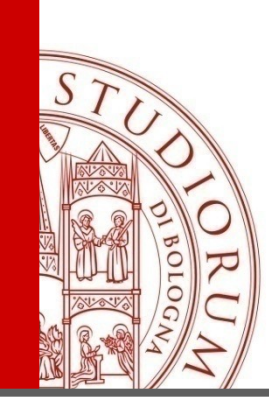
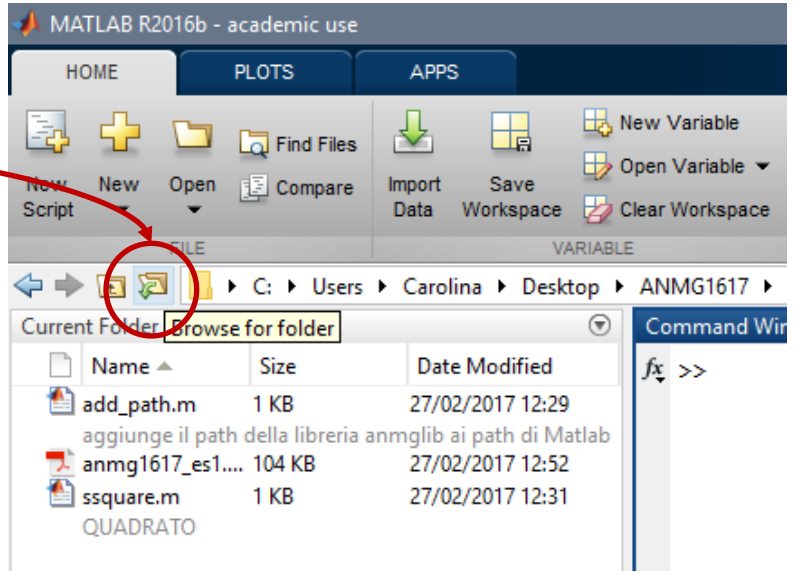


Valutazione Polinomiale e Curve 2D

Esercitazione 4

Ogni Volta all'inizio

1. Usando
Browse for Folder
rendere corrente la cartella
matlab_mnc2324_4



2. Dalla command window dare il comando
>>add_path <invio>

Ora siamo pronti per lavorare ☺

Polinomi in Matlab/Octave

Un polinomio in Matlab/Octave viene rappresentato mediante un array contenente i suoi coefficienti.

a è un array riga di lunghezza $n+1$ che contiene i coefficienti del polinomio $p(x)$ di grado n ordinati con potenze decrescenti:

$$p(x) = a(1)*x^n + a(2)*x^{n-1} + \ldots + a(n)*x^1 + a(n+1)$$

per esempio, il polinomio

$$1x^3 + 2x^2 + 7x + 5$$

viene rappresentato mediante l'array

$$a = [1\ 2\ 7\ 5]$$

Function polyval

Il comando Matlab polyval determina il risultato della valutazione del polinomio $p(x)$ in corrispondenza di un'ascissa o di un'array di ascisse

a = [1 2 7 5];

x = linspace (0, 1, 100);

y = polyval (a, x);

y contiene i valori del polinomio $p(x)$ in corrispondenza degli elementi dell'array x

anmglib_4.0 e funzioni

Nella libreria anmglib_4.0 le funzioni nella base di Bernstein sono memorizzate in una structure di Matlab. Una structure è un contenitore di più dati chiamati campi. Ogni campo può contenere qualsiasi tipo di dato. Si può accedere ai dati di un campo usando la notazione

NomeStruttura.NomeCampo

Se chiamiamo bern la structure di una funzione nella base di Bernstein, allora avremo:

bern.deg : campo scalare per il grado

bern.cp : campo array per i punti di controllo (array (n+1) x 1)

bern.ab : campo array per intervallo di definizione (array di 2)

