Metodi Numerici per il Calcolo

Esercitazione 7: Equazioni non lineari e Applicazioni alle curve

A.A.2024/25

Scaricare dalla pagina web del corso l'archivio matlab_mnc2425_7.zip e scompattarlo nella propria home directory. Verrà creata una cartella con lo stesso nome contenente alcuni semplici script e function Matlab/Octave. Si svolga la seguente esercitazione che ha come obiettivo quella di sperimentare metodi per determinare le radici di equazioni non lineari.

A. Radici di Equazioni

Si considerino le seguenti funzioni test di cui si vogliono determinare gli zeri o radici dell'equazione associata (nella cartella sono presenti le function che le implementano insieme alle function delle derivate prime):

`zfun01` $f(x) = (4x-7)/(x-2) \quad x \in [1, 1.9]$
`zfun02` $f(x) = 1 + 3/x^2 - 4/x^3 \quad x \in [0.5, 6]$
`zfun03` $f(x) = 1 - 2.5x + 3x^2 - 3x^3 + 2x^4 - 0.5x^5 \quad x \in [0, 3]$
`zfun04` $f(x) = x^n - 2 \quad x \in [0, 2] \quad n = 2, 3, 4$
`zfun05` $f(x) = x^3 - 3x + 2 \quad x \in [-2.5, 2]$
`zfun06` $f(x) = 1/x - 2 \quad x \in (0, 4]$

1. **Metodo di bisezione**
   La function `main_bisez.m` fa uso della function `bisez.m` che implementa il metodo di bisezione. Viene effettuato il grafico della funzione $f(x)$ così da poter localizzare visivamente gli zeri della funzione nell'intervallo e poter definire l'intervallo di innesco del metodo.

   - Analizzare i codici e sperimentare con alcune funzioni test la ricerca degli zeri modificando la tolleranza di arresto e l'intervallo di innesco.
   - Modificare le function `main_bisez.m` e `bisez.m` affinché producano in stampa il numero di iterazioni effettuate. Mediamente quante iterazioni vengono compiute per la tolleranza 10^{-15}?

2. **Metodi di Newton e delle secanti**
   Le function `main_tangmet.m` e `main_secmet.m` utilizzano le function `tangmet.m` e `secmet.m` che implementano rispettivamente i metodi di Newton e delle secanti. Viene effettuato il grafico della funzione $f(x)$ così da poter localizzare visivamente gli zeri della funzione nell'intervallo.

   - Analizzare i codici, e sperimentare con alcune funzioni test la ricerca degli zeri modificando la tolleranza di arresto.

- Dopo aver analizzato il grafico di una funzione test stimare possibili iterati iniziali.

3. **Metodo di Newton e ordine di convergenza**
   Chiamando la funzione `tangmet.m` con `ftrace` ad 1 si hanno tutti gli iterati calcolati dal metodo di Newton; si modifichi lo script `main_tangmet.m` (lo si chiami `main_tangmet_ordconv.m`) affinché produca una tabella con le seguenti informazioni:

k	x_k	$e_k = x_k - x^*$	$\lvert e_{k+1}\rvert/\lvert e_k\rvert$	$\lvert e_{k+1}\rvert/\lvert e_k\rvert^2$

   Si analizzino i risultati delle ultime due colonne nel caso di zeri semplici o multipli della funzione test.

4. Le function `bezier_clipping.m` e `lane_riesenfeld.m` della libreria `anmglib_4.1` implementano due differenti algoritmi per determinare tutti gli zeri di una funzione polinomiale nella base di Bernstein nell'intervallo di definizione. Si completi lo script `main_zeri_bern.m` per determinare gli zeri del polinomio `zfun05` rappresentato nella base di Bernstein. (Come si può ottenere la funzione `zfun5` nella base di Bernstein?) Si disegnino anche i punti di controllo della funzione.

B. Interrogazione di curve 2D di Bézier

1. Data una curva 2D di Bézier si determinino i suoi **punti estremi**, cioè i punti della curva a tangente verticale e orizzontale e si disegnino insieme al primo ed ultimo punto della curva. Lo script si chiami `main_bezier_estremi.m`. (Sugg. si determinino gli zeri delle derivate prime delle componenti la curva). Dopo aver realizzato quanto richiesto si modifichi lo script affinché determini il bounding-box della curva. Lo script si chiami `main_bezier_bbox.m`. (Sugg. sarà il più piccolo rettangolo che contiene i punti estremi e il primo ed ultimo punto della curva).

2. Data una curva 2D di Bézier si determini il suo tight bounding-box (vedi corso on-line su curve di Bézier). Lo script si chiami `main_bezier_tight_bbox.m`. (Sugg. si realizzi una function secondaria (`curv2_align.m`) che determini la trasformazione geometrica che porta il primo punto della curva nell'origine degli assi e l'ultimo punto di controllo sull'asse x; si determini il bounding-box di questa curva aligned come fatto nell'esercizio precedente, quindi si applichi la trasformazione inversa al bounding-box trovato).

