniziale

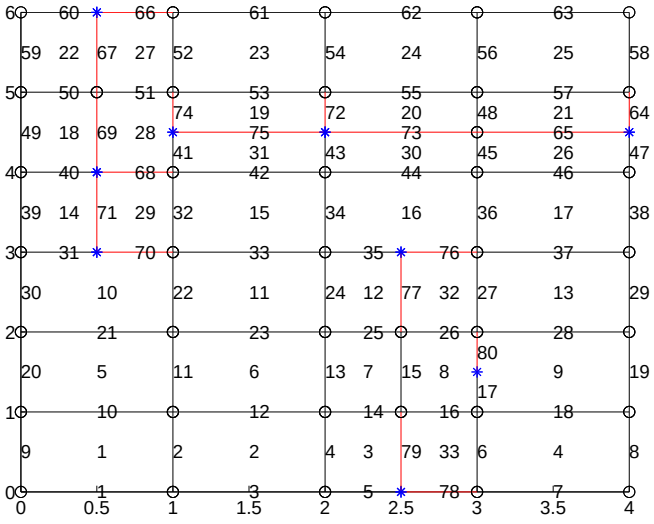

Figura 3.7: Inserimento nodo (3,1.5) con algoritmo 1

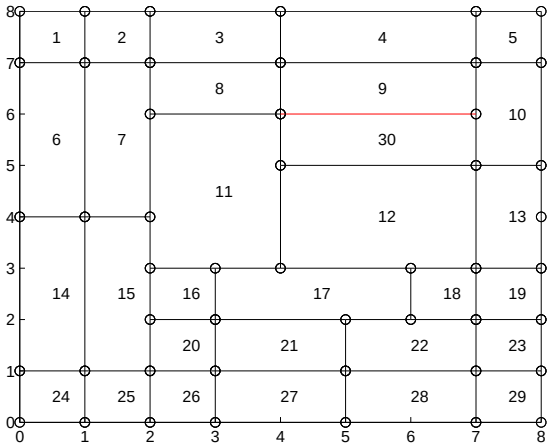

Figura 3.8: T-Mesh

il quale per essere inserito richiede a sua volta l'inserimento di alcuni nodi, tra cui $(4, 4)$.

In conclusione l'algoritmo proposto rappresenta una prima evoluzione poco efficiente dell'algoritmo di knot insertion per le superfici NURBS. L'algoritmo infatti sfrutta le proprità della T-Mesh per raffinare localmente la superficie, ma non garantisce la convergenza in ogni situazione. Esistono infatti dei casi in cui non è possibile inserire un punto di controllo all'interno di una T-Mesh.

3.3 Algoritmo di Knot Insertion 2

L'esigenza di poter sempre inserire un generico punto di controllo all'interno di una qualsiasi T-Mesh ha portato alla creazione di un nuovo algoritmo di knot insertion che converge in ogni situazione, inserendo al massimo un numero finito di punti di controllo oltre a quello richiesto [2].
Questo secondo algoritmo di knot insertion si basa principalmente sul concetto di *Raffinamento di una funzione blending* che viene ora introdotto.

Raffinamento di una Blending Function

Il raffinamento di funzioni blending consiste nello scrivere una funzione base B esistente come combinazione lineare di nuove funzioni base.
Si consideri il caso di funzioni base B-Spline cubiche. Sia $s = [s_0, s_1, s_2, s_3, s_4]$ un vettore nodale e sia $\tilde{s}$ un vettore di m nodi contenente gli elementi di s. La funzione base $N[s_0, s_1, s_2, s_3, s_4](s)$ può essere scritta come combinazione lineare di $m-4$ funzioni base B-Spline che saranno definite dalle $m-4$ stringhe di lunghezza 5 formate da elementi consecutivi di $\tilde{s}$.
Il caso più semplice da analizzare, che sarà anche quello maggiormente utilizzato nell'algoritmo di knot insertion, è quello del raffinamento di una funzione base mediante l'inserimento di un nodo k all'interno del vettore $s = [s_0, s_1, s_2, s_3, s_4]$, ovvero $k > s_0$ e $k < s_4$. Si hanno quindi 4 possibili vettori $\tilde{s}$ composti ciascuno da 6 elementi. A seconda della posizione di k all'interno del vettore $\tilde{s}$ la funzione base associata a s si scrive come combinazione lineare di due nuove funzioni base che hanno vettori nodali contenenti k.

. Se $\tilde{s} = [s_0, k, s_1, s_2, s_3, s_4]$:

$$N[s_0, s_1, s_2, s_3, s_4](s) = c_0 N[s_0, k, s_1, s_2, s_3](s) + d_0 N[k, s_1, s_2, s_3, s_4](s)$$

$$\text{dove } c_0 = \frac{k - s_0}{s_3 - s_0} \quad \text{e} \quad d_0 = 1.$$

. Se $\tilde{s} = [s_0, s_1, k, s_2, s_3, s_4]$:

$$N[s_0, s_1, s_2, s_3, s_4](s) = c_1 N[s_0, s_1, k, s_2, s_3](s) + d_1 N[s_1, k, s_2, s_3, s_4](s)$$

$$\text{dove } c_1 = \frac{k - s_0}{s_3 - s_0} \quad \text{e} \quad d_1 = \frac{s_4 - k}{s_4 - s_1}.$$

. Se $\tilde{s} = [s_0, s_1, s_2, k, s_3, s_4]$:

$$N[s_0, s_1, s_2, s_3, s_4](s) = c_2 N[s_0, s_1, s_2, k, s_3](s) + d_2 N[s_1, s_2, k, s_3, s_4](s)$$

$$\text{dove } c_2 = \frac{k - s_0}{s_3 - s_0} \quad \text{e} \quad d_2 = \frac{s_4 - k}{s_4 - s_1}.$$

. Se $\tilde{s} = [s_0, s_1, s_2, s_3, k, s_4]$:

$$N[s_0, s_1, s_2, s_3, s_4](s) = c_3 N[s_0, s_1, s_2, s_3, k](s) + d_3 N[k, s_1, s_2, s_3, k, s_4](s)$$

dove $c_3 = 1$ e $d_3 = \dfrac{s_4 - k}{s_4 - s_1}$.

Nel caso bivariato, se è possibile scrivere una funzione base $B(s,t)$ come prodotto di due funzioni base $N(s)$ e $N(t)$, il ragionamento è identico. Se si aggiungono due nodi k_0 e k_1 rispettivamente ad s e a t allora la funzione base $B(s,t)$ si scriverà come combinazione lineare di 4 nuove funzioni base ottenute combinando le formule appena elencate.
Una funzione base T-Spline $B(s,t)$, che è data dal prodotto di due funzioni base univariate, può essere quindi raffinata e scritta come combinazione lineare di nuove funzioni base.

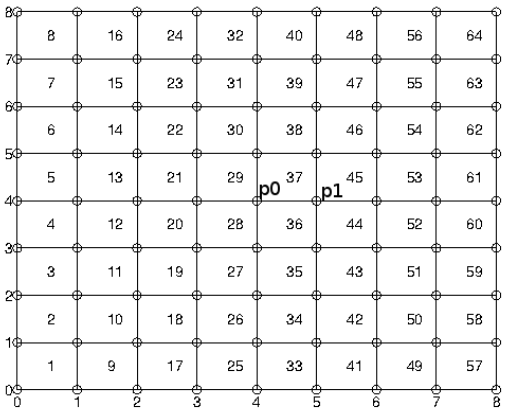

Figura 3.9: Raffinamento delle funzioni base associate ai nodi p_0 e p_1

Esempio 3.7. Si osservi la figura 3.9 in cui è mostrata una superficie NURBS. I punti p_0 e p_1 hanno coordinate parametriche rispettivamente $(4,4)$ e $(5,4)$. I loro vettori nodali in s e in t sono:

$$s_{p_0} = (2,3,4,5,6) \quad t_{p_0} = (2,3,4,5,6)$$

$$s_{p_1} = (3, 4, 5, 6, 7) \quad t_{p_1} = (2, 3, 4, 5, 6)$$

Se si inserisce il nodo $p = (3.5, 4)$ nella T-Mesh le funzioni base dei 2 punti considerati vengono entrambe modificate perché $s_{p0_0} < 3.5 < s_{p0_4}$ e $s_{p1_0} < 3.5 < s_{p1_4}$. Nel primo caso la coordinata s del nodo p è compresa tra il secondo e il terzo valore mentre nel secondo caso è compresa tra il primo e il secondo valore.
Applicando le formule scritte in precedenza si ottiene quindi che:

. Per p_0 si ha $\tilde{s} = [2, 3, 3.5, 4, 5, 6]$. I coefficienti di raffinamento sono:
$c_1 = \dfrac{k - s_0}{s_3 - s_0} = \dfrac{1.5}{3} = 0.5$
$d_1 = \dfrac{s_4 - k}{s_4 - s_1} = \dfrac{2.5}{3} = \dfrac{5}{6}$

. Per p_1 si ha $\tilde{s} = [3, 3.5, 4, 5, 6, 7]$. I coefficienti di raffinamento sono:
$c_0 = \dfrac{k - s_0}{s_3 - s_0} = \dfrac{0.5}{3} = \dfrac{1}{6}$
$d_0 = 1$

Di conseguenza le vecchie funzioni base in s associate a p_0 e p_1 si riscriveranno come combinazione delle nuove funzioni base in s così:

$$N[2, 3, 4, 5, 6](s) = 0.5N[2, 3, 3.5, 4, 5](s) + \frac{5}{6}N[3, 3.5, 4, 5, 6](s)$$

$$N[3, 4, 5, 6, 7](s) = \frac{1}{6}N[3, 3.5, 4, 5, 6](s) + N[3.5, 4, 5, 6, 7](s)$$

Come si può vedere la vecchia funzione base associata a p_0 si riscrive come combinazione delle nuove funzioni base associate a p e a p_0, mentre la vecchia funzione base associata a p_1 si riscrive come combinazione delle nuove funzioni base associate a p_1 e a p_0.
Si vede che se viene inserito un nodo in s non vengono modificate le funzioni base in t e analogamente se si inserisce un nodo in t non si modificano le funzioni base in s.
Si osserva infine che l'inserimento del nodo $(3.5, 4)$ ha modificato anche le funzioni base dei punti di coordinate $(2, 4)$ e $(3, 4)$, che non sono stati considerati nell'esempio. Il ragionamento fatto per i punti p_0 e p_1 può essere

ripetuto anche per i 2 punti non considerati e si ottiene che le vecchie funzioni base dei punti $(2, 4)$ e $(3, 4)$ vengono riscritte come combinazione lineare delle nuove funzioni base associate a tali punti e al nodo $(3.5, 4)$.

L'algoritmo di knot insertion proposto ora si basa sul raffinamento delle funzioni base. Ogni funzione base appartenente alla T-Mesh iniziale viene riscritta come combinazione lineare di nuove funzioni base appartenenti alla T-Mesh finale. Si ha così che

$$P(s,t) = \frac{\sum_{i=1}^{n} P_i w_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)} \tag{3.3}$$

diventa

$$P(s,t) = \frac{\sum_{i=1}^{m} \tilde{P}_i \tilde{w}_i \tilde{B}_i(s,t)}{\sum_{i=1}^{m} \tilde{w}_i \tilde{B}_i(s,t)} \tag{3.4}$$

dove $m > n$ e dove ogni funzione base vecchia $B_i(s,t)$ viene espressa come

$$B_i(s,t) = \sum_{j=1}^{m} c_{i,j} \tilde{B}_j(s,t) \tag{3.5}$$

Nell'equazione della nuova superficie i coefficienti $\tilde{P}_j$ (e così anche i pesi $\tilde{w}_i$) sono determinati da

$$\tilde{P}_j = \sum_{i=1}^{n} c_{i,j} P_i(s,t) \tag{3.6}$$

L'algoritmo è suddiviso in due fasi, una topologica e una geometrica. Nella fase topologica si determinano e si inseriscono tutti i punti di controllo necessari per l'inserimento del nodo richiesto. Nella fase geometrica invece, si riscrive l'equazione della superficie andando a determinare i nuovi valori da associare ai punti di controllo e ai loro corrispondenti pesi.

Fase topologica

Inizialmente ad ogni punto di controllo P_i della T-Mesh è associata una funzione base, determinata univocamente da due vettori nodali s_i e t_i. I vettori nodali sono costruiti rispettando la seguente regola, già introdotta nel capitolo precedente:

. Gli elementi centrali di s_i e di t_i sono le coordinate parametriche di P_i in s e t rispettivamente.

. I due elementi successivi a s_{i2} in s_i sono le coordinate dei primi due s-edges o punti di controllo intersecati dal raggio $R(\alpha) = (s_{i2} + \alpha, t_{i2})$ nello spazio parametrico (con $\alpha > 0$). Analogamente si determinano i due elementi successivi a t_{i2} in t_i.

. I due elementi mancanti nei vettori s_i e t_i si calcolano con un procedimento identico al caso precedente, con l'unica differenza che in questo caso $\alpha < 0$.

L'inserimento di nuovi punti di controllo crea violazioni della regola appena detta, poichè si modificano i vettori nodali associati ad alcune funzioni base. Inoltre non è detto che i vettori nodali presenti consentano di raffinare correttamente le funzioni base nella fase geometrica.
Per essere più precisi, esistono 3 tipi di violazioni che si possono creare:

A. Nella T-Mesh attuale una funzione base ha i vettori nodali che non rispettano la regola precedentemente citata.

B. Una funzione base ha un vettore nodale che non le consente di essere miscelata con un'altra funzione base.

C. Un punto di controllo non ha una funzione base associata.

Questi 3 tipi di violazioni vengono risolti ad ogni passo dell'algoritmo inserendo nodi aggiuntivi o modificando i vettori nodali dei punti di controllo presenti.

A questo punto è possibile descrivere la struttura della fase topologica dell'algoritmo:

1. Si inseriscono tutti i punti di controllo richiesti.
2. Si controllano tutte le violazioni presenti e si determinano gli eventuali punti di controllo da aggiungere.
3. Si ripetono i passi precedenti fino a quando non sono risolte tutte le violazioni e non ci sono più nodi da inserire.

Osservazione 18. La sequenza proposta giunge necessariamente a termine in un numero finito di operazioni.
Questo fatto si dimostra immediatamente considerando che nella situazione peggiore l'algoritmo aggiunge tanti punti di controllo quanti sono quelli necessari per ottenere una superficie NURBS.
Non è possibile che vengano inseriti infiniti punti perché, per come sono state definite le violazioni, non si richiede mai di inserire punti di controllo con coordinate parametriche s_i e t_i diverse dalle coordinate parametriche in s e in t dei punti della T-Mesh e del primo punto da inserire.

Osservazione 19. L'algoritmo è applicabile per ogni punto della superficie. Infatti, contrariamente a quanto accade nel primo algoritmo, i punti di controllo richiesti vengono inseriti per primi e solo successivamente si controllano le violazioni che si creano. Inoltre le violazioni non richiedono mai di togliere punti di controllo, ma solamente di aggiungerne.

Si vede ora un primo importante esempio, che consente di capire il funzionamento dell'algoritmo.

Esempio 3.8. Si consideri la T-Mesh rappresentata in figura 3.10. Si vuole inserire il nodo $(1.5, 4)$, segnato con un asterisco in figura.

1. Si inserisce il nodo $(1.5, 4)$ nel lato 66.
2. Si determinano tutte le violazioni presenti. In particolare sono presenti 4 violazioni di tipo A e 2 violazioni di tipo B.

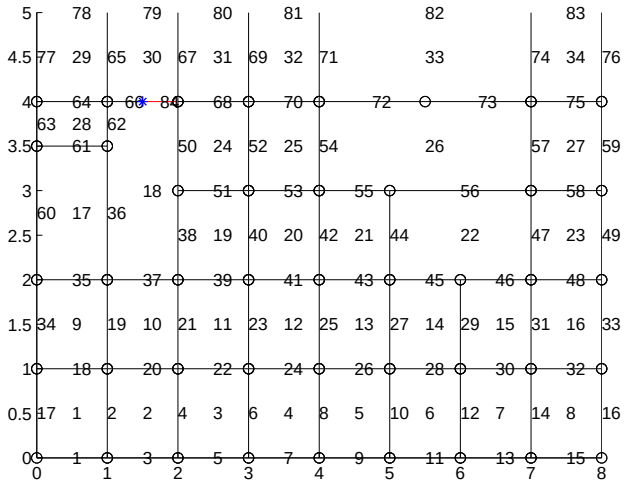

Figura 3.10: Inserimento nodo $(1.5, 4)$

Le violazioni di tipo A riguardano i punti di controllo di coordinate $(0, 4), (1, 4), (2, 4)$ e $(3, 4)$. I vettori nodali di questi punti non rispettano più la regola di costruzione dei vettori nodali a causa dell'inserimento di $(1.5, 4)$.