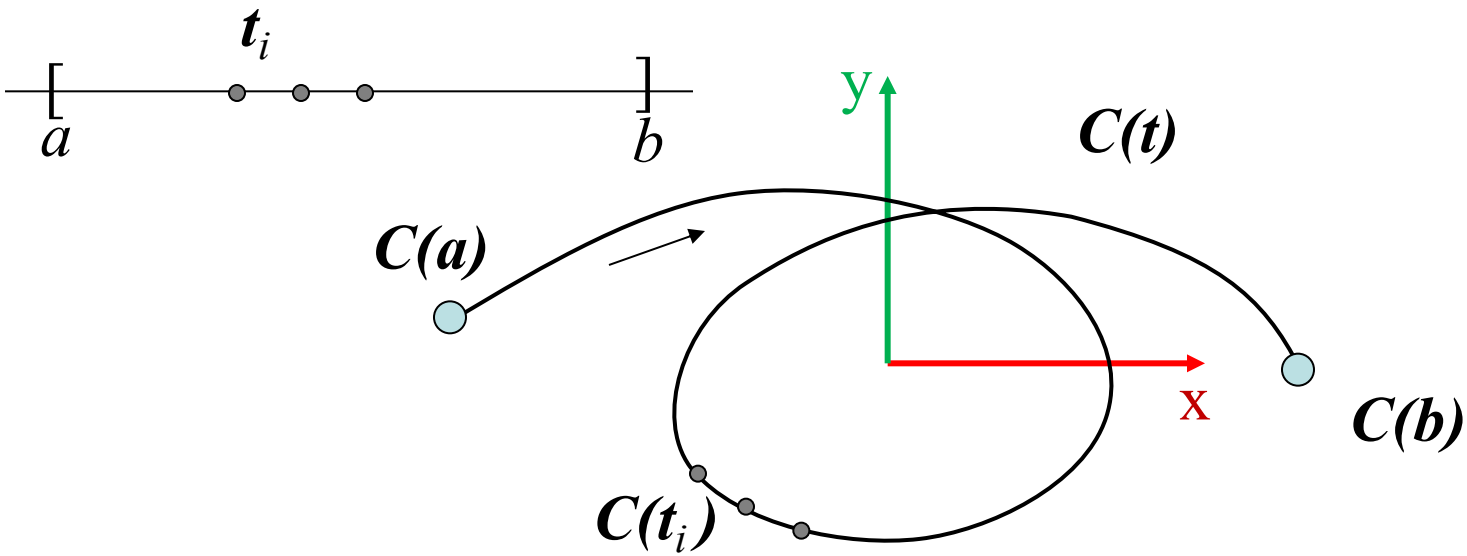
Esercizi B.3, B.4 e B.5

Si tratta di tre esercizi sulla valutazione di funzioni polinomiali nella base di Bernstein utilizzando gli Alg.1 e Alg.2 implementati in funzioni del toolbox anmglib_4.0.

Si poi abbiamo la necessità di determinare e valutare le funzioni derivate prime, si può procedere in più modi differenti.

Curve 2D in forma parametrica

Una curva nel piano è una funzione vettoriale
$C(t) = (x(t), y(t))^T$ del parametro $t \in [a,b]$
Al variare di t, le coordinate $(x(t), y(t))$ rappresentano un punto 2D che si muove lungo la curva



$C([a,b])$ definisce un segmento di curva.

Rappresentazione di curve per punti

Si vuole disegnare una curva 2D data in forma parametrica.

La curva è definita da infiniti punti e non è possibile determinarli e disegnarli tutti in un tempo finito!

Allora si valutano *np* punti della curva in corrispondenza di *np* valori t_i dell'intervallo di definizione *[a , b]* e si disegna la spezzata da loro definita come approssimazione della curva.

Un problema non semplice è determinare un valore opportuno per *np* affinché si abbia una buona rappresentazione.

Esercizio B.1

Si disegni la seguente curva piana chiusa (cardioide).

$$C(t) = (2\cos(t) - \cos(2t),\ 2\sin(t) - \sin(2t))^T \quad t \in [0, 2\pi]$$

1) Per usare la function curv2_plot si deve scrivere una function Matlab con l'espressione parametrica della curva (la chiameremo c2_cardioide.m)

Lo script scardio.m :

- chiama una funzione per aprire una finestra grafica
- chiama una funzione per disegnare gli assi
- chiama una funzione per disegnare la curva

Creare uno script

Usare l'Editor per completare c2_cardioide.m:

```
function [x,y]=c2_cardioide(t)
%espressione parametrica della cardioide con
%t in [0,2*pi]
x = 2*cos(t)-cos(2*t);
y = 2*sin(t)-sin(2*t);
end
```

- Salvare la function usando 'salva con nome' e dando 'c2_cardioide' come nome

Eseguire lo script

Dalla Command Window dare il comando:

```
>>scardio
```

- MATLAB cerca nella cartella corrente se esiste un M-file con questo nome
- Se lo trova lo carica in memoria e lo esegue, cioè esegue istruzione per istruzione il contenuto dello script.

