UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Elaborato finale per l'esame di **Calcolo Numerico**

Modellazione grafica: Image Fitting con Interpolazione Lineare a Tratti, Curve di Bézier, B-Spline, NURBS.

Anno Accademico 2022/2023

Candidato:

Francesco Scognamiglio M63001364
Felice Micillo M63001377

Indice

Introduzione

Negli ultimi decenni, l'evoluzione delle tecnologie ha rivoluzionato il modo in cui raccogliamo, analizziamo e utilizziamo i dati. Questi sono diventati una risorsa preziosa per molteplici contesti, come la scienza, l'ingegneria, la medicina, l'arte e persino l'intrattenimento. Tuttavia, la domanda fondamentale è: come possiamo rappresentare graficamente e modellare in modo accurato l'andamento dei dati raccolti? È qui che entra in gioco la **modellazione grafica**, una disciplina informatica che si occupa della costruzione di modelli matematici per descrivere in modo preciso e attendibile il profilo di un'immagine o dei dati.

Uno degli strumenti fondamentali utilizzati in questa disciplina sono le curve di Bézier, le B-spline e i NURBS (Non-Uniform Rational B-Splines). Queste curve sono funzioni matematiche che consentono di approssimare una serie di punti di controllo, tracciando una curva fluida e regolare che passa attraverso tali punti. Il processo di fitting delle curve di Bézier, B-spline e NURBS implica la ricerca di una funzione che migliori possibile la descrizione dell'andamento dei dati. Questo significa trovare una curva che si adatti in modo ottimale ai punti di controllo forniti, rispecchiando accuratamente il comportamento dei dati originali. Questo tipo di approccio consente di individuare pattern, tendenze e irregolarità nei dati, facilitando la valutazione delle caratteristiche del problema e la qualità delle informazioni raccolte.

Capitolo 1: Rappresentazione dei dati

L'obiettivo della modellazione è quello di avvicinarsi quanto più possibile alla realtà; quindi, si vuole fare in modo di costruire un modello che sia attendibile. Per farlo, si possono seguire due approcci differenti:

1. ***Modello interpolante***: si costruisce il modello mediante una curva passante per i punti assegnati e si assume che l'errore sui dati sia trascurabile; Preso un insieme finito di dati $S=(x_i,y_i)$ $i=1,\ldots,n$ appartenenti ad un intervallo I, tale che $\{x_i\}\subset I$, con x_i detti nodi, definiamo l'interpolazione quel procedimento tramite cui si costruisce una funzione $f(x)$ tale che nei nodi (x_i) $i=1,\ldots,n$ siano soddisfatte determinate condizioni, dette **condizioni di interpolazione**. Queste condizioni devono essere rispettate dalla $f(x)$ e/o delle sue derivate. Quindi, con l'interpolazione, è richiesto che la funzione passi per i punti assegnati.
2. ***Modello approssimante***: si costruisce il modello mediante una curva che non è vincolata a passare per i punti assegnati e si assume, in questo caso, che l'errore dei dati non sia trascurabile. Con l'approssimazione intendiamo quel procedimento tramite cui, dato un insieme di n punti distinti $S=(x_i,y_i)$ $i=1,\ldots,n$ appartenenti ad un intervallo I, tale che $\{x_i\}\subset I$, con xi detti nodi, si costruisce una funzione $f(x)$ tale che non passi per necessariamente per i punti assegnati ma si discosti di un certo ϵ, ovvero tale che $|f(x_i)-y_i(i)|<\epsilon$ con $i=1\ldots n$.

1.1 Interpolazione di Lagrange e di Hermite

La condizione di interpolazione di Lagrange viene definita nel seguente modo:

Sia F uno spazio di funzioni di variabili reale o complesse ed assegnati

- N valori distinti reali p complessi $(x)_{i=1,\dots,n}$
- N valori distinti reali p complessi $(y)_{i=1,\dots,n}$

Si cerca una funzione che soddisfi la condizione di interpolazione di Lagrange $f(x) \in F : f(x_i) = y_i \; i = 1, \; \dots, \; n$

Questa relazione definita dalla funzione f(x) deve essere valida per ogni nodo dell'insieme ed è nota come **condizione di interpolazione di Lagrange**.

In ciascun nodo sono assegnate condizioni sulla funzione $f(x)$ e sulle sue derivate; se è assegnata in un nodo la condizione sulla derivata di ordine q, deve essere assegnata anche la condizione sulle derivate dall'ordine 0 fino all'ordine $(q-1)$.

Per le stesse condizioni di Lagrange, se volessimo imporre che anche la curva debba avere una certa pendenza, diamo informazioni sulla tangente in alcuni punti. Queste condizioni aggiuntive sulle derivate portano all'interpolazione di Hermite.

Definita in modo formale abbiamo:

dato un insieme di nodi $\{x_i\} i=0\dots n$ e noti i valori $\{y_i\}$ e di eventuali derivate, con l'**interpolazione di Hermite** si cerca la funzione che nei punti $\{x_i\}$ assume gli stessi valori sia della funzione che delle sue derivate.

Dati, inoltre, n interi positivi $l_1, l_2, \dots, l_n$ (che rappresentano l'ordine di derivazione desiderato per ciascun punto) tali che:

$$\sum_{i=1}^{n} l_i = m$$

Ed m valori y_i^j i=1,…,n e j=0,1..,l_{i-1}, si vuole determinare la funzione:

$$f^{(j)}(x_i) = y_i^j \quad con\, i = 1, \dots, n \; e \; j = 0, \dots, l_{i-1}$$

Le condizioni di interpolazione di Hermite prevedono quindi l'assegnazione di un vincolo sulle derivate per ogni nodo e, come detto prima, nel caso in cui viene assegnato sulla derivata di ordine q, per un certo nodo, allora devono essere assegnati i vincoli di derivata per tutte le altre di ordine inferiore.

1.2 Interpolazione Polinomiale

Il nostro elaborato si concentrerà sull'interpolazione polinomiale, perché più vantaggiosa dal punto di vista computazionale.

Sia $f(x)$ la funzione interpolante, consideriamo una combinazione di $\boldsymbol{n}$ **funzioni linearmente indipendenti** $\boldsymbol{\varphi_i}$ definite in un certo intervallo $[a,b]$. Assegnati i valori (x_i,y_i) con $x_i \in [a,b]$ tali che i punti siano distinti tra loro, possiamo scrivere:

$$f(x) = \sum_{i=0}^{n} a_i \varphi_i(x)$$

La difficoltà consiste nel determinare i coefficienti a_i. Ricordando che nel caso di interpolazione polinomiale la generica funzione interpolante $f(x)$ è definita come un polinomio $p(x)$ e che le funzioni φ_i che costituiscono la base sono anch'esse dei polinomi, il tipo di rappresentazione di tali funzioni conduce a diverse forme di rappresentazione del polinomio. Esistono 3 rappresentazioni per un polinomio $p(x)$:

- **Forma standard**

$$p(x) = \sum_{i=0}^{n-1} a_i \varphi_i(x) \text{ con } \varphi_i(x) = x^{i-1}$$

- **Forma di Lagrange**

$$\varphi_i = \prod_{\substack{j=1 \\ i \neq j}}^{n} \frac{(x-x_j)}{(x_i-x_j)} = l_i(x) \qquad \text{ed } a_i = y_i \rightarrow p(x) = y_1 l_1(x) + y_2 l_2(x) + \cdots + y_n l_n(x)$$

- **Forma di Newton**

$$\varphi_i = \prod_{j=1}^{i}(x - x_j) \rightarrow p(x) = a_0 + a_1(x - x_1) + \cdots + a_{n-1}(x - x_1)\ldots(x - x_{n-1})$$

Si utilizza la forma di Newton perché risulta essere più conveniente a livello computazionale; infatti, la forma standard dà origine ad una matrice mal condizionata, quindi non utilizzabile. La seconda, quella di Lagrange, dà luogo ad una complessità computazionale particolarmente elevata.

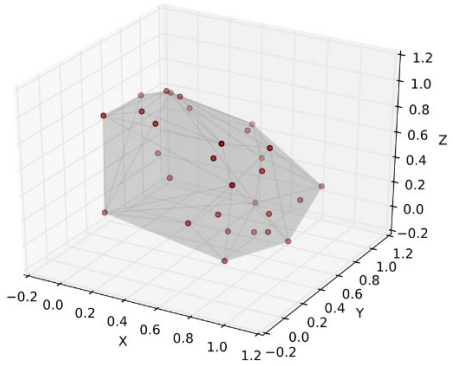
1.3 Punti di controllo di Bézier

Definisco i punti di controllo come le variabili che costituiranno la matrice del nostro problema. La matrice è composta da vari vincoli che caratterizzano il problema come per esempio:

- Vincoli di grandezza: (larghezza massima, lunghezza massima, altezza massima, etc.)
- Vincoli di Lagrange: per i quali si crea una funzione che soddisfi la condizione $f(x_i) = y_i$
- Interpolazione di Hermite: vincoli simili a quelli di Lagrange, ma con l'aggiunta dell'imposizione che la curva debba avere una certa pendenza aggiungendo al modello la tangente su alcuni punti.

Tutte queste richieste che abbiamo imposto alla funzione attraverso diversi vincoli, si traducono in termini matematici attraverso i punti di controllo, che delimitano un particolare poligono chiamato convex hull o inviluppo convesso.

Bèzier nel 1960 introdusse i punti di controllo per i problemi di interpolazione, come un modo naturale per stabilire la dimensione dell'oggetto da disegnare. Fissati opportunamente questi punti di controllo, dobbiamo costruire il più piccolo poligono convesso contenente tali punti.



Quindi tutte le curve sono costruite operando unicamente sui punti di controllo (e quindi sul convex hull) e tutti i punti di controllo sono disposti a partire dai vincoli imposti alla curva. Si va a definire quindi una matrice, dove la prima riga definisce tutti i nodi di interpolazione, mentre dalla seconda riga, sulla colonna associata al nodo i-esimo, abbiamo le condizioni di interpolazione.

x_1	x_2	…	x_{n-}	x_n
y_1^0	y_2^0	…	y_{n-}^0	y_n^0
y_1^1	y_2^1	…	y_{n-}^1	y_n^1
	y_2^2	…	y_{n-}^2	
	y_2^3	…		

1.4 Rappresentazione di una curva

Dopo aver considerato le varie principali forme di interpolazione polinomiale esistenti, in grafica, si individua prima il grado del polinomio che determina la curva che si vuole costruire, e poi si sceglie come base un polinomio che abbia lo stesso grado della curva scelta. In generale, una curva si può rappresentare in due modi:

- **Rappresentazione non parametrica:** Nella rappresentazione non parametrica non è consentito rappresentare curve chiuse o con punti multipli, in quanto questa rappresentazione si basa su una trasformazione univoca di x in y. Per caratterizzare la curva abbiamo bisogno di calcolare i suoi coefficienti; quindi, qualsiasi trasformazione sulla curva può essere ottenuta applicando la trasformazione al vettore dei coefficienti.

$$y = \sqrt{1-x^2} \quad x \in [0,1]$$

- **Rappresentazione parametrica:** Nella rappresentazione parametrica, invece, esiste un parametro t, che definisce il punto di coordinate x(t), y(t), che individuano un punto P(t), che si sposta sulla curva. Questa rappresentazione ha il vantaggio di poter rappresentare anche curve chiuse o con punti multipli, cosa che con la rappresentazione non parametrica risulta impossibile.

$$\begin{cases} x = x(t) \\ y = y(t) \end{cases} \quad C(t) = (x(t), y(t)) \quad t \in [a, b]$$

Al variare di t in $[a, b]$, le coordinate (x(t), y(t)) individuano un punto C(t) che si sposta sulla curva. Un esempio interessante è la circonferenza:

$$\begin{cases} x = \cos(t) \\ y = \sin(t) \end{cases} \quad t \in [0, \frac{\pi}{2}]$$

Nei problemi di grafica, si usa proprio la rappresentazione parametrica delle curve perché è molto più flessibile ed inoltre si usano i polinomi come base:

$$C(t) = \sum_{i=0}^{N} a_i f_i(t) \quad t \in [a, b]$$

Capitolo 2: Le curve di Bézier

Le curve di Bézier sono particolari curve parametriche, dall'andamento morbido, definite partendo da un numero finito di punti. Questi punti sono chiamati **punti di controllo** e la curva di Bézier è disegnata all'interno del **convex hull** o **inviluppo convesso**, cioè il più piccolo poligono convesso che contiene tutti questi punti. La curva di Bézier è il centro di massa del convex hull.

Il **centro di massa** (CM) di un oggetto è il punto dell'oggetto in cui è concentrato il suo peso, cioè dove si può assumere che la forza di gravità sia applicata.

Per un insieme finito di punti può essere definito come:

$$CM = \frac{\sum m_i p_i}{\sum m_i}$$

con masse m_i e p_i posizioni delle relative masse.

Dato un insieme di punti di controllo P_i i=0,…N, per stabilire chi svolge il ruolo delle masse, la curva di Bézier deve rispettare le seguenti proprietà:

- Deve interpolare il primo punto $P0$ e l'ultimo Pn, ma non quelli intermedi;
- I suoi punti devono trovarsi interamente all'interno del convex hull;
- La somma CM deve definire un punto nel piano;
- La curva deve essere indipendente dal sistema di riferimento cartesiano (cioè cambiando sistema ottengo sempre la stessa forma della curva).

Da queste proprietà scaturisce che la somma delle m_i deve essere 1 e che quindi l'equazione della curva di Bézier assume la forma:

$$CM = \frac{\sum m_i p_i}{\sum m_i} = \sum m_i p_i$$

Quest'ultima rappresenta l'equazione della curva di Bézier.

2.1 Proprietà delle curve di Bézier

Le curve di Bézier sono disegnate nello spazio a partire da un numero finito di punti e devono rispettare un certo numero di proprietà. In particolare, data un insieme di n punti di controllo P_i i=0,…,N la curva di Bézier:

- è tangente in $P0$ al segmento $P0P1$
- è tangente in PN al segmento $PN-1PN$
- la derivata k-esima in $P0$ dipende dai primi k + 1 punti
- la derivata k-esima in PN dipende dagli ultimi k + 1 punti
- la curvatura in $P0$ dipende da $P0P1P2$
- la curvatura in PN dipende da $PN-2PN-1PN$

Il motivo per cui sono necessarie tutte queste condizioni è legato al fatto che per ogni punto di controllo P_i i=0,…,N si costruisce un polinomio e, siccome ciascun polinomio ha grado n, il numero totale di condizioni è pari a $(n+1)^2$ condizioni.