Le violazioni di tipo B invece riguardano i punti di controllo di coordinate $(1, 4)$ e $(2, 4)$. Le vecchie funzioni base associate a questi punti di controllo non possono essere attualmente scritte come combinazione lineare delle nuove funzioni base associate ai medesimi punti e al punto inserito. È necessario quindi aggiungere i punti $(1.5, 3.5)$ e $(1.5, 3)$.

3. Si inseriscono i punti $(1.5, 3.5)$ e $(1.5, 3)$. In figura 3.11 si vede che i punti di coordinate $(0, 3.5), (1, 3.5), (3, 3)$ e $(2, 3)$ presentano violazioni di tipo A. Non sono presenti violazioni di tipo B, di conseguenza non ci sono più punti da inserire e l'algoritmo termina la parte topologica.

Fase geometrica

La seconda parte dell'algoritmo riguarda il calcolo dei valori di tutti i punti di controllo della superficie e dei corrispondenti pesi. In questa fase non

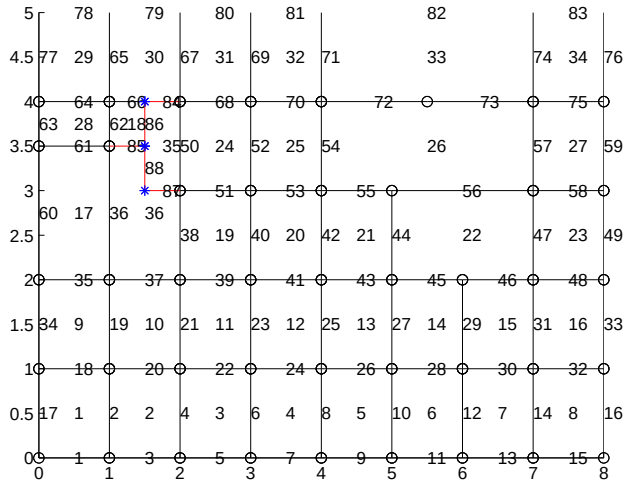

Figura 3.11: Inserimento nodi $(1.5, 3.5)$ e $(1.5, 3)$

vengono aggiunti punti di controllo alla superficie.
Il calcolo dei valori dei punti di controllo segue le regole definite per il raffinamento di funzioni base:

1. Si scrive ogni vecchia funzione base come combinazione lineare delle nuove funzioni base presenti, esattamente come scritto nell'equazione (3.5). I coefficienti della combinazione saranno determinati utilizzando le equazioni di raffinamento delle funzioni blending.

2. Si moltiplica sia a destra che a sinistra dell'uguale per il valore del punto di controllo P_i, che si indica sempre con P_i e si ottiene:

$$P_i B_i(s,t) = \sum_{j=1}^{m} P_i c_{i,j} \tilde{B}_j(s,t) \tag{3.7}$$

   Ogni coefficiente $P_i c_{i,j}$ associato alla funzione base $\tilde{B}_j(s,t)$ è un addendo della sommatoria relativa al nuovo coefficiente $\tilde{P}_j$, che è descritta nell'equazione (3.6). Analogamente si comportano i pesi.

3. Unendo le equazioni ottenute per ogni i si può scrivere l'uguaglianza:

$$\frac{\sum_{i=1}^{n} P_i w_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)} = \frac{\sum_{i=1}^{m} \tilde{P}_i \tilde{w}_i \tilde{B}_i(s,t)}{\sum_{i=1}^{m} \tilde{w}_i \tilde{B}_i(s,t)}$$

che verifica quanto scritto nelle equazioni (3.3) e (3.4).

Osservazione 20. Subito si possono adottare alcuni accorgimenti per migliorare l'efficienza di questa seconda parte dell'algoritmo:

. I punti di controllo che variano il loro valore sono quelli che cambiano dominio di influenza dopo la fase topologica dell'algoritmo. Di conseguenza tutti i punti di controllo che non sono considerati nella fase topologica non devono necessariamente essere considerati nella fase geometrica.

. Una funzione base $B_i(s,t)$ associata ad un punto di controllo P_i coinvolto nella fase topologica si scrive come combinazione lineare delle nuove funzioni base $\tilde{B}_j(s,t)$ che hanno dominio di influenza contenuto nel dominio di influenza associato a $B_i(s,t)$. È possibile quindi determinare un sottoinsieme ristretto di funzioni di base effettivamente coinvolte nella sommatoria.

Esempio 3.9. Si consideri ad esempio la figura 3.12. Il punto di controllo con coordinate nodali $(1,4)$ (segnato con un pallino rosso), prima dell'inserimento dei nodi (contrassegnati con gli asterischi), aveva un dominio di influenza rappresentato dall'area evidenziata. Dopo il knot insertion i punti che hanno dominio di influenza contenuto nel dominio iniziale di $(1,4)$ sono: $(1,4), (1.5,4)$ e $(1.5,3.5)$. Di conseguenza la funzione base associata a $(1,4)$ si scriverà come combinazione lineare delle nuove funzioni base associate ai 3 punti appena citati.

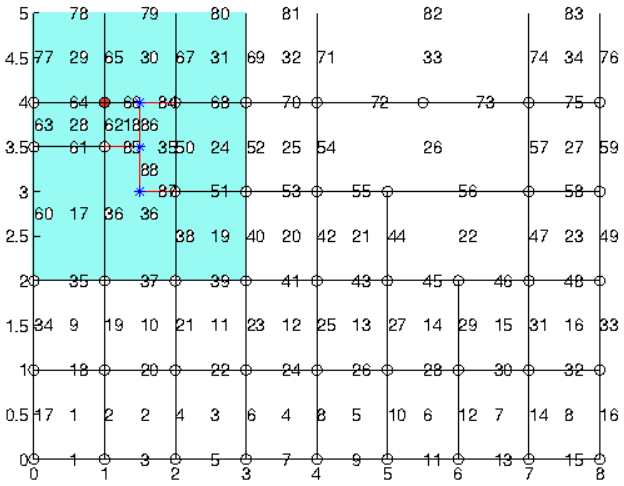

Figura 3.12: Nodo $(1, 4)$, dominio di influenza pre-inserimento

La condizione appena posta è necessaria ma non sufficiente. Esistono casi particolari in cui una nuova funzione base che ha dominio di influenza contenuto nel dominio di $B_i(s, t)$ non è coinvolta nella combinazione lineare.

Esempio 3.10. Si consideri la figura 3.13. Il punto di coordinate $(8, 3)$, che aveva come dominio di influenza iniziale il rettangolo $[5, 10] \times [1, 5]$, si può scrivere come combinazione lineare delle funzioni base associate ai punti $(8, 3), (7, 3)$ e $(6, 3.5)$. L'ultimo punto però, non essedo collegato agli altri due da nessun lato, non ha influenza nella combinazione lineare.

Il secondo algoritmo di knot insertion proposto risolve in maniera assai efficiente i problemi presentati dal primo algoritmo visto. La convergenza dell'algoritmo è sempre garantita in un numero finito di passi. Nel caso peggiore, come si è visto, l'algoritmo regolarizza la T-Mesh della superficie, rendendo la T-Spline una NURBS.
Inoltre si è visto che il numero di punti di controllo che l'algoritmo 2 inserisce è solitamente minore rispetto a quelli richiesti dall'algoritmo 1.

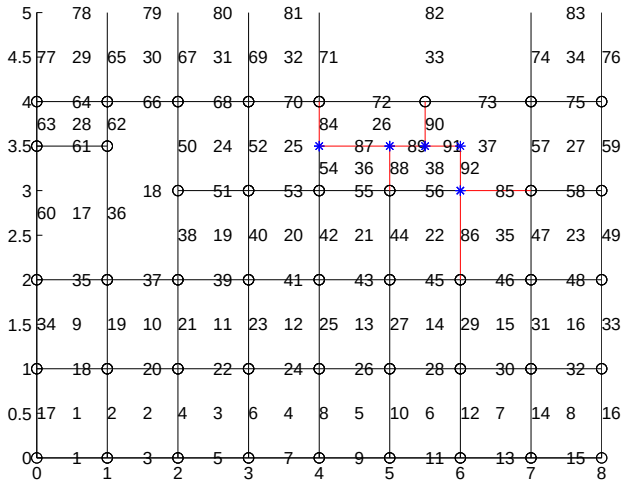

Figura 3.13: Algoritmo 2: inserimento nodo $(4, 3.5)$

Esempio 3.11. Una prima conferma di quanto appena detto si ha riprendendo l'esempio 3.5, in cui è stato applicato l'algoritmo 1 per inserire il nodo $(3, 1.5)$. Applicando l'algoritmo 2 nelle stesse condizioni si ottiene la superficie presentata in figura 3.14, in cui viene inserito un solo punto di controllo oltre a quello richiesto inizialmente. In questo caso quindi invece che 8 punti di controllo ne viene inserito solamente 1.

3.4 Algoritmi di raffinamento locale

Gli algoritmi di raffinamento locale di una superficie sono una generalizzazione degli algoritmi di knot insertion. Questi algoritmi, che non modificano la superficie, sono utilizzati per raffinare la mesh di controllo associata alla superficie. Il raffinamento della mesh di controllo viene effettuato per poter poi modificare la superficie in una regione localmente ristretta utilizzando algoritmi di knot insertion che modificano la superficie.
Esistono due tipi di raffinamento locale:

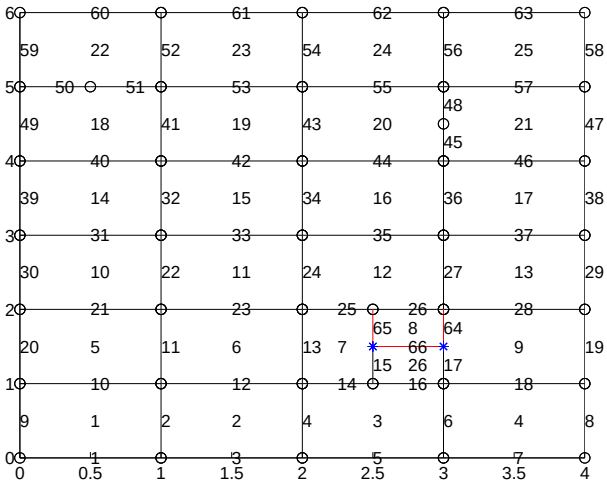

Figura 3.14: Algoritmo 2: inserimento nodo $(3, 1.5)$

. Face Refinement: si inseriscono nuovi nodi nei punti medi dei lati l_i della faccia che si vuole raffinare.

. Knot Refinement: si inseriscono nuovi nodi nei punti medi dei lati l_i che hanno il nodo scelto come estremo.

Gli algoritmi di raffinamento locale per superfici T-Spline che vengono introdotti in questo capitolo hanno la seguente struttura:

. Si determinano i punti medi di tutti i lati l_i coinvolti nel raffinamento e si richiama l'algoritmo di knot insertion per inserire i nuovi nodi.

Gli algoritmi di raffinamento locale differiscono a seconda dell'algoritmo di knot insertion che utilizzano per inserire i nodi. Le proprietà di questi algoritmi sono quindi le stesse dei corrispondenti algoritmi di knot insertion.

Osservazione 21. Variando l'ordine di raffinamento delle facce o dei nodi della superficie l'algoritmo 1 produce risultati differenti, mentre invece l'algoritmo 2 ottiene sempre lo stesso risultato.
Questo fatto si spiega semplicemente considerando che l'algoritmo 1 di knot

insertion inserisce un nodo solo alla volta e richiede che siano soddisfatte delle condizioni prima dell'inserimento di tale nodo. Di conseguenza variare l'ordine di inserimento dei nodi può modificare la sequenza delle operazioni che vengono fatte e può variare le condizioni che si devono rispettare ad ogni passo.

In conclusione quindi gli algoritmi di raffinamento che utilizzano il secondo algoritmo di knot insertion dovrebbero essere più efficienti dei corrispondenti algoritmi che utilizzano il primo algoritmo di knot insertion presentato.

Capitolo 4

Realizzazione degli algoritmi di valutazione e knot insertion

Obiettivo principale di questa tesi è la realizzazione degli algoritmi di editing delle superfici T-spline introdotti nel capitolo precedente. Per creare questi algoritmi è stata progettata una particolare struttura dati che permette di ottenere dalla preimmagine di una T-Mesh tutte le informazioni necessarie per la valutazione, l'inserimento di nodi e il raffinamento locale di una superficie T-Spline.
Sono stati implementati entrambi gli algoritmi di knot insertion precedentemente descritti. Il primo algoritmo, che è molto meno efficiente del secondo e non converge in ogni situazione, richiede semplicemente di applicare ricorsivamente la regola per stabilire se è possibile inserire il punto desiderato.
Il secondo algoritmo invece è definito da alcune semplici linee guida, che sono le tre possibili violazioni introdotte nel capitolo precedente. Queste violazioni non sono però sufficienti per definire un'unica struttura dell'algoritmo. Per questo motivo sono stati fatti alcuni confronti tra differenti implementazioni con l'obiettivo di determinare, se possibile, una sequenza di passi che risulti più efficiente di altri.
Nella prima parte del capitolo si introduce la struttura dati creata. Successivamente si spiegano nel dettaglio tutti gli aspetti della realizzazione degli

algoritmi di valutazione e knot insertion. Infine si effettuerà un confronto tra i due algoritmi di raffinamento locale introdotti nel capitolo precedente.

4.1 Struttura dati

La struttura dati ideata contiene tutte le informazioni necessarie per modificare o valutare una generica superficie T-Spline. In particolare sono presenti le coordinate parametriche dei punti di controllo e le informazioni relative ai lati e alle facce della T-Mesh. Inoltre ad ogni punto di controllo è associato un vettore di indici relativi ai lati che consente di determinare sulla preimmagine della T-Mesh i vettori nodali del punto stesso.
Le superfici T-Spline vettoriali, come si è detto nei capitoli precedenti, permettono di rappresentare anche oggetti di genere maggiore di 1. In questi casi non è sempre possibile costruire globalmente la preimmagine di una T-Mesh nel dominio parametrico (s, t). Poichè però sia l'algoritmo di valutazione che il secondo algoritmo di knot insertion considerano solo localmente la superficie, è sempre possibile definire la struttura dati relativa ad una parte della superficie che ammette preimmagine della T-Mesh nel dominio parametrico (s, t). Per questo motivo d'ora in poi si considereranno superfici T-Spline di tipo funzione che ammettono preimmagine della T-Mesh in (s, t).
La struttura dati appena introdotta è composta da quattro matrici e due vettori. Più precisamente si hanno:

. Una matrice V contenente tante righe quanti sono i punti di controllo della T-Mesh e 6 colonne. Le prime due colonne contengono le coordinate parametriche del vertice, le tre colonne successive contengono le coordinate x, y, z dei punti di controllo e la sesta colonna invece contiene il valore del peso associato al punto di controllo.

. Una matrice L relativa ai lati della T-Mesh. Ogni riga della matrice contiene gli indici associati ai vertici estremi del lato, le loro coordinate parametriche e un valore che indica se il lato è considerato orizzontale (2) o verticale (1) nella preimmagine della T-Mesh.

. Una matrice F contenente le informazioni relative alle facce che costituiscono la T-Mesh. In particolare sulla i-esima riga sono presenti gli indici dei vertici che formano la faccia.
Poichè le T-Spline non hanno una griglia regolare come le NURBS, esistono facce con un numero diverso di vertici ed esisterà almeno una faccia f_i con m vertici, dove m è il massimo numero di vertici di una faccia. Le righe della matrice F associate a facce con un numero di vertici minore di m sono completate con tanti zeri quanti sono i vertici che mancano per arrivare a m.

. Una matrice M con tante righe quanti sono i punti di controllo della T-Mesh in cui sono contenute le informazioni necessarie per calcolare i vettori nodali dei vertici della griglia.
Più precisamente si è visto che i vettori nodali associati ad un punto di controllo contengono le coordinate parametriche dei primi due s-edge (t-edge) sopra (a destra) e dei primi due s-edge (t-edge) sotto (a sinistra) il punto stesso. Di conseguenza tali vettori si determinano conoscendo i primi 8 lati che si intersecano muovendosi dal punto considerato nelle 4 direzioni principali. Esistono due possibili casi: o si interseca un vertice di un lato o si interseca un punto interno ad un lato. Nel primo caso, per le regole della T-Mesh, è presente un lato che congiunge il punto di controllo considerato e il vertice intersecato. Nel secondo caso invece si attraversa una faccia. Nel primo caso è sufficiente memorizzare l'indice associato al lato che congiunge i due punti, mentre nel secondo caso si memorizza l'indice del lato intersecato, cambiandolo di segno per evidenziare che si tratta di un lato intersecato internamente.
Nella matrice sono memorizzati gli indici associati a questi 8 lati partendo dai 2 lati a sinistra del punto e muovendosi in senso antiorario.

