

Metodi Numerici per il Calcolo

Esercitazione 4:
Funzioni Polinomiali e Curve 2D
(Curve di Bézier)
A.A.2024/25

Scaricare dalla pagina web del corso l'archivio `matlab_mnc2425_4.zip` e scompattarlo nella propria home directory. Verrà creata una cartella con lo stesso nome contenente script e function utili per questa esercitazione che ha come obiettivo sperimentare la valutazione numerica di funzioni polinomiali e il disegno di curve 2D.

A. Valutazione numerica di funzioni polinomiali

1. Errore Algoritmico (Algoritmo di Ruffini-Horner)

Completare la function `poly_eval.m` con l'algoritmo di Horner per valutare un polinomio in base canonica, in corrispondenza di una lista di ascisse. Si utilizzi poi lo script `spoly_eval.m` che richiama tale function e valuta il seguente polinomio sia in precisione single che double:

$$p(x) = x^3 - 39x^2 + 504x - 2158 \quad x \in [10, 16]$$

Considerando il risultato ottenuto in precisione double come esatto, si calcola e rappresenta graficamente l'errore algoritmico. Si analizzi il risultato e si individui in corrispondenza di quali ascisse si hanno i valori di maggior errore; si dia una spiegazione.

(**Sugg.** si valuti il polinomio nell'intervallo indicato in ascisse che siano numeri finiti; poiché i coefficienti del polinomio sono numeri interi, gli eventuali errori numerici saranno solo di tipo algoritmico).

2. Errore Inerente

Si utilizzi lo script `spoly_eval_linear.m` che richiama la function `poly_eval.m` e valuta il seguente polinomio in precisione double

$$p(x) = -x + 100 \quad x \in [100, 101]$$

sia con dati double che single. Considerando i risultati ottenuti dai dati double come esatti (elaborazione in precisione double) e quelli ottenuti dai dati single come quelli calcolati (elaborazione in precisione single), si determina e rappresenta graficamente l'errore inerente.

3. **Polinomi nella base di Bernstein (valutazione con Alg.1)**

Completare lo script `sbernst.m` per valutare e rappresentare graficamente un polinomio nella base di Bernstein mediante valutazione delle funzioni base e successiva combinazione lineare. Utilizzare i polinomi definiti nella function `def_pol.m`.

(Sugg. si utilizzi la function `bernst_val` del toolbox `anmglib.4.1`)

4. **Polinomi nella base di Bernstein (valutazione con Alg.2)**

Completare lo script `sdecast.m` per valutare e rappresentare graficamente un polinomio nella base di Bernstein mediante l'algoritmo di de Casteljau. Utilizzare i polinomi definiti nella function `def_pol.m`. (Sugg. si utilizzi la function `decast_val` del toolbox `anmglib.4.1`)

5. **Valutazione polinomiale della derivata prima**

Modificare i due script `sbernst.m` e `sdecast.m` per valutare e rappresentare graficamente la derivata prima del polinomio in due modi differenti. Gli script si chiamino `sbernst_der.m` e `sdecast_der.m`. (Sugg. si utilizzino le function `bernst_valder` e `decast_valder` del toolbox `anmglib.4.1`)

B. Disegno di curve piane

1. Si rappresenti graficamente la seguente curva piana chiusa (cardioide)

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

insieme al sistema di assi cartesiani. Lo script si chiama `scardio.m`.

(Sugg. La function `curv2_plot` del toolbox `anmglib.4.1` valuta e disegna una curva in forma parametrica di cui sia nota l'espressione analitica; si realizzi una function `c2_cardioide.m` che contenga l'espressione analitica della curva)

2. Si modifichi lo script `scardio.m` per disegnare, in una seconda finestra, le curve ottenute dalla curva cardioide per rotazione degli angoli $\theta_i = \frac{2\pi}{n}i$ per $i = 1, \dots, n$ con $n = 7$; ognuna sia disegnata con un differente colore. Lo script si chiami `scardio_trans.m`.

3. Si modifichi lo script `scardio.m` per disegnare le seguenti curve piane:

$$C(t) = (t + 3, t)^T \quad t \in [-3, 3]$$

$$C(t) = (6t - 9t^2 + 4t^3, -3t^2 + 4t^3)^T \quad t \in [-0.5, 1.5]$$

$$C(t) = (t + \sin(2t), t + \cos(5t))^T \quad t \in [-3\pi, 3\pi]$$

$$C(t) = (t \cos(t), t \sin(t))^T \quad t \in [0, 16]$$

Lo script si chiami `scurve_2D.m`.

4. Si disegni la curva 2D di Bézier definita nel file `c2_bezier.db` insieme alla sua poligonale di controllo. Lo script si chiami `sbezcurv2d.m`. Si utilizzino le funzioni `curv2_bezier_load` e `curv2_bezier_plot` del toolbox `anmglib_4.1`.
(Sugg. si analizzi il file dati `c2_bezier.db`).
5. Si disegni la curva 2D di Bézier definita nel file `c2_bezier_heart.db`. In una seconda finestra si disegnino le curve ottenute per rotazione degli angoli $\theta_i = \frac{2\pi}{n}i$ per $i = 1, \dots, n$ con $n = 7$; ognuna sia disegnata con un differente colore. Lo script si chiami `sbezcurv2d_trans.m`.
6. Si esamini lo script `sppbezplot.m` che legge il file `c2_ppbez_esse.db` contenente una curva di Bézier a tratti e la disegna insieme alla sua poligonale di controllo. Si modifichi lo script per estrarre e disegnare ogni tratto di Bézier con un colore differente.
(Sugg. si analizzi il file dati `c2_ppbez_esse.db`).