Attenzione

Una volta ottenuto uno script che produce risultati corretti, lo si usi per effettuare altre esecuzioni con dati differenti;

Per esempio, sullo script scardio.m si modifichino gli estremi dell'intervallo di definizione, il numero di punti di plotting e la line specification.

Modificare lo script

Usare l'Editor per modificare scardio.m:

```
open_figure(1);
axis_plot(2);
a=0;  b=pi;  np=20;
curv2_plot('c2_cardioide',a,b,np,'b-');
```

- Salvare lo script usando 'salva'
- Eseguire più volte assegnado differenti valori per a, b ed np.

Disegno della tangente

Si disegni la curva piana chiusa (cardioide), il punto $P_0 = C(t_0)$ e la retta tangente alla curva nel punto P_0.

1) Per la retta tangente di una curva si deve scrivere una function Matlab con l'espressione della sua derivata (la chiameremo cp2_cardioide.m)

2) Si scriva lo script scardio_tan.m tale che:

- chiami una funzione per aprire una finestra grafica
- chiami una funzione per disegnare gli assi
- chiami una funzione per disegnare la curva
- chiama una funzione per disegnare il punto P_0
- chiami una funzione per disegnare la retta tangente in P_0

Retta in forma parametrica definita da P_0 e v

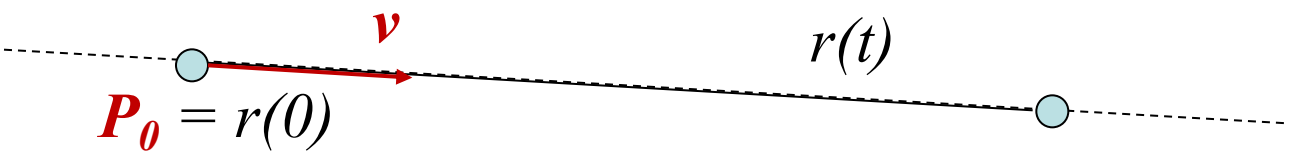
$$r(t) = (\, x(t),\, y(t)\,)^T = P_0 + t\, v \quad t \in [0,\, 1]$$

$$P_0 = (x_0,\, y_0)^T$$

$$v = (x_v,\, y_v)^T$$

$$x(t) = x_0 + t\, x_v$$

$$y(t) = y_0 + t\, y_v$$



se $P_0 = C(t_0)$, $v = C'(t_0)$

allora retta tangente alla curva $C(t)$

Creare uno script

Usare l'Editor per completare cp2_cardioide.m:

```
function [x,y,xp,yp]=cp2_cardioide(t)
%espressione della cardioide e della derivata
x = 2*cos(t)-cos(2*t);
y = 2*sin(t)-sin(2*t);
xp= … ;
yp= … ;
end
```

- Salvare la function

Derivata di una funzione

Dobbiamo saper fare la derivata di una funzione!

Matlab ci aiuta anche in questo caso; da command window dare:

```
>>syms t
>>diff(2*cos(t)-cos(2*t))
ans =
2*sin(2*t) - 2*sin(t)
>>diff(2*sin(t)-sin(2*t))
ans =
2*cos(t) - 2*cos(2*t)
```

Modificare lo script

Usare l'Editor per modificare scardio.m in scardio_tan.m

```
open_figure(1);
axis_plot(2);
a=0;  b=pi;  np=20;  tbar=pi/2;
curv2_plot('c2_cardioide',a,b,np,'b-');
[P0(1),P0(2)]=c2_cardioide(tbar);
point_plot(P0,'bo');
[P0(1),P0(2),vT(1),vT(2)]=cp2_cardioide(tbar);
%il vettore vT e' normalizzato?
line_plot(P0,vT,-3,3,'r-');
```

