

Bézier & B-Spline Toolbox

Documentazione esterna

App MATLAB per il calcolo di curve di Bézier e Spline

Elaborato di Calcolo Numerico (A.A. 2019/20)

Prof.ssa: D'Amore Luisa

Studenti: Castaldo Alessandro mat. M63/0946
Cesarano Carmine mat. M63/0948
Allocca Roberto mat. M63/1045



UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Bézier & B-Spline Toolbox

Documentazione esterna

Il toolbox BBS (Bézier & B-Spline) è un'applicazione MATLAB progettata ed implementata nell'ambito dell'elaborato d'esame di Calcolo Numerico. Il software permette di utilizzare e comparare vari metodi per la costruzione di curve parametriche mediante l'utilizzo di una libreria di routine MATLAB sviluppata ad-hoc.

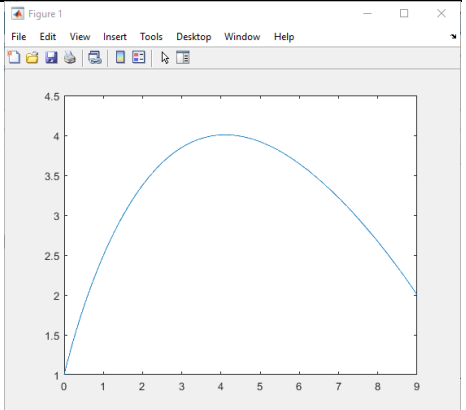
In particolare, tramite un interfaccia grafica user-friendly, è permessa all'utente la scelta tra vari metodi:

- Bézier CUSTOM – costruzione di una curva di Bézier con approccio personalizzato
- Spline CUSTOM – costruzione di una curva Spline con approccio personalizzato
- Bézier MATLAB – costruzione di una curva di Bézier con routines MATLAB
- Spline MATLAB – costruzione di una curva di Spline con routines MATLAB

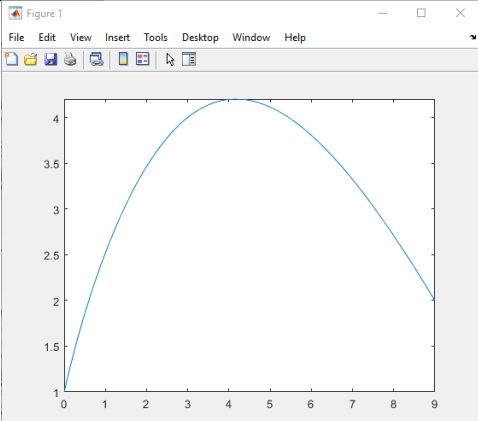
1. Librerie di routines

Ognuno dei quattro metodi introdotti utilizza una delle routine incluse nella libreria sviluppata ad-hoc. I metodi 'CUSTOM' implementano gli algoritmi ex-novo, mentre i metodi 'MATLAB' riutilizzano funzioni pre-esistenti.

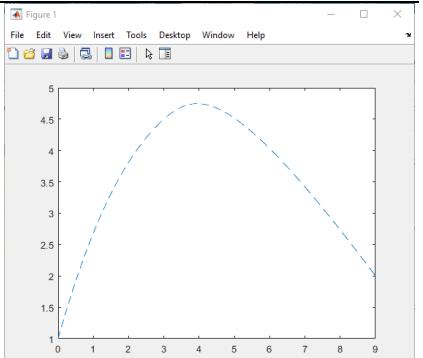
Routine 'Bezier_custom.m'

Scopo	La routine permette di calcolare e plottare i punti della curva di Bézier dato un vettore di control points in ingresso.
Specifiche	<i>tempo = Bezier_custom (x,y,app.UIAxes)</i>
Descrizione	<p>La routine implementa un algoritmo ad-hoc che, a partire dai punti di controllo forniti in input, calcola prima i polinomi di Bernstein e, successivamente, i valori di tutti i punti della curva di Bézier. Questi ultimi vengono restituiti all'interno della matrice A, dove:</p> <ul style="list-style-type: none"> • $A(1, i) = x_i$ • $A(2, i) = y_i$ <p>Oltre che ritornare i punti della curva, la routine plotta tali punti sulla figure UIAxes, fornita in input, e presente nell'interfaccia utente.</p>
Lista dei parametri	<p>Ingresso:</p> <ul style="list-style-type: none"> • x: vettore riga delle coordinate x dei punti di controllo • y: vettore riga delle coordinate y dei punti di controllo • app.UIAxes: figure presente nell'interfaccia utente <p>Uscita:</p> <ul style="list-style-type: none"> • Plotting della curva di Bézier e del poligono di controllo • Tempo: tempo di esecuzione dell'algoritmo
Esempi d'uso	<pre>>> x = [0 1 3 5 9]; >> y = [1 3 5 5 2]; >> tempo = Bezier_custom(x,y,app);</pre> 

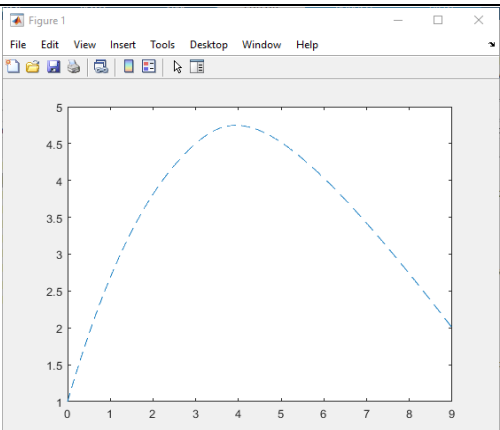
Routine 'Bezier_MATLAB.m'