Dunque, fissati n+1 punti di controllo, il grado del polinomio che definisce la curva di Bézier è pari ad n, e la curva è definita conoscendo:

- derivata zero in $P0$
- derivata zero in Pn
- derivata prima in $P0$
- derivata prima in Pn
- derivata seconda in $P0$
- derivata seconda in Pn

…

- la derivata n-ma in $P0$
- la derivata n-ma in Pn

2.2 Limiti delle curve di Bézier e l'apporto dei polinomi di Bernstein

Dalle precedenti proprietà enunciate, si potrebbe pensare che, aumentare dei punti di controllo aumenti la qualità del modello realizzato. In realtà ciò non si verifica a causa di una proprietà dei polinomi, che afferma, che le loro oscillazioni aumentano all'aumentare del loro grado, dunque, in molti casi l'errore cresce al crescere di n, e questo evento prende il nome di **fenomeno di Runge.** La conclusione è quindi che i polinomi di grado elevato oscillano fortemente tra i dati, fornendo un modello **poco affidabile**, e usare invece polinomi di grado basso fornisce pochi dati e quindi la creazione di un modello **poco attendibile**

Quindi, le limitazioni del polinomio interpolante sono:

- impossibilità di garantire la convergenza del polinomio alla funzione
- legame tra grado del polinomio e numero di nodi
- aumento dell'oscillazione al crescere del grado del polinomio

Un modo per risolvere il problema della convergenza del polinomio interpolante è quello di utilizzare come funzioni base dei polinomi a tratti di grado n chiamati **polinomi di Bernstein**, costruendo per ogni punto di controllo un polinomio.

I polinomi di Bernstein sono definiti come segue:

Un polinomio di Bernstein $P(x)$ di grado n è dato dalla formula:

$$P(x) = \sum_{k=0}^{n} c_k B_{k,n}(x)$$

dove gli $B_k^n(.)$ sono elementi della base dei polinomi di Bernstein definiti da:

$$B_{i,n}(x) = \binom{n}{i} x^i (1-x)^{n-i} \quad \text{se } x \in [0,1]$$

o, più in generale:

$$B_{i,n}(x) = \binom{n}{i} \frac{(b-x)^{n-i}(x-a)^i}{(b-a)^n} \quad \text{se } x \in [a,b]$$

Dato che ∀ P_i i=0,…,n si costruisce un polinomio e siccome ciascun polinomio ha grado n (n+1 incognite), il numero totale di condizioni è pari a $(n+1)^2$ condizioni cioè punti di controllo del polinomio più grado del polinomio. A questo punto, in generale, dato un insieme di punti di controllo P_i i=0,…,n, la corrispondente curva di Bézier è:

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

Dove il polinomio di Bernstein di grado n è :

$$B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i \quad \text{con} \quad t \in [0,1]$$

Per poter definire correttamente le curve di Bézier, parliamo di alcuni aspetti fondamentali. Le funzioni base utilizzate sono curve parametriche che dipendono dal parametro t per $t \in [0,1]$ e devono assumere valore non negativo in tale intervallo. Esse formano una combinazione convessa, ovvero la loro somma ∀ t è sempre pari ad 1.

Esempio di curva di grado 1:

Dati due punti di controllo p_0 = 1-t e p_1 = t, la curva di Bézier determinata da questi due punti coincide con l'interpolazione lineare del segmento che li congiunge:

$$P(t) = (1-t)P_0 + tP_1$$

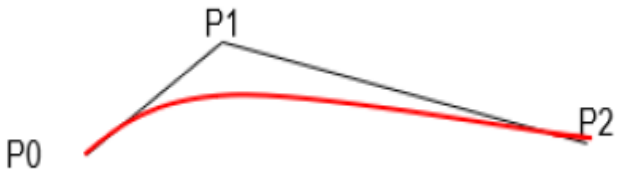
Esempio di curva di grado 2:

Per definizione l'equazione della curva sarà la seguente:

$$C(t) = P_0 B_{0,2}(t) + P_1 B_{1,2}(t) + P_2 B_{2,2}(t) \text{ con} \quad t \in [0,1]$$

Dove i polinomi di Bernstein di grado 2 sono definiti come:

$$B_{0,2}(t) = (1-t)^2$$
$$B_{1,2}(t) = 2t(1-t)$$
$$B_{2,2}(t) = t^2$$



Esempio di curva di grado 3:

Per definizione l'equazione della curva sarà la seguente:

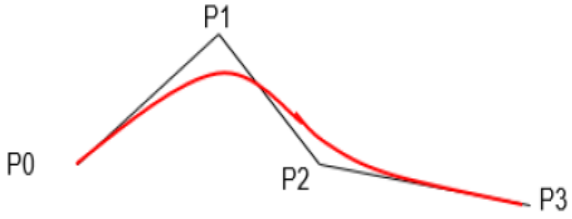
$$C(t) = P_0 B_{0,3}(t) + P_1 B_{1,3}(t) + P_2 B_{2,3}(t) + P_3 B_{3,3}(t) \text{ con } t \in [0,1]$$

Dove i polinomi di Bernstein di grado 3 sono definiti come:

$$B_{0,3}(t) = (1-t)^3$$
$$B_{1,3}(t) = 3(1-t)^2 t$$
$$B_{2,3}(t) = 3(1-t)t^2$$
$$B_{3,3}(t) = t^3$$

Quindi:

Per $t = 0$ allora $C(0) = P_0$, per definizione $P_0 B_{0,3}(0) = 1$
Per $t = 1$ allora $C(1) = P_3$, per definizione $P_3 B_{3,3}(1) = 1$



Per ogni valore di t tra 0 e 1, tutte le funzioni di base sono non nulle e quindi contribuiscono al valore di C(t) sulla curva.

2.3 Proprietà e svantaggi delle curve di Bézier

- Deve interpolare il primo punto $P0$ e l'ultimo punto PN;
- I suoi punti devono trovarsi interamente all'interno dell'inviluppo convesso. Conseguenza che i polinomi di Bernstein sono positivi nell'intervallo [0,1] e che la loro somma deve essere pari ad 1, ossia sono una combinazione convessa;
- Tangente in P0 al segmento P0P1 ed è tangente in Pn al segmento Pn-1Pn. Questo lo si può dimostrare, calcolando la derivata della curva agli estremi, ossia per t=0 e t=1.

Tuttavia, le curve di Bézier presentano però degli svantaggi:

- Se si aggiunge all'insieme dei vertici di controllo un solo vertice, è necessario ricalcolare completamente l'equazione parametrica della curva (cioè, si dice che le curve sono "globali", in quanto non si possono apportare modifiche locali). Per ottemperare a questo problema, si potrebbe optare per l'uso di una interpolazione a tratti tramite cui, le funzioni interpolanti si ottengono suddividendo l'insieme assegnato dei punti in sottoinsiemi aventi ascisse consecutive e costruendo su ciascun sottoinsieme il relativo polinomio interpolante;
- Un altro svantaggio è che la sostituzione di un solo punto di controllo, comporta il cambiamento di forma dell'intera curva;
- Per poter rappresentare accuratamente forme complesse è necessario un grado elevato della curva, però computazionalmente queste curve sono più onerose da calcolare.

2.4 Algoritmo di De Casteljau

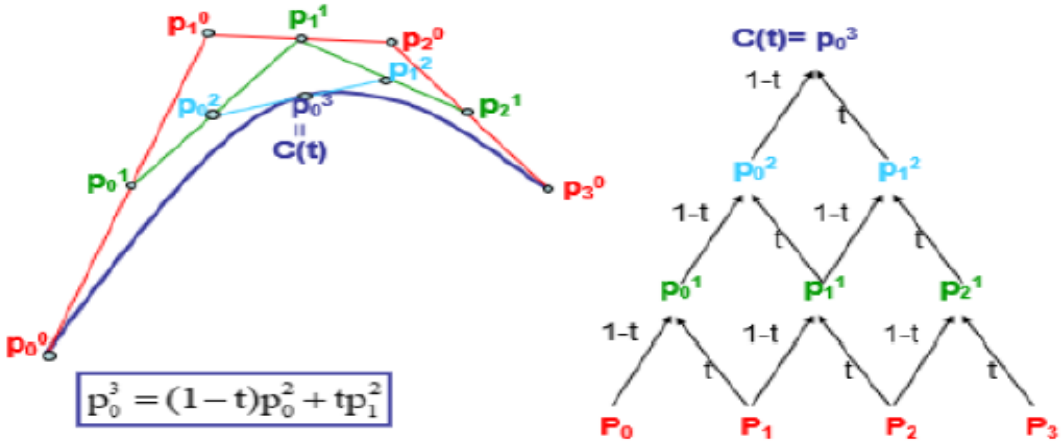
L'algoritmo di De Casteljau è un algoritmo ricorsivo sebbene lento numericamente stabile, per calcolare un punto C(t), su una curva di Bézier per $t \in [0,1]$. Se la curva di Bézier è di grado **n**, l'algoritmo di de Casteljau ha **n** iterazioni.

L'esecuzione dell'algoritmo si articola nel seguente modo:

Inizia con n+1 punti e ad ogni iterazione, il numero di punti considerati si riduce di 1. All'n-esima iterazione, l'unico punto rimasto è il punto C(t) appartenente alla curva di partenza.

Ogni lato del poligono viene diviso in due parti, formando un nuovo punto. I punti collegati tra loro determinano un nuovo poligono con n-1 lati. L'algoritmo è iterato finchè non si ottiene un solo lato, il quale diviso in due parti, permette di determinare un unico punto corrispondente a quello sulla curva.

Esempio su una curva di grado 3:



Relazione di ricorsione:

$$P_i^j(t) = (1-t)P_i^{j-1}(t) + tP_{i+1}^{j-1}(t)$$
$$\text{con } t \in [0,1], j = 1, ..., n \text{ e } i = 0, 1, ..., n-j$$

Per poter eseguire agevolmente i calcoli, è utile avvalersi di uno schema computazionale triangolare per eseguire i passaggi dell'algoritmo, e quindi per i vari livelli delle poligonali generate. Iterando questo algoritmo per ogni t in [0,1], si ottiene l'intera curva.

2.5 Fenomeno di Runge

Come accennato in precedenza, l'utilizzo dei polinomi presenta un inconveniente legato al fatto che le loro oscillazioni aumentano all'aumentare del grado. Questa caratteristica rappresenta una limitazione significativa per l'impiego dei polinomi nei processi di interpolazione, in quanto il grado delle curve di Bézier deve corrispondere al numero dei punti di controllo meno uno. Ciò significa che all'aumentare del numero di punti di controllo, ovvero dei dati disponibili per un determinato fenomeno, l'errore di

interpolazione cresce a causa dell'instabilità dei polinomi. Tale problematica diventa evidente osservando il fenomeno noto come "fenomeno di Runge".

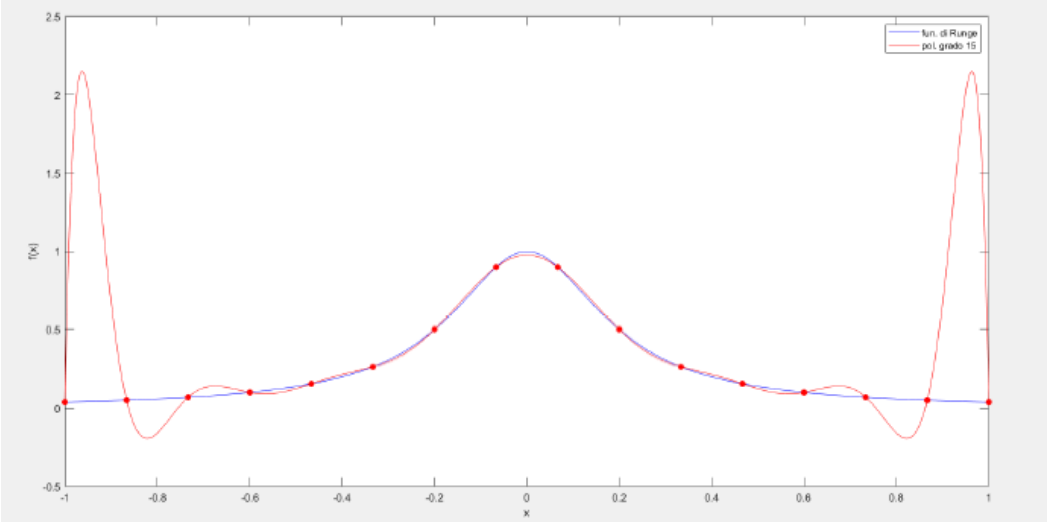
Per poter vedere al meglio questo fenomeno, possiamo prendere in considerazione la funzione di Runge:

$$f(x) = \frac{1}{1 + 25x^2}$$

Runge scoprì che interpolando questa funzione su un insieme di punti x_i,$con\ i$=1,…,n , equidistanti nell'intervallo [−1,1], con un polinomio $Pr(x)$ di grado $r \leq n$, l'interpolazione risultante oscilla in ampiezza verso gli estremi dell'intervallo (in questo caso −1 e +1). Si dimostra inoltre, che l'errore fra $f(x)$ e $Pn(x)$ tende all'infinito all'aumentare del grado del polinomio:

$$\lim_{n\to\infty} \left(\max_{x\in[-1,1]} |f(x) - P_n(x)| \right) = +\infty$$

Se si va a graficare la funzione in questione e la curva interpolante di grado 15, possiamo notare le oscillazioni di quest'ultima:



Capitolo 3: Curve B-Spline

Le curve di Beizer portano con loro un problema relativo alla loro globalità, ossia modificando un punto viene modificata l'intera curva, non consentendone dunque una rappresentazione precisa di alcune curve. Ciò è dovuto al fatto che le funzioni di base sono definite e sono non nulle in tutto l'intervallo di definizione della curva. Tale problema viene risolto usando **funzioni polinomiali a tratti interpolanti,** ossia funzioni in cui ogni tratto è un polinomio di grado n e per ogni punto di controllo si costruisce un polinomio. Il problema di questa soluzione, tuttavia, è che nei punti di raccordo (punti finali degli intervalli) la funzione potrebbe presentare delle irregolarità (ad esempio derivata discontinua). Dobbiamo quindi trovare un compromesso tra polinomi di grado moderatamente basso e sufficiente regolarità dell'intera funzione risultante.

Questo compromesso è realizzato con le funzioni **Spline**.

3.1 Funzioni Spline

Dato un insieme di nodi n (detti breakpoints) le funzioni spline sono funzioni polinomiali a tratti tali che, presi intervalli consecutivi:

$\forall [x_i, x_{i+1}] \quad i = 1, \dots, n-1 \rightarrow s(x) = s_i(x) \in \pi_m, \; s(x)$ è derivabile fino all'ordine m – 1 continue in $[x_i, x_{i+1}]$.

Una spline è una funzione, costituita da un insieme di polinomi raccordati tra loro, il cui scopo è interpolare in un intervallo un insieme di punti (detti nodi della spline), in modo tale che la funzione sia continua almeno fino ad un dato ordine di derivate in ogni punto dell'intervallo.

Esistono varie modalità di generazione delle spline, ma, contrariamente ai metodi numerici di Newton e Lagrange, i cui polinomi hanno un grado che dipende dal numero dei polinomi considerati, esse utilizzano polinomi di grado p fissato per unire tutti gli n nodi, con $p<n$. Quindi una spline non genera un solo polinomio interpolatore, ma un insieme di polinomi che si raccordano tra loro, in modo tale che la funzione spline finale sia continua in tutti i nodi. Inoltre, le curve spline sono delle curve composte da altre curve, per tale motivo sono anche dette curve composite.