. Due vettori s e t contenenti tutte le coordinate parametriche in s e in t rispettivamente dei vertici che compongono la T-Mesh. I primi e gli

ultimi due elementi dei vettori contengono le coordinate parametriche di punti di controllo fittizi inseriti per poter calcolare i vettori nodali di ogni punto di controllo della T-Mesh.

Esempio 4.1. Per comprendere meglio ciò che si è detto si consideri la figura 4.1, in cui è rappresentata la preimmagine di una T-Mesh nello spazio parametrico (s, t). Ad ogni lato e ad ogni faccia è associato un indice. Le facce della T-Mesh sono 12 e i lati sono 36. I vertici sono 25 e sono numerati partendo dall'angolo in alto a sinistra e muovendosi verso destra. La struttura

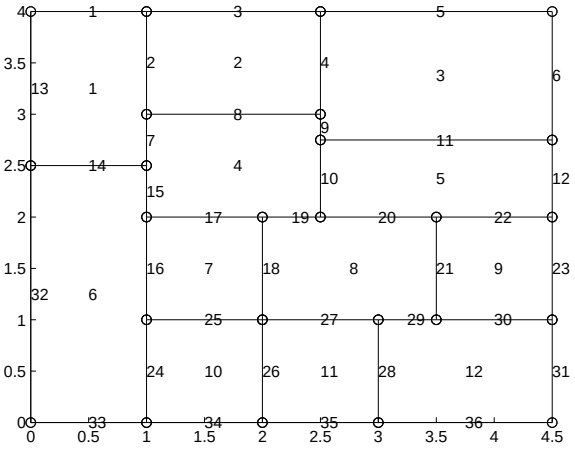

Figura 4.1: Preimmagine di una T-Mesh

dati ideata ha una matrice V con 25 righe e 6 colonne. La matrice L ha 36 righe e 7 colonne. La riga associata al lato 1 è ad esempio:

```
L(1,:) = [1,  2,  0,  4,  1,  4,  2]
```

I primi due elementi sono gli indici associati agli estremi del lato, il terzo e il quarto elemento sono le coordinate parametriche del primo vertice, il quinto e il sesto sono le coordinate del secondo vertice. L'ultimo elemento indica che il lato è orizzontale.
La matrice F ha 12 righe e 7 colonne perché la faccia 4 è composta da 7 vertici. Un esempio è:

```
F(4,:) = [5  6  7  13  12  11  10]
F(5,:) = [7  8  15  14  13  0  0]
```

La riga 5 ha due valori nulli perché la faccia 5 è composta solamente da 5 vertici.
La matrice M ha 25 righe e 8 colonne. Alcune righe sono:

```
M(1,:) = [0  0  13  32  1  3  0  0]
M(5,:) = [-13  0  7  15  8  -6  2  0]
M(14,:) = [19  20  21  -36  22  0  -11  -5]
M(17,:) = [25  -32  26  0  27  29  18  -8]
```

Come si vede dall'esempio proposto, il vertice 1 che ha coordinate $(0, 4)$ ha valori diversi da 0 solamente andando verso il basso e verso destra. Questo accade perché il punto si trova ai bordi della T-Mesh. I valori 13 e 32 indicano che, muovendosi verso il basso, bisogna percorrere i lati 13 e 32 per intersecare i primi due vertici che determinano i primi due elementi del vettore nodale in t relativo al vertice 1. Analogamente i valori 1 e 3 indicano i primi 2 lati che si devono percorrere per determinare gli ultimi due elementi del vettore nodale associato a 1 in s. Quando i valori sono uguali a 0 i vettori nodali vengono determinati utilizzando i vettori generali s e t.
Il nodo 5 ha come primo elemento -13, quindi il secondo elemento del vettore nodale in s associato a 5 è dato dalla coordinata parametrica s dei due vertici de lato 13. Nello stesso modo vengono calcolati i vettori nodali di tutti gli altri punti.

4.2 Algoritmo di valutazione

L'algoritmo di valutazione realizzato valuta separatamente le singole facce della superficie. Ad ogni passo l'algoritmo costruisce una griglia di discretizzazione della faccia da valutare e calcola i valori che la superficie assume in tutti i singoli punti della griglia.
Come si è detto nel capitolo precedente, per evitare calcoli inutili e costosi,

nella valutazione di una faccia della superficie vengono utilizzati solamente i punti di controllo che hanno dominio di influenza che interseca l'interno della faccia stessa. Di conseguenza, prima di richiamare l'algoritmo vero e proprio vengono creati dei *Patch*, strutture di dati contenenti le informazioni necessarie per la valutazione delle facce. Ogni patch è dedicato ad una faccia della superficie e contiene l'insieme dei punti di controllo interessati nella valutazione. Per ogni singolo punto di controllo sono poi memorizzate le seguenti informazioni:

. Coordinate spaziali x, y, z, w del punto.

. Coordinate parametriche s, t del punto.

. Knot vector in s e in t associati al punto di controllo.

Nella fase che precede la valutazione viene effettuato un ciclo su tutte le facce in cui viene richiamata la funzione `create_patch` che riempie i campi della struttura `patches`:

```
for i=1:numero_facce
  create_patch(mesh,patches,i);
end
numero_patch=numero_facce
```

Successivamente viene richiamato l'algoritmo di valutazione, che ha la seguente struttura:

```
function(Surface)=surface_evaluation(Mesh)
  for i=1:numero_patch
     G=scanconv(patch,i);
     for j=1:numero_punti_griglia_G
      Surface=valuta_punto(G[j],patch,i);
     end
  end
end
```

Per ogni elemento dell'array `patches` viene richiamata la funzione `scanconv` che crea la griglia di valutazione `G` della faccia. Si è scelto di utilizzare l'algoritmo di scanconv perché questo ha la particolarità di riuscire a creare griglie di valutazione anche per facce non rettangolari, che presentano punti ad L. Successivamente si calcola il valore della superficie in ognuno dei punti della griglia `G`, utilizzando la formula che caratterizza le superfici T-Spline, presentata nel capitolo 1 (definizione 2.18).

4.3 Algoritmo di knot insertion 1

Il primo algoritmo di knot insertion, come si è visto, inserisce un nodo all'interno di un lato della T-Mesh solamente se viene rispettata la seguente regola:

> Un generico punto P_3 appartenente ad un t-edge (s-edge) di una superficie può essere inserito se e solo se i vettori nodali t_1, t_2, t_4, t_5 (s_1, s_2, s_4, s_5) associati ai quattro punti P_1, P_2, P_4, P_5 precedenti e successivi a P_3 nella direzione s sono tra loro uguali.

L'algoritmo realizzato ha quindi una struttura ricorsiva molto semplice:

```
function(M,L,V,F)=tmesh_knot_insertion1(M,L,V,F,ntins,nodo)
  nodes=violation_control(M,L,V,F,ntins,nodo);
  condizione=control_algorithm(ntins);
  if condizione
     for i=1:num_nodes
      (M,L,V,F)=tmesh_knot_insertion1(M,L,V,F,ntins,nodes(i));
     end
     insert(nodo);
  else errore!
end
```

Viene passata in input e restituita in output l'intera struttura dati relativa alla T-Mesh, che viene aggiornata ogni volta che viene effettivamente inserito

un nuovo punto di controllo. La funzione `tmesh_knot_insertion1` prende poi in input il nodo da inserire e una matrice `ntins` in cui sono memorizzati i nodi da inserire. La matrice `ntins` viene utilizzata per controllare che l'algoritmo possa terminare correttamente in un numero finito di passi.
Prima di inserire il nodo viene richiamata la funzione `violation_control` che restituisce in output una matrice `nodes` contenente i nodi che devono essere inseriti prima di poter inserire il nodo desiderato.
Successivamente viene effettuato un controllo sulla matrice `ntins` per stabilire se l'algoritmo può terminare correttamente. Nel caso in cui l'algoritmo entri in un ciclo infinito viene stampato un messaggio di errore e viene bloccato il processo. In caso affermativo invece viene richiamata ricorsivamente la funzione di knot insertion per inserire, uno per volta, tutti i nodi della matrice `nodes`. Solo se tutti gli inserimenti dei nodi vengono effettuati correttamente viene inserito il nodo inizialmente fornito in input.

4.4 Algoritmo di knot insertion 2

Il secondo algoritmo di knot insertion, basato sul raffinamento delle funzioni base, è suddiviso tra fase topologica e fase geometrica. La parte topologica ha una struttura ricorsiva in cui ad ogni passo vengono inseriti immediatamente tutti i punti di controllo necessari e successivamente vengono determinate le violazioni che si creano e gli eventuali punti di controllo aggiuntivi da inserire. Nella fase geometrica invece vengono miscelate le funzioni base per ottenere una nuova combinazione di funzioni base, punti di controllo e pesi che lascia invariata la superficie.

Fase topologica

La fase topologica dell'algoritmo è la più interessante dal punto di vista dell'implementazione. È possibile infatti variare la struttura dell'algoritmo senza violare le linee guida date, ottenendo risultati diversi ma comunque corretti. Questo fatto si verifica sia perché l'algoritmo non inserisce neces-

sariamente il numero minimo di punti aggiuntivi che rendono valido l'inserimento dei nodi iniziali sia perché l'inserimento di più punti di controllo aggiuntivi lascia invariata la superficie.
Più precisamente le possibili strutture diverse dell'algoritmo si differenziano nel momento in cui si richiama ricorsivamente la funzione di inserimento dei nodi che devono essere aggiunti alla T-Mesh per evitare violazioni di tipo B (capitolo 3.3).
L'algoritmo introdotto teoricamente nel capitolo precedente ha la seguente struttura:

```
function(M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,nodo)
  inserisci_nodi(nodo);
  nodes=controlla_violazioni(M,L,V,F,nodo);
  (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,nodes)
end
```

La funzione `tmesh_knot_insertion2` prende in input la struttura dati e il vettore dei nodi da inserire e restituisce in output la struttura dati modificata.
Nell'algoritmo le fasi di `inserisci_nodi` e `controlla_violazioni` hanno una struttura leggermente più complessa poichè si deve poter gestire l'inserimento di più nodi contemporaneamente e per ognuno dei nodi inseriti si devono controllare sia le violazioni di tipo A che le violazioni di tipo B.
Esiste un'unica funzione, chiamata `general_node_insertion`, che inserisce tutti i nodi e controlla immediatamente le violazioni di tipo A che si creano richiamando la funzione `det_A_violations`. Successivamente le violazioni di tipo B sono determinate dalla funzione `det_B_violations` che viene richiamata più volte nella funzione principale. La funzione `general_node_insertion` ha la seguente struttura:

```
function(M,L,V,F)=general_node_insertion(M,L,V,F,nodo)
  for i=1:num_nodi
    (M,L,V,F,num1)=insert_nodo(M,L,V,F,nodo(i));
```

```
    det_A_violations(num1);
  end
end
```

Ad ogni passo la funzione di inserimento nodi varia a seconda che si debba inserire un nodo appartenente ad un lato o un nodo interno ad una faccia della T-Mesh. In entrambi i casi viene determinato un vettore `num1` contenente gli indici dei nodi appartenenti al primo ring dell'i-esimo nodo inserito. Questo vettore viene poi utilizzato nella funzione `det_A_violations` per determinare i nodi per cui è presente una violazione di tipo A, ovvero tutti i nodi che hanno variato i loro vettori nodali dopo l'inserimento dell'i-esimo nodo.
Spesso la sequenza di controlli effettuati per risolvere le molteplici violazioni che si creano richiede l'inserimento e il controllo di un nodo già presente nella T-Mesh. La funzione di inserimento nodi evita di inserire più volte un nodo della griglia, ma determina i nodi adiacenti al nodo considerato così da poter controllare in seguito eventuali violazioni di tipo B presenti.
La parte centrale dell'algoritmo di knot insertion riguarda la determinazione delle violazioni di tipo B, che consentono di stabilire se devono essere aggiunti ulteriori nodi alla T-Mesh. Nell'algoritmo le violazioni di tipo B vengono controllate dalla funzione `det_B_violations` che ha la seguente struttura:

```
function(ntins,bviol)=det_B_violations(num,k,ss,tt,ntins,bviol)
  if k==1
    if ss(num(1))!=ss(num(2))
    è presente una violazione di tipo B
    (ntins,bviol)=determina_nodi_da_inserire(ntins,bviol)
    end
  else if tt(num(1))!=tt(num(2))
    è presente una violazione di tipo B
    (ntins,bviol)=determina_nodi_da_inserire(ntins,bviol)
       end
  end
```

```
end
```

Il parametro k stabilisce se si stanno analizzando due estremi di un lato orizzontale (k=2) o verticale (k=1). Due vertici di un lato verticale presentano una violazione di tipo B se non hanno lo stesso vettore nodale in s, in quanto nella fase geometrica non è possibile combinare linearmente le loro funzioni base per ottenere una vecchia funzione base. Analogamente due estremi di un lato orizzontale determinano una violazione di tipo B se non hanno gli stessi vettori nodali in t. Se è presente una violazione di tipo B si determinano i nodi che devono essere inseriti nella T-Mesh per risolvere la violazione e si inseriscono nella matrice `ntins`. La matrice `bviol` tiene invece memoria di tutte le violazioni di tipo B trovate, che verranno in seguito controllate per verificare che siano stati effettivamente inseriti tutti i nodi richiesti dall'algoritmo.

Esempio 4.2. Per comprendere meglio il comportamento dell'algoritmo e vedere un semplice esempio in cui il controllo delle violazioni legate ad un nodo già presente risulta fondamentale per l'algoritmo si consideri la figura 4.2 in cui si vuole inserire nella T-Mesh il nodo di coordinate parametriche $(3, 1.5)$.

L'algoritmo richiama la funzione `general_node_insertion` in cui viene inserito il nodo $(3, 1.5)$ e vengono determinati i nodi che presentano violazioni di tipo A. Questi nodi hanno coordinate: $(3, 0), (3, 1), (3, 2)$ e $(3, 3)$. Successivamente vengono controllate le eventuali violazioni di tipo B, richiamando ripetutamente la funzione `det_B_violations`. Come si è visto in precedenza la funzione che determina tali violazioni considera coppie di vertici appartenenti ad uno stesso lato e determina se è possibile effettuare una combinazione lineare tra le nuove funzioni base associate a questi nodi.
In questo esempio la vecchia funzione base associata al nodo $(3, 0)$ si dovrebbe scrivere come combinazione delle nuove funzioni base associate ai nodi $(3, 0)$ e $(3, 1)$, estremi del lato verticale 6. Poichè però i vettori nodali in s delle funzioni base associate a questi due nodi non coincidono si ha una violazione di tipo B che richiede l'inserimento del nodo $(2.5, 1)$, già presente

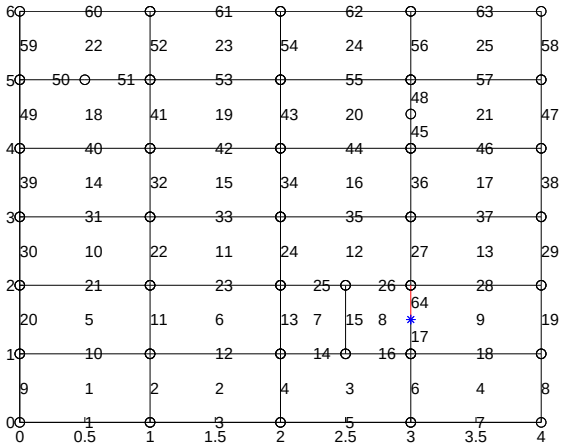

Figura 4.2: Inserimento nodo (3,1.5)

nella T-Mesh. La vecchia funzione base associata a $(3, 0)$ si scriverà quindi come combinazione delle nuove funzioni base associate ai nodi $(3, 0)$, $(3, 1)$ e $(2.5, 1)$. Lo stesso ragionamento si può applicare ai nodi $(3, 2)$,$(3, 3)$, per cui l'algoritmo richiede di aggiungere il nodo $(2.5, 2)$, già presente nella T-Mesh. Le coppie di nodi $(3, 1), (3, 1.5)$ e $(3, 2), (3, 1.5)$ non presentano invece violazioni di tipo B.

L'algoritmo viene richiamato ricorsivamente per inserire i nodi $(2.5, 1)$ e $(2.5, 2)$. Nella funzione `general_node_insertion` viene effettuato un controllo su tali nodi e vengono determinati i nodi che potrebbero causare violazioni di tipo B. In questo caso particolare poichè la funzione base associata al nodo $(2.5, 1)$ deve essere miscelata con la funzione base associata al nodo $(3, 1)$ e i due nodi non hanno lo stesso vettore nodale in t, l'algoritmo aggiunge alla T-Mesh il nodo $(2.5, 1.5)$. Analogamente i nodi $(2.5, 2)$ e $(3, 2)$ presentano una violazione di tipo B che si risolve inserendo il nodo $(2.5, 1.5)$. L'inserimento dell'ultimo nodo risolve tutte le violazioni presenti e l'algoritmo termina.