Scopo	La routine permette di calcolare e plottare i punti della curva di Bézier dato un vettore di control points in ingresso.
Specifiche	<i>tempo = Bézier_MATLAB (x,y,app.UIAxes)</i>
Descrizione	<p>La routine fa uso delle funzioni MATLAB 'bernsteinMatrix()' e 'simplify()' per costruire i polinomi di Bernstein e calcolare i punti della curva di Bézier, restituiti all'interno della matrice A. Come prima:</p> <ul style="list-style-type: none"> • $A(1, i) = x_i$ • $A(2, i) = y_i$ <p>La routine plotta tali punti sulla figure UIAxes fornita in input.</p>
Lista dei parametri	<p>Ingresso:</p> <ul style="list-style-type: none"> • x: vettore riga delle coordinate x dei punti di controllo • y: vettore riga delle coordinate y dei punti di controllo • app.UIAxes: figure presente nell'interfaccia utente <p>Uscita:</p> <ul style="list-style-type: none"> • Plotting della curva di Bézier e del poligono di controllo • Tempo: tempo di esecuzione dell'algoritmo
Esempi d'uso	<pre>>> x = [0 1 3 5 9]; >> y = [1 3 5 5 2]; >> tempo = Bezier_MATLAB(x,y,app);</pre> 

Routine 'B_Spline_custom.m'

Scopo	La routine permette di calcolare e plottare i punti della curva Spline dato un vettore di control points in ingresso e dato il grado desiderato per la curva spline.
Specifiche	<i>tempo = B_Spline_custom (x,y,k,app.UIAxes)</i>
Descrizione	<p>La routine implementa un algoritmo ad-hoc che, a partire dai punti di controllo forniti in input, calcola dapprima il vettore dei <i>knot point</i> e successivamente, tramite la funzione di supporto 'Spline()', calcola tutti i punti della curva Spline, tenendo conto anche del grado della curva, preso in input dalla routine.</p> <p>La matrice A conterranno i punti calcolati che verranno opportunamente plottati sulla figure UIAxes.</p>
Lista dei parametri	<p>Ingresso:</p> <ul style="list-style-type: none"> • x: vettore riga delle coordinate x dei punti di controllo • y: vettore riga delle coordinate y dei punti di controllo • k: grado della curva Spline • app.UIAxes: figure presente nell'interfaccia utente <p>Uscita:</p> <ul style="list-style-type: none"> • Plotting della curva Spline e del poligono di controllo • Tempo: tempo di esecuzione
Esempi d'uso	<pre>>> x = [0 1 3 5 9]; >> y = [1 3 5 5 2]; >> k = 3; >> tempo = B_Spline_custom(x,y,k,app);</pre> 

Routine 'B_Spline_MATLAB.m'

Scopo	La routine permette di calcolare e plottare i punti della curva B-Spline dato un vettore di control points in ingresso. Il grado della curva calcolata è $(n-1)$ dove n è il numero di control points.
Specifiche	<i>tempo = B_Spline_MATLAB (x,y,app.UIAxes)</i>
Descrizione	<p>La routine fa uso delle funzioni MATLAB 'linspace()' per costruire il vettore di <i>knot point</i> e 'bsplinepolytraj()' successivamente per calcolare i punti della curva B-Spline, restituiti all'interno della matrice A. dove:</p> <ul style="list-style-type: none"> • $A(1, i) = x_i$ • $A(2, i) = y_i$ <p>La routine plotta tali punti sulla figure UIAxes.</p>
Lista dei parametri	<p>Ingresso:</p> <ul style="list-style-type: none"> • x: vettore riga delle coordinate x dei punti di controllo • y: vettore riga delle coordinate y dei punti di controllo • app.UIAxes: figure presente nell'interfaccia utente <p>Uscita</p> <ul style="list-style-type: none"> • Plotting della curva Spline e del poligono di controllo • Tempo: tempo di esecuzione
Esempi d'uso	<pre>>> x = [0 1 3 5 9]; >> y = [1 3 5 5 2]; >> tempo = B_Spline_MATLAB(x,y,app);</pre> 

2. Usabilità: GUI

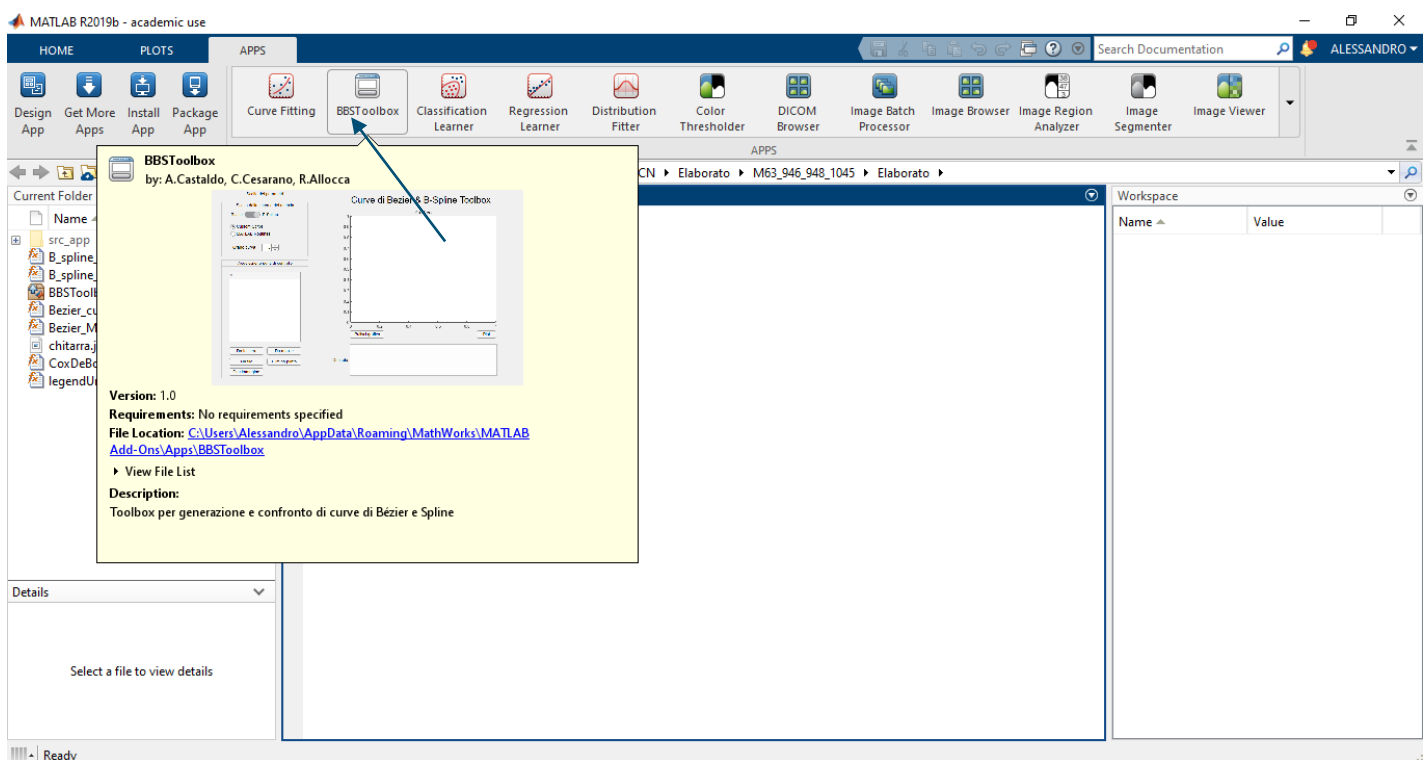
Il toolbox, realizzato attraverso lo strumento App Designer di MATLAB, è installabile nel pannello delle App nella dashboard di MATLAB.

