Algoritmi di tipo De Casteljau per curve e superfici di Bézier razionali

PROGETTO PER L'ESAME DI NUMERICAL METHODS FOR GRAPHICS

DI ALBERTO BILIOTTI

MATRICOLA: 7109894

Scopo del progetto

In questo progetto è stato deciso di implementare un algoritmo tipo-De Casteljau in grado di valutare uno specifico punto di una curva razionale di Bézier.

Successivamente tale algoritmo è stato sfruttato per poter valutare anche punti sulle superfici razionali di Bézier nello spazio sfruttando le proprietà delle superfici tensor-product.

Per fare ciò mi sono basato sul paper: «On de Casteljau-type algorithms for rational Bézier curves» di Zbyněk Šír e Bert Jüttler.

Perché?

Usare un algoritmo di tipo-De Casteljau è utile al fine di rappresentare le curve razionali di Bézier in quanto permette di calcolare i punti della curva razionale mantenendo i principali vantaggi derivanti dall'utilizzo dell'algoritmo di De Casteljau «base» tra cui, in particolare, il fatto di essere numericamente stabile e la relativa semplicità di implementazione.

Dati in input

Il lavoro da me analizzato descrive un algoritmo che prende in input, oltre ai punti che costituiranno il polinomio di controllo della curva desiderata ed il parametro t in cui valutare la curva, al posto del vettore contenente i pesi dei punti, una sequenza di n fattori lineari del tipo:

$$L_i(t) = a_i(1-t)+b_i t$$

Il cui prodotto di tutti gli n fattori in t si dimostra essere nient'altro che la somma dei pesi moltiplicati per i polinomi di Bernstein corrispondenti valutati in t.

Il mio codice prende quindi in input due vettori contenenti i termini a e b dei coefficienti lineari (che avranno lunghezza pari al numero di punti di controllo desiderati meno uno) definiti in precedenza da cui successivamente andrò a ricavare il valore dei pesi corrispondenti (solo per maggior chiarezza, potrei fare anche a meno di calcolarli in quanto non necessari per l'esecuzione) sfruttando la ricorrenza:

$$w_i^n = a_i \frac{n-i}{n} w_i^{n-1} + b_i \frac{i}{n} w_{i-1}^{n-1}$$

Con $w_0^0 = 1$ e $w_i^n = 0$ se $i < 0$ o $i > n$

Funzionamento dell'algoritmo

L'algoritmo, similmente a quello di De Casteljau, calcola il valore della curva razionale c in t partendo dagli n+1 punti di controllo:

$P_i^0(t) = P_i$ per i $= 0, \ldots,$ n

E ad ogni passo restituisce un insieme di punti, con un numero di elementi minore di uno rispetto alla precedente iterazione, ricavati con la formula:

$P_i^j(t) = a_j \frac{1-t}{L_j} P_i^{j-1}(t) + b \frac{t}{L_j} P_{i+1}^{j-1}(t)$ per j = 1, . . . , n e i = 0, . . . , n − j.