3.2 Funzioni B-Spline

Le funzioni adoperate per costruire una spline sono dette B-spline.

Le B-spline sono funzioni linearmente indipendenti a supporto compatto pari al numero dei nodi + 1, così definite:

$$B_{i,h}(t),\ i = 1,2,3 \ldots \quad h = grado$$

Mentre per indicare la base per lo spazio delle Spline di grado h sul vettore dei nodi usiamo

$$\{B_{0,h}(t), \ldots, B_{n,h}(t)\}$$

Con le funzioni B-spline entrano in gioco i nodi che svolgono un ruolo significativo. Sia T= (t0, t1, …) il **vettore dei nodi**.

Il supporto locale delle B-spline significa che:

• la funzione B-spline di grado 0 $B_{i,0}(t)$ ha supporto compatto nell'intervallo $[t_i, t_{i+1}]$

• la funzione B-spline di grado 1 $B_{i,1}(t)$ ha supporto compatto nell'intervallo $[t_i, t_{i+2}]$

• la funzione B-spline di grado h $B_{i,h}(t)$ ha supporto compatto nell'intervallo $[t_i, t_{i+h+1}]$

Le **curve B-Spline** si distinguono dalle spline perché sono, in realtà, un caso particolare in cui come curve polinomiali vengono adoperate le curve di Beziér.

Per cui, le b-splines "incollano" tra di loro più curve di Bézier e si dividono principalmente in due differenti categorie:

- B-Spline **uniformi**: parametrizzate su intervalli della medesima dimensione;
- B-Spline **non uniformi**: parametrizzate su intervalli che hanno dimensioni differenti.

La differenza tra le Spline e le B-Spline sta nel fatto che quest'ultime possono essere funzioni diverse in base all'intervallo; invece, le Spline sono identiche nei vari sotto-intervalli. Questa caratteristica permette una buona approssimazione dei dati che stiamo interpolando.

3.3 Proprietà delle curve B-Spline

In analogia alle curve di Bézier, le curve B-spline sono definite come una combinazione lineare degli n+1 punti di controllo ($P_i \; con \; i = 1, \dots, n$) moltiplicate le funzioni B-spline di grado h (ottenute in maniera ricorsiva) definite su un intervallo di nodi.

$$C(t) = \sum_{i=0}^{n} P_i B_{i,h}(t)$$

Dove Pi = punti di controllo e $B_{i,h}(t)$ = funzione B-spline di grado h definita in $[t_i, t_{i+h+1}]$.

I polinomi in questo caso, non sono quelli di Bernstein, ma quelli calcolati attraverso la formula ricorsiva di de Boor:

$$B_{i,h}(x) = \frac{x - t_i}{t_{i+h} - t_i} B_{i,h-1}(x) + \frac{t_{i+n+1} - x}{t_{i+h+1} - t_{i+1}} B_{i+1,h-1}(x) \quad h > 0$$

La funzione base è:

$$B_{i,0}(x) = \begin{cases} 1 & \forall x \in [t_i, t_i + 1) \\ 0 & altrove \end{cases}$$

I parametri di tali curve sono:

- n + 1 punti di controllo P0, … , Pn
- grado h
- vettore dei nodi
- $T=(t_0, t_1, t_2, \dots, t_{n+h+1})$, $t_i < t_{i+1}$

Ci servono queste informazioni sui nodi proprio perché, rispetto a quelle di Bézier, non richiedono informazioni solo sui punti di controllo.

Le funzioni B-Spline sono definite a partire dai nodi T, che sono una sequenza di numeri reali non decrescente (non è detto che debba essere strettamente crescente: alcuni nodi possono coincidere). La cardinalità dell'insieme T dipende dal numero dei punti di

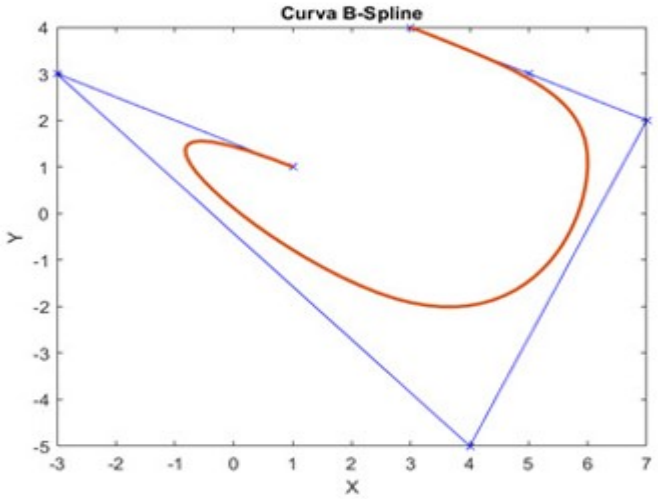
controllo e dal grado p delle funzioni B-Spline, infatti:

m=n+p+1

ovvero:

#$nodi$=#$punti$+$grado$+1

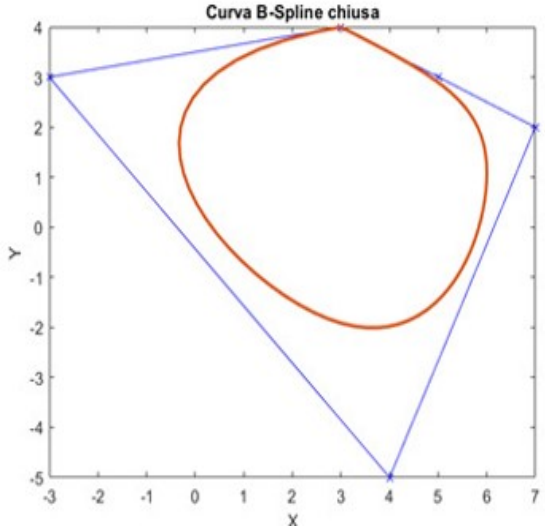
In generale la B-Spline non passa per i punti di controllo. Se vogliamo far passare la curva per i punti di controllo, allora dobbiamo aumentare la molteplicità di un nodo. Se la molteplicità di un nodo è pari a h+1, la curva passerà per quel punto. Modificando la molteplicità dei nodi, il poligono rimane invariato ma viene modificata la forma della curva generata.

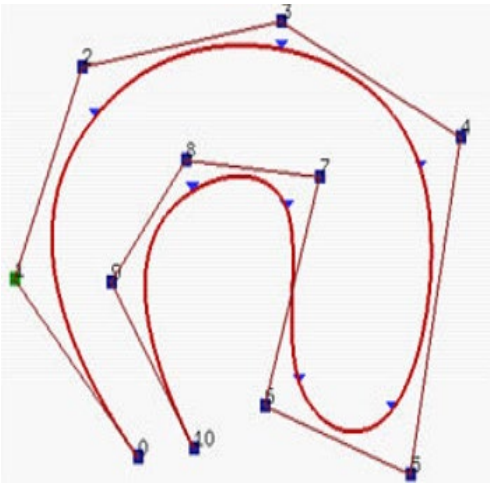


Ulteriori proprietà delle curve sono:

- Le curve non sono chiuse a meno che l'ultimo punto di controllo P_n coincide con il primo P_0;
- La curva non interpola alcun punto di controllo;
- Si può modificare la curva in modo che sia tangente al primo e all'ultimo segmento nel primo e nell'ultimo punto di controllo, imponendo che il primo e l'ultimo nodo abbiano molteplicità h+1;
- Controllo locale: come conseguenza del supporto compatto delle B-spline, se P_j cambia, $C(t)$ risulta modificata solo in corrispondenza dei punti di definizione della B-spline;
- Invarianza: è possibile modificare la curva applicando trasformazioni geometriche ai punti di controllo e poi riapplicare la formula per rivalutare la curva;

- Dato il grado h, all'aumentare della molteplicità di un nodo la curva si avvicina al nodo multiplo fino ad interpolare il punto di controllo corrispondente. In quanto aumentando la molteplicità di un nodo si riduce il supporto della B-spline definita in quel nodo, di conseguenza il risultato tende al suo valore massimo.





Capitolo 4: Curve NURBS

Una forte limitazione delle B-spline è che, essendo polinomi, esse non riescono a rappresentare senza errore le curve descritte da funzioni trigonometriche oppure una circonferenza, le quali devono essere necessariamente approssimante da polinomi (di Taylor). Queste funzioni non possono essere rappresentate da un numero finito di polinomi, però posso essere descritte da **polinomi razionali**. Per questa ragione si decise di estendere le B-spline ai polinomi razionali, introducendo così le NURBS (Non Uniform Rational B-Spline).

Le curve NURBS, a differenza delle B-spline, hanno un parametro in più che ne modifica la forma, ovvero i pesi w_i, che rappresentano la capacità del punto P_i di attirare a sé la curva.

Possiamo definire le curve nel seguente modo:

- Razionale: se i punti di controllo non hanno tutti lo stesso peso;
- Non razionale: se i punti di controllo hanno tutti lo stesso peso.

Possiamo dunque definire le curve NURBS partendo dalla defizione della curva B-Spline, aggiungendole opportunamente i pesi. Ci servono dunque quattro informazioni di base:

1. Grado;
2. Punti di controllo $P_0, \dots, P_n$;
3. $B_{i,h}(t)$ le funzioni B-spline di grado h sul vettore dei nodi T;
4. Pesi $w_0, \dots, w_n$.

$$C(t) = \frac{\sum_{i=0}^{n} w_i P_i B_{i,h}(t)}{\sum_{i=0}^{n} w_i B_{i,h}(t)} \qquad t \in [t_h, t_{n+h+1}]$$

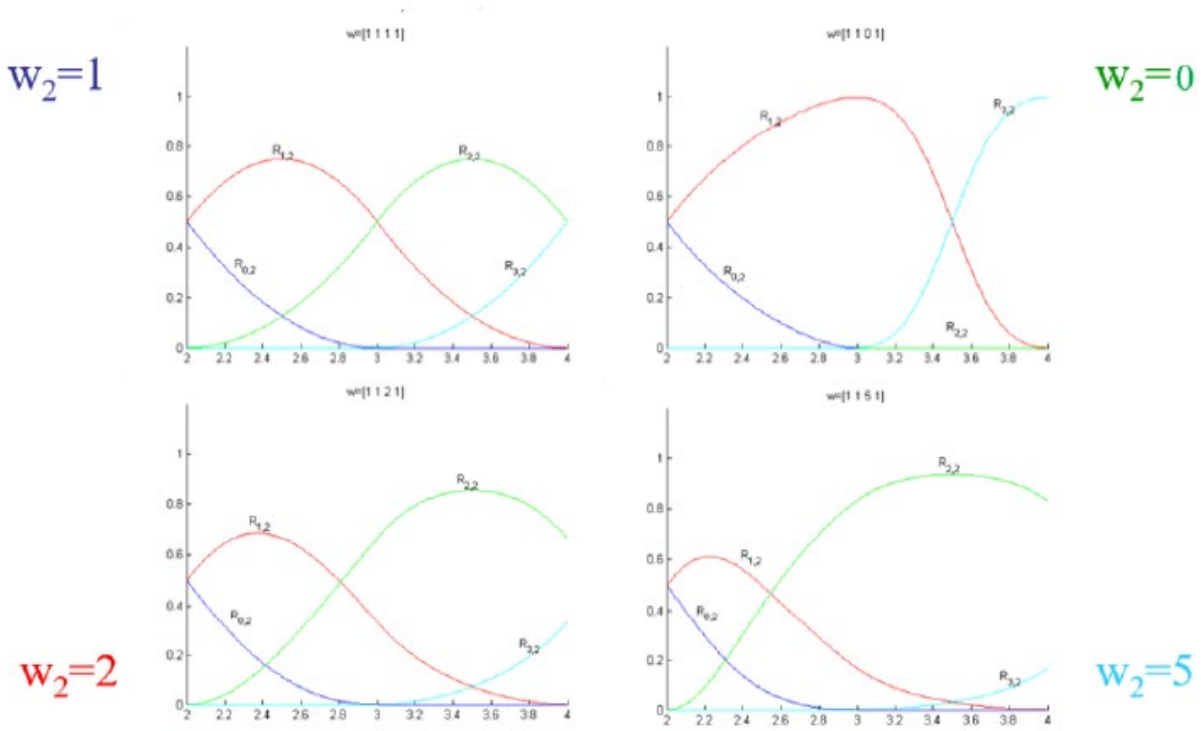
In alternativa è possibile riscrivere l'equazione nella seguente forma

$$C(t) = \sum_{i=0}^{n} P_i R_{i,h}(t) \quad t \in [t_h, t_{n+h+1}]$$

dove $R_{i,h}(t)$ sono le funzioni di base B-spline razionali di grado h sul vettore dei nodi T e pesi $w_0, \ldots, w_n$.

$$R_{i,h}(t) = \frac{w_i B_{i,h}(t)}{\sum_{i=0}^{n} w_i B_{i,h}(t)} \quad t \in [t_h, t_{n+h+1}]$$

Aumentando e diminuendo il valore del peso w_i, la curva avvicinerà o allontanerà sempre più al punto di controllo P_i. Al tendere del valore del peso w_i all'infinito, la curva passerà esattamente per il punto di controllo P_i,



4.1 Proprietà delle NURBS

- Partizionamento dell'unità: $\sum_i B_{i,h} = 1$
- Relazione con B-spline: se tutti i pesi $w_i = 1$,la curva NURBS degenera in B-spline
- Differenziabilità: tra i nodi, la curva è di classe è $C\infty$. Sui nodi, la curva appartiene a
- C^{k-z} con z= molteplicità del nodo. Se si aumenta la molteplicità di un nodo, il livello di continuità decresce, mentre all'aumentare di k, aumenta.
- Supporto locale

- Non negatività
- L'ultima proprietà delle curve NURBS è l'invarianza alle trasformazioni affini, dato un generico punto di controllo P, si definisce precisamente come A[P]=L[P]+T.

4.2 Trasformazioni Affini

Una trasformazione affine è una trasformazione lineare seguita da una traslazione. Dato un generico punto di controllo P, si definisce precisamente come A[P] = L[P] + T. Grazie a questa trasformazione lineare, è possibile modificare la curva andando a modificare la posizione dei punti di controllo, cioè la curva si muove se si muovono i punti di controllo.

Sfruttando la linearità delle NURBS si può applicare questa trasformazione sui punti e quindi di conseguenza sulla curva, cioè, applicare una trasformazione sulla curva oppure sui punti produce lo stesso risultato.

Infatti, data C(u) la generica curva NURBS allora avremo che:

$$A[C(u)] = L[C(u)] + T$$

dalla definizione di curva NUURBS si ottiene che:

$$L[C(u)] = \Sigma_i\{L[P_i]R_{i,h}(u)\} + T$$

Prossimo riscrivere l'uguaglianza come segue:

$$A[C(u)] = L[C(u)] + T = L[C(u)] = \Sigma_i\{L[P_i]R_{i,h}(u)\} + T$$

Applicando invece la stessa trasformazione ai punti, avremo:

$$\sum_i A[P_i]R_{i,h}(u) = \sum_i \{L[P_i] + T\}R_{i,h}(u) = \sum_i \{L[P_i]R_{i,h}(t)\} + T\sum_i R_{i,h}(t)$$
$$= \sum_i \{L[P_i]R_{i,h}(u)\} + T$$

Combinando le due espressioni $A[C(u)] = \Sigma_i A[P_i]R_{i,h}(u)$

Come possiamo notare, il risultato è analogo sia operando direttamente sulla curva che operando sui punti di controllo, quindi in conclusione, servono solo i punti di controllo per modificare la curva.