L'installazione può essere effettuata cliccando sul file  `BBSToolbox.mlappinstall`

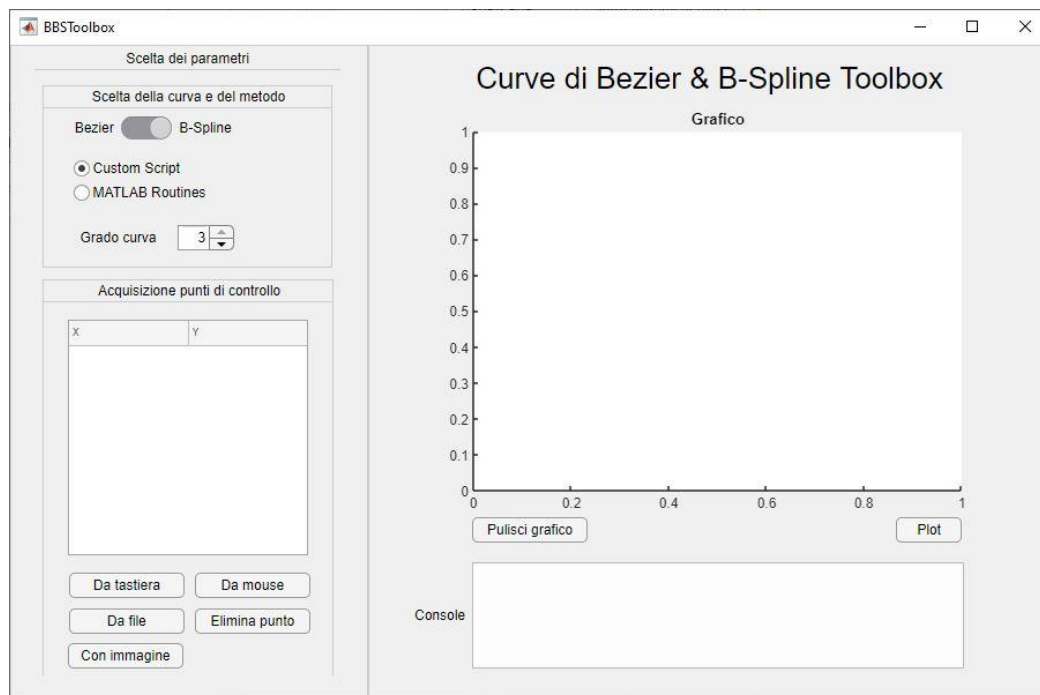
A questo punto, all'avvio di MATLAB, è necessario cliccare sul tasto *install* della finestra proposta:



A questo punto l'app risulta correttamente installata e reperibile nell'ambito del task *app* della dashboard di MATLAB, come visibile nella figura successiva:



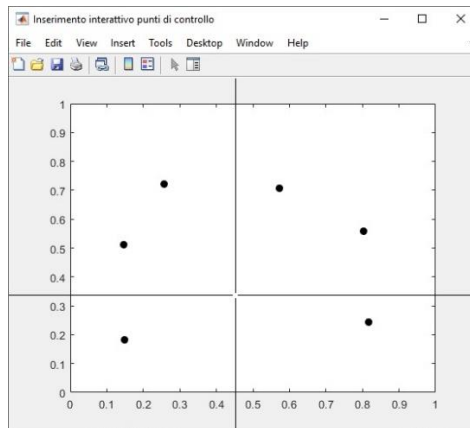
Cliccando sull'apposita icona, l'app esibisce la seguente interfaccia utente:



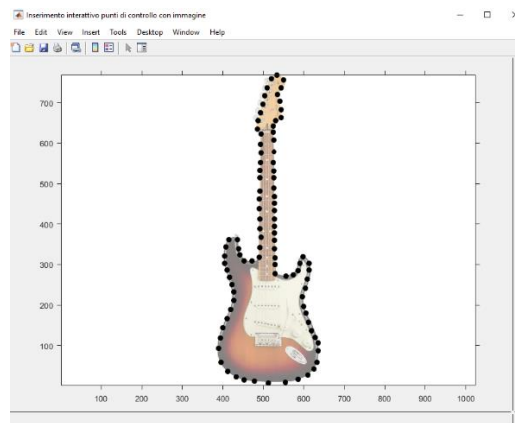
Attraverso la sidebar “scelta dei parametri” è possibile fornire tutti gli input necessari al tool:

- **Scelta della metodo di tracciamento della curva**
 1. B  zier
 2. B-Spline
- **Routines di libreria**
 1. Custom Script (per B-Spline Custom    possibile selezionare il grado della curva)
 2. MATLAB
- **Punti di controllo**
 1. *Acquisizione da tastiera*: i punti sono inizializzati a (0,0) e possono essere modificati da tastiera attraverso la tabella interattiva.
 2. *Acquisizione da file*: i punti sono caricati attraverso un file “.txt”, fornendo una colonna per ciascuna coordinata. L’importazione del file    guidata mediante una finestra di esplora risorse di Windows.

3. *Acquisizione da mouse*: i punti sono acquisiti da mouse mediante l'apertura di una nuova figura.