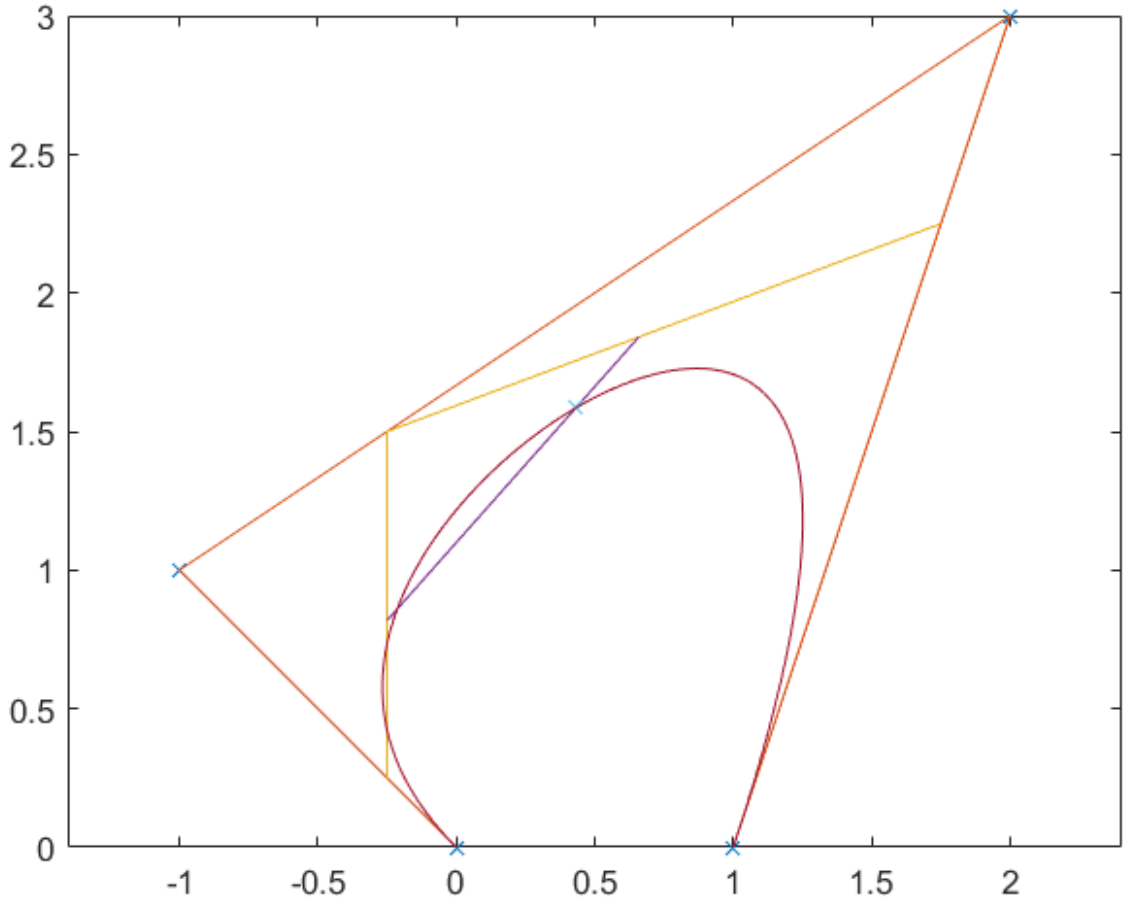
Fino ad arrivare al passo n nel quale l'unico punto rimasto non sarà altro che il valore della curva razionale di Bézier in t:

$P_0^n(t) = c(t)$

Possiamo notare come l'algoritmo perda la proprietà di tangenza tipica dell'algoritmo di De Casteljau «tradizionale», ossia l'ultimo segmento che connette i due punti generati dalla penultima esecuzione non è più, in generale, tangente alla curva in esame.

Ciò comporta anche la perdita della possibilità di usufruire dell'algoritmo di subdivision, visto per le curve di Bézier, che sfruttava i punti generati durante l'esecuzione dell'algoritmo di De Casteljau tradizionale.

Questo esempio è identico a quello riportato nello studio in analisi.



$t = 0.5$

Coefficienti lineari: $L_1(t) = 3(1-t) + t, L_2(t) = 6(1-t) + 5t, L_3(t) = (1-t) + 3t$

Pesi corrispondenti: $w_o = 18, w_1 = 25, w_2 = \frac{68}{3}, w_3 = 15$

Algoritmo per le superfici razionali di Bézier

Come anticipato, partendo dall'algoritmo per la valutazione delle curve razionali visto in precedenza, è possibile descrivere un algoritmo in grado di calcolare iterativamente i punti di una superficie razionale di Bézier.

Per fare ciò è necessario fornire in input, oltre al net dei punti di controllo (contenuto in una matrice bidimensionale in quanto sarà il codice stesso a dividere le righe) e i parametri u e v in cui valutare la superficie, due matrici le cui righe contengono i termini a e b utili per definire i pesi di ogni punto rispetto alla propria riga del net di controllo oltre a due vettori che conterranno i termini dei coefficienti utili per assegnare un peso ad ogni riga.

La mia implementazione calcolerà quindi i pesi delle righe e dei relativi punti stampandoli a video e salvandoli in una matrice chiamata W (come nell'algoritmo per le curve questo passaggio non è utile per l'esecuzione ma può aiutare per comprendere meglio i risultati).

Per calcolare in maniera ottimale il valore dei pesi è però necessario che i coefficienti lineari diano luogo ad una rappresentazione dei pesi dei punti nelle righe e delle righe stesse in forma standard, ossia con i due pesi «estremi» di uguale valore (per fare ciò è necessario che i due vettori dei termini dei coefficienti lineari contengano gli stessi valori ma in ordine inverso).