4.3 Confronto Curve

Qui di seguito, è riportata una tabella riassuntiva, contenente tutte quelle che sono le principali caratteristiche di queste curve, e dunque le loro differenze:

	BEZIER	B-SPLINE	NURBS
La curva è contenuta all'interno dell'involucro convesso del poligono di controllo	SI	SI	SI (a patto che il peso associato ai punti del poligono di controllo sia maggiore di 0)
Legame tra il grado della curva rispetto al poligono di controllo	Grado = Numero di vertici - 1	Indipendente dal poligono di controllo	Indipendente dal poligono di controllo
La curva tocca il poligono di controllo?	SI, nel primo e nell'ultimo punto	NO, ma può essere fatto, collassando i nodi quanto il grado della curva.	NO, ma può essere fatto, collassando i nodi quanto il grado della curva.
La forma della curva e del poligono sono in qualche modo correlate?	NO, solo in modo molto approssimato	SI e li si può far coincidere	SI e li si può far coincidere
Si può avere una trasformazione affine sulla curva?	SI, operando sui vertici del poligono di controllo	SI, operando sui vertici del poligono di controllo	SI, operando sui vertici del poligono di controllo..
Si può avere il controllo locale sulla curva?	NO	SI	SI
Si possono disegnare cerchi ed ellissi?	NO	NO	SI

Capitolo 5: Implementazione dei modelli di interpolazione

Abbiamo implementato degli script in Matlab che effettuano la risoluzione del problema di interpolazione sui modelli descritti teoricamente nei capitoli precedenti.

Inoltre, abbiamo sviluppato una applicazione con interfaccia grafica che permette di risolvere il problema di interpolazione in maniera pratica ed intuitiva (Capitolo 6).

5.1 Lineare a tratti

Di seguito il codice dello script Matlab per la risoluzione del problema di interpolazione tramite il modello lineare a tratti.

(Il codice è contenuto nel file 'lineare_a_tratti.m').

```
clc;
clear;

% Modello di interpolazione lineare a tratti
figure ('Name', 'Seleziona i punti di controllo');
imshow ('Ferrari_488_Spider.jpg');
[x, y] = getpts ();
close;

% Configurazione degli assi con asse Y invertito
figure ('Name', 'Lineare a tratti');
hold on;
axis ('equal');
set (gca, 'Ydir', 'reverse');
nodes = plot (x, y, '*');
set (nodes, 'Color', 'k', 'LineWidth', 3);

% Interpolazione lineare a tratti
hold on
s = 1 : length(x);  % Vettore degli indici dei nodi
t = 1 : 0.1 : length(x); % Vettore in cui calcolare i punti intermedi
p1 = interp1 (s, x, t); % Calcolo dei punti intermedi sulla coordinata x
p2 = interp1 (s, y, t); % Calcolo dei punti intermedi sulla coordinata y
lineare = plot (p1, p2); % Plot dell'interpolazione lineare a tratti
set (lineare, 'Color', 'r', 'LineWidth', 2);
```

Guida all'utilizzo:

Indicando il percorso di un'immagine nella funzione 'imshow()', è possibile tracciare i punti di controllo su di essa.

Una volta tracciati i punti di controllo e premendo Invio, verrà generata una figura contenente il plot della lineare a tratti.

Descrizione del codice:

Lo script Matlab utilizza la funzione 'getpts()' per consentirci di tracciare i punti su un'immagine, restituendo due vettori che rappresentano le coordinate dei punti selezionati.

Nella fase di configurazione degli assi per il plot del modello, vengono impostati in modo che partano dall'origine utilizzando 'axis(equal)', e inoltre viene invertito l'asse delle Y tramite 'set(gca, 'Ydir', 'reverse')', al fine di rappresentare correttamente i punti di controllo.

I punti di controllo vengono quindi tracciati all'interno degli assi utilizzando la funzione 'plot()'.

Procediamo con la generazione del modello. Viene definito un vettore 's' che contiene gli indici di tutti i nodi di controllo e un vettore 't' che rappresenta il vettore di query richiesto dalla funzione 'interp1()', utilizzata per eseguire l'interpolazione lineare a tratti. Il vettore 't', in base al passo scelto (nel codice 0.1), indica quanti punti devono essere presenti tra il primo punto di controllo e l'ultimo.

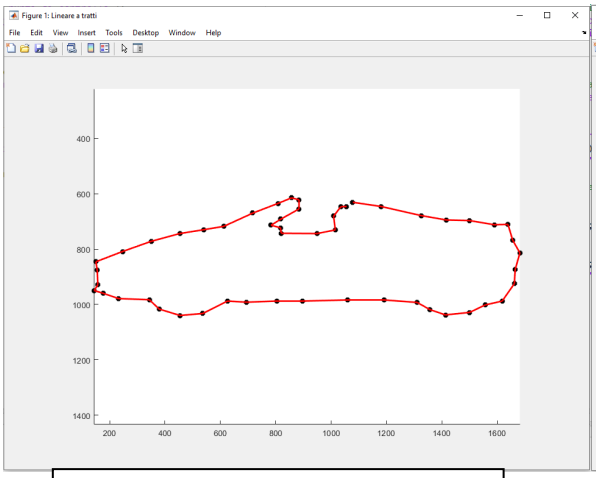
Aumentando il numero di punti (passo basso), il modello si adatta bene passando attraverso tutti i punti di controllo, mentre diminuendo i punti (passo alto), il modello non si adatta bene e potrebbe non passare attraverso tutti i punti di controllo.

La funzione 'interp1()' prende in ingresso il vettore degli indici 's', il vettore 'x' dei valori delle coordinate x dei punti di controllo e il vettore di query points 't'. La funzione restituisce tutti i punti interpolati che si trovano tra un punto di controllo e il successivo.

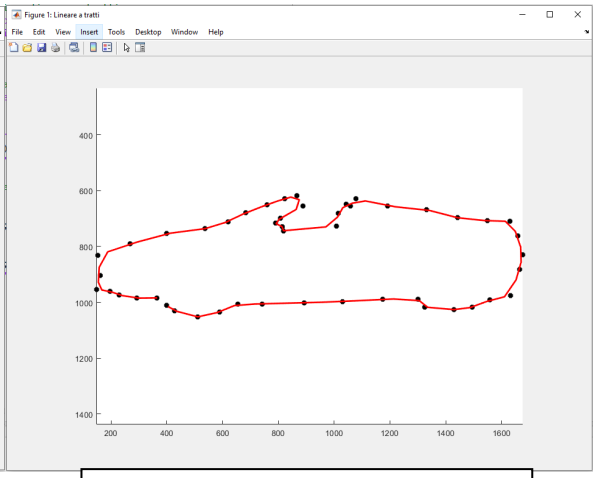
Eseguiamo la stessa operazione anche per le coordinate y e infine tracciamo i punti sugli assi.

Il risultato è soddisfacente e rispetta tutte le proprietà teoriche di questo modello.

Immagini di esempio:



Lineare a tratti con passo 0.1



Lineare a tratti con passo 0.9

5.2 Curva di Bezier

Di seguito il codice dello script Matlab per la risoluzione del problema di interpolazione tramite la curva di Bezier.

(Il codice è contenuto nel file 'bezier_curve.m').

```
clc;
clear;

% Modello di interpolazione curve di Bezier
figure ('Name', 'Seleziona i punti di controllo');
imshow ('Ferrari_488_Spider.jpg');
[x, y] = getpts ();
close;

% Configurazione degli assi con asse Y invertito
figure ('Name', 'Interpolazione curva di Bezier');
hold on;
axis ('equal');
set (gca, 'Ydir', 'reverse');
nodes = plot (x, y, '*');
set (nodes, 'Color', 'k', 'LineWidth', 2);

% Curve di Bezier
hold on
P = zeros(length(x), 2); % Definisco una matrice P con le coordinate dei punti di c.
for i = 1 : length (x)
    P (i,1) = x (i);
    P (i,2) = y (i);
end
syms t;
B = bernsteinMatrix (length(x)-1, t);  % Realizzo la matrice dei polinomi di Bernstein
% Moltiplico la matrice di bernstein e i punti di controllo e ottengo la curva di
% Bezier
bezierCurve = simplify (B*P);
bezier = fplot (bezierCurve(1), bezierCurve(2), [0,1]); % Plotto la curva di Bezier
set (bezier, 'Color', 'r', 'LineWidth', 2);

% Plot dei polinomi di Bernstein
figure('Name', 'Polinomi della matrice di Bernstein')
axis('equal')
axis([0 1 0 1])
hold('on')
% Definisco l'intervallo in cui definire i polinomi da plottare
t_values = linspace(0, 1, 50);
% Plotto ogni polinomio
for i = 1:(length(x))
    y = subs(B(:, i), t_values);
    plot(t_values, y);
end
hold off;
legend(cellstr(strcat('B(', num2str((0:length(x)-1)'), ',', ...
    num2str(length(x)-1), ')')));
```

Guida all’utilizzo:

Indicando il percorso di un'immagine nella funzione 'imshow()', è possibile tracciare i punti di controllo su di essa.

Una volta tracciati i punti di controllo e premendo Invio, verrà generata una figura contenente il plot della curva di Bezier e una figura contenente il plot di tutti i polinomi di Bernstein.

Descrizione del codice:

Dopo aver tracciato i punti di controllo sull'immagine indicata, viene definita una matrice dei punti di controllo 'P' che contiene le coordinate dei punti tracciati.

Viene quindi definito un simbolo 't' che rappresenterà la variabile per i polinomi di Bernstein che saranno calcolati successivamente.

La funzione 'bernsteinMatrix()' restituisce una matrice contenente tutti i polinomi di Bernstein in base al numero di punti di controllo meno uno.

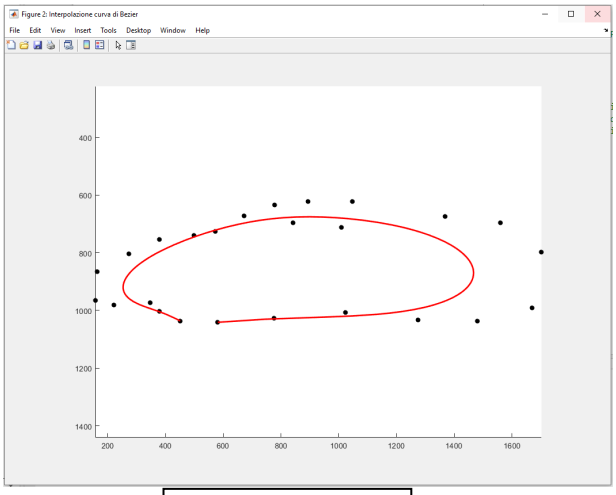
Per ottenere la curva di Bezier, viene eseguita la moltiplicazione tra le matrici 'P' (punti di controllo) e 'B' (polinomi di Bernstein) ottenendo un vettore contenente due equazioni che rappresentano la curva di Bezier rispetto all'asse x e all'asse y.

Successivamente, la curva di Bezier viene plottata in modo che la variabile 't' vari nell'intervallo [0, 1], come stabilito dalla teoria.

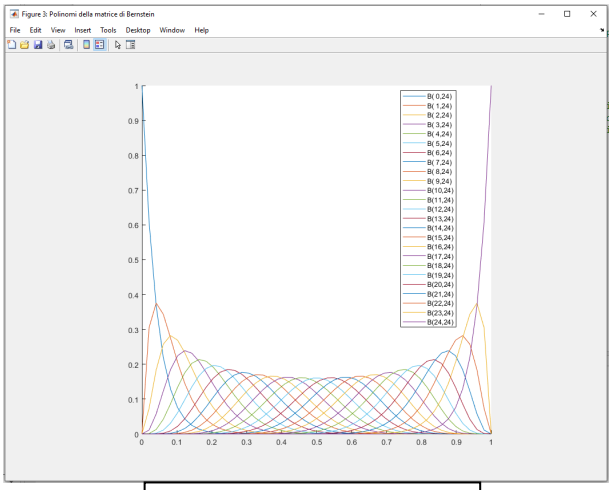
Inoltre, viene eseguito il plot dei polinomi di Bernstein contenuti nella matrice 'B'. Per effettuare il plot dei polinomi, viene definito un vettore 't_values' che definisce i punti in cui calcolare i polinomi. Utilizzando la funzione 'subs()', il valore di 't' (la variabile dei polinomi) viene sostituito con gli elementi del vettore 't_values', consentendo di plottare ogni polinomio sugli assi.

L'implementazione di questo codice fornisce una rappresentazione visiva della curva di Bezier e dei polinomi di Bernstein associati ai punti di controllo tracciati sull'immagine.

Immagini di esempio:



Curva di Bezier



Polinomi di Bernstein

5.3 Curve B-Spline

Di seguito il codice dello script Matlab per la risoluzione del problema di interpolazione tramite le curve B-Spline.

(Il codice è contenuto nel file 'bspline_curve.m').

```
clc;
clear;

% Modello di interpolazione Curve B-Spline
figure ('Name', 'Seleziona i punti di controllo');
imshow ('Ferrari_488_Spider.jpg');
[x, y] = getpts ();
close;

% Configurazione degli assi con asse Y invertito
figure ('Name', 'B-Spline');
hold on;
axis ('equal');
set (gca, 'Ydir', 'reverse');
nodes = plot (x, y, '*');
set (nodes, 'Color', 'k', 'LineWidth', 2);

% B-Spline con algoritmo di DeBoor
n = 4;
grado = n-1;
P = [x,y]';
% Knots a punti fissi
t = [zeros(grado + 1, 1)' sort(rand(size(x, 1) - grado - 1, 1))' ...
    ones(grado + 1, 1)'];
% Knots a punti variabili
%t = sort(rand(length(x) + grado + 1, 1))'

Y = bspline_deboor_fun(n,t,P);
% Plot della curva B-Spline sui punti
hold on;
plot(Y(1,:), Y(2,:), 'r');
hold off;

% Plot delle funzioni base della B-Spline
figure('Name','Bik')
hold on;
axis('equal')
N = bspline_funzioni_base_fun(grado, t, size(x,1));
for j = 1 : size(x, 1)
    plot(linspace(1, 9, 10*size(x, 1)), N(:, j));
end
hold off;
legend(cellstr(strcat('B(', num2str((0:length(x)-1)'), ',', ...
    num2str(length(x)-1), ')')));
```

Guida all'utilizzo:

Indicando il percorso di un'immagine nella funzione 'imshow()', è possibile tracciare i punti di controllo su di essa.

Una volta tracciati i punti di controllo e premendo Invio, verrà generata una figura contenente il plot della curva B-Spline e una figura contenente il plot di tutte le funzioni di base.