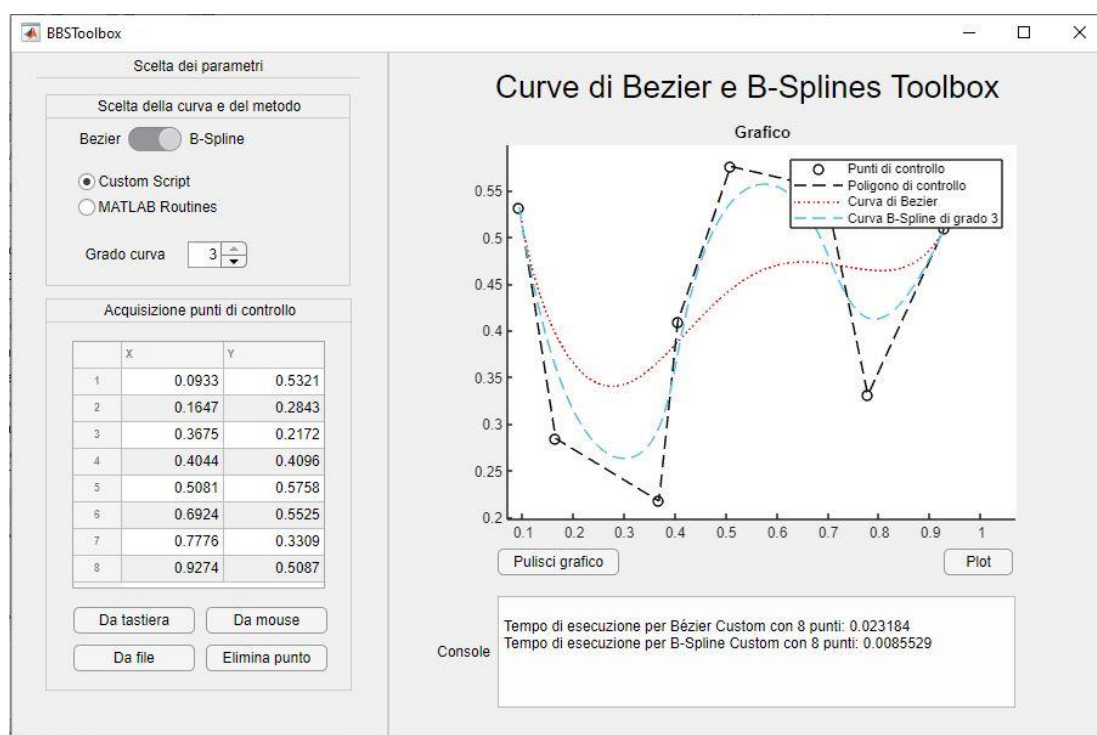


4. *Acquisizione da mouse con immagine*: i punti sono acquisibili da mouse, come nel caso precedente, ma con la guida di un'immagine di default in trasparenza.



5. *Elimina punto*: è possibile eliminare uno o più punti, selezionandoli nella tabella.

- **Grafico curve**: utilizzato dalle funzioni della libreria per rappresentare le curve, opportunamente calcolate, in seguito alla pressione del pulsante "*plot*". Gli assi sono ridimensionati automaticamente in funzione della curva. Il grafico è corredato di una legenda recante gli elementi plottati, oltre che il metodo utilizzato. Nell'ambito del plotting viene utilizzata la routine "*legendUnq*", appositamente importata, per consentire la stampa della legenda del grafico, evitando eventuali ripetizioni di labels.



- **Console:** mostra il dettaglio sui tempi di esecuzione delle routine utilizzate.

Console

Tempo di esecuzione per Bézier Custom con 10 punti: 0.017455 s
Tempo di esecuzione per Bézier MATLAB con 10 punti: 0.56612 s
Tempo di esecuzione per B-Spline Custom con 10 punti: 0.010033 s
Tempo di esecuzione per B-Spline MATLAB con 10 punti: 0.0009162 s

3. Robustezza: Casi di errore

Di seguito è illustrata brevemente la gestione degli errori.

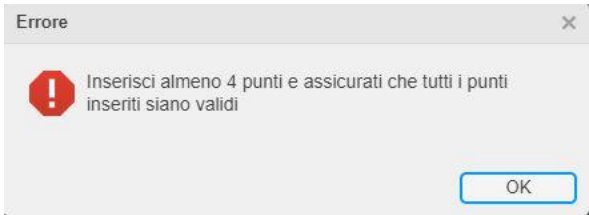
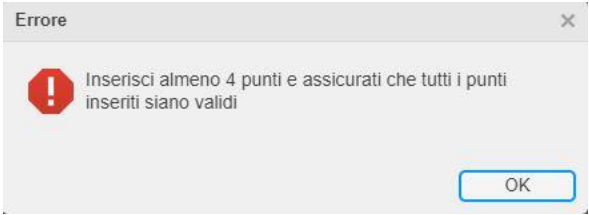
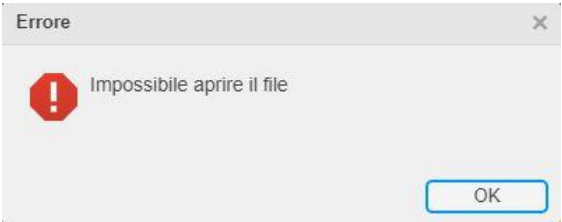
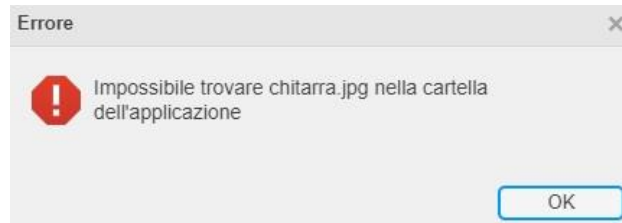
Numero minimo punti di controllo (messaggio errore)	Indipendentemente dal metodo scelto, ci si assicura che siano $k+1$, dove k è il grado della curva. 
Grado curva (adattamento campo)	Bézier MATLAB: fissato a 3 Bézier Custom: è fissato a 3 (cubica) B-Spline MATLAB: fissato a 3 B-Spline Custom: a scelta nel range [1-3]
NaN (messaggio errore)	E' effettuato un controllo sull'inserimento di coordinate non in formato numerico. 
Inserimento da file (messaggio errore)	E' effettuato un controllo all'atto dell'importazione della matrice dei punti da file, relativamente al suo formato. 
Inserimento da file (messaggio errore)	Controllo effettuato all'atto dell'importazione della matrice di punti nel caso in cui non sia stato selezionato alcun file. 