Funzionamento dell’algoritmo per le superfici

L’algoritmo, per calcolare i punti di una superficie razionale di Bézier, non farà altro che applicare l’algoritmo di tipo De Casteljau visto in precedenza per le curve su ogni riga, con i coefficienti lineari specificati, per poi calcolare il punto desiderato sfruttando un’ultima volta l’algoritmo di tipo De Casteljau sui punti calcolati prima con i coefficienti lineari che rappresentano i pesi associati ad ogni riga del net di controllo.

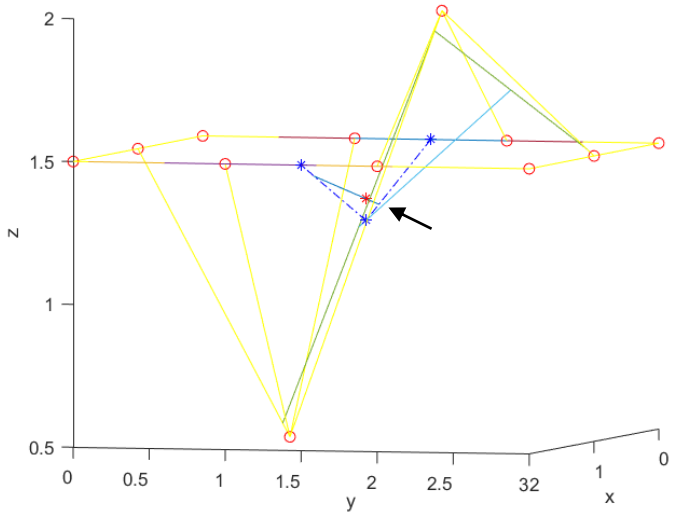
Questo procedimento è reso possibile dal fatto che stiamo rappresentando una superficie di tipo tensor-product. Questo ci permette effettuare la valutazione dei punti della superficie applicando l’algoritmo per la valutazione delle curve lungo le dimensioni da noi desiderate. Nel nostro caso è stato quindi scelto di effettuare prima solo la valutazione per righe ed in fondo applicare un’unica volta l’algoritmo lungo l’unica colonna di punti rimasta.

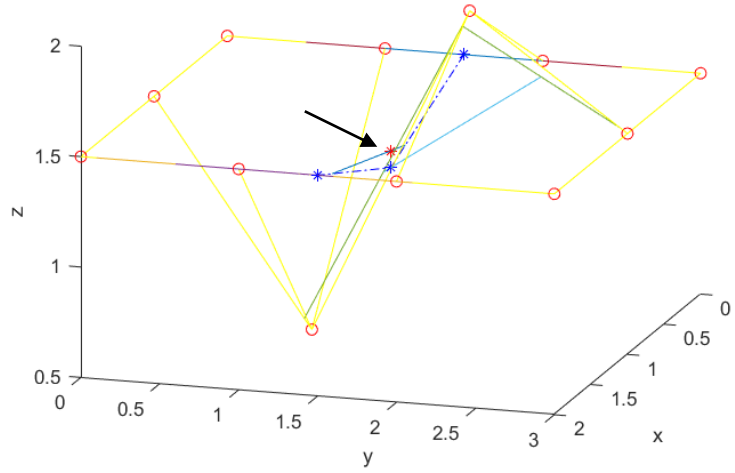
Esempio di applicazione dell'algoritmo su un net di controllo composto di 9 punti con pesi e parametri indicati di seguito:

```
xyz=[0 0 1.5; 0 1 1.5; 0 2 1.5; 0 3 1.5;
     1 0 1.5;  1 1 0.5; 1 2 2; 1 3 1.5;
     2 0 1.5; 2 1 1.5; 2 2 1.5; 2 3 1.5];
```

```
an=[1,1,1;1/2,10,10;3,2,1/3]
bn=[1,1,1;10,10,1/2;1/3,2,3]
am=[1/2,2];
bm=[2,1/2];
```

```
Peso righe del net dei punti di controllo:
    1.0000
    2.1250
    1.0000

Peso dei punti del net dei punti di controllo per riga:
    0.2500    0.2500    0.2500    0.2500

    0.0624    0.4376    0.4376    0.0624

    0.1144    0.3856    0.3856    0.1144

Risultato:
    1.0000    1.5000    1.3377
```