Modificare lo script

Usare l'Editor per modificare scardio_tan.m in scardio_tan_lib.m e disegnare il vettore tangente alla curva in un punto

```
open_figure(1);
axis_plot(2);
a=0;  b=pi;  np=20;  tbar=pi/2;
curv2_plot('c2_cardioide',a,b,np,'b-');
[px,py]=curv2_plot('c2_cardioide',tbar,tbar,1,'bo');
[tx,ty]=curv2_tan_plot('cp2_cardioide',tbar,tbar,1,'r-');
.....
```

- Salvare il file nella cartella

Disegno della normale

Si disegni la curva piana chiusa (cardioide), il vettore tangente alla curva nel punto P_0 applicato a P_0 quindi un versore normale applicato a P_0.

1) La function curv2_tan_plot vuole come parametron il nome di una function MATLAB con l'espressione della curva e della sua derivata (esattamente come la cp2_cardioide.m di prima)

2) Si scriva lo script scardio_tan_norm.m tale che:
   - chiami una funzione per aprire una finestra grafica
   - chiami una funzione per disegnare gli assi
   - chiami una funzione per disegnare la curva
   - chiami una funzione per il vettore tangente in P_0

anmglib_4.0 e curve 2D di Bézier

Nella libreria anmglib_4.0 le curve 2D di Bézier (funzioni vettoriali nella bse di Bernstein) sono memorizzate in una structure di Matlab. Una structure è un contenitore di più dati chiamati campi. Ogni campo può contenere qualsiasi tipo di dato. Si può accedere ai dati di un campo usando la notazione

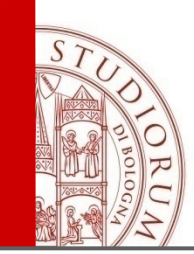
NomeStruttura.NomeCampo

Se chiamiamo bez la structure di una curva 2D di Bèzier, allora avremo:

bez.deg : campo scalare per il grado

bez.cp : campo array per i punti di controllo (array o (n+1)x2)

bez.ab : campo array per intervallo di definizione (array di 2)

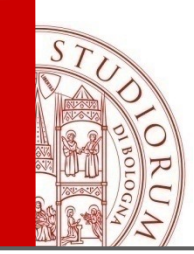


Esercizio B.3

Testo. si completi lo script sbezcurv2d.m per disegnare una curva 2D di Bézier insieme alla sua poligonale di controllo; la curva è definita nel file c2_bezier.db

Soluzione: utilizzare il commando help per le funzioni curv2_bezier_load e curv2_bezier_plot; si analizzino i parametri necessari e si completi lo script.

Domanda: visualizzando la curva a punti che informazioni si possono ottenere? Che relazione c'é con i punti di controllo?

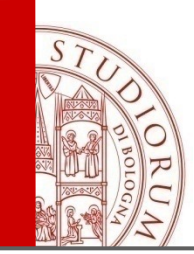


Esercizio B.4

Obiettivo: applicare una trasformazione geometrica ad una curva 2D di Bézier.

Soluzione: si salvi lo script sbezcurv2d.m con nome sbezcurv2d_trans.m e lo si modifichi per ottenere i disegni richiesti.

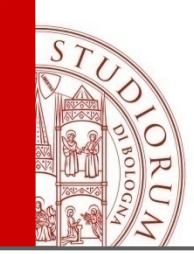
Si rammenti la proprietà di invarianza per trasformazioni affini delle curve 2D di Bézier.



Esercizio B.5

Testo. Con riferimento alla curva dell'esercizio B.3 (file c2_bezier.db), si disegni il vettore tangente negli estremi e nel punto centrale (t=0, t=1 e t=0.5).

Soluzione: si utilizzi il commando help per esaminare la funzione curv2_bezier_tan_plot del toolbox anmglib_4.0



Esercizio B.6

Si completi lo script sbezcurv2d_subdiv.m per suddividere una curva di Bézier; si utilizzi la function decast_subdiv del toolbox anmglib_4.0.