Nella figura 4.3 si vede la T-Mesh finale, in cui sono stati inseriti i nodi $(3, 1.5)$ e $(2.5, 1.5)$.

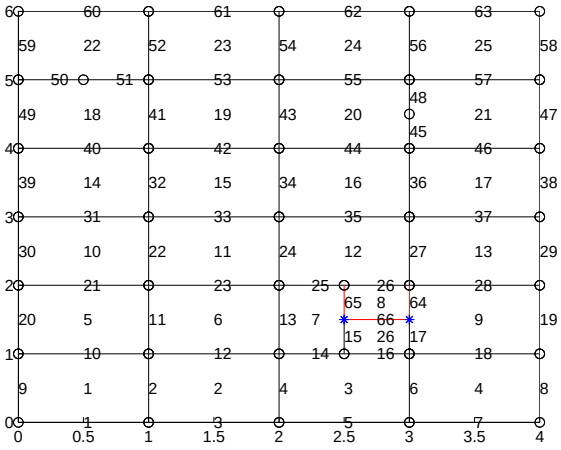

Figura 4.3: Inserimento nodo (3,1.5), T-Mesh finale

Come si è detto in precedenza, la funzione `det_B_violations` viene richiamata più volte nell'algoritmo per determinare eventuali violazioni di tipo B. Esistono due parti distinte dell'algoritmo che separano il controllo delle violazioni relative ai nodi appena inseriti dal controllo delle violazioni riguardanti i nodi già presenti nella T-Mesh e che devono essere nuovamente controllati. Una delle possibili strutture dell'algoritmo può essere quindi schematizzata nel seguente modo:

```
function(M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,nodo)
  (M,L,V,F)=general_node_insertion(M,L,V,F,nodo);
  for i=1:num_nodi
    if nodo(i) è stato appena inserito
(ntins,bviol)=det_B_violations(nodo(i),ntins,bviol);
    else
(ntins,bviol)=det_B_violations(nodo(i),ntins,bviol);
    end
  (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,ntins)
end
```

Questa prima struttura determina tutte le violazioni di tipo B associate sia ai nodi appena inseriti sia ai nodi già presenti che sono stati controllati e successivamente aggiunge tutti i nuovi nodi che devono essere inseriti.
Una funzione simile è la seguente:

```
function(M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,nodo,pass)
  (M,L,V,F)=general_node_insertion(M,L,V,F,nodo);
  for i=1:num_nodi
    if nodo(i) è stato appena inserito
        (ntins,bviol)=det_B_violations(nodo(i),ntins,bviol);
        (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,ntins,pass+1)
    else
        (ntins,bviol)=det_B_violations(nodo(i),ntins,bviol);
        (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,ntins,pass+1)
    end
  if pass==0
    for i=1:num_bviol
      (ntins,bviol)=det_B_violations(bviol(i),ntins,bviol);
    end
      (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,ntins,pass)
  end
end
```

In questo caso è stata aggiunta una variabile `pass`, inizialmente posta uguale a 0, che tiene conto del livello di profondità in cui ci si trova, ovvero conta il numero di volte in cui l'algoritmo è stato chiamato ricorsivamente. Ogni volta che vengono controllate le violazioni relative ad un nodo si richiama ricorsivamente la funzione di inserimento nodi aumentando la variabile `pass` di 1. Alla fine dell'algoritmo, quando `pass`=0, è necessario ricontrollare tutte le violazioni di tipo B trovate per determinare se ci sono ulteriori nodi da aggiungere e richiamare la funzione `tmesh_knot_insertion2` un'ultima volta per inserire tali nodi.

Esiste poi un'altra possibile soluzione dell'algoritmo, simile ad entrambe le strutture proposte:

```
function(M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,nodo,pass)
  (M,L,V,F)=general_node_insertion(M,L,V,F,nodo);
  for i=1:num_nodi
    if nodo(i) è stato appena inserito
        (ntins,bviol)=det_B_violations(nodo(i),ntins,bviol);
        (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,ntins,pass+1)
    else
        (ntins,bviol)=det_B_violations(nodo(i),ntins,bviol);
    end
  if pass==0
    for i=1:num_bviol
      (ntins,bviol)=det_B_violations(bviol(i),ntins,bviol);
    end
      (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,ntins,pass)
  end
end
```

Quest'ultima funzione richiama ricorsivamente la `tmesh_knot_insertion2` sia dopo il controllo delle violazioni associate ai nodi appena aggiunti sia quando viene effettuato il controllo finale. Non vengono invece inseriti (almeno inizialmente) i nodi aggiuntivi determinati dal controllo dei nodi preesistenti.
Tutte le strutture proposte rispettano le linee guida dell'algoritmo, in quanto per ogni nodo determinano le violazioni di tipo B ad esso associate e inseriscono tutti i nodi necessari per risolvere tali violazioni.
È stato effettuato un piccolo test sulle superfici a disposizione per stabilire se esiste una configurazione che si può considerare migliore delle altre tra le tre proposte.
È stato analizzato l'inserimento del nodo $(1.5, 4)$ nella T-Mesh presentata in figura 4.4.

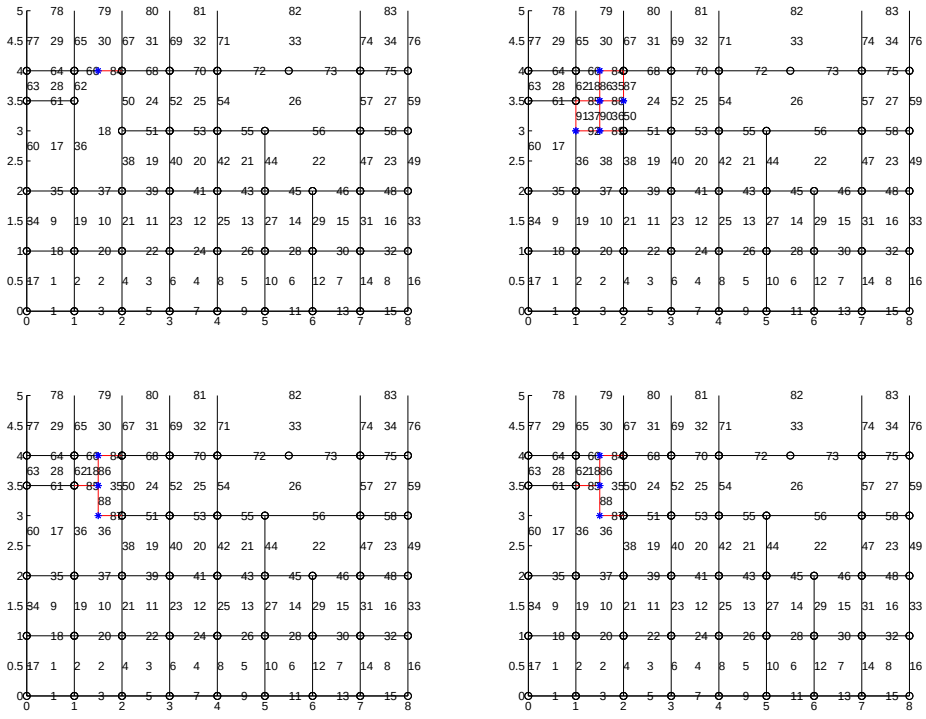

Figura 4.4: Confronto tra diverse strutture dell'algoritmo 2 di knot insertion
Fig a: T-Mesh e nodo da inserire, Fig b: Configurazione 1
Fig c: Configurazione 2, Fig d: Configurazione 3

La prima configurazione proposta, che meno si discosta dalle linee guida dell'algoritmo, inserisce 2 nodi in più rispetto agli altri due algoritmi, e mostra la tendenza a suddividere la faccia 18 in più facce rettangolari senza nodi a L.

Successivamente, sempre nella stessa superficie, è stato studiato l'inserimento del nodo $(4, 3.5)$. I tre algoritmi hanno prodotto gli stessi risultati, mostrati nelle figure 4.4b, 4.4c e 4.4d. Nell'ultimo caso proposto è stato esaminato l'inserimento del nodo $(2, 3)$ nella T-Mesh presentata in figura 4.4a. Le tre versioni hanno prodotto risultati simili, anche se la prima versione inserisce due nodi aggiuntivi rispetto alle altre, evidenziando ancora una volta

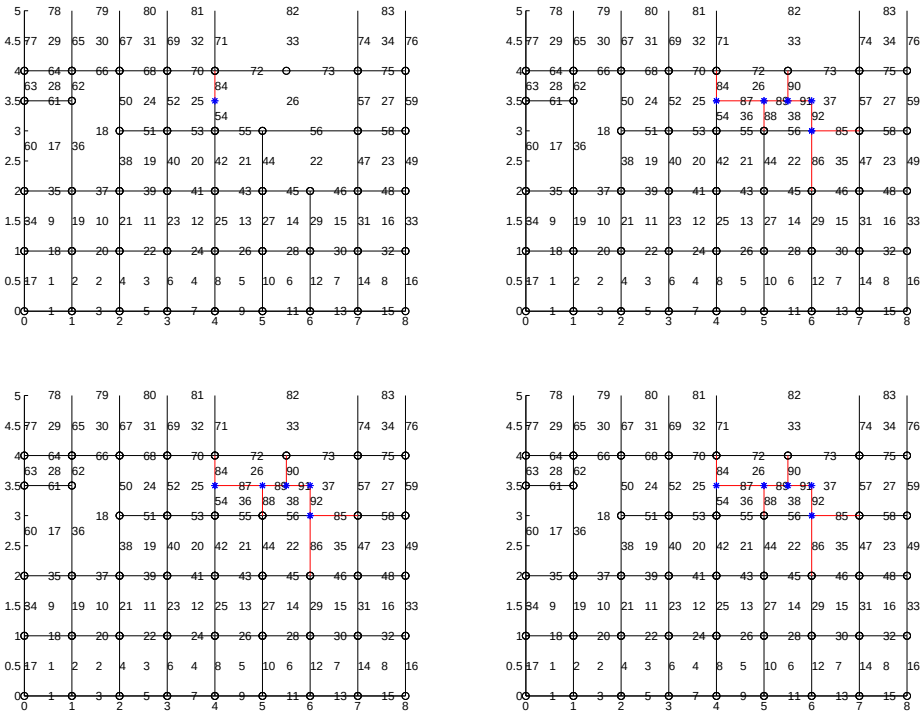

Figura 4.5: Confronto tra diverse strutture dell'algoritmo 2 di knot insertion
Fig a: T-Mesh e nodo da inserire, Fig b: Configurazione 1
Fig c: Configurazione 2, Fig d: Configurazione 3

la tendenza mostrata in precedenza.
Gli altri test effettuati non hanno mostrato differenze tra i vari algoritmi. In conclusione quindi si osserva che le ultime due versioni proposte a volte inseriscono meno nodi rispetto alla prima, che però cerca di limitare la presenza di nodi a L. In generale comunque il comportamento dei tre algoritmi risulta assai simile.

Fase geometrica

La fase geometrica dell'algoritmo è la parte in cui vengono calcolati i nuovi valori dei punti di controllo della T-Mesh. In questa fase dell'algoritmo non vengono inseriti nuovi punti di controllo. Seguendo le regole di raffinamento

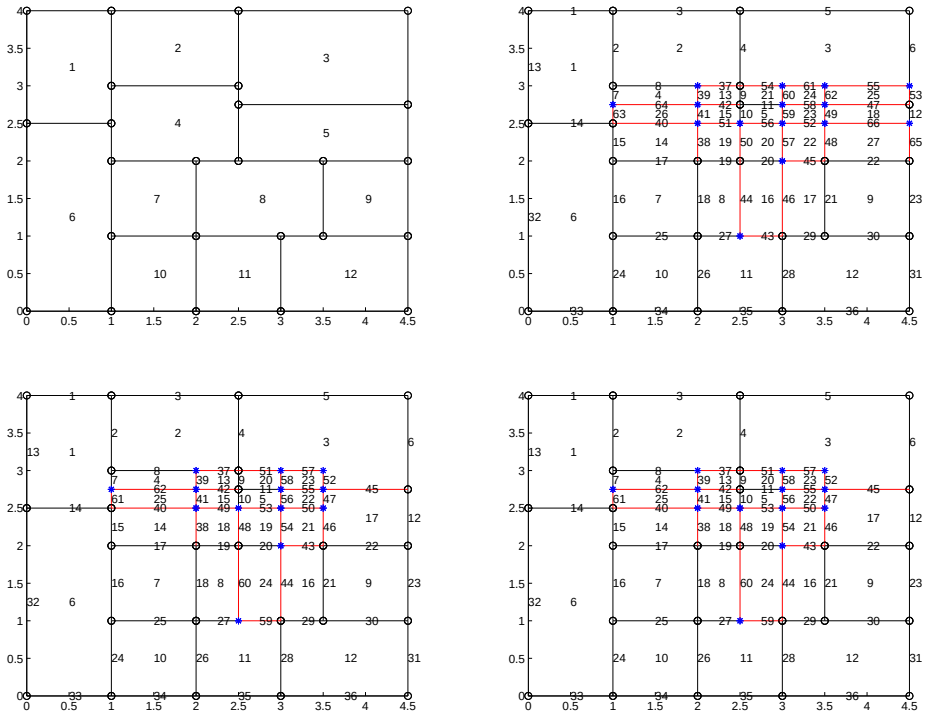

Figura 4.6: Confronto tra diverse strutture dell'algoritmo 2 di knot insertion
Fig a: T-Mesh e nodo da inserire, Fig b: Configurazione 1
Fig c: Configurazione 2, Fig d: Configurazione 3

di funzioni blending (capitolo 3.3) i nuovi punti di controllo presenti nella T-Mesh vengono scritti come combinazione lineare dei vecchi.
L'algoritmo prevede la modifica di tutti e soli i nodi coinvolti nella fase topologica. Tali nodi sono quelli che hanno causato violazioni di tipo A o B e che quindi hanno visto modificato il proprio dominio di influenza come conseguenza dell'inserimento di un nuovo nodo. Tutti gli altri nodi non vengono presi in considerazione, di conseguenza il loro valore rimane inalterato.
Esattamente come si è osservato nella descrizione teorica, per ogni nodo p che ha subito modifiche è necessario determinare l'insieme Q_p dei nuovi nodi q_i che hanno dominio di influenza strettamente contenuto nel vecchio dominio

di influenza di p. L'algoritmo determina quindi per ogni nodo p l'insieme Q_p dei nodi da miscelare per ottenere nuovi coefficienti di moltiplicazione che saranno utilizzati in seguito per combinare linearmente i nodi rispettando l'equazione (3.7).
L'algoritmo ha quindi la seguente struttura:

```
function(V)=calcolate_cp(M,L,V,M0,L0,V0)
  for i=1:num_nodi
      V(i,3)=0;
      V(i,4)=0;
  end
  for i=1:num_nodi_iniziali
    h=calcola_sstt(M,L,V,i);
    k=calcola_sstt(M0,L0,V0,i);
    if h!=k
        f=determina_nodi_da_combinare(M,M0,i);
        for j=1:num_nodi_f
        stnew(j)=calcola_sstt(M,L,V,f(j));
        stold(j)=calcola_sstt(M0,L0,V0,f(j));
        newcoeff=calcola_nuovi_coeff(newcoeff,f,stnew,stold,i);
        V=aggiorna_coefficienti(V,V0,i,f,newcoeff);
    end
  end
end
```

La funzione che calcola i nuovi valori dei punti di controllo prende in input le nuove matrici M, L, V aggiornate dopo la fase topologica e le vecchie matrici $M0, L0, V0$ che rappresentano la T-Mesh prima del knot insertion. La funzione restituisce poi in output la matrice V aggiornata.
All'inizio della funzione vengono azzerati i valori di tutti i punti di controllo della T-Mesh e dei loro pesi.
Il corpo principale della funzione è costituito da un ciclo che prende in esame tutti i vecchi nodi della T-Mesh. Per ognuno di questi nodi vengono

determinati i nuovi e i vecchi vettori nodali. Se tali vettori restano invariati prima e dopo la fase topologica allora l'algoritmo non modifica la riga della matrice V relativa a tali nodi. Se invece le matrici h e k, contenenti i vettori nodali prima e dopo l'inserimento dei nodi, sono diverse l'algoritmo richiama la funzione `determina_nodi_da_combinare` che costruisce un vettore `f` in cui sono elencati tutti i nodi che devono essere combinati per ottenere i nuovi coefficienti. Una volta determinato `f` viene chiamata la funzione `calcola_nuovi_coeff` che inserisce nella matrice `newcoeff` i valori dei nuovi coefficienti calcolati. Infine la funzione `aggiorna_coefficienti` aggiorna i valori dei punti di controllo della matrice V.
La funzione `determina_nodi_da_combinare` controlla se il nuovo dominio di ogni nodo della T-Mesh è contenuto nel vecchio dominio di influenza del nodo considerato. Successivamente ogni nodo con questa proprietà viene aggiunto al vettore `f`.
La funzione `aggiorna_coefficienti` invece è composta da poche righe:

```
function(V)=aggiorna_coefficienti(V,V0,i,f,newcoeff)
  for j=1:num_nodi_f
      if(f(j) è stato modificato)
      V(f(j),3)=V(f(j),3)+V0(i,3)*newcoeff(j);
      V(f(j),4)=V(f(j),4)+V0(i,4)*newcoeff(j);
      end
  end
end
```

