Metodi Numerici per il Calcolo

Esercitazione 1: Ambiente MATLAB e script

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es1.zip e scompattarlo nella propria home directory. Verrà creata una cartella con lo stesso nome contenente script, function e file dati utili per questa esercitazione che ha come obiettivo conoscere l'ambiente e il linguaggio MATLAB.

A. Realizzare uno script per effettuare i calcoli indicati.

1. Definire la seguente matrice:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 5 & 6 \\ 5 & 6 & 7 & 8 \\ 7 & 8 & 9 & 0 \end{pmatrix}$$

   quindi:

   (a) fare una copia della matrice `A` e chiamarla `B`;

   (b) copiare la prima riga di `A` in un vettore `a`;

   (c) sostituire la prima riga di `A` con l'ultima;

   (d) copiare il vettore `a` nell'ultima riga di `A`;

   (e) definire `P` come la matrice identità 4×4 con la prima e quarta riga scambiate;

   (f) definire `C` come il prodotto di `P` per `B`;

   (g) stampare le matrici `A` e `C` e confrontarle.

   Lo script si chiami `sA1.m`.

   Dal confronto fra le matrici `A` e `C` cosa si può dire? Che effetto ha la matrice `P` quando la si premoltiplica per un'altra matrice? E se la si postmoltiplica?

B. Risolvere i seguenti problemi con funzioni predefinite di MATLAB/Octave: creare degli script file

1. Calcolare il massimo, il minimo ed il valore medio degli elementi di un assegnato vettore numerico.

   Lo script si chiami `smmm.m`.

   Traccia: **Input**: si consideri il seguente vettore `[3,7,5,1,4,9,2,8]`, oppure gli elementi dati da `fix(100.*rand([1,10]))`;
   Output: stampare il massimo, il minimo e la media dei valori usando il comando `fprintf`.

2. Costruire una tabella di `n` valori delle funzioni seno, coseno e della somma dei loro quadrati nell'intervallo $[0, 2\pi]$. Si determinino e stampino i valori minimo, massimo, indice del valore minimo ed indice del valore massimo fra gli `n` valori calcolati per le funzioni seno, coseno e somma dei loro quadrati, quindi si riportino in stampa.
   Traccia: **Input**: valore per `n`, quindi determinare `n` ascisse equispaziate nell'intervallo indicato e valutare le funzioni indicate;
   Output: fornire in stampa i valori calcolati organizzati in una tabella con la seguente intestazione:

   ```
   i      x      sin(x)      cos(x)      sin(x)^2+cos(x)^2
   ```

   a fine tabulazione produrre in stampa i valori: indice del valore minimo, valore minimo, indice del valore massimo e valore massimo, per le funzioni seno, coseno e somma dei loro quadrati.

   Lo script si chiami `stabella.m`
   Si esegua lo script e guardando le stampe prodotte fare un commento sugli indici dei valori minimo e massimo relativi ai valori della somma dei loro quadrati.

3. La gittata di un oggetto lanciato ad un angolo θ rispetto all'asse x con una velocità iniziale v_0, trascurando la resistenza dell'aria, è data da:

   $$R(\theta) = \frac{v_0^2}{g} \sin(2\theta)$$

   per $0 \leq \theta \leq \pi/2$. Sia $g = 9.81 m/s^2$ e la velocità iniziale v_0 sia 100 m/s. Tabulando i valori della gittata tra $0 \leq \theta \leq \pi/2$ con un incremento di 0.05, verificare che la gittata massima si ottiene per $\theta = \pi/4$.

   Lo script si chiami `sgittata.m`.

   Successivamente si provi per velocità iniziale metà e doppia. Spiegare quanto si è trovato.

Metodi Numerici per il Calcolo

Esercitazione 2: Script, function, grafici e toolbox anmglib_5.0 per il disegno

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es2.zip e scompattarlo nella propria home directory. Verrà creata una cartella con lo stesso nome contenente script, function e file dati utili per questa esercitazione che ha come obiettivo imparare il linguaggio MATLAB e un po' di programmazione grafica.

A. Risolvere i seguenti problemi realizzando function e script file

1. Si consideri lo script `smmm.m` dell'Esercitazione 1; si realizzi una function di nome `mm_vect.m` che determini i valori massimo e minimo di una lista di valori (vettore numerico) ed uno script `smm_vect.m` che la richiami.
   Traccia: Lo script principale `smm_vect.m` definisca una lista di valori (per es. `[3,7,5,1,4,9,2,8]`, oppure `fix(100.*rand([1,10]))`), quindi chiami la function `mm_vect.m` e stampi i valori di ritorno.