Immagine non presente
(messaggio errore)

È effettuato un controllo sull'esistenza, nella directory dell'applicazione, dell'immagine di sfondo utilizzata nell'ambito del metodo di acquisizione punti "con immagine".

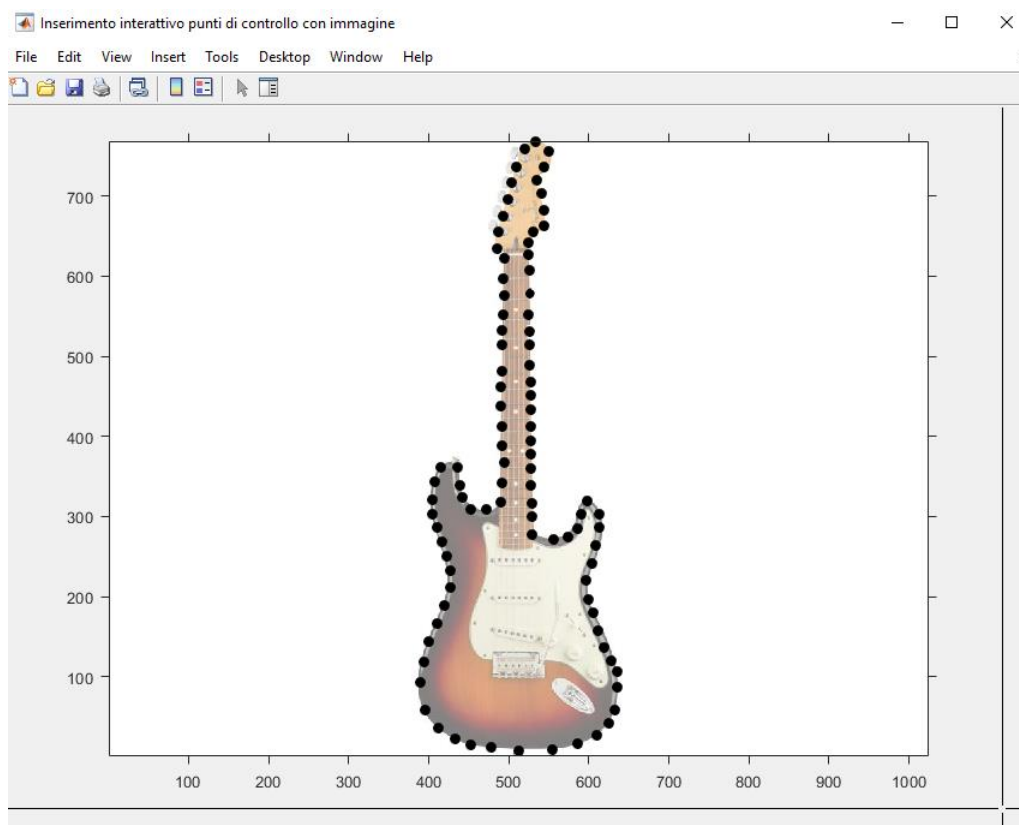


Elimina punto
(operazione abortita)

È effettuato un controllo che abortisce l'operazione di eliminazione dei punti nel caso in cui non sia stato selezionato alcun punto.

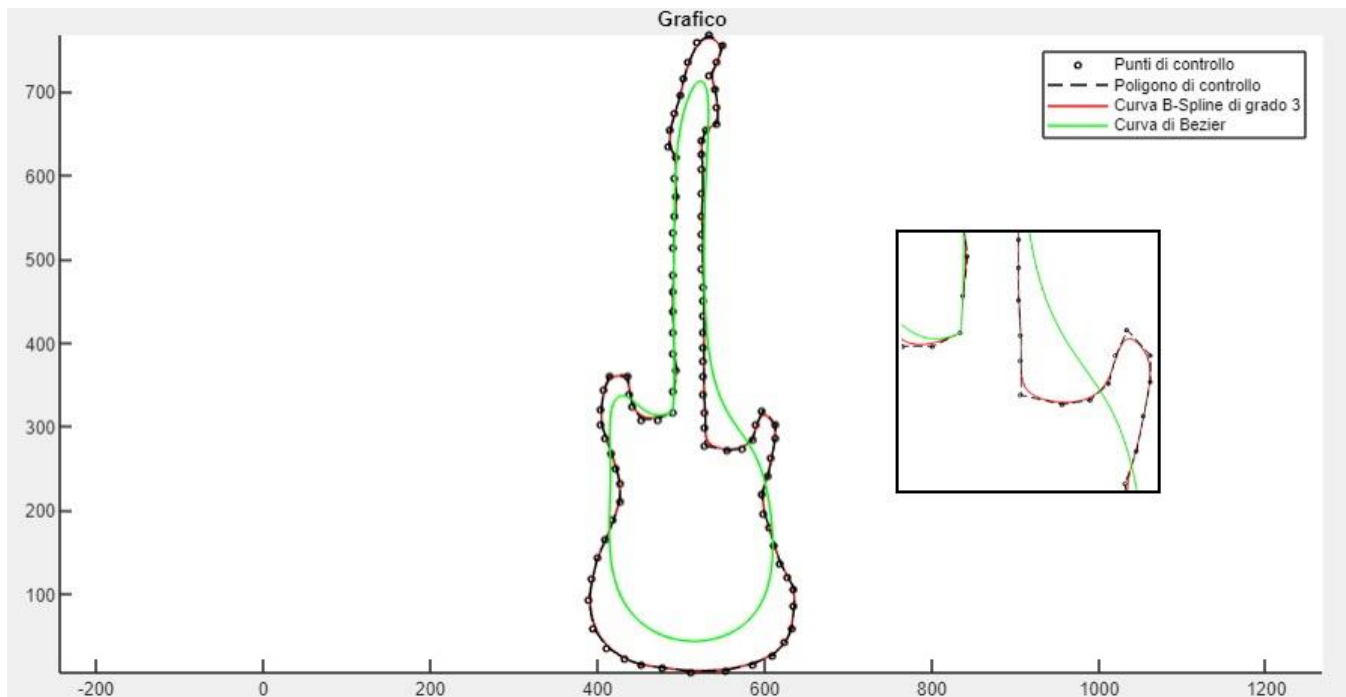
4. Accuratezza del metodo

Di seguito è mostrato un esempio d'uso del tool utilizzando la modalità di acquisizione "con immagine" per valutare visivamente l'accuratezza dei due metodi.