Il ciclo principale della funzione scorre tutti gli elementi di `f`. Se il coefficiente associato al nodo `f(j)` è stato modificato si aggiorna il valore `V(f(j),3)` aggiungendo `V0(i,3)*newcoeff(j)`. In questo modo il valore del punto di controllo è ottenuto esattamente come nell'equazione (3.6). Per i pesi il procedimento è identico.
La funzione più importante è la `calcola_nuovi_coeff`, che determina i coefficienti moltiplicativi `newcoeff` associati ad ogni nodo. Questa funzione ha una struttura ricorsiva in cui ad ogni passo si simula l'inserimento di un nuovo

nodo che modifica i vettori nodali associati ai punti di controllo considerati. Ogni volta che si inserisce un nuovo nodo si utilizzano le formule presenti nel capitolo 3.3 per determinare i coefficienti di raffinamento che vengono poi inseriti nella matrice `newcoeff`.
L'algoritmo prosegue fino a quando non sono stati simulati gli inserimenti di tutti i nodi aggiuntivi necessari per passare dalla configurazione iniziale a quella finale.
La struttura dell'algoritmo si può quindi riassumere così:

```
function(newcoeff)=calcola_nuovi_coeff(newcoeff,f,stnew,stold,nod);
  molt=fabs(newcoeff(nod));
  [nodins,nodint]=det_nodi_da_inserire(stnew,stold,nod);
  [coefs,coeft,stold1]=calcola_coeff(nodins,nodint,stold);
  for i=1:size(coefs,1)
    for(j=1:size(coeft,1)
       l=det_nodo(coefs(i,1),coeft(j,1),stold1)
       newcoeff(l)=molt*coefs(i,2)*coeft(j,2);
       if stold1(l)!=stnew(l)
  (newcoeff1)=calcola_nuovi_coeff(newcoeff,f,stnew,stold1,l);
       else newcoeff1=newcoeff;
       end
       p=valori_mod(newcoeff,newcoeff1);
    end
  end
  newcoeff=aggiorna_newcoeff(newcoeff,p);
  return newcoeff;
end
```

Inizialmente la funzione `det_nodi_da_inserire` determina i nodi da inserire in s e in t, mentre la funzione `calcola_coeff` simula l'inserimento di tali nodi calcolando i nuovi coefficienti `coefs` e `coeft` e aggiornando la matrice `stold`. Le matrici `coefs` e `coeft` sono composte da 2 colonne: la prima contiene la coordinata parametrica del nodo inserito, mentre nella seconda

c'è il valore del coefficiente di raffinamento. Nel caso in cui non sia necessario inserire un nodo in s la matrice `coefs` contiene la coordinata s del nodo che si sta considerando e il coefficiente moltiplicativo 1. Lo stesso accade per t.
Successivamente, effettuando un ciclo sui valori di `coefs` e `coeft`, si determina il nodo di cui deve essere aggiornato il coefficiente, si stabilisce se devono essere aggiunti ulteriori nodi ed eventualmente si richiama ricorsivamente la funzione `calcola_nuovi_coeff`.
L'ultima parte della funzione memorizza i valori modificati in un vettore `p` e aggiorna correttamente il vettore `newcoeff`.

Esempio 4.3. Si consideri la T-Mesh in figura 4.3, che rappresenta il risultato dell'inserimento del nodo di coordinate $(3, 1.5)$. Il nodo p_0 di coordinate $(3, 0)$ aveva inizialmente dominio di influenza dato dal rettangolo $[1, 5] \times [-2, 2]$. I nodi che hanno dominio di influenza contenuto in quello di p_0 sono: $p_0 = (3, 0)$, $p_1 = (3, 1)$ e $p_2 = (2.5, 1)$.
L'algoritmo richiama quindi la funzione `calcola_nuovi_coeff` passando in input il vettore `f`=$[p_0, p_1, p_2]$, le matrici `stnew` e `stold` con i vettori nodali dei 3 punti considerati, il vettore newcoeff con valori uguali a -1 e `nod`=p_0.
Confrontando `stnew` e `stold` relative al nodo p_0 l'algoritmo determina che deve essere inserito solamente il nodo $(3, 1.5)$ in t. La funzione `calcola_coeff` ritorna quindi le seguenti matrici:

$$coefs = (3 \quad 1) \qquad coeft = \begin{pmatrix} 0 & 1 \\ 1 & \frac{1}{6} \end{pmatrix}$$

La matrice `stold1` è ottenuta dalla matrice `stold` modificando i vettori nodali dei nodi interessati dall'inserimento di $(3, 1.5)$ con i vettori nodali utilizzati per simulare l'inserimento. In particolare p_0 e p_1 avranno come vettore nodale in s il vettore nodale iniziale di p_0. Questo avviene perché in questo modo si riesce a determinare correttamente come vengono combinate le funzioni base.
Nel doppio ciclo vengono aggiornati i coefficienti di p_0 e p_1. Si vede poi che nel caso di p_1 il vettore nodale in s determinato dalla funzione `calcola_coeff`

non coincide col corrispondente vettore nodale nella matrice `stnew`. Di conseguenza bisogna richiamare ricorsivamente la funzione `calcola_nuovi_coeff` per simulare l'inserimento del nodo $(2.5, 1)$.
In questo caso il coefficiente `molt` ha valore $\frac{1}{6}$. L'inserimento del nodo $(2.5, 1)$ produce le matrici:

$$coefs = \begin{pmatrix} 2.5 & \frac{1}{2} \\ 3 & \frac{5}{6} \end{pmatrix} \qquad coeft = (1 \quad 1)$$

Il doppio ciclo aggiorna i coefficienti di p_1 e p_2. Non ci sono altri inserimenti da simulare quindi la funzione ritorna il vettore `newcoeff` modificato.
Quando l'algoritmo ritorna, viene aggiornato il vettore `p` e si ritorna `newcoeff`, che avrà i seguenti valori:

$$newcoeff = \begin{pmatrix} 1 & \frac{5}{36} & \frac{1}{12} \end{pmatrix}$$

Infine la funzione `aggiorna_coefficienti` calcola i nuovi termini della sommatoria.
Il procedimento viene ripetuto per tutti i nodi e si determinano così tutti i nuovi valori dei punti di controllo.

4.5 Algoritmi di raffinamento locale

Gli algoritmi di raffinamento delle superfici, come si è detto nel capitolo precedente, possono effettuare il raffinamento di un singolo punto di controllo o di una faccia. Nel primo caso si inseriscono nodi nei punti medi dei lati che hanno il punto considerato come estremo. Nel secondo caso invece si inseriscono dei nodi nei punti medi dei lati della faccia che si vuole raffinare. I due tipi di algoritmi hanno quindi una struttura estremamente simile.
Si vede di seguito l'algoritmo di face refinement:

```
function(M,L,V,F)=face_refinement(M,L,V,F,face)
  edges=determina_edges(M,L,V,F,face);
  nodes=determina_midpoints(edges);
```

```
  (M,L,V,F)=tmesh_knot_insertion2(M,L,V,F,nodes);
end
```

La funzione `face_refinement` prende in input la struttura dati della T-Mesh e un vettore contenente le facce da raffinare. Successivamente si determinano i lati di ogni faccia e si calcolano i loro punti medi. Infine viene richiamato l'algoritmo di knot insertion per inserire i nuovi punti di controllo.
L'algoritmo di knot refinement prende in input un vettore `nodo` relativo ai nodi da raffinare e richiama una funzione `determina_edges1`, analoga a quella dell'algoritmo di face refinement, che determina i lati che hanno come estremi i nodi del vettore `nodo`. Il resto dell'algoritmo è identico a quello del raffinamento di una faccia.
Gli algoritmi di knot refinement e face refinement si distinguono poi per il tipo di algoritmo di knot insertion che viene utilizzato. Come si è detto nel capitolo precedente, ci si aspetta che questi algoritmi presentino tutti i pregi e i diffetti degli algoritmi di knot insertion che utilizzano. Per verificare la correttezza di tale ragionamento è stato effettuato un confronto tra i due algoritmi di knot refinement e face refinement che utilizzano i due algoritmi di knot insertion visti.

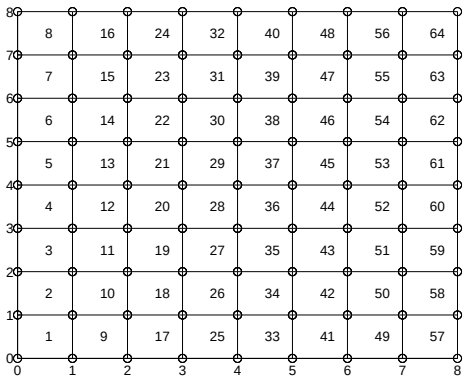

Figura 4.7: Knot Refinement: Superficie NURBS

Knot Refinement

Per confrontare i due algoritmi knot refinement, è stato riprodotto il test effettuato da W.Sederberg in [1] in cui viene raffinato l'intorno di un nodo appartenente ad una superficie NURBS. In quella situazione è stato utilizzato solamente il primo algoritmo di knot insertion, mentre in questo caso si utilizzano entrambi gli algoritmi.
Si consideri la figura 3.7, che mostra una superficie NURBS. Si raffina il no-

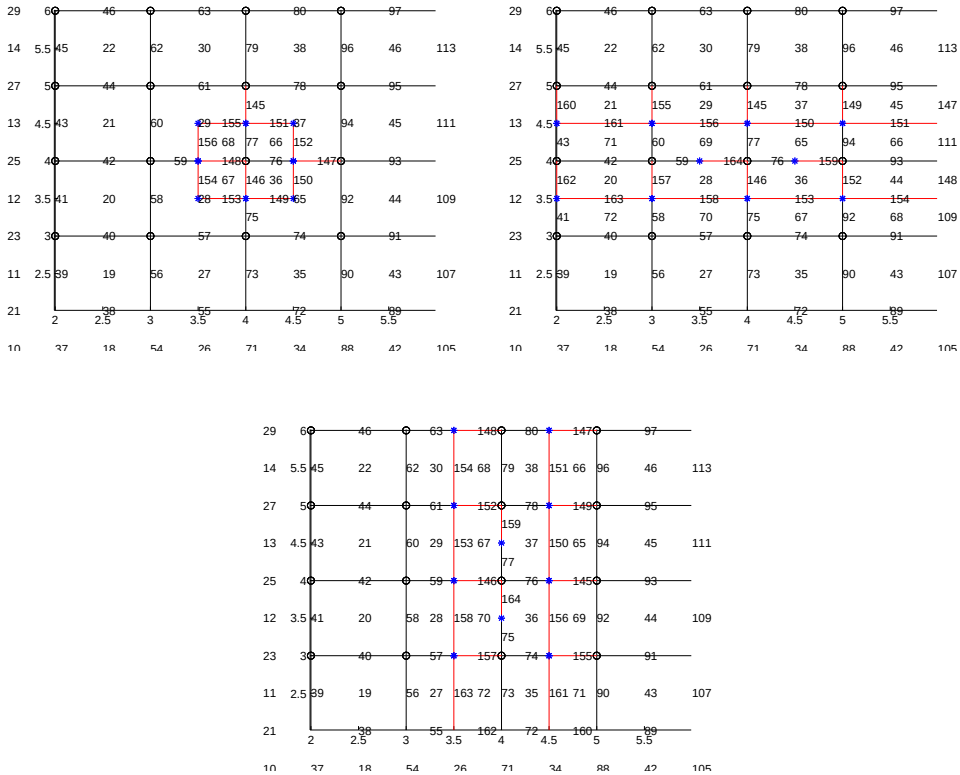

Figura 4.8: Knot Refinement nodo $(4, 4)$.
Fig a: Alg.1 ordinamento decrescente
Fig b: Alg.1 ordinamento crescente, Fig c: Algoritmo 2

do $(4, 4)$, inserendo nuovi nodi nei punti medi dei lati che hanno $(4, 4)$ come estremo. A seconda dell'ordinamento scelto per l'inserimento dei punti il primo algoritmo ottiene il risultato nelle figure 3.8a e 3.8b. Il secondo algoritmo

invece produce sempre lo stesso risultato, mostrato in figura 3.8c.
Successivamente è stato nuovamente raffinato il nodo $(4, 4)$ con entrambi gli algoritmi. I risultati ottenuti sono proposti nelle figure 3.9a, 3.9b, 3.9c. I

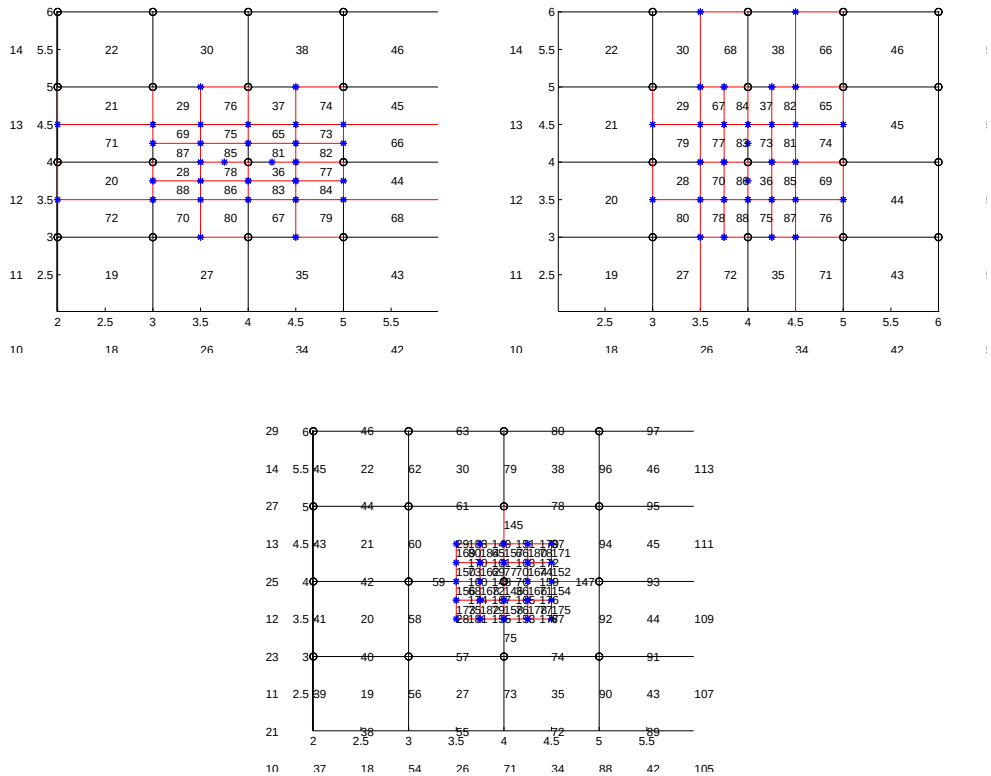

Figura 4.9: Doppio Knot Refinement nodo $(4, 4)$.
Fig a: Alg.1 ordinamento decrescente, Fig b: Alg.1 ordinamento crescente
Fig c: Algoritmo 2

risultati mostrano come il secondo algoritmo risulti migliore rispetto al primo in quanto inserisce un numero minore di nodi aggiuntivi e delimita maggiormente l'intorno del punto in cui viene modificata la struttura della griglia di controllo.

Face Refinement

Anche per il face refinement sono stati effettuati alcuni test per confrontare

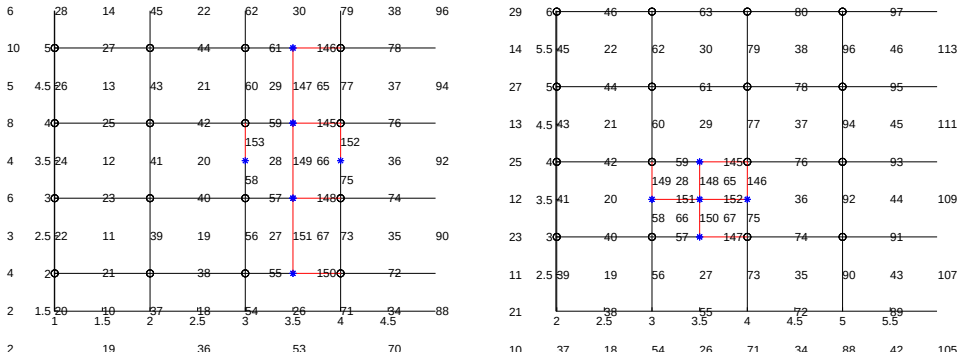

Figura 4.10: Confronto tra Algoritmo 1 e Algoritmo 2 di Face Refinement: raffinamento faccia 28

i due algoritmi di knot insertion.
Come primo esperimento si considera la superficie NURBS rappresentata in figura 3.7, di cui si raffina la faccia 28.