Descrizione del codice:

Una volta selezionati i punti di controllo, vengono definite le seguenti variabili per l'applicazione delle B-Spline:

- 'n': il grado della curva, che definisce il tipo di B-Spline da utilizzare (2 per lineare, 3 per quadratica, 4 per cubica, ecc.).
- 'grado': 'n-1', utilizzato per calcolare la dimensione del vettore dei nodi.
- 'P': la matrice dei punti di controllo che contiene le coordinate x e y.
- 't': il vettore dei nodi, definito in due modi a seconda se si desidera fissare i punti di controllo o meno. Se i punti di controllo sono fissati, il vettore sarà costituito dai valori 0 e 1 per i primi e ultimi nodi, mentre i valori centrali saranno numeri casuali compresi tra 0 e 1. Se i punti non sono fissati, il vettore dei nodi sarà composto interamente da numeri casuali che variano da 0 a 1. In entrambi i casi, il vettore dei nodi deve essere ordinato in modo crescente.

Una volta definite le variabili, possiamo applicare l'algoritmo di de Boor per ottenere la curva B-Spline utilizzando la funzione 'bspline_deboor_fun()'. Ottenuta la curva, la plottiamo.

Successivamente, procediamo al plot delle funzioni di base. Le funzioni di base della curva B-Spline vengono calcolate in base al numero di punti di controllo, al grado e al vettore dei nodi. Per ottenere le funzioni di base, utilizziamo la funzione 'bspline_funzioni_base_fun()'. Una volta ottenute le funzioni, vengono plottate.

Attraverso l'implementazione di queste operazioni, otteniamo la curva B-Spline desiderata e il plot delle funzioni di base associate ai punti di controllo selezionati.

Descrizione della funzione 'bspline_deboor_fun()':

Di seguito il codice della funzione.

(Il codice è presente nel file 'bspline_deboor_fun.m').

```
function [C] = bspline_deboor_fun(n, t, P)
    % Fase di validazione dei parametri in ingresso.
    validateattributes(n, {'numeric'}, {'positive','integer','scalar'});
    d = n-1;  % Grado della B-Spline.
    % Il vettore dei nodi deve essere composto da numeri reali.
    validateattributes(t, {'numeric'}, {'real','vector'});
    assert(all(t(2:end) >= t(1:end-1)), ...
        'bspline:deboor:InvalidArgumentValue', ...
        'Il vettore dei nodi deve essere in ordine crescente.');
    % La matrice dei punti di controllo deve essere composto da numeri reali.
    validateattributes(P, {'numeric'}, {'real','2d'});
    % Il numero dei punti di controllo deve essere uguale al numero di elementi
    % del vettore t meno il grado inserito + 2.
    nctrl = numel(t)-(d+1);
    assert(size(P,2) == nctrl, 'bspline:deboor:DimensionMismatch', ...
        ['Numero dei punti di controllo scorretti, %d dati, %d ' ...
        'richiesti.'], size(P,2), nctrl);
    % Vettore U dove valutare la curva B-Spline.
    % Viene creato un vettore 'U' che rappresenta una sequenza di punti in cui valutare
    % la curva B-Spline.
    U = linspace(t(d+1), t(end-d), 10*size(P,2));
    % Questi punti sono distribuiti uniformemente tra il primo nodo e l'ultimo nodo.
    m = size(P,1);
    t = t(:).';
    U = U(:);
    S = sum(bsxfun(@eq, U, t), 2);
     % L'indice I indica i punti di controllo che sono coinvolti nel calcolo della
     % B-Spline ottenuto.
    % tramite la funzione 'bspline_deboor_interval' (spiegazione sotto)
    I = bspline_deboor_interval(U,t);

    Pk = zeros(m,d+1,d+1);
    a = zeros(d+1,d+1);
    C = zeros(size(P,1), numel(U));
    for j = 1:numel(U) % Ricorsione di de Boor
        u = U(j);
        s = S(j);
        ix = I(j);
        Pk(:) = 0;
        a(:) = 0;

        Pk(:, (ix-d):(ix-s), 1) = P(:, (ix-d):(ix-s));
        h = d - s;

        if h > 0
            for r = 1:h
                q = ix-1;
                for i = (q-d+r):(q-s)
                    a(i+1,r+1) = (u-t(i+1)) / (t(i+d-r+1+1)-t(i+1));
                    Pk(:,i+1,r+1) = (1-a(i+1,r+1)) * Pk(:,i,r) + ...
                        a(i+1,r+1) * Pk(:,i+1,r);
                end
            end
            C(:,j) = Pk(:,ix-s,d-s+1);
        elseif ix == numel(t)
            C(:,j) = P(:,end);
        else
            C(:,j) = P(:,ix-d);
        end
    end
end
```

```
% Questa funzione restituisce l'indice del nodo nel vettore dei nodi t che non è
inferiore al valore u.
function ix = bspline_deboor_interval(u,t)
    % Se il nodo ha molteplicità maggiore di 1, viene restituito l'indice più alto.
    i = bsxfun(@ge, u, t) & bsxfun(@lt, u, [t(2:end) 2*t(end)]);
    [row,col] = find(i);
    [row,ind] = sort(row);
    ix = col(ind);
end
```

La funzione 'bspline_deboor_fun()' implementa l'algoritmo di de Boor per valutare le curve B-Spline. Queste curve sono definite da un insieme di punti di controllo che determinano la loro forma.

L'algoritmo consente di calcolare il valore di una curva B-Spline in un punto specifico utilizzando la "ricorsione" per ottenere tale valore.

La funzione ha i seguenti argomenti di input:

- 'n': l'ordine della B-Spline (2 per la lineare, 3 per la quadratica, 4 per la cubica, ecc.).
- 't': il vettore dei nodi (knot vector).
- 'P': la matrice dei punti di controllo.

La funzione restituisce l'output seguente:

- 'C': i punti della curva B-Spline.

La funzione inizia con una fase di validazione dei parametri di input, garantendo che siano nel formato corretto e soddisfino i requisiti.

Successivamente, viene creato un vettore 'U' che rappresenta una sequenza di punti in cui valutare la curva B-Spline. Questi punti sono distribuiti uniformemente tra il primo nodo e l'ultimo nodo. Viene quindi eseguito un ciclo su tutti i punti di valutazione nel vettore 'U'. Per ogni punto, viene determinato l'intervallo corrispondente nel vettore dei nodi utilizzando la funzione 'bspline_deboor_interval()', che restituisce l'indice del nodo che non è inferiore al valore del punto.

Successivamente, viene eseguita la ricorsione di de Boor per calcolare il valore della curva B-Spline nel punto desiderato. Questo coinvolge la selezione dei punti di controllo corrispondenti all'intervallo determinato, l'interpolazione tra i punti di controllo successivi e la riduzione dell'intervallo verso destra. I risultati vengono memorizzati nella matrice 'C' che contiene i punti della curva B-Spline.

La funzione ausiliaria 'bspline_deboor_interval()' viene utilizzata per ottenere l'indice del nodo nel vettore dei nodi che non è inferiore al valore dato. Questa funzione supporta il calcolo dell'intervallo di de Boor.

Complessivamente, la funzione 'bspline_deboor_fun()' implementa l'algoritmo di de Boor per calcolare le curve B-Spline e restituisce i punti della curva risultante.

Descrizione della funzione 'bspline_funzioni_base_fun()':

Di seguito il codice della funzione.

(Il codice è presente nel file 'bspline_funzioni_base_fun.m').

```
function N = bspline_funzioni_base_fun(h,t,n)
Z= linspace( t(1), t(end), 10*n);% Vettore di punti in cui viene calcolata la curva.
% Inizializzazione del vettore bidimensionale che contiene le funzioni di base.
N=zeros(size(Z,2),n);
N_h=zeros(size(Z,2),size(t,2));% Matrice di supporto al calcolo delle funzioni di base.

% Formula di ricorsione di Cox-De Boor.
for i=1:size(Z,2)
    z = Z(i); % La variabile z è il punto in cui viene calcolata la curva.
% Funzioni di base grado 0.
% Costrutto for per tutti i numeri di nodi-1 (l'ultimo nodo non avrà un valore
successivo su cui costruire l'intervallo).
    for j=1:size(t,2)-1
        % Se il punto del linespace è maggiore del j-esimo nodo e minore del j+1esimo.
        if z>=t(j)&&z<t(j+1)
            % Setta il valore ad 1 nella j-esima colonna della matrice di supporto.
            N_h(i,j) = 1;
        else
            N_h(i,j) = 0; % Altrimenti lo setta a 0.
        end
    end
% Funzioni di base di grado successivo.
    for k=2:h    % Iterazione per i gradi successivi.
        for j=1:size(t,2)-k
            if N_h(i,j)==0
                add1 = 0;
            else
                add1 = (((z-t(j))/(t(j+k-1)-t(j)))*N_h(i,j));
            end
            if N_h(i,j+1)==0
                add2 = 0;
            else
                add2 =(((t(j+k)-z)/(t(j+k)-t(j+1)))* N_h(i,j+1));
            end
            N_h(i,j)=add1+add2;
        end
    end

end
N = N_h(:,1:n);
end
```

La funzione 'bspline_funzioni_base_fun()' calcola le funzioni di base della curva B-Spline a partire dal vettore dei nodi fornito in ingresso.

La funzione ha i seguenti argomenti di input:

- 'h': il grado della curva B-Spline.
- 't': il vettore dei nodi (knot vector).
- 'n': il numero dei punti di controllo.

La funzione restituisce l'output seguente:

- 'N': una matrice contenente le funzioni di base.

Il codice inizia creando un vettore 'Z' che rappresenta una sequenza di punti in cui viene calcolata la curva B-Spline. Questi punti sono distribuiti uniformemente tra il primo nodo e l'ultimo nodo.

Vengono quindi inizializzate le matrici 'N' e 'N_h'. 'N' conterrà le funzioni di base della curva B-Spline, mentre 'N_h' è una matrice di supporto utilizzata per calcolare le funzioni di base.

Successivamente, viene eseguito un ciclo su tutti i punti nel vettore 'Z'. Per ogni punto, viene eseguito il calcolo delle funzioni di base tramite la formula di ricorsione di Cox-De Boor.

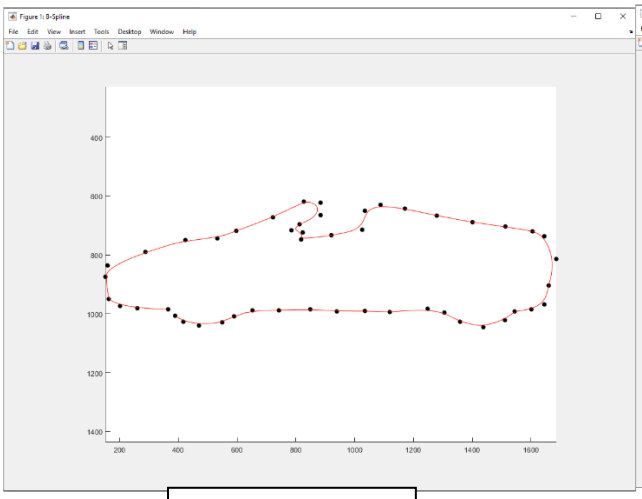
Inizia con il calcolo delle funzioni di base di grado 0. Viene utilizzato un ciclo for per iterare su tutti i numeri di nodi meno uno. Se il punto 'z' è compreso tra il j-esimo e il j+1-esimo nodo, viene impostato il valore della funzione di base nella colonna corrispondente della matrice di supporto 'N_h' a 1; altrimenti, viene impostato a 0.

Successivamente, vengono calcolate le funzioni di base per i gradi successivi utilizzando un altro ciclo for. Viene effettuata una serie di calcoli per ogni coppia di punti di controllo, in cui si considerano i valori delle funzioni di base di grado precedente nella matrice di supporto 'N_h'. Questi calcoli seguono la formula di ricorsione di Cox-De Boor.

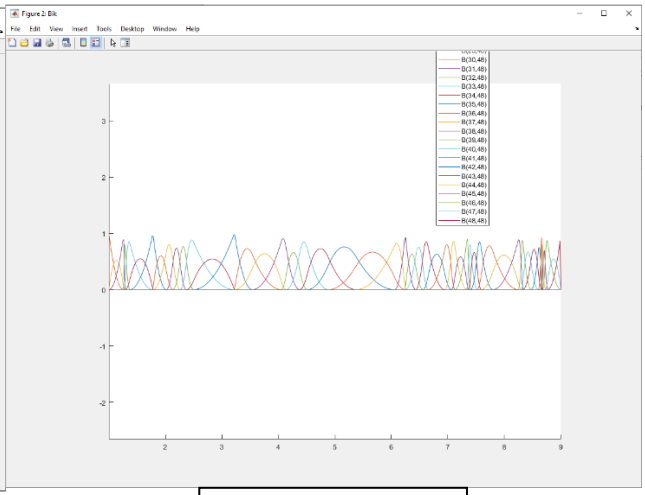
Alla fine del ciclo, la matrice 'N_h' contiene le funzioni di base complete. La matrice 'N' viene quindi assegnata come una porzione di 'N_h' che include solo le prime 'n' colonne, corrispondenti ai punti di controllo.

In definitiva, la funzione 'bspline_funzioni_base_fun()' calcola le funzioni di base della curva B-Spline necessarie per costruire la curva stessa. Queste funzioni di base sono calcolate in base al grado della curva, al vettore dei nodi e al numero dei punti di controllo.

Immagini di esempio:



Curva B-Spline



Funzioni di base

5.4 Curve NURBS

Di seguito il codice dello script Matlab per la risoluzione del problema di interpolazione tramite le NURBS.

(Il codice è contenuto nel file ‘nurbs.m’).

```
clc;
clear;

% Modello di interpolazione NURBS
figure ('Name', 'Seleziona i punti di controllo');
imshow ('Ferrari_488_Spider.jpg');
[x, y] = getpts ();
close;

% Configurazione degli assi con asse Y invertito
figure ('Name', 'NURBS');
hold on;
axis ('equal');
set (gca, 'Ydir', 'reverse');
nodes = plot (x, y, '*');
set (nodes, 'Color', 'k', 'LineWidth', 2);

%Nurbs con algoritmo di DeBoor
n = 4;
grado = n-1;
P = [x,y]';
t = [zeros(grado + 1, 1)' sort(rand(size(x, 1) - grado - 1, 1))' ...       % Knots a
punti fissi
    ones(grado + 1, 1)'];
                                                                           % Knots a
punti variabili
%t = sort(rand(length(x) + grado + 1, 1))'
w = 10*rand(size(x, 1), 1)';                                               % Vettore
casuale dei pesi
Y = nurbs_deboor_fun(n,t,P,w);
hold on;
plot(Y(1,:), Y(2,:), 'r');
hold off;

%Plot delle funzioni base
figure('Name','Nik')
hold on;
axis('equal')
N = bspline_funzioni_base_fun(grado, t, size(x,1));
R = (N.*w)./sum(N.*w);
for j = 1 : size(x, 1)
    plot(linspace(1, 9, 10*size(x, 1)), R(:, j));
end
hold off;
legend(cellstr(strcat('N(', num2str((0:length(x)-1)'), ',', ...
    num2str(length(x)-1), ')')));
```

Guida all'utilizzo:

Indicando il percorso di un'immagine nella funzione 'imshow()', è possibile tracciare i punti di controllo su di essa.