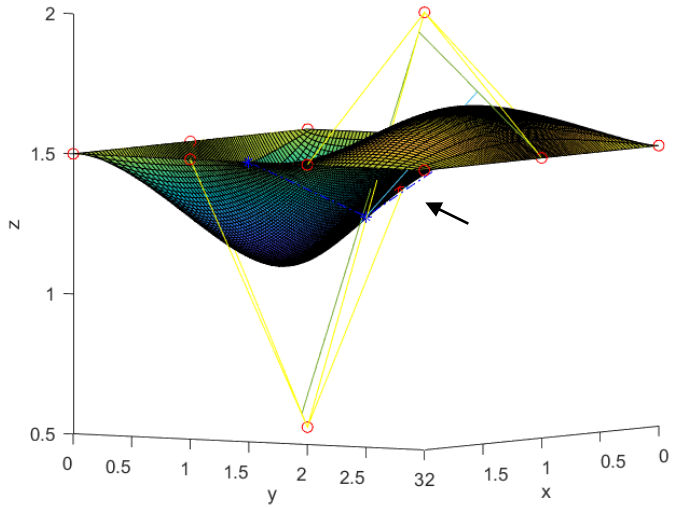
$u = 0.5, \qquad v = 0.5$

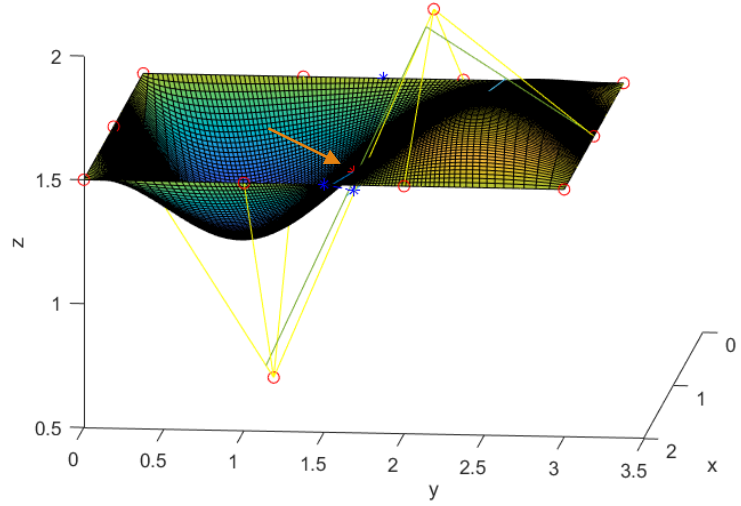
Esempio di applicazione dell'algoritmo su un net di controllo composto di 9 punti con pesi e parametri indicati di seguito con anche il resto della superficie disegnata:

```
xyz=[0 0 1.5; 0 1 1.5; 0 2 1.5; 0 3 1.5;
     1 0 1.5;  1 1 0.5; 1 2 2; 1 3 1.5;
     2 0 1.5; 2 1 1.5; 2 2 1.5; 2 3 1.5];
```

```
an=[1,1,1;1/2,10,10;3,2,1/3]
bn=[1,1,1;10,10,1/2;1/3,2,3]
am=[1/2,2];
bm=[2,1/2];
```

```
Peso righe del net dei punti di controllo:
    1.0000
    2.1250
    1.0000

Peso dei punti del net dei punti di controllo per riga:
    0.2500    0.2500    0.2500    0.2500

    0.0624    0.4376    0.4376    0.0624

    0.1144    0.3856    0.3856    0.1144

Risultato:
    1.0000    1.5000    1.3377
```





$u = 0.5, \qquad v = 0.5$

Conclusioni

Dalle slide mostrate in precedenza è possibile osservare come entrambi i codici permettano di valutare in maniera numericamente stabile e relativamente rapida e semplice le curve o le superfici di Bézier, pur non permettendo di passare in input direttamente i valori dei pesi desiderati per ogni punto del polinomio o del net di controllo.

Tuttavia è comunque possibile conoscere i pesi assegnati ai punti dati i coefficienti lineari passati in input in modo da poter comunque modellare con relatività semplicità la curva o la superficie razionale a nostro piacere.

Lo studio preso in esame suggerisce inoltre che in futuro questo metodo potrebbe essere sia esteso per valutare patch triangolari razionali sia ampliato con lo sviluppo di un metodo simile per poter calcolare punti di curve o superfici B-Spline razionali.