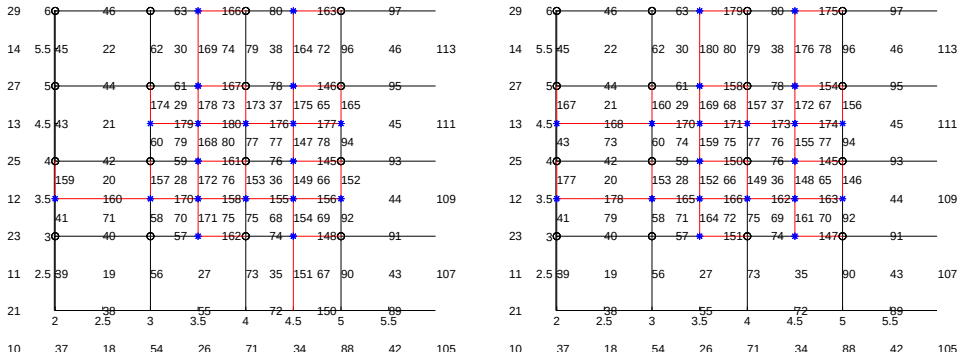

Figura 4.11: Confronto Algoritmi di Face Refinement: Fig a: Alg.1 facce 28 29 35 36 ,Fig b: Alg.1 facce 28 35 29 36

Entrambi gli algoritmi convergono ad una soluzione che lascia invariata la superficie. Le immagini mostrano la regione della T-Mesh interessata dal raffinamento. L'algoritmo 1, rappresentato a sinistra, inserisce 6 nuovi nodi nelle due facce 29 e 27, dividendole in parti uguali. L'algoritmo 2 invece

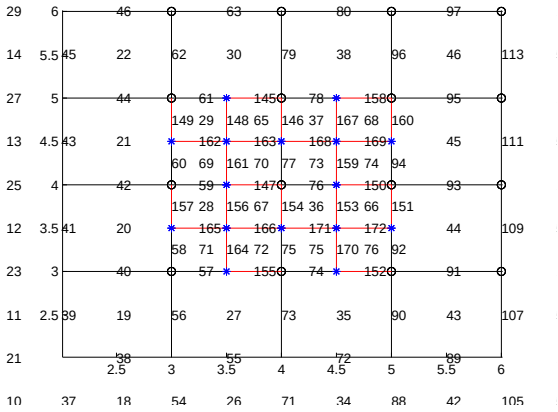

Figura 4.12: Confronto Algoritmi di Face Refinement: Algoritmo.2

inserisce 5 nodi, tutti nella faccia 28, dividendola in quattro parti uguali.
.tsp In questo esempio l'algoritmo 2 risulta migliore rispetto all'algoritmo 1 poichè inserisce solamente i punti medi dei lati e il punto centrale della faccia da raffinare.
Si consideri ancora la superficie NURBS rappresentata in figura 3.7. Si effettua ora il raffinamento delle facce $28, 29, 36, 37$ che costituiscono un *ring* attorno al nodo di coordinate $(4, 4)$.
I due algoritmi di face refinement producono i risultati proposti nelle figure 3.11 e 3.12. Per ognuno dei raffinamenti si ingrandisce l'immagine così da mostrare solamente la regione sottoposta a modifiche.
In questo caso l'algoritmo 2 modifica la T-Mesh solamente nelle facce da raffinare, mentre l'algoritmo 1 richiede l'inserimento di nodi aggiuntivi anche nelle facce adiacenti a quelle raffinate. L'algoritmo 2 risulta quindi più performante dell'algoritmo 1.
Si consideri ora invece la superficie rappresentata in figura 4.13, di cui si vogliono raffinare le facce 11 e 12. In questo caso solamente il secondo algoritmo converge alla soluzione, mentre il primo entra in un ciclo infinito.
Si può osservare come, a causa della particolarità della superficie, l'algoritmo 2 debba inserire vari nodi nelle facce adiacenti a quelle da raffinare.

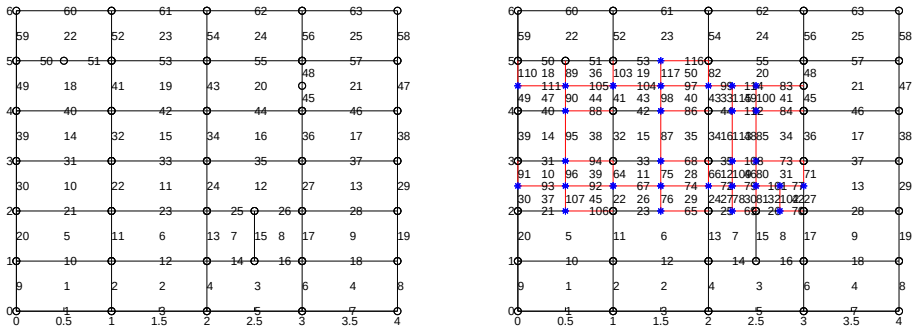

Figura 4.13: Face refinement, facce 11,12

Questi ed altri test effettuati su altre superfici dimostrano chiaramente che gli algoritmi di raffinamento che utilizzano il secondo algoritmo di knot insertion risultano più corretti ed efficienti rispetto a quelli che utilizzano il primo algoritmo. Ciò accade perché il secondo algoritmo fornisce sempre il medesimo risultato anche variando l'ordine di inserimento dei nodi e limita maggiormente la regione della T-Mesh che modifica.

Capitolo 5

Esempi di modellazione di superfici T-Spline

Gli algoritmi descritti e realizzati nei capitoli precedenti sono stati inseriti in un unico software che consente di visualizzare, modificare e interrogare curve e superfici importate da file.
È stata creata una sezione dedicata alle T-Spline che consente di caricare una superficie da un file in formato `.tsp` e di visualizzare la superficie e la sua mesh di controllo. Sono stati poi aggiunti al menù principale alcuni comandi che permettono di scegliere il lato o la faccia della T-Mesh che si desidera raffinare. Selezionando una di queste opzioni vengono richiamati i corrispondenti algoritmi di raffinamento, viene modificata la superficie e viene visualizzata la nuova superficie e la sua griglia di controllo aggiornata. È possibile ripetere il procedimento più volte ottenendo raffinamenti successivi della superficie.
Nella prima parte del capitolo si spiega brevemente il funzionamento del software dedicato alle T-Spline, evidenziando alcuni degli accorgimenti adottati per ottenere informazioni non presenti nel file `.tsp`.
Nella seconda parte invece si mostrano alcuni esempi di raffinamento di superfici T-Spline effettuati utilizzando il software prodotto.

5.1 Software di modellazione di superfici T-Spline

La sezione dedicata alle superfici T-Spline legge un file in formato `.tsp` (vedi appendice A) e inserisce le informazioni contenute nel file in una struttura `Model`. Per ogni vertice vengono memorizzate in un vettore le coordinate x, y, z, w dei punti di controllo. In un altro vettore vengono poi inserite alcune informazioni relative al vertice come ad esempio la valenza, l'indice della faccia di riferimento e, se il punto di controllo è una T-junction, l'indice della faccia per la quale il vertice è una T-junction.
Per ogni lato sono salvati gli indici associati ai vertici estremi, l'intervallo nodale ad esso assegnato e gli indici delle facce che contengono il lato.
Per ogni faccia sono memorizzati gli indici dei vertici e dei lati che la compongono e un valore intero che consente di determinare se la faccia è interna, di bordo o esterna alla T-Mesh.
Le informazioni memorizzate nella struttura `Model` vengono mantenute durante tutto l'utilizzo del programma. Tali informazioni sono necessarie ma non sufficienti per poter valutare e raffinare la superficie, per questo motivo viene creata una ulteriore struttura `Mesh` in cui vengono inizialmente inserite solo le informazioni relative ai vertici, ai lati e alle facce interne o di bordo della superficie. Successivamente viene creata un'orientazione della superficie, dividendo coerentemente i lati in orizzontali e verticali. Sfruttando l'orientazione della superficie, vengono poi memorizzate le informazioni sui nodi interni di valenza 2 e si creano i vettori nodali associati ai punti di controllo della T-Mesh. Infine vengono salvate nella struttura `Mesh` anche tutte le informazioni di adiacenza tra i vertici, i lati e le facce che costituiscono la T-Mesh.
Quando la struttura `Mesh` è completa, per poter visualizzare la superficie viene creata una struttura `Patches`, formata da un vettore di strutture `Patch`. Le strutture `Patch`, che sono tante quante le facce della superficie, contengono al loro interno tutte le informazioni necessarie per la valutazione della

faccia a cui si riferiscono.
L'algoritmo di valutazione valuta singolarmente tutte le facce della superficie. Per ogni faccia si crea una discretizzazione del dominio parametrico e si utilizza un particolare algoritmo che determina i punti della discretizzazione che sono interni alla faccia. Successivamente per ognuno dei punti della discretizzazione si effettua la valutazione e si aggiorna una struttura `Model` (differente da quella utilizzata per memorizzare le informazioni del file .tsp) creata solamente per la visualizzazione grafica. Quando tutte le facce sono state valutate viene rappresentata graficamente la struttura `Model` ottenuta.

Come si è detto in precedenza, per poter modellare la superficie sono state aggiunte al menù principale del software alcune opzioni che consentono all'utente di scegliere i lati o le facce da raffinare.
Dopo che sono state determinate le facce o lati da modificare, analogamente a quanto accade per la valutazione, le informazioni contenute nella struttura `Model` vengono copiate nella struttura `Mesh`, che viene immediatamente aggiornata e completata. Successivamente viene costruita una struttura `Patch` contenente le informazioni relative alla regione della superficie coinvolta nel raffinamento, viene creata la struttura dati necessaria per modificare tale regione e viene raffinata la superficie.
Infine vengono aggiornate le strutture `Mesh` e `Model` e viene valutata la nuova superficie, che sarà uguale alla precedente ma con una nuova mesh di controllo.
L'aggiornamento della struttura `Model` consente di effettuare più raffinamenti consecutivi su una stessa superficie e di salvare la nuova superficie ottenuta in un file `.tsp`.

5.2 Esempi di valutazione di superfici T-Spline

Il progetto realizzato consente di rappresentare graficamente sia superfici aperte che chiuse. Ad esempio si considerino le figure 5.1a e 5.1b che

mostrano un cono e una sfera. Per ogni faccia è stata creata la medesima discretizzazione del dominio parametrico.

L'algoritmo di valutazione realizzato consente poi di rappresentare grafi-

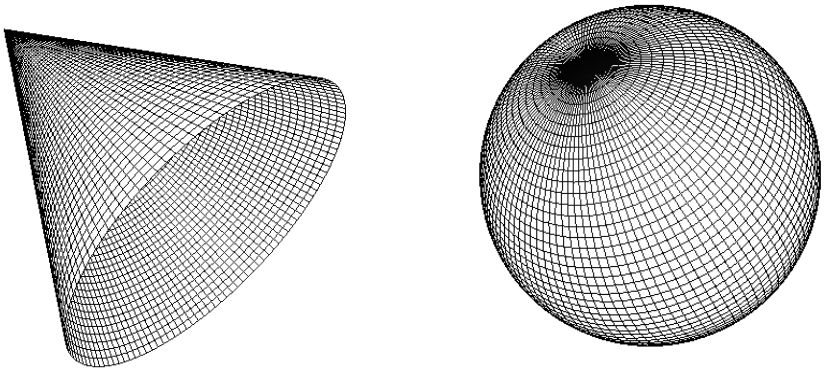

Figura 5.1: Rappresentazione di superfici T-Spline

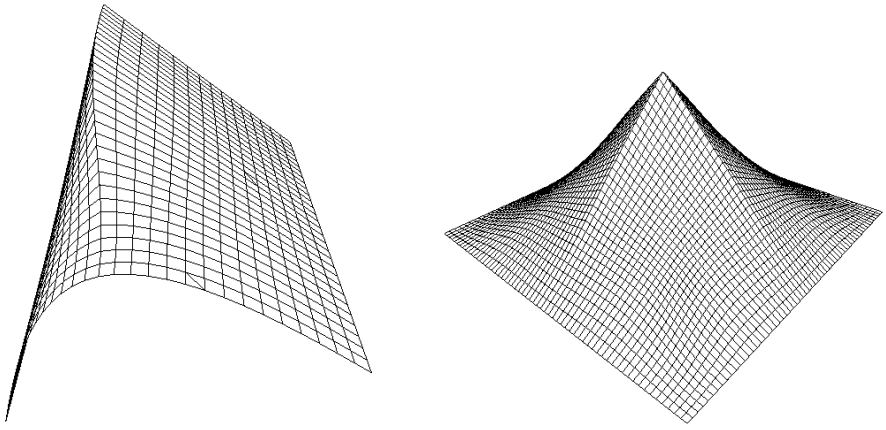

Figura 5.2: Rappresentazione di superfici T-Spline con nodi multipli

camente anche superfici con nodi di molteplicità maggiore di 1, come ad esempio le superfici presentate in figura 5.2. Si osserva che la presenza di no-

di multipli diminuisce la continuità della superficie in alcuni punti, creando creste o spigoli.

5.3 Esempi di raffinamenti di superfici T-Spline

Il software creato per modellare le superfici T-Spline consente di raffinare la superficie in due modi:

. Knot Insertion: si inserisce un nodo nel punto medio del lato selezionato.

. Face Refinement: si inseriscono dei nodi nei punti medi di tutti i lati della faccia selezionata.

Entrambi i metodi possono essere applicati a più lati o facce contemporaneamente.

Knot Insertion

L'algoritmo di knot insertion può essere applicato per aggiungere punti di controllo alla T-Mesh senza rischiare di estendere eccessivamente la regione di superficie che si vuole raffinare.
Un primo esempio di applicazione dell'algoritmo di knot insertion è presentato in figura 5.3. Nelle figure 5.3a e 5.3b sono rappresentate la mesh e la superficie iniziali. La mesh è un semplice esempio di griglia di controllo di una superficie NURBS in cui tutti i punti di controllo sono situati su uno stesso piano ad eccezione del punto centrale che è posto più in alto rispetto agli altri.
Si è scelto di inserire un nodo sul lato A, che ha come estremo il punto centrale. Osservando la T-Mesh in figura 5.3c e la corrispondente superficie in figura 5.3d si nota che l'inserimento di tale punto di controllo ha modificato le coordinate dei due estremi del lato considerato ma ha lasciato invariata la superficie.

Un esempio leggermente più complesso di utilizzo dell'algoritmo di knot

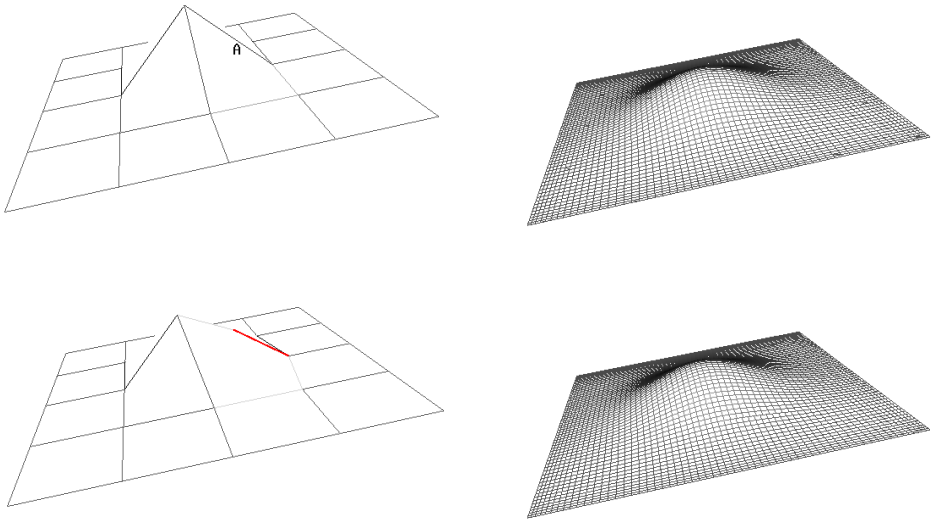

Figura 5.3: Knot Insertion: fig a: Mesh , fig b: Superficie
fig c: Mesh raffinata , fig d: Superficie raffinata

insertion è proposto nella figura 5.4 in cui si raffinano contemporaneamente i lati A e B della T-Mesh in figura 5.4a.
La nuova mesh di controllo che si ottiene, mostrata in figura 5.4c, presenta 2 nuovi vertici, 2 nuove facce e 4 nuovi lati, segnati in rosso. In questo esempio l'inserimento di più punti di controllo su lati che hanno estremi in comune ha creato un raffinamento della faccia contenente entrambi i lati. La superficie che si ottiene, proposta in figura 5.4d, è uguale a quella iniziale anche se presenta una discretizzazione del dominio parametrico differente rispetto a quella iniziale poiché sono state aggiunte nuove facce.