Una volta tracciati i punti di controllo e premendo Invio, verrà generata una figura contenente il plot della NURBS e una figura contenente il plot di tutte le funzioni di base.

Descrizione del codice:

Una volta selezionati i punti di controllo, vengono definite le seguenti variabili per l'applicazione delle NURBS:

- 'n': il grado della curva, che definisce il tipo di B-Spline da utilizzare (2 per lineare, 3 per quadratica, 4 per cubica, ecc.).
- 'grado': 'n-1', utilizzato per calcolare la dimensione del vettore dei nodi.
- 'P': la matrice dei punti di controllo che contiene le coordinate x e y.
- 't': il vettore dei nodi, definito in due modi a seconda se si desidera fissare i punti di controllo o meno. Se i punti di controllo sono fissati, il vettore sarà costituito dai valori 0 e 1 per i primi e ultimi nodi, mentre i valori centrali saranno numeri casuali compresi tra 0 e 1. Se i punti non sono fissati, il vettore dei nodi sarà composto interamente da numeri casuali che variano da 0 a 1. In entrambi i casi, il vettore dei nodi deve essere ordinato in modo crescente.
- 'w': il vettore dei pesi, è composto da valori randomici e la dimensione è pari al numero di punti di controllo selezionati.

Una volta definite le variabili, possiamo applicare l'algoritmo di de Boor per ottenere le NURBS, tramite la funzione 'nurbs_deboor_fun()'. Ottenuta la curva, la plottiamo.

Successivamente, procediamo al plot delle funzioni di base. Le funzioni di base delle NURBS vengono calcolate in base al numero di punti di controllo, al grado e al vettore dei nodi. Per ottenere le funzioni di base, utilizziamo la funzione 'bspline_funzioni_base_fun()'. Una volta ottenute le funzioni devono essere normalizzate per cui moltiplichiamo ogni funzione per il suo corrispettivo peso e lo dividiamo per la somma delle funzioni moltiplicate il loro peso. Infine, le funzioni ottenute le plottiamo.

Attraverso l'implementazione di queste operazioni, otteniamo le NURBS e il plot delle funzioni di base associate ai punti di controllo selezionati.

Descrizione della funzione 'nurbs_deboor_fun()':

Di seguito il codice della funzione.

(Il codice è presente nel file 'nurbs_deboor_fun.m').

```
function [C] = nurbs_deboor_fun(n,t,P,w)
       w = transpose(w(:));
       P = bsxfun(@times, P, w);
       P = [P ; w];

       [Y] = bspline_deboor_fun(n,t,P);

       C = bsxfun(@rdivide, Y(1:end-1,:), Y(end,:));
end
```

La funzione 'nurbs_deboor_fun()' calcola le curve NURBS utilizzando l'algoritmo di de Boor, lo stesso algoritmo utilizzato per il calcolo delle B-Spline.

Prima di poter applicare l'algoritmo di de Boor, è necessario effettuare una trasformazione dei punti di controllo a due dimensioni, utilizzati per definire le B-Spline, in punti di controllo a tre dimensioni. Questa trasformazione avviene come segue:

- Le prime due dimensioni dei punti di controllo a tre dimensioni corrispondono alle coordinate dei punti di controllo originali moltiplicate per il peso del punto di controllo stesso.
- La terza dimensione rappresenta il peso del punto di controllo stesso.

Dopo aver effettuato la trasformazione dei punti di controllo a tre dimensioni, viene applicato l'algoritmo di de Boor ai punti di controllo a tre dimensioni per calcolare le curve NURBS. I risultati ottenuti sono proiettati nuovamente in due dimensioni per ottenere le curve NURBS finali.

In sintesi, la trasformazione dei punti di controllo a tre dimensioni incorpora i pesi dei punti di controllo e consente l'utilizzo dell'algoritmo di de Boor per il calcolo delle curve NURBS, che sono rappresentate in due dimensioni.

La funzione 'nurbs_deboor_fun()' ha i seguenti argomenti di input:

- 'n': il grado della curva NURBS.

- 't': il vettore dei nodi (knot vector).
- 'P': la matrice dei punti di controllo.
- 'w': il vettore dei pesi, con una dimensione uguale al numero dei punti di controllo.

La funzione restituisce l'output seguente:

- 'C': la curva NURBS risultante.

All'interno del codice, viene prima trasposto il vettore dei pesi in modo che sia un vettore colonna. Successivamente, la matrice dei punti di controllo viene moltiplicata per il vettore dei pesi utilizzando la funzione 'bsxfun(@times, P, w)', in modo che le prime due dimensioni dei punti di controllo siano moltiplicate per il peso corrispondente.

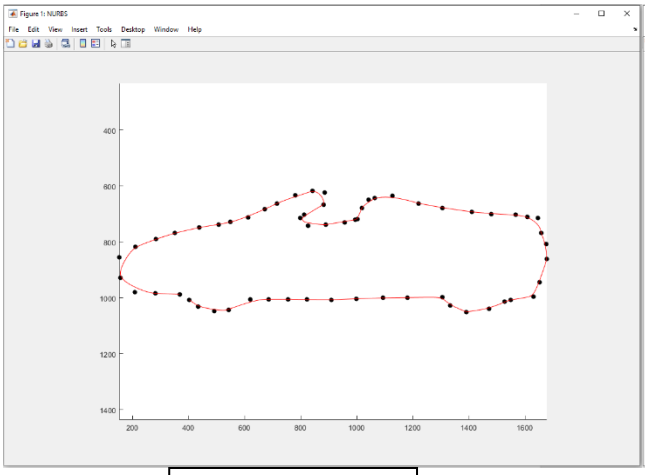
Successivamente, viene aggiunta una terza dimensione alla matrice dei punti di controllo utilizzando '[P ; w]'. In questo modo, i punti di controllo sono rappresentati in tre dimensioni, con le prime due dimensioni corrispondenti alle coordinate moltiplicate per il peso e la terza dimensione corrispondente al peso stesso.

Successivamente, viene chiamata la funzione 'bspline_deboor_fun()' per calcolare la curva utilizzando l'algoritmo di de Boor. Questa funzione restituisce i punti della curva B-Spline.

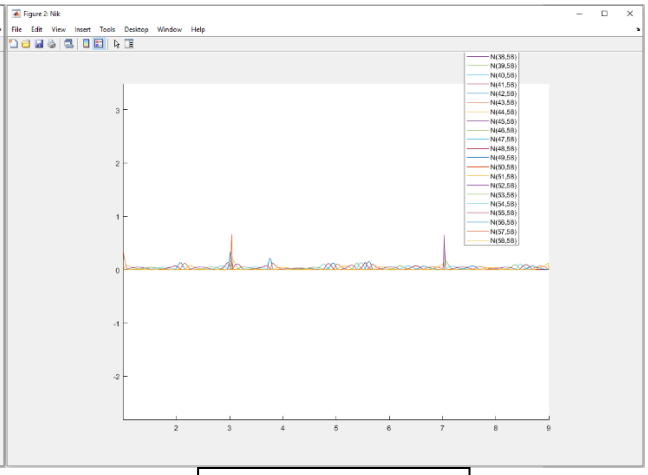
Infine, la matrice dei punti della curva B-Spline viene divisa per il valore dell'ultima riga corrispondente ai pesi dei punti di controllo utilizzando 'bsxfun(@rdivide, Y(1:end-1,:), Y(end,:))'. Questa operazione proietta i punti della curva B-Spline risultante nuovamente in due dimensioni, ottenendole curve NURBS finali.

Complessivamente, la funzione 'nurbs_deboor_fun()' trasforma i punti di controllo a due dimensioni in punti di controllo a tre dimensioni incorporando i pesi dei punti di controllo. Successivamente, utilizza l'algoritmo di de Boor per calcolare le curve NURBS, che vengono proiettate nuovamente in due dimensioni per ottenere il risultato finale. Questa funzione è utilizzata nel contesto di calcolo delle curve NURBS all'interno del file 'nurbs.m' e nell'applicazione UI.

Immagini di esempio:



Curva NURBS



Funzioni di base

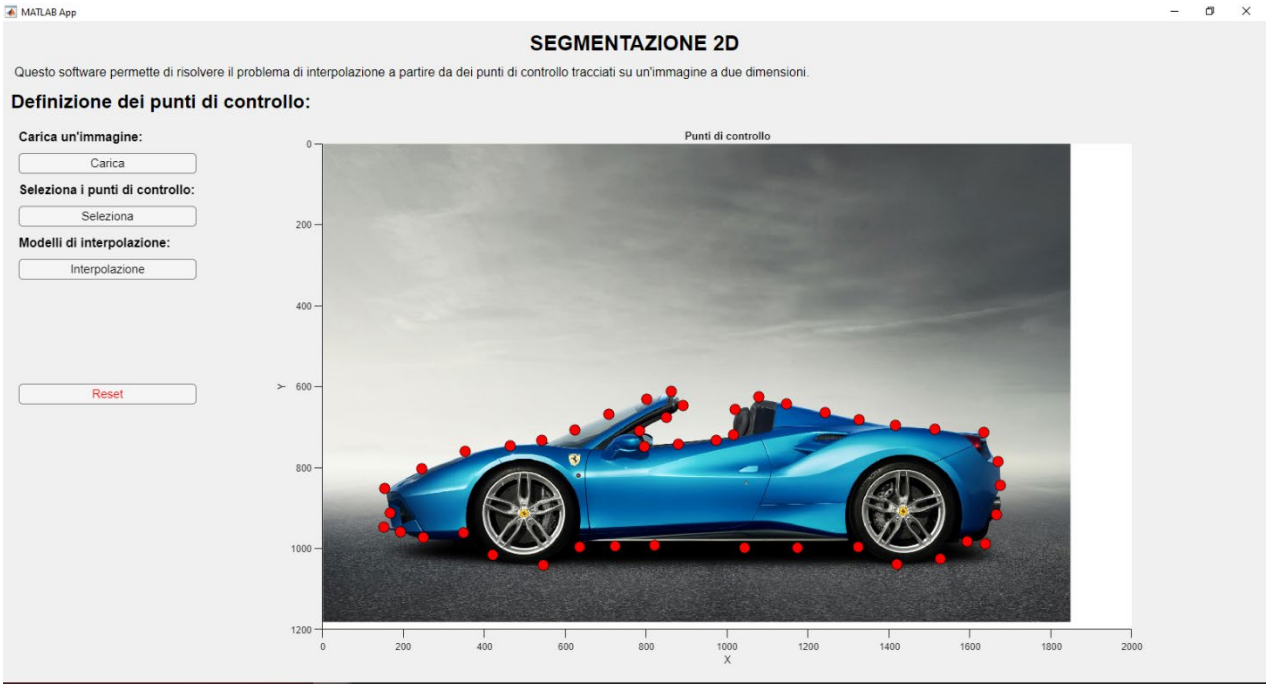
Capitolo 6: Applicazione con intefaccia grafica

Per la risoluzione dei problemi di interpolazione documentati precedentemente, abbiamo sviluppato un'applicazione in Matlab che permette di applicare le varie metodologie in maniera semplice ed intuitiva.

L'applicazione può essere eseguita aprendo il file 'app1.mlapp' oppure può essere installata tra le app di Matlab tramite il pacchetto di installazione 'Segmentazione2D.mlappinstall' presente nella cartella.

6.1 Selezione dei punti di controllo

All'avvio l'applicazione mostra una finestra dove è possibile caricare un'immagine e selezionare i punti di controllo.



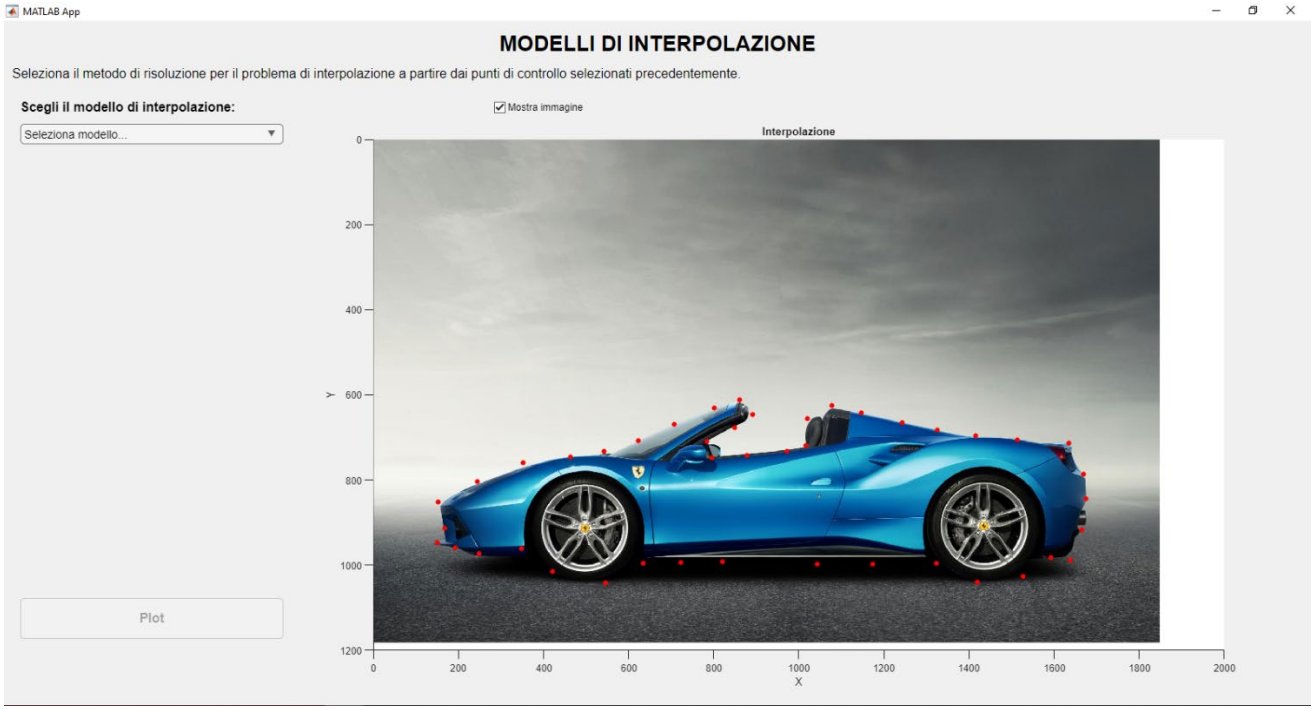
Se i punti di controllo vengono selezionati al di fuori dell'area definita dagli assi, viene

mostrato un errore con il ripristino dell'immagine originale.

Una volta selezionati i punti di controllo, cliccando sul bottone 'Interpolazione', si può procedere con l'applicazione dei vari modelli visti in precedenza.

6.2 Interpolazione

Una volta cliccato il bottone 'Interpolazione', verrà generata una finestra dove è possibile selezionare un modello tramite il menu a tendina.