Dai risultati ottenuti è possibile confrontare visivamente l'accuratezza dei due metodi utilizzati (si lascia la percezione dell'accuratezza all'aspetto visivo dei risultati, date le difficoltà algoritmiche e computazionali di approcciare ad un metodo sistematico):

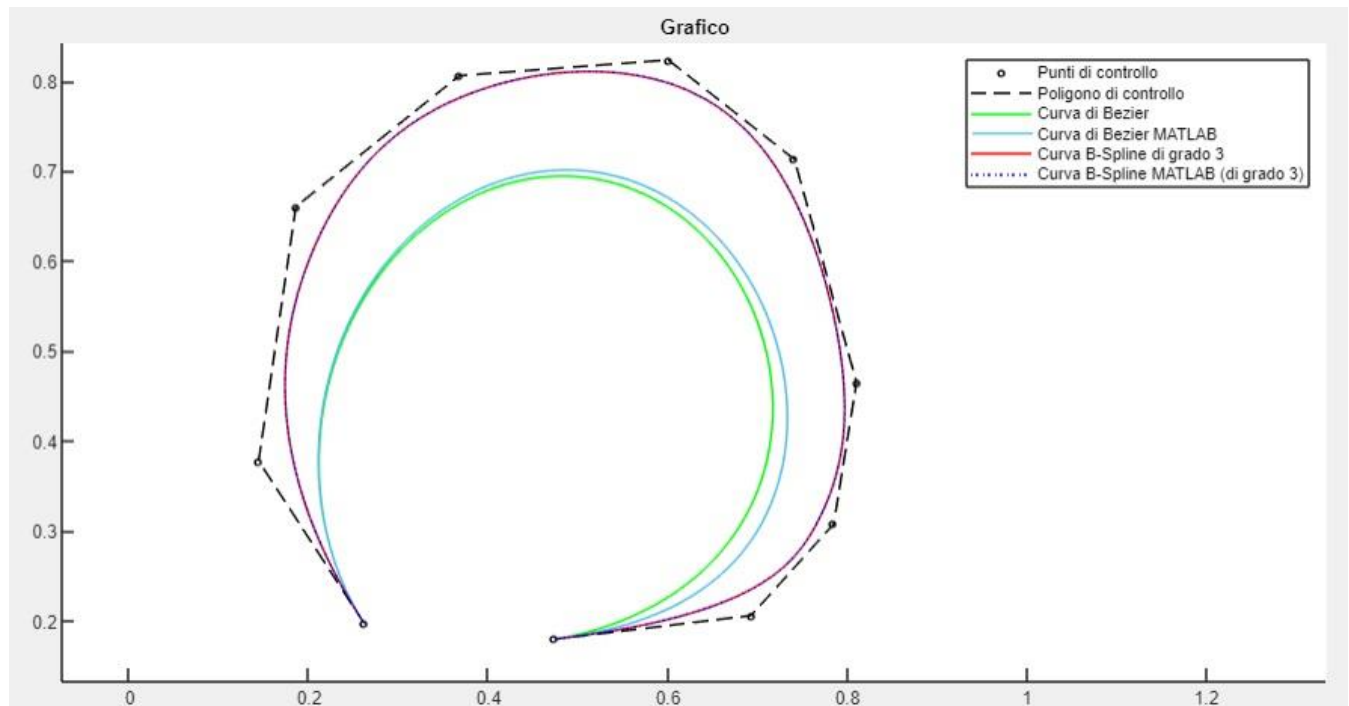
- Curva di Bézier (custom) in verde
- Curva Spline cubica (custom) in rosso



Come è possibile apprezzare ad occhio nudo, le curve Spline, a differenza delle curve di Bézier, grazie alla proprietà di controllo locale, permettono di approssimare in maniera più precisa il poligono di controllo, quasi interpolandone i punti.

5. Accuratezza dell'algoritmo

In questa sezione si vogliono sottolineare, invece, le differenze di accuratezza nell'utilizzo di tutte e quattro le routines, a cui vengono dati in input gli stessi punti di controllo.



Relativamente alle routines che calcolano la curva Spline, entrambe, sia quella custom che quella MATLAB, forniscono il medesimo risultato. Le curve, infatti, risultano perfettamente sovrapposte.

Viceversa, relativamente alle routines che calcolano le curve di Bézier, sembrerebbe, ad un primo confronto, che quella custom approssimi peggio il poligono di controllo, rispetto a quella basata sulle funzioni di libreria MATLAB. Tuttavia, si è notato che quest'ultima risulta instabile se utilizzata con un grado superiore a 35 (raggiunto con 36 punti di controllo).

Di seguito è mostrato un esempio che rappresenta tale problematica.