Face Refinement

L'algoritmo di face refinement ha l'obiettivo di suddividere le facce inserendo nuovi punti di controllo nei punti medi dei lati, creando così una mesh di controllo più fine nella regione che si vuole modellare.
Si consideri la superficie in figura 5.5, che è la stessa vista nel primo esem-

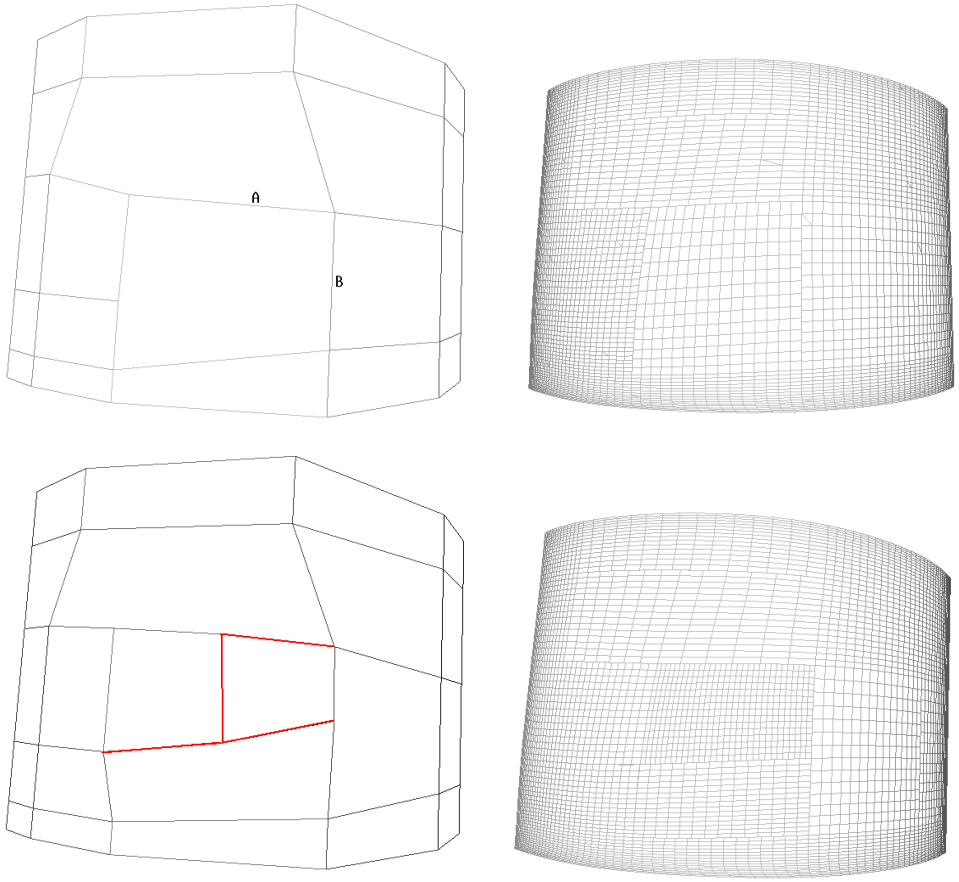

Figura 5.4: Knot Insertion: fig a: Mesh , fig b: Superficie
fig c: Mesh raffinata , fig d: Superficie raffinata

pio relativo al knot insertion. Sono state raffinate le 4 facce centrali della T-Mesh. Essendo la T-Mesh iniziale una griglia regolare il raffinamento delle facce centrali suddivide ogni faccia in 4 inserendo 4 nodi nei punti medi dei lati e un punto centrale. Non vengono inseriti nodi doppi, di conseguenza la T-Mesh che si ottiene avrà 16 nuovi punti di controllo e 12 nuove facce, come si può vedere in figura 5.5c. La superficie resta sempre invariata, anche se cambia la discretizzazione del dominio utilizzata per valutare la superficie. Successivamente è stata ulteriormente raffinata una delle facce che hanno

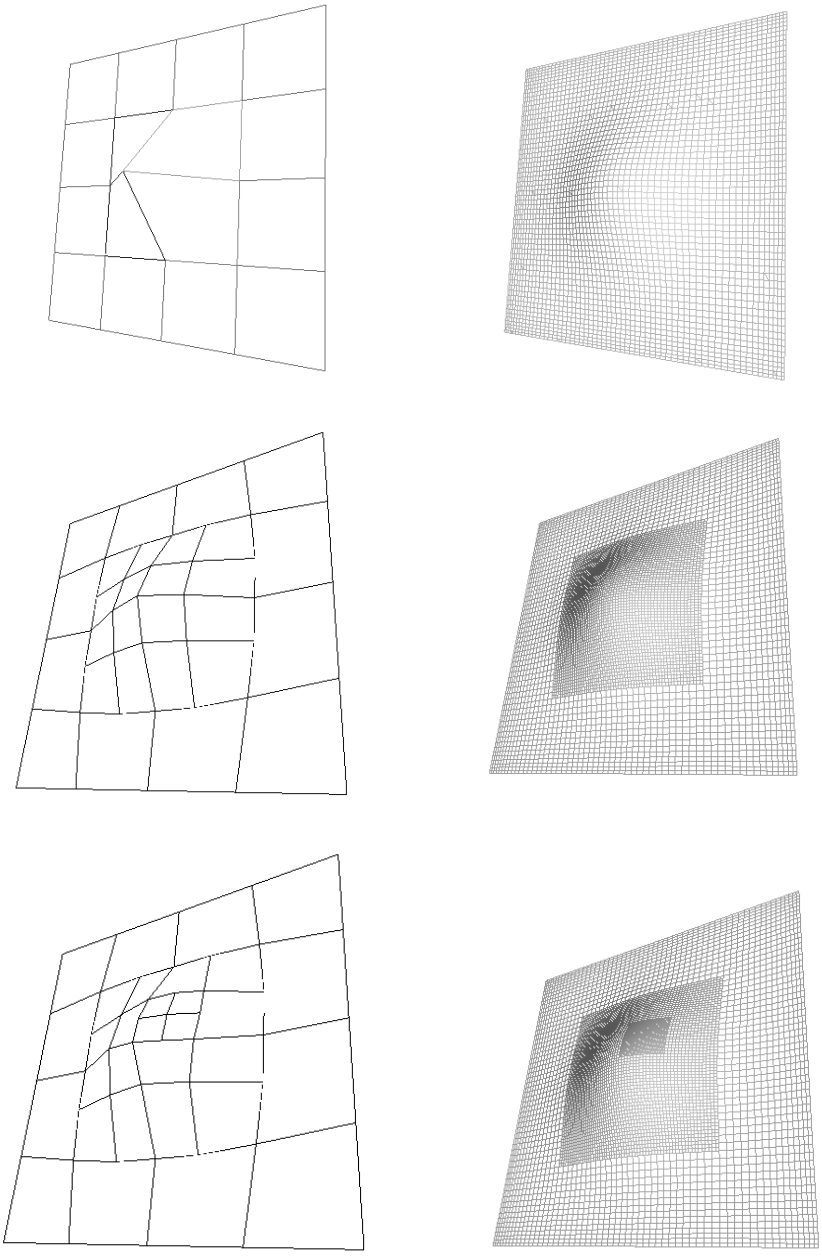

Figura 5.5: Face Refinement: fig a: Mesh , fig b: Superficie
fig c: Mesh raffinata 1 , fig d: Superficie raffinata 1
fig e: Mesh raffinata 2, fig f: Superficie raffinata 2

come vertice il punto di controllo centrale. La T-Mesh ottenuta è mostrata in figura 5.5e. L'ultimo raffinamento effettuato ha nuovamente suddiviso una faccia in 4 facce più piccole inserendo 5 nodi.
Con questo procedimento è stata ottenuta una nuova mesh di controllo, più fine di quella iniziale, che consente di modellare localmente la superficie, modificandola in una regione maggiormente limitata rispetto a quanto consentito in precedenza.

Si consideri ora la figura 5.6 in cui è presentata la superficie T-Spline vista in precedenza nel secondo esempio relativo al knot insertion. Sono state raffinate le facce A e B, mostrate in figura 5.6a, inserendo nuovi nodi nei punti medi dei lati di ognuna delle facce. La T-Mesh ottenuta, visibile in figura 5.6c, presenta 12 nuove facce che la rendono più fine nella regione interessata dal raffinamento. Come negli altri casi la superficie non è stata modificata.
Successivamente è stata raffinata ulteriormente una delle nuove facce inserite inserendo 5 nuovi nodi (uno per ogni lato e il nodo centrale) e lasciando invariata la superficie.
Osservando attentamente questo esempio e il precedente si vede che, per quanto riguarda l'inserimento di nodi interni alla faccia, l'algoritmo di face refinement ha un comportamento che dipende strettamente dalla struttura della faccia su cui si effettua il raffinamento. Ad esempio si può notare come le due facce A e B, che presentano la stessa struttura (5 lati e un vertice a T) siano state raffinate nello stesso modo. Analogamente la faccia con 4 lati raffinata nel secondo passo dell'ultimo esempio presenta la stessa struttura delle facce con 4 lati raffinate nel primo esempio. In questo caso non sono stati considerati i nodi aggiuntivi inseriti fuori dalle facce raffinate in quanto si suppone che chi effettua un raffinamento sia interessato esclusivamente alle facce che sceglie di raffinare.
Si può quindi ipotizzare che sia possibile prevedere il comportamento dell'algoritmo all'interno della faccia interessata dal raffinamento semplicemente conoscendo la struttura della faccia. Questo fatto consentirebbe di ottenere un preciso raffinamento della faccia desiderata modificandone prima la strut-

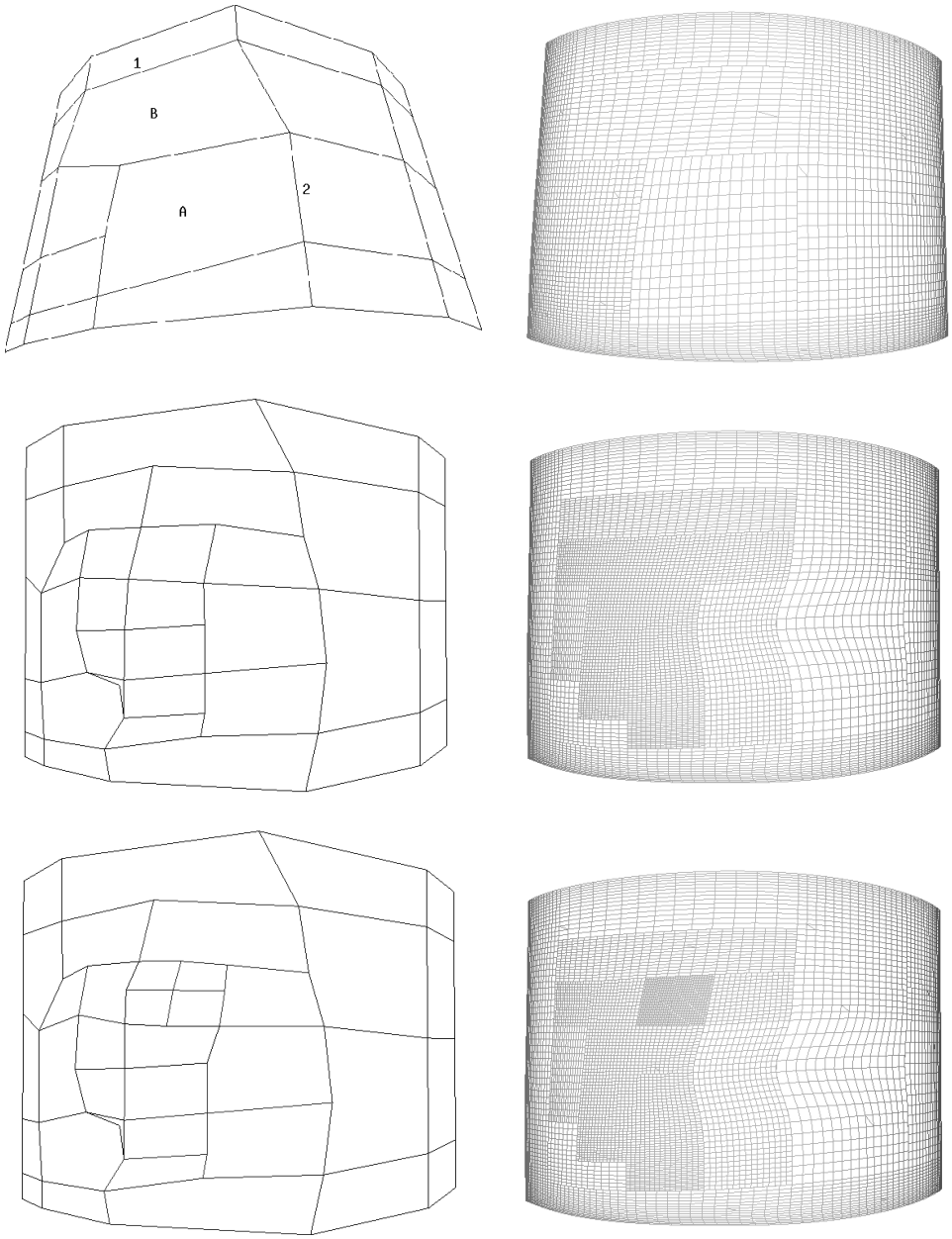

Figura 5.6: Face Refinement: fig a: Mesh , fig b: Superficie
fig c: Mesh raffinata 1 , fig d: Superficie raffinata 1
fig e: Mesh raffinata 2, fig f: Superficie raffinata 2

tura inserendo nodi con l'algoritmo di knot insertion.
Ad esempio si consideri la figura 5.7 in cui sono presentati due raffinamenti consecutivi della superficie analizzata nel caso precedente. Il primo raffinamento effettuato è un doppio knot insertion nei punti medi dei lati 1 e 2 in figura 5.6a, mentre il secondo raffinamento è un face refinement delle facce A e B della figura 5.7a.
Si vede immediatamente che il fatto di aver precedentemente raffinato la superficie inserendo 2 nodi nei punti medi dei lati 1 e 2 ha permesso di ottenere un raffinamento più preciso della stessa area di superficie mostrata in precedenza senza dover aggiungere nodi esterni alle due facce raffinate.

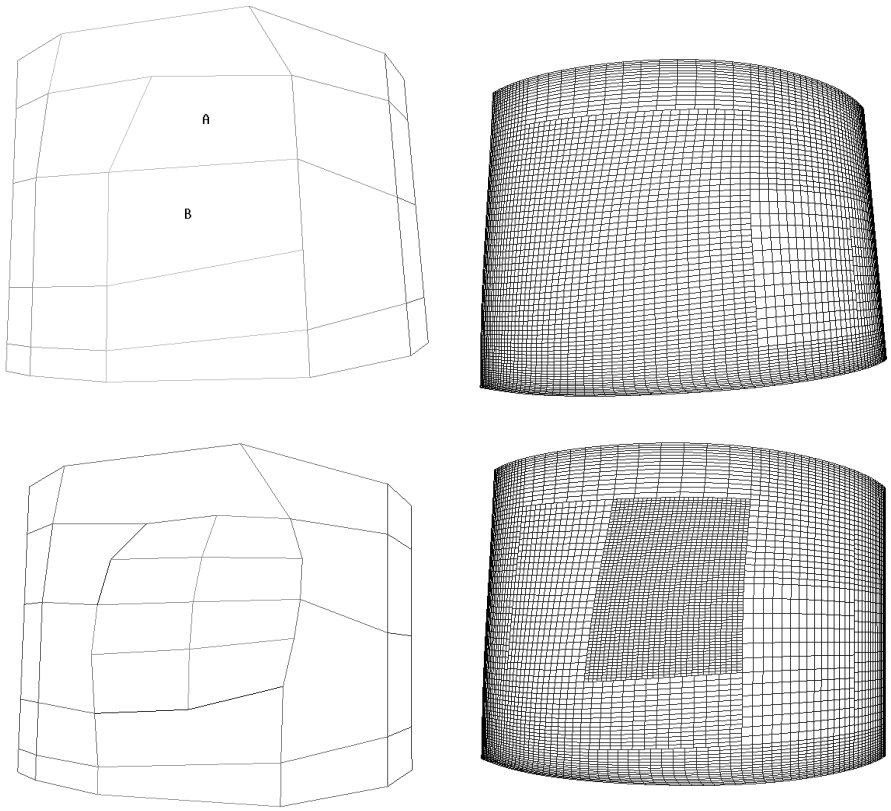

Figura 5.7: Face Ref: fig a:Mesh raffinata 1, fig b: Superficie raffinata 1
fig c: Mesh raffinata 2 , fig d: Superficie raffinata 2

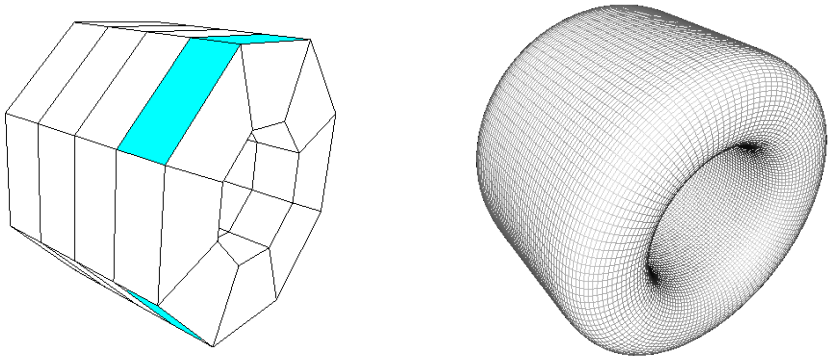

Figura 5.8: Face Ref: fig a:Mesh , fig b: Superficie

Si vede ora un ultimo esempio di modellazione di una superficie chiusa. Nella figura 5.8a è rappresentata la T-Mesh associata ad una superficie che rappresenta un bullone con sezione a corona circolare. Sono stati effettuati vari raffinamenti consecutivi con l'obiettivo di rendere più fine una regione della T-Mesh.
Il primo passo di modellazione è stato effettuato raffinando le 3 facce della T-Mesh originale (figura 5.8a) colorate in azzurro. Esattamente come negli altri esempi l'algoritmo ha raffinato ogni faccia inserendo 4 nodi nei punti medi dei suoi lati e un nodo al centro. La superficie ottenuta è mostrata in figura 5.9a.
Successivamente è stata raffinata la faccia colorata in giallo nella figura 5.9a. Il risultato ottenuto, presentato in figura 5.9c, consente di vedere che l'algoritmo ha raffinato la faccia esattamente come le altre, ma ha dovuto inserire alcuni nodi aggiuntivi, segnati con 1 e 2, per rendere valido il raffinamento.
Infine è stata raffinata la faccia colorata in verde nella figura 5.9c e il risultato ottenuto è rappresentato nella figura 5.9e. Come negli altri esempi ogni passo della modellazione ha lasciato invariata la superficie.