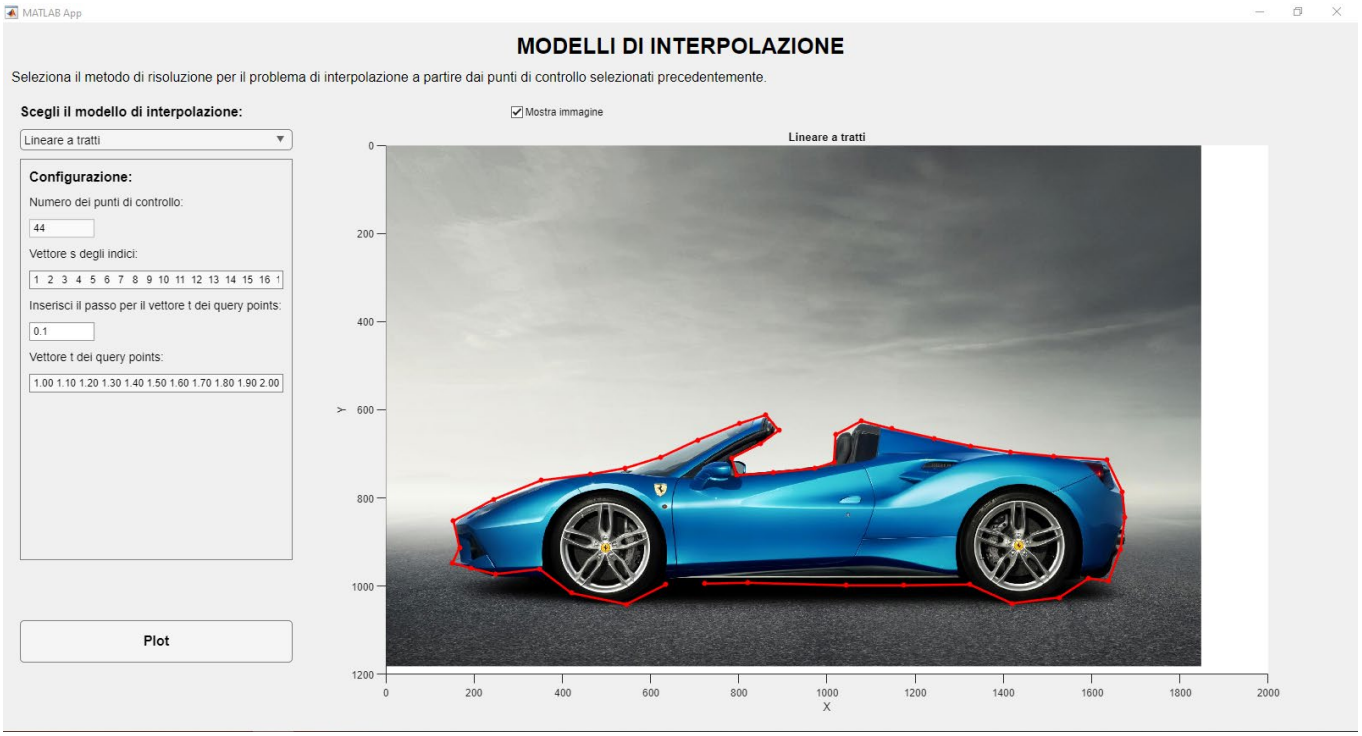
In base al modello scelto, verrà generato un pannello di configurazione differente in modo tale da poter applicare eventuali variazioni ai parametri.

6.3 Lineare a tratti

Selezionando il modello lineare a tratti, verrà mostrato un pannello di configurazione che presenta le seguenti informazioni:

- Il numero dei punti di controllo;
- Il vettore ‘s’ degli indici dei punti di controllo;
- Il passo utilizzato per il vettore ‘t’ dei query points, il quale è modificabile;
- Il vettore ‘t’ dei query points.

Cliccando sul pulsante Plot, verrà effettuata l’interpolazione.



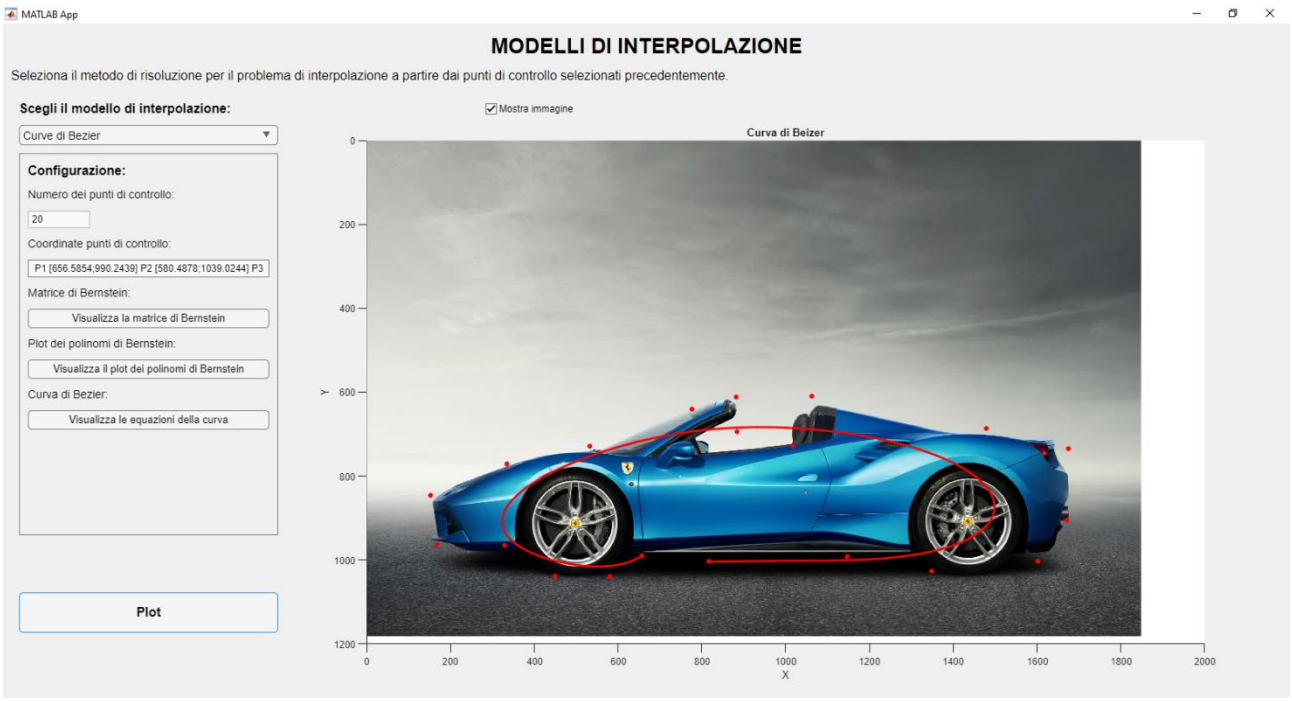
Modificando il passo dei query points, il vettore t verrà generato automaticamente e cliccando il pulsante Plot, possiamo vedere il risultato direttamente sugli assi.

6.4 Curva di Bezier

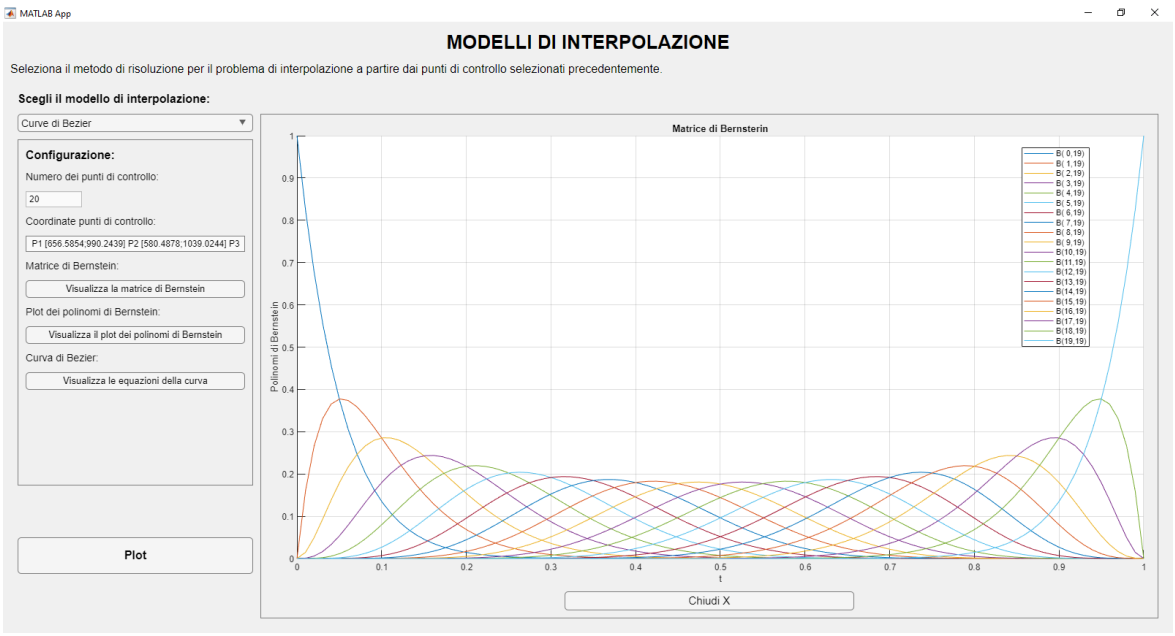
Selezionando dal menù a tendina il modello della curva di Bezier, verrà mostrato il pannello di configurazione con i seguenti parametri:

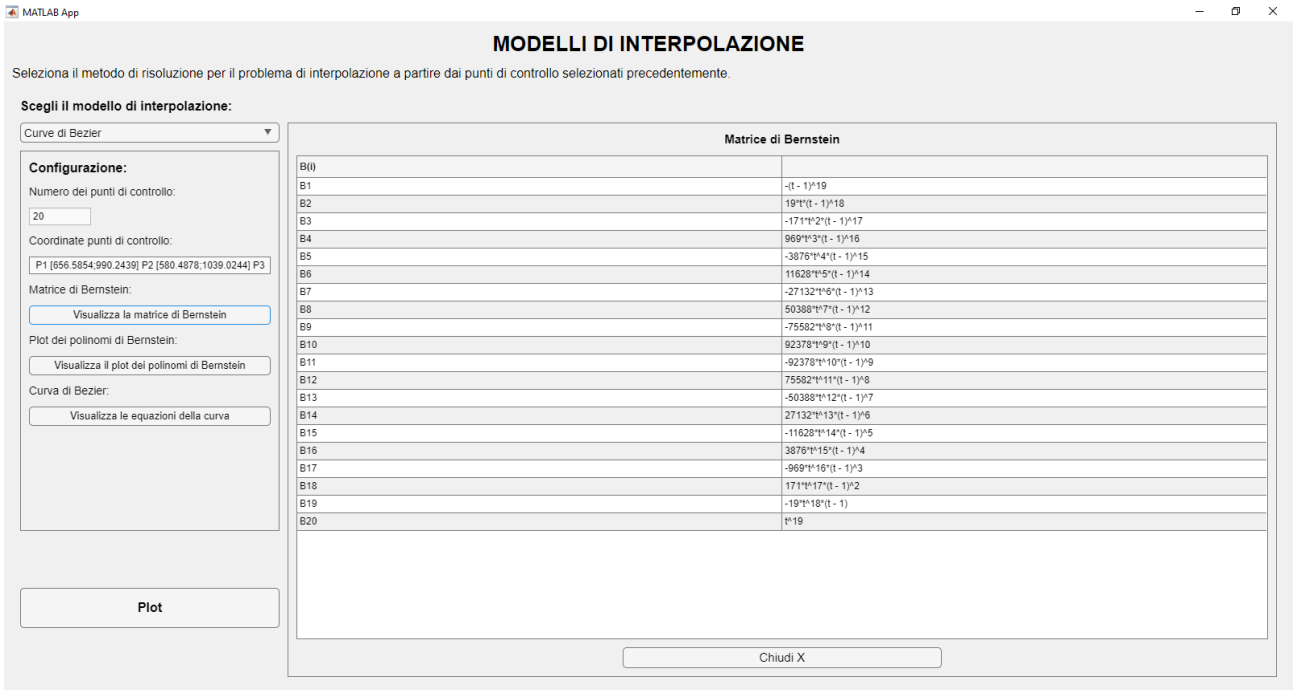
- Il numero dei punti di controllo;
- La matrice dei punti di controllo.

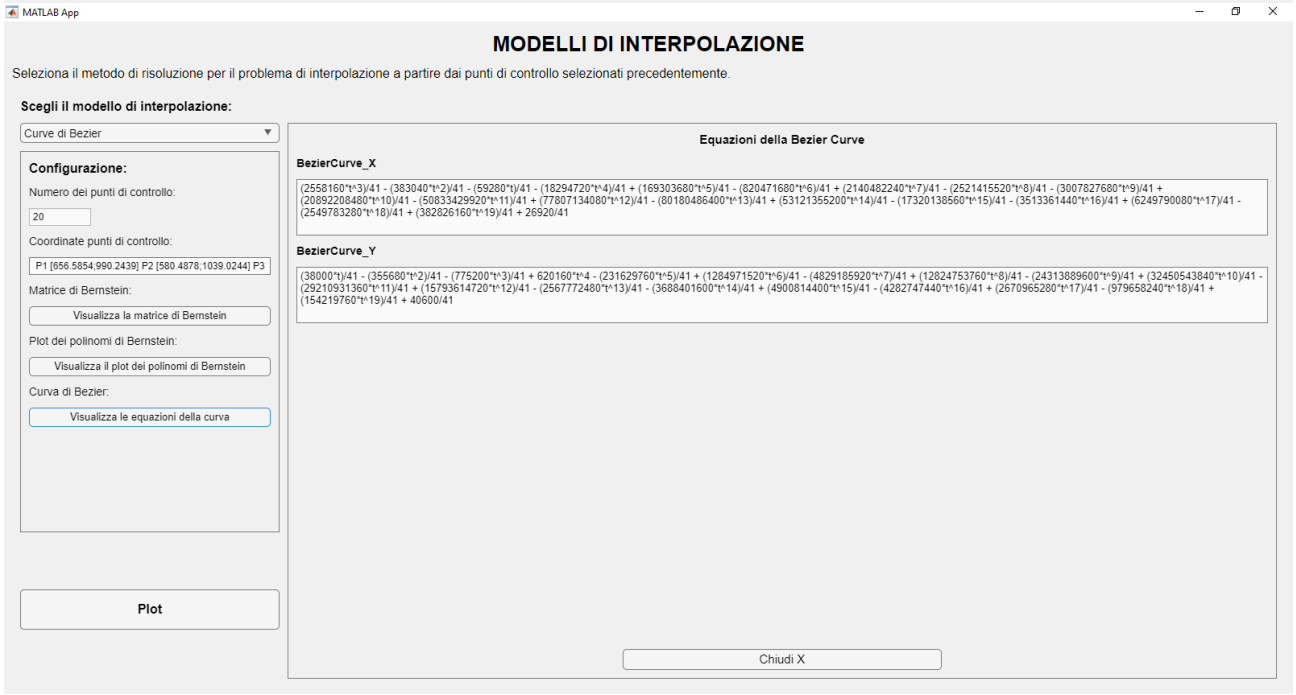
Cliccando il pulsante Plot, verrà visualizzata la curva direttamente sugli assi.