3. Date due curve 2D di Bézier che si intersecano in due punti, si determini la curva di Bézier a tratti che definisce la regione chiusa fra le due. Lo script si chiami `main_bezier_intersect.m`. (Sugg. si utilizzino le function

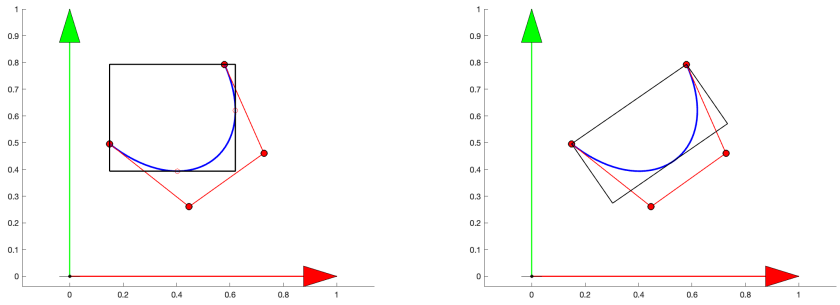

Figura 1: Bounding box (a sinistra) e tight-boundinbox a destra

`decast_subdiv` e `curv2_ppbezier_join`).

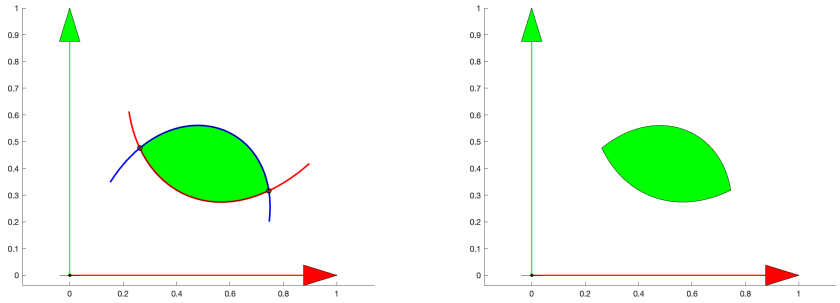

Figura 2: Curve da intersecare (a sinistra); regione chiusa (a destra)

Si osservi che le function della libreria `anmglib_4.1` che gestiscono curve di Bézier a tratti (suffisso `ppbezier`) possono essere utilizzate anche per curve di Bézier, a meno della function `curv2_ppbezier_load`.

4. Fare uno script per riprodurre il disegno a destra di Figura 3. Si intepolino le due seguenti funzioni con due curve di Bézier di grado 4:

$$y = (2 - 2x^2 - \sqrt{1 - 3x^2 + 3x^4 - x^6})/(3 + x^2), \quad x \in [-1, 1]$$
$$y = (1.8 - 2x^2 + \sqrt{1 - 3x^2 + 3x^4 - x^6})/(3 + x^2), \quad x \in [-1, 1]$$

Calcolare e stampare il MaxErr di interpolazione per entrambe le funzioni. quindi si intersechino e si determini la curva di Bézier a tratti, base del disegno. La piccola circonferenza al centro sia ottenuta per interpolazione di Hermite con una curva di Bézier cubica a tratti; abbia centro l'origine e

raggio 1/4. Lo script si chiami `sppbezinterp_curv2d_bicorn.m`. (Sugg. si segua quanto fatto nell'esercizio precedene e si utilizzino le function `curv2_intersect`, `decast_subdiv` e `curv2_ppbezier_join`).

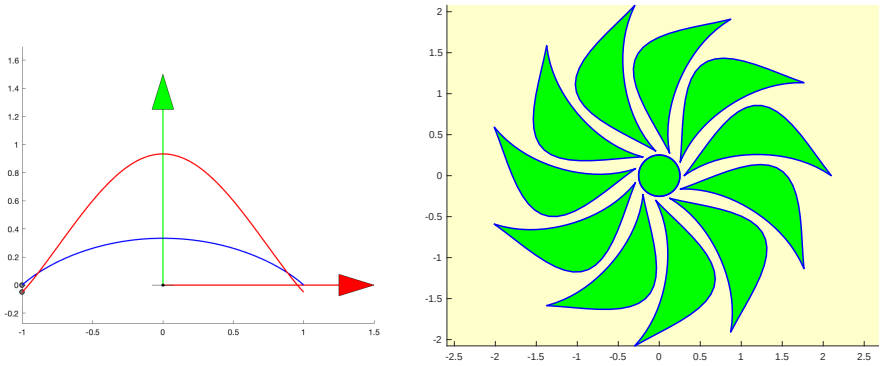

Figura 3: Funzioni da interpolare (a sinistra); disegno da riprodurre (a destra)