Gli esempi mostrati e altri test effettuati mostrano come utilizzare gli algoritmi realizzati per modellare superfici aperte o chiuse ed evidenziano le

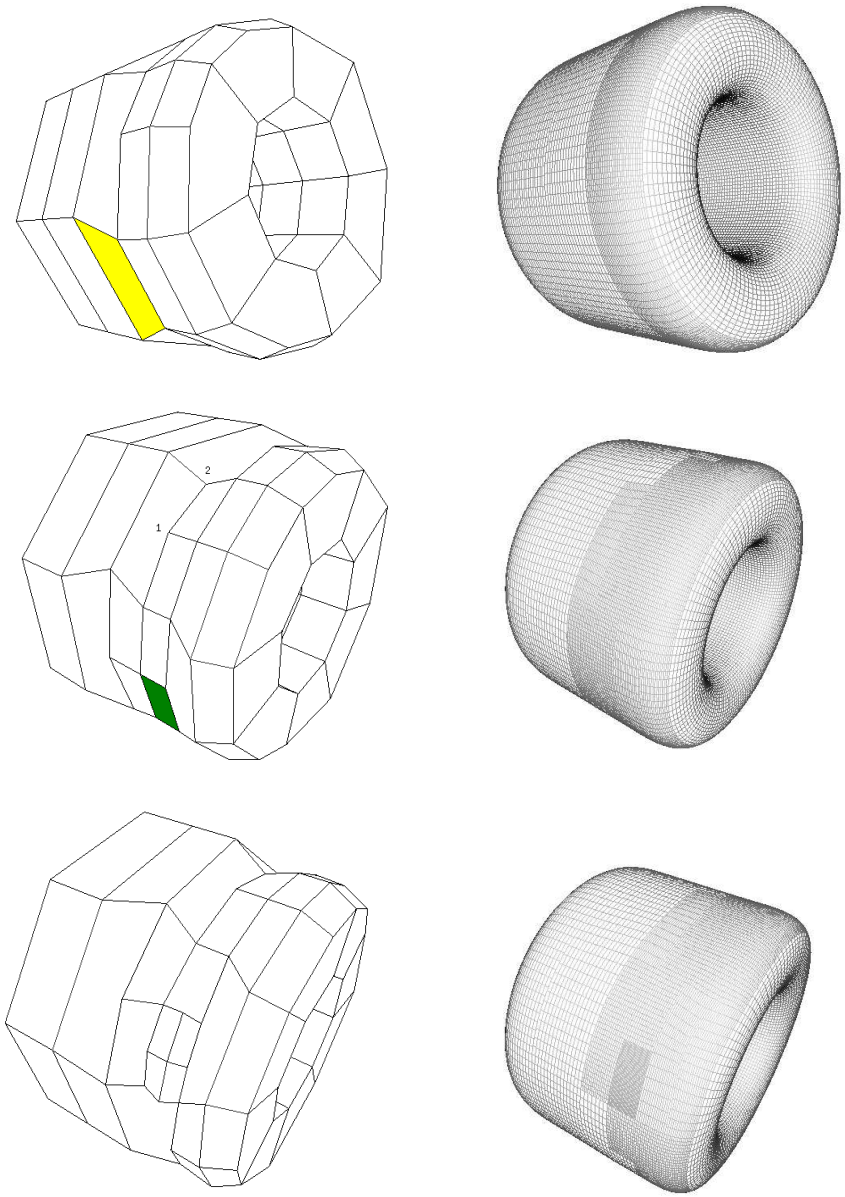

Figura 5.9: Face Ref: fig a: Mesh raffinata 1
fig b: Superficie raffinata 1
fig c: Mesh raffinata 2 , fig d: Superficie raffinata 2
fig e: Mesh raffinata 3 , fig f: Superficie raffinata 3

situazioni in cui gli algoritmi risultano più efficaci. In particolare si è visto come sia possibile utilizzare l'algoritmo di knot insertion per modificare la struttura di una faccia in modo tale da renderne successivamente più semplice il raffinamento.

Conclusioni

In questa tesi sono stati studiati e realizzati i principali metodi numerici per la visualizzazione e la modellazione di superfici T-Spline.
Le poche informazioni presenti nei documenti sulle T-Spline hanno richiesto un notevole lavoro di comprensione degli algoritmi di valutazione e knot insertion, che sono stati dettagliatamente spiegati sia dal punto di vista teorico che dal punto di vista implementativo nella tesi.
È stato creato un algoritmo di valutazione per superfici T-Spline che consente di rappresentare correttamente superfici di genere 0 o 1. Successivamente sono stati realizzati due algoritmi di knot insertion e i corrispondenti algoritmi di face refinement. Gli algoritmi sono stati testati e confrontati in modo tale da determinare l'algoritmo più efficiente.
L'ultima parte del lavoro ha riguardato la progettazione e l'implementazione di un software in linguaggio C per la visualizzazione e la modellazione di T-Spline che utilizza gli algoritmi realizzati e descritti in precedenza. Questo programma è stato utilizzato poi per creare alcuni esempi di modellazione di superfici T-Spline aperte e chiuse.
È possibile sviluppare ulteriormente questo lavoro rendendo più generali sia l'algoritmo di valutazione che il secondo algoritmo di knot insertion. In questo lavoro infatti sono state studiate solamente superfici T-Spline senza nodi straordinari con valenza maggiore di 4, di conseguenza tutte le superfici analizzate sono limitate al genere 0 o 1. Il secondo algoritmo di knot insertion è stato realizzato per superfici senza nodi a valenza maggiore di 4 e senza nodi doppi, ovvero nodi legati da un lato a cui è associato un knot interval nullo.

Nonostante le limitazioni in cui ci si è messi gli algoritmi proposti sono così generali da poter essere estesi in modo semplice anche nel caso di vertici a valenza > 4 e nodi doppi.

Appendice A

Struttura dei file *.tsp*

Il formato *.tsp* è una particolare tipologia di file ideata da Matthew T. Sederberg e Thomas Finningam e utilizzata per salvare e caricare superfici T-Spline su software commerciali come *Rhinoceros* e *Maya*.
La struttura di un documento *.tsp* deriva da quella di un file in formato *.obj* e consente di descrivere precisamente la mesh di controllo di una superficie T-Spline. Se file *.tsp* è riferito ad una superficie aperta vengono memorizzate le informazioni relative alla T-Mesh e ad un anello di facce che circondano la superficie, in modo tale da riuscire a determinare i vettori nodali associati a tutti i vertici della mesh di controllo, compresi quelli di bordo.
Un documento *.tsp* è suddiviso in varie liste che contengono la struttura della T-Mesh. Più precisamente le informazioni principali sono contenute nelle liste relative ai vertici, ai lati e alle facce della mesh. La struttura di un file *.tsp* è la seguente:

```
\sharp TS0102
subdiv sl
extractor et

-v <vx>  <vy>  <vz>  <w> <fr>  <val>

-kl  <ki1>  <ki2> ...
```

```
-l <v0>  <v1>  <f0>  <f1>  <nkl>

-f <flag>  <l0/tv0>  <l1/tv1>  <l2/tv2>  ...  <ln/tvn>
```

Le prime 3 righe contengono l'intestazione al documento. In particolare la prima riga contiene una stringa che si riferisce alla versione di file *.tsp* attualmente utilizzata. Le due righe seguenti invece contengono i parametri `sl` e `et` che determinano rispettivamente il numero di suddivisioni locali da processare e il tipo di algoritmo di computazione di superfici utilizzato.
I file utilizzati in questo lavoro di tesi hanno come parametri impostati 0 e 1.

Lista dei vertici

La sequenza dei vertici contiene le coordinate x, y, z, w e alcune informazioni relative a tutti e soli i punti di controllo della T-Mesh. Non sono considerati i vertici delle facce esterne alla T-Mesh che non appartengono alla mesh di controllo. Ogni elemento della lista è preceduto dalla stringa `-v` e ha la seguente struttura:

```
-v <vx>  <vy>  <vz>  <w> <fr>  <val>
```

I valori `vx`, `vy`, `vz` e `w` contengono le coordinate spaziali del vertice considerato e il peso ad esso associato. Il parametro `fr`, contiene l'indice della faccia di riferimento del vertice considerato, ovvero l'indice di una delle facce che contengono il vertice. L'ultimo valore `val` contiene la valenza del vertice, ovvero il numero di lati che hanno come estremo il vertice stesso.
La lista può contenere anche vertici cancellati che sono memorizzati ponendo solamente la stringa `-v` senza alcuna informazione affianco. Ad esempio un vertice cancellato si scrive come:

```
-v
```

Quando si studia un file .tsp si deve fare attenzione ai vertici cancellati perchè anche se non appartengono alla T-Mesh ad essi è associato comunque un

indice dato dalla posizione che occupano nella lista. Di conseguenza quando si analizzano i lati gli indici dei vertici devono essere calcolati correttamente considerando anche eventuali vertici cancellati.

Lista degli intervalli nodali

La lista degli intervalli nodali è costituita da un vettore di valori reali preceduti dalla stringa `-kl`. Ogni valore corrisponde ad un intervallo nodale che può essere associato ad uno o più lati della T-Mesh. Un esempio di vettore è il seguente:

```
-kl 0  1  0.5
```

In questo caso i lati della T-Mesh possono avere intervallo nodale 0,1 o 0.5. È possibile memorizzare più volte uno stesso intervallo nodale, inoltre se sono presenti più di 5 intervalli nodali i valori vengono memorizzati su più righe e ogni riga può contenere al massimo 5 intervalli nodali.

Lista dei lati

La sequenza relativa ai lati contiene le informazioni strettamente necessarie per determinare la posizione di ogni lato nella T-Mesh. Ogni elemento della lista può essere scritto quindi come:

```
-l <v0>  <v1>  <f0>  <f1>  <nkl>
```

Gli elementi `v0` e `v1` sono gli indici associati ai veritici estremi del lato. I valori `f0` e `f1` sono gli indici associati alle facce che contengono il lato. Il valore `nkl` contiene l'indice assegnato all'intervallo nodale del lato nel vettore `-kl`.
Se il documento è relativo ad una superficie aperta nella lista dei lati sono memorizzati tutti i lati della T-Mesh e tutti i lati delle facce che costituiscono un anello attorno alla superficie. I lati delle facce esterne hanno almeno un vertice posto uguale a -1 per indicare che un estremo è dato da un vertice esterno alla mesh.

Lista delle facce

Le facce della mesh di controllo sono descritte utilizzando i lati che le costituiscono. La riga relativa ad una faccia di 4 lati è la seguente:

```
-f <flag>  <l0/tv0>  <l1/tv1>  <l2/tv2>  <l3/tv3>
```

Il valore di `flag` determina il tipo di faccia che si sta considerando: le facce interne alla mesh sono memorizzate con il valore 0, le facce di bordo con il valore 2 mentre le facce appartenenti all'anello esterno sono memorizzate con il valore 6.
Gli indici `l0,l1,l2,l3` sono gli indici dei lati che formano la faccia. I parametri `tv0,tv1,tv2,tv3` assumono valore 1 se il primo vertice del lato a cui si riferiscono è una giunzione a T per la faccia che si sta considerando, mentre invece sono posti uguali a 0 se il vertice non è una giunzione a T. Nel caso di una faccia a 4 lati i parametri `tv0,tv1,tv2,tv3` sono necessariamente uguali a 0, mentre invece se la faccia ha più di 4 lati è quasi sempre presente almeno un valore posto uguale a 1.

In alcuni file *.tsp* vengono memorizzate alcune informazioni aggiuntive come ad esempio le coordinate u, v della texture da applicare alla superficie. Tali coordinate vengono memorizzate nella lista `-vt` che ha la seguente struttura:

```
-vt <u>  <v>
```

Nel caso in cui siano presenti le informazioni relative alla texture nella lista delle facce viene memorizzato anche un indice relativo ad un elemento del vettore `-vt` da associare al primo vertice del lato che si sta considerando.

Bibliografia

[1] T.W.Sederberg, J.Zheng, A.Bakenov, A.Nasri; *T-Spline and T-NURCCs*, 2003, ACM Transactions on Graphics - TOG, vol. 22, no. 3, pp. 477-484

[2] T.W.Sederberg, J.Zheng, T.Lyche; *T-Spline Simplification and Local Refinemnent*, 2004, ACM Transactions on Graphics - TOG , vol. 23, no. 3, pp. 276-283

[3] A.Krishnamurthy, Y.Yasui; *Efficient Rendering of NURBS and T-Spline surfaces*,

[4] T.W.Sederberg, X.Li, H.Lin, H.Ipson, G.T.Finnigan *Watertight Trimmed NURBS*, 2008, ACM Transactions on Graphics - TOG , vol. 27, no. 3

[5] T.J.R.Hughes, J.A.Cottrell, Y.Bazilevs; *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, 2005, Computer Methods in Applied Mechanics and Engineering, vol. 194, no. 39, pp. 4135-4195

[6] T.W.Sederberg, T.J.R.Hughes, X.Li, J.Zheng, M.A.Scott; *On linear independence of T-Spline blending functions*, 2011 Computer Aided Geometric Design - CAGD

[7] F.Roman; *T-Spline: un passo oltre le NURBS* 2011 Tesi di laurea magistrale, Università di Torino

[8] X.Li, J.Deng, F.Chen; *Polynolmial splines over general T-Meshes*, 2010, The Visual Computer - VC , vol. 26, no. 4, pp. 277-286

[9] X.Li, H.Wang, Y.He, X.Gu, H.Qin; *Geometry-aware domain decomposition for T-Spline-based manifold modeling*, 2009, Computers & Graphics - CG , vol. 33, no. 3, pp. 359-368

[10] T.W.Sederberg, M.T.Sederberg; *T-Splines: A technology for Marine Design with Minimal Control Points*, 2010, Chesapeake Powerboat Symposium, Annapolis

Ringraziamenti

Ringrazio infinitamente il professor Casciola per la grande disponibilità e cortesia con cui mi ha guidato in questo lavoro e per tutto l'aiuto che mi ha fornito durante la stesura di questa tesi.
Ringrazio Michele Antonelli che mi ha aiutato a superare alcune fasi assai complicate del mio lavoro e non mi ha mai fatto mancare la sua disponibilità.
Ringrazio la mia famiglia e tutti i miei amici che mi hanno sopportato e supportato nei momenti difficili che ho vissuto in questi mesi.
Ringrazio infine Carlotta che mi ha sempre sostenuto nei momenti più difficili, aiutandomi ad andare avanti e ad affrontare e superare tutti i problemi che ho incontrato.