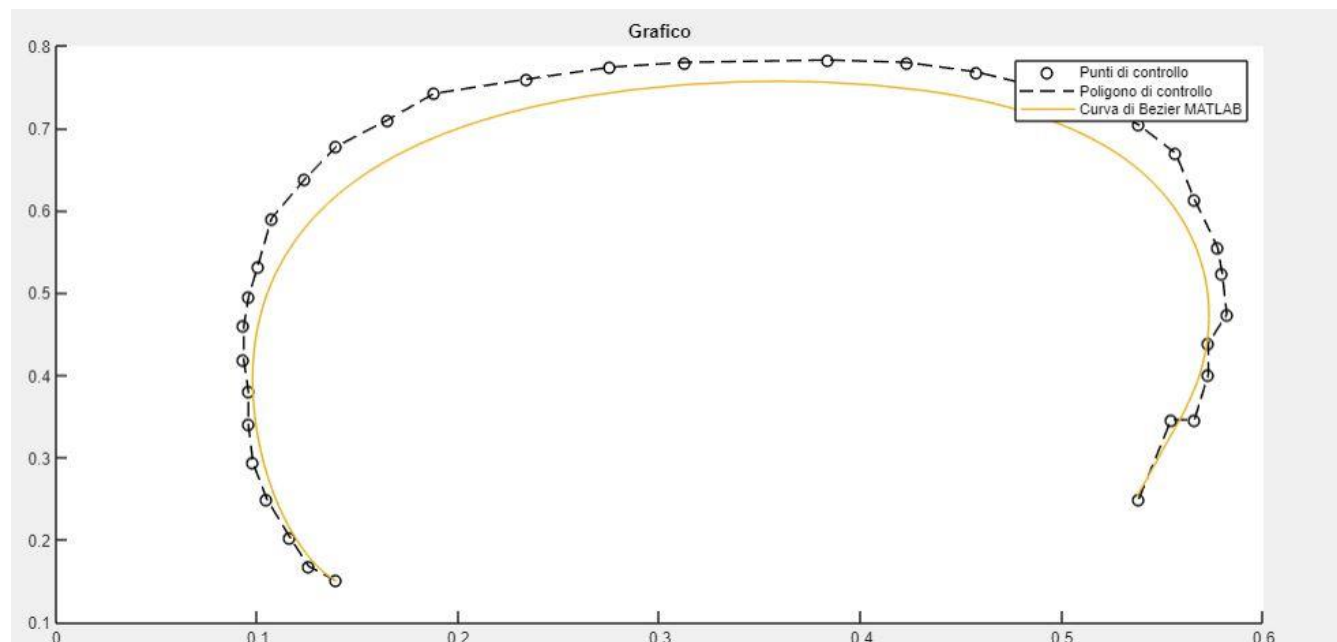


Figura 1. Curva di Bézier di grado 34 (MATLAB)

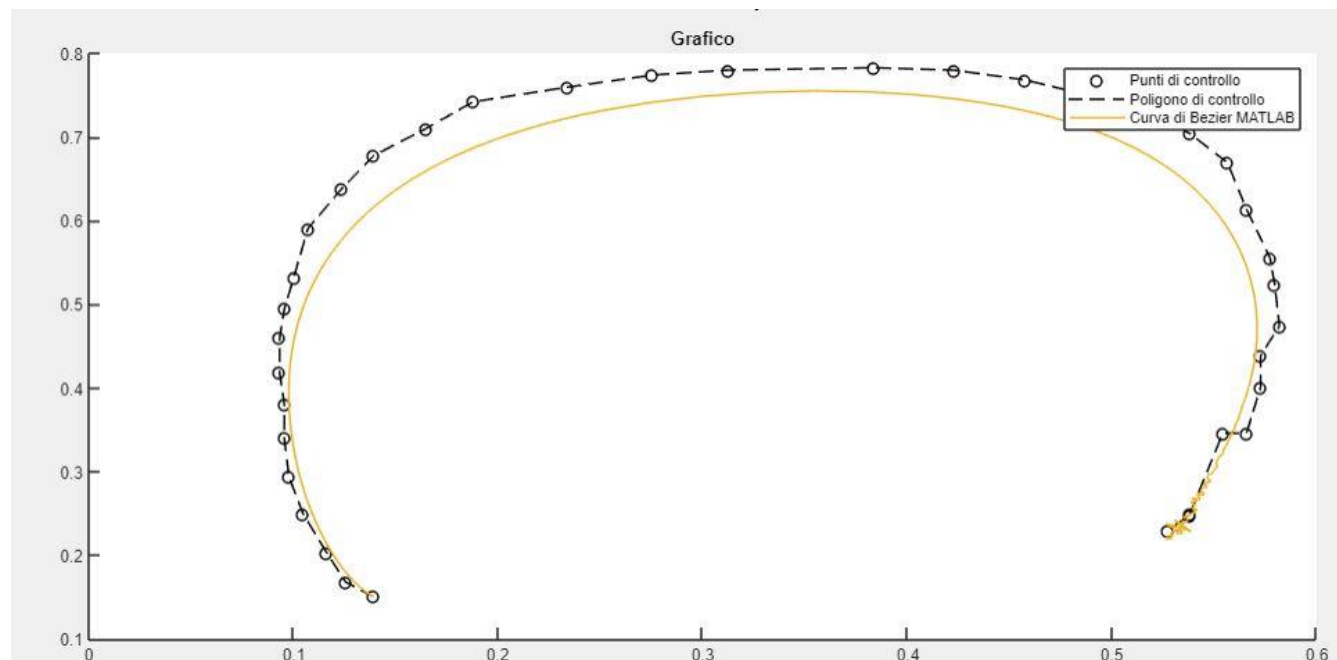


Figura 2. Curva di Bézier di grado 36 (MATLAB)