2. Si consideri lo script `stabella.m` dell'Esercitazione 1 in cui si chiedeva di costruire una tabella di `n` valori delle funzioni seno, coseno e della somma dei loro quadrati nell'intervallo $[0, 2\pi]$. Si realizzi un grafico delle funzioni seno e coseno.

   Lo script si chiami `stabella_plot.m`

3. Si consideri lo script `sgittata.m` dell'Esercitazione 1 in cui si chiedeva di tabulare i valori della gittata di un oggetto lanciato ad angoli $0 \leq \theta \leq \pi/2$; si chiede di fare tre plot di tali valori della gittata nei casi di velocità iniziale $v_0 = 100, 150, 200$ e di verificare dai grafici che la gittata massima si ottiene sempre per $\theta = \pi/4$.

   Lo script si chiami `sgittata_plot.m`

4. Sapendo che l'equazione parametrica di una circonferenza di centro $C = [cx, cy]$ e raggio r è data da:

$$\begin{cases} x = r \cdot cos(t) + cx \\ y = r \cdot sin(t) + cy \quad t \in [0, 2\pi], \end{cases}$$

   si modifichi lo script dell'esercizio A.2 per definire n punti di una circonferenza di centro l'origine e raggio $r = 5$; quindi si disegni la poligonale definita da questi punti con un colore.

Lo script si chiami `scircle_plot.m`
(Sugg. si realizzi una function secondaria di nome `circle` per definire l'equazione parametrica della circonferenza.)

Disegnare poi 12 circonferenze di stesso raggio, con centri sulla circonferenza precedente, e tangenti fra di loro.

Lo script si chiami `scircles_plot.m`

B. Toolbox `anmglib_5.0` per il disegno
Scaricare dalla pagina web del corso l'archivio `anmglib_5.0.zip` e scompattarlo nella propria home directory. Verrà creata una cartella con lo stesso nome contenente il toolbox `anmglib_5.0`. Si esegua lo script `add_path` presente nella cartella `mnc2526_es2` per poter utilizzare le function del toolbox. Ogni esercizio consiste nel realizzare uno script che richiami opportunamente alcune fra le seguenti function per realizzare il disegno richiesto:

`open_figure`	`axis_plot`	`point_plot`	`point_fill`
`line_plot`	`circle2_plot`	`rectangle_plot`	`curv2_plot`

1. Si riscriva il codice per risolvere l'esercizio A.4 utilizzando le function `circle2_plot`, `point_plot` e `point_fill` del toolbox.
   Lo script si chiami `scircles_plot_lib.m`

2. Realizzare uno script per leggere il file `paperino.txt` contenente i vertici/punti 2D di una poligonale rappresentante un disegno 2D, quindi si visualizzi la poligonale, il suo bounding-box e il sistema di assi cartesiani, utilizzando le function `point_plot` e `rectangle_plot` del toolbox. Successivamente si riempia la poligonale con un colore usando la function `point_fill` del toolbox.
   Lo script si chiami `sload_plot.m`.
   (Sugg. Il bounding-box di una poligonale è il più piccolo rettangolo con i lati paralleli agli assi che contiene tutti i vertici della poligonale.)

3. Si rappresenti graficamente la seguente curva piana in forma parametrica (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 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)

4. Si rappresentino graficamente le funzioni `seno` e `coseno` dell'esercizio A.3 come curve in forma parametrica utilizzando la function `curv2_plot` del toolbox.
   Lo script si chiama `ssin_cos_plot.m`.

Metodi Numerici per il Calcolo

Esercitazione 3: Numeri Finiti e Trasformazioni Geometriche

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es3.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 l'aritmetica finita (Numeri Finiti) e le trasformazioni geometriche con il toolbox `anmglib_5.0` per il disegno in MATLAB.

A. Numeri Finiti in precisione BASIC single e BASIC double
MATLAB usa di default la precisione double e si può passare in precisione single mediante l'utilizzo della funzione `single()`. Poter alternare in un codice le precisioni single e double può essere molto utile a fini didattici.
I seguenti esercizi vogliono mettere in pratica alcuni concetti visti a lezione per rafforzare la propria comprensione; i vari script vanno modificati e rieseguiti più volte.

1. Lo script `scompute_u.m` calcola l'unità di arrotondamento U, sia in precisione single che double, mediante la seguente definizione operativa (ANSI/IEEE std.754):

   U è il più grande numero finito positivo tale che $