Inoltre, vi è la possibilità di visualizzare la matrice di Bernstein, il plot dei polinomi di Bernstein e le equazioni della curva di Bezier, tramite gli appositi pulsanti.




MATLAB App
MODELLI DI INTERPOLAZIONE
Seleziona il metodo di risoluzione per il problema di interpolazione a partire dai punti di controllo selezionati precedentemente.
Scegli il modello di interpolazione:
Curve di Bezier
Configurazione:
Numero dei punti di controllo:
20
Coordinate punti di controllo:
P1 [656.5854;990.2439] P2 [580.4878;1039.0244] P3
Matrice di Bernstein:
Visualizza la matrice di Bernstein
Plot dei polinomi di Bernstein:
Visualizza il plot dei polinomi di Bernstein
Curva di Bezier:
Visualizza le equazioni della curva
Plot
Matrice di Bernstein
B(i)
B1 -(t - 1)^19
B2 19*t*(t - 1)^18
B3 -171*t^2*(t - 1)^17
B4 969*t^3*(t - 1)^16
B5 -3876*t^4*(t - 1)^15
B6 11628*t^5*(t - 1)^14
B7 -27132*t^6*(t - 1)^13
B8 50388*t^7*(t - 1)^12
B9 -75582*t^8*(t - 1)^11
B10 92378*t^9*(t - 1)^10
B11 -92378*t^10*(t - 1)^9
B12 75582*t^11*(t - 1)^8
B13 -50388*t^12*(t - 1)^7
B14 27132*t^13*(t - 1)^6
B15 -11628*t^14*(t - 1)^5
B16 3876*t^15*(t - 1)^4
B17 -969*t^16*(t - 1)^3
B18 171*t^17*(t - 1)^2
B19 -19*t^18*(t - 1)
B20 t^19
Chiudi X

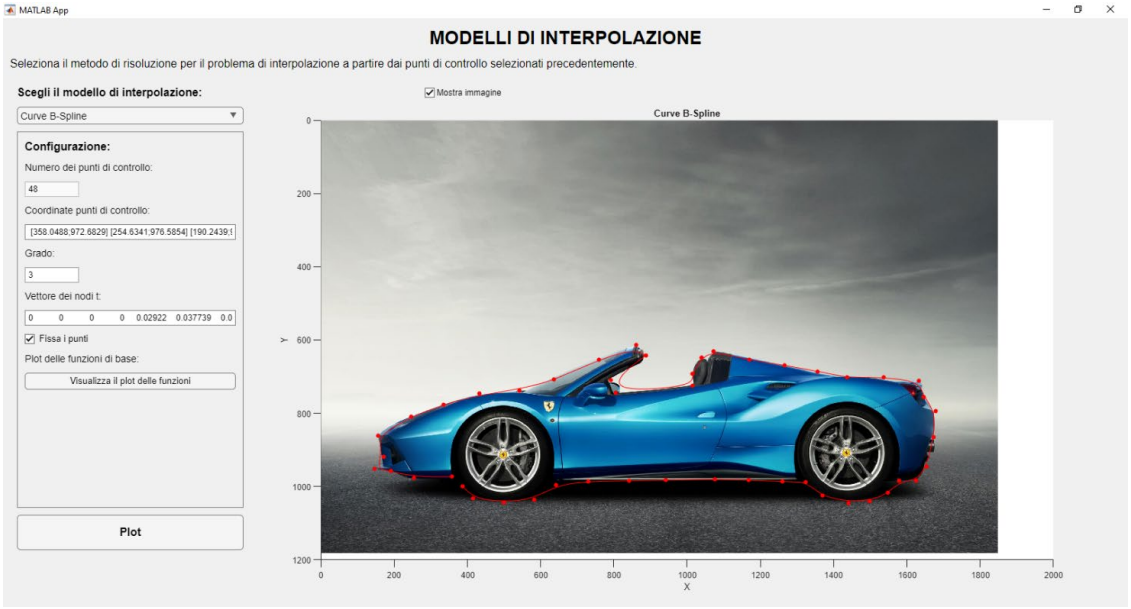

MATLAB App
MODELLI DI INTERPOLAZIONE
Seleziona il metodo di risoluzione per il problema di interpolazione a partire dai punti di controllo selezionati precedentemente.
Scegli il modello di interpolazione:
Curve di Bezier
Configurazione:
Numero dei punti di controllo:
20
Coordinate punti di controllo:
P1 [656.5854;990.2439] P2 [580.4878;1039.0244] P3
Matrice di Bernstein:
Visualizza la matrice di Bernstein
Plot dei polinomi di Bernstein:
Visualizza il plot dei polinomi di Bernstein
Curva di Bezier:
Visualizza le equazioni della curva
Plot
Equazioni della Bezier Curve
BezierCurve_X
(2558160*t^3)/41 - (383040*t^2)/41 - (59280*t)/41 - (18294720*t^4)/41 + (169303680*t^5)/41 - (820471680*t^6)/41 + (2140482240*t^7)/41 - (2521415520*t^8)/41 - (3007827680*t^9)/41 + (20892208480*t^10)/41 - (50833429920*t^11)/41 + (77807134080*t^12)/41 - (80180486400*t^13)/41 + (53121355200*t^14)/41 - (17320138560*t^15)/41 - (3513361440*t^16)/41 + (6249790080*t^17)/41 - (2549783280*t^18)/41 + (382826160*t^19)/41 + 26920/41
BezierCurve_Y
(38000*t)/41 - (355680*t^2)/41 - (775200*t^3)/41 + 620160*t^4 - (231629760*t^5)/41 + (1284971520*t^6)/41 - (4829185920*t^7)/41 + (12824753760*t^8)/41 - (24313889600*t^9)/41 + (32450543840*t^10)/41 - (29210931360*t^11)/41 + (15793614720*t^12)/41 - (2567772480*t^13)/41 - (3688401600*t^14)/41 + (4900814400*t^15)/41 - (4282747440*t^16)/41 + (2670965280*t^17)/41 - (979658240*t^18)/41 + (154219760*t^19)/41 + 40600/41
Chiudi X

6.5 Curve B-Spline

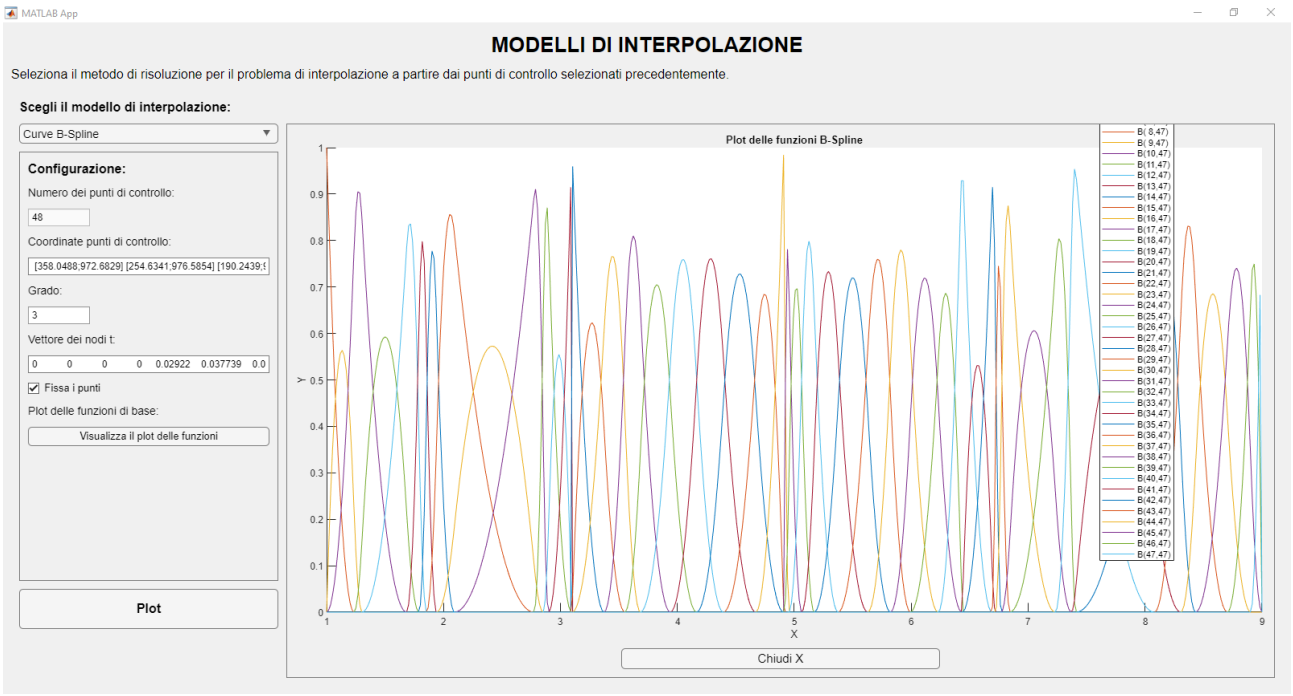
Selezionando Curve B-Spline dal menù a tendina, verrà mostrato il pannello di configurazione contenente le seguenti informazioni:

- Numero dei punti di controllo
- Vettore dei punti di controllo dove ogni punto presenta il formato [x;y]. Questo vettore è modificabile e in base alla modifica del punto, questo verrà posizionato sugli assi. Se viene alterato il formato del nodo oppure viene eliminato o aggiunto un nodo, verrà restituito un errore;
- Il grado della curva, questo vettore è modificabile in base a quale tipo di curva vogliamo (1 lineare, 2 quadratica, 3 cubica…);
- Il vettore t dei nodi, questo vettore viene generato automaticamente in base al grado inserito. Su questo vettore abbiamo inserito dei controlli per verificare se il formato è corretto. Il vettore deve essere in ordine crescente e deve contenere [(numero di nodi)-1]+grado+2 elementi;
- Fissa i punti, è un checkbox che genera un vettore t in base a se vogliamo fissare i punti di controllo o meno. Se il valore è check allora il vettore t sarà costituito da valori iniziali pari a 0, valori finali pari ad 1 e valori centrali casuali, sempre ordinato in modo crescente. Se il valore è uncheck, il vettore dei nodi sarà costituito da valori casuali compresi tra 0 e 1, sempre ordinato in modo crescente.

Cliccando sul pulsante Plot, possiamo visualizzare la curva B-Spline sugli assi.



Nel pannello di configurazione vi è la possibilità, tramite l'apposito bottone, di visualizzare le funzioni di base sugli assi.



6.6 Curve NURBS

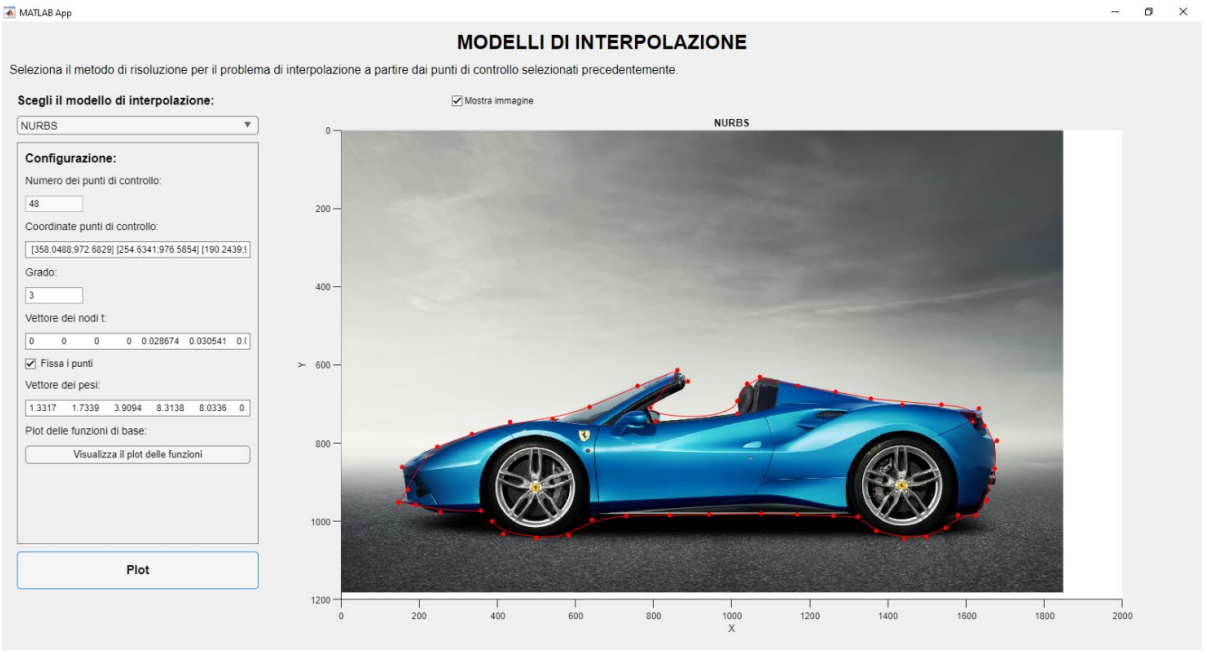
Selezionando NURBS dal menù a tendina, verrà visualizzato un pannello di configurazione con le seguenti informazioni:

- Numero dei punti di controllo;
- Vettore dei punti di controllo dove ogni punto presenta il formato [x;y]. Questo vettore è modificabile e in base alla modifica del punto, questo verrà posizionato sugli assi. Se viene alterato il formato del nodo oppure viene eliminato o aggiunto un nodo, verrà restituito un errore;
- Il grado della curva, questo vettore è modificabile in base a quale tipo di curva vogliamo (1 lineare, 2 quadratica, 3 cubica…);
- Il vettore t dei nodi, questo vettore viene generato automaticamente in base al grado inserito. Su questo vettore abbiamo inserito dei controlli per verificare se il formato è corretto. Il vettore deve essere in ordine crescente, deve contenere [(numero di nodi)-1]+grado+2 elementi;
- Fissa i punti, è un checkbox che ci genera un vettore t in base a se vogliamo fissare i

punti di controllo o meno. Se il valore è check allora il vettore t sarà costituito da valori iniziali pari a 0, valori finali pari ad 1 e valori centrali casuali, sempre ordinato in modo crescente. Se il valore è uncheck, il vettore dei nodi sarà costituito da valori casuali compresi tra 0 e 1, sempre ordinato in modo crescente;

- Vettore dei pesi 'w'. Questo vettore è generato automaticamente ma è anche modificabile, l'unico controllo effettuato è che deve contenere un numero di pesi pari al numero di punti di controllo selezionati.

Cliccando sul pulsante Plot è possibile visualizzare la curva sugli assi.



Inoltre, è possibile visualizzare le funzioni di base tramite l'apposito bottone presente nel pannello di configurazione.