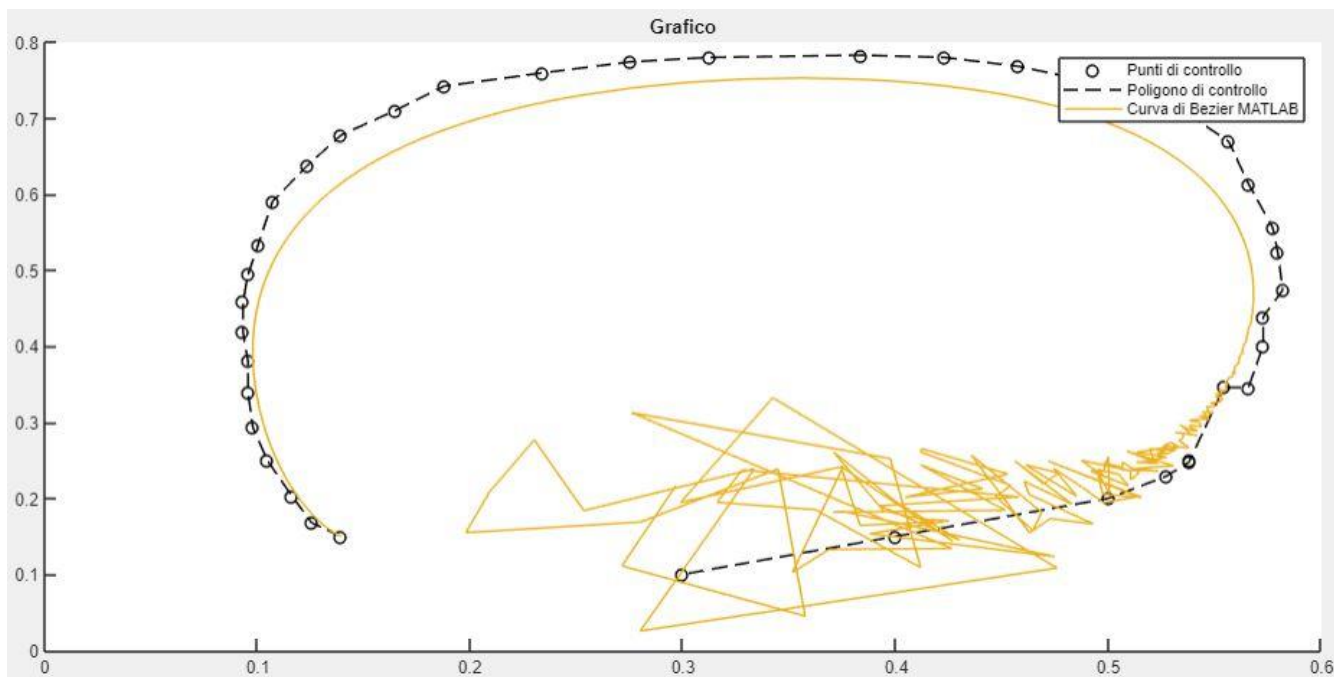


Figura 3. Curva di Bézier di grado 39 (MATLAB)

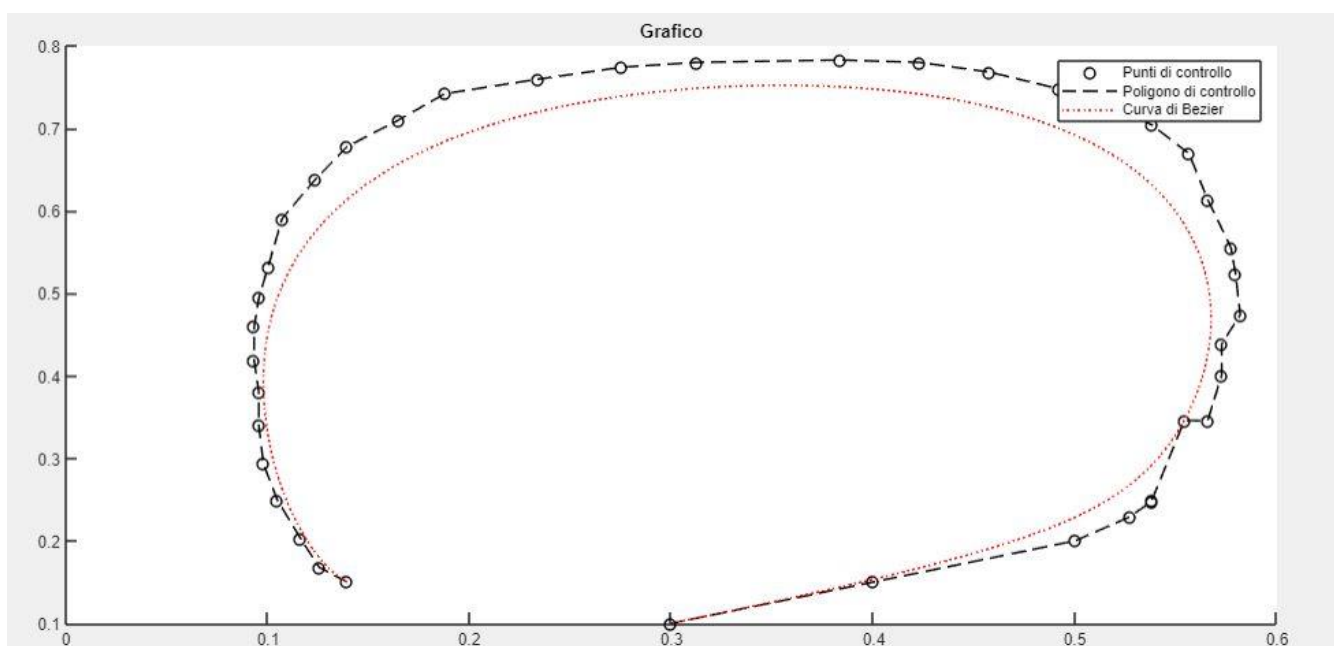
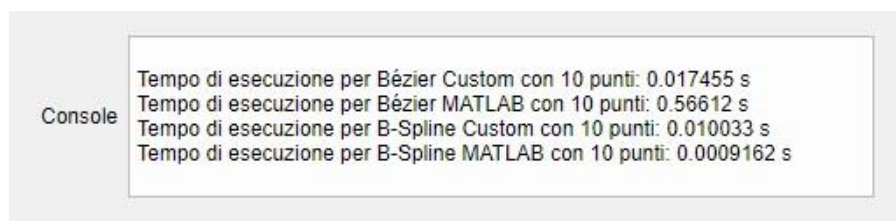


Figura 4. Curva di Bézier di grado 39 (custom)

6. Efficienza: tempi di esecuzione

Di seguito è mostrata un'istantanea, a scopo illustrativo, della *console* nella quale sono mostrati in output i tempi di esecuzione per ognuna delle routine utilizzate (misurati in secondi).



È stata effettuata inoltre una valutazione qualitativa della complessità computazionale di ciascuna routine, in funzione del numero di punti di controllo in input. Non sono stati riportati i tempi di esecuzione della routine “*Bézier MATLAB*” che, data la scarsa affidabilità dell’algoritmo, sono sembrati poco significativi.

numero punti	10	10^2	10^3	10^4	10^5
Bézier custom	0,058	13,12	173,34	1738	18543
B-Spline custom	0,05	0,59	0,53	5,157	58,66
B-Spline MATLAB	0,002	0,01	0,011	0,12	0,76