1. Si definisca una curva 2D di Bézier e la si disegni insieme alla sua poligonale di controllo;
2. Si utilizzi la decast_subdiv (utilizzare l'help) per suddividerla nel punto tbar=0.5*(a+b)
3. Disegnare le due curve trovate, le relative poligonali di controllo e i vettori tangenti negli estremi

Esercizio B.6

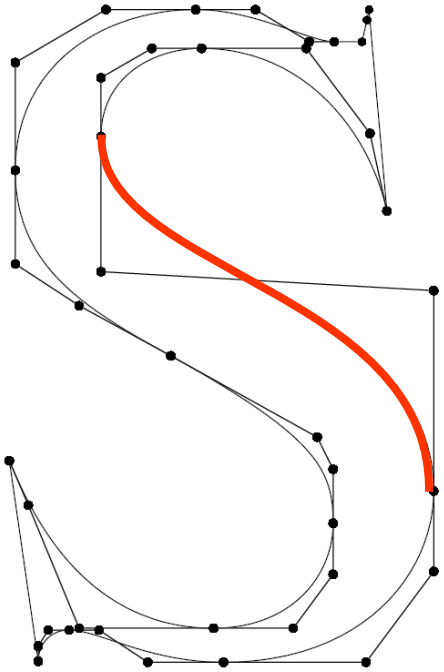
Rispondere alle seguenti domande:

1. Quanti punti di controllo definiscono la prima curva, quanti la seconda?
2. Qual è l'intervallo parametrico della prima curva? e quello della seconda?
3. Se si definisce come intervallo parametrico della prima [0,2] e della seconda [2,6] cosa succede?
4. E se si usa [0,1] per entrambe?
5. Le due curve che ordine di continuità hanno nel punto in cui sono state suddivise?

Curva di Bézier a tratti

Una curva complessa in forma, può essere pensata in più tratti, ciascuno dei quali rappresentabile con una curva 2D di Bézier di grado in genere basso (vedi figura con polinomi di grado 3).

Vogliamo definire formalmente un nuovo tipo di curva che chiameremo *piecewise curve (curva di Bézier a tratti) ottenuta* unendo insieme più curve 2D di Bézier di stesso grado.



Curva di Bézier a tratti

Sia data una partizione di [a,b] mediante una sequenza di punti u_i detti nodi, tali che:

$$u_0=a < u_1 < u_2 < \ldots < u_k < u_{k+1}=b$$

Allora una curva di Bézier a tratti di grado n viene definita come:

$$C(u)=\begin{cases} C_0(u) & \text{per } u \in [u_0, u_1] \\ C_1(u) & \text{per } u \in [u_1, u_2] \\ \ldots & \\ \ldots & \\ C_k(u) & \text{per } u \in [u_k, u_{k+1}] \end{cases}$$

$$C_i(u) = \sum_{j=0}^{n} P_j^i B_j^n(u)$$

con $C_i(u)$ curva di Bézier di grado n.

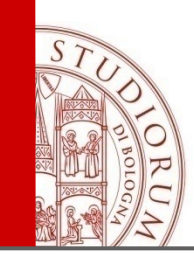
Curva di Bézier a tratti

Una curva di Bezier a tratti (per esempio di nome ppP) è memorizzata in una structure di Matlab formata da un campo per il grado (ppP.deg), una lista di punti di controllo (ppP.cp) e da una lista di nodi (ppP.ab).

ppP.cp ha dimensione (ncp x 2), con ncp il numero dei punti di controllo;

Se il loro numero è ncp ed il grado è n, la curva è formata da (ncp-1)/n tratti (chiamiamolo np);

Allora ppP.ab ha dimensione np+1.



Esercizio B.7

1. Si complete lo script sppbezplot per caricare la curva a tratti definite nel file c2_ppbez_esse.db.
2. La curva 2D di Bezier a tratti (ppP) viene memorizzata in una struttura formata da una lista di punti di controllo (ppP.cp), da una lista di nodi (ppP.ab) e da un grado (ppP.deg).
3. Come sono memorizzate nella lista ppP.cp i punti di controllo? Se il loro numero è ncp ed il grado è g, quanti tratti ha la curva?
4. Si disegnino i singoli tratti e per ogni tratto si disegnino i vettori tangenti negli estremi.

Esercizio B.7

Domanda:

Esaminando come sono i vettori tangenti di due tratti nel punto di raccordo cosa si può dire sulla continuità della curva a tratti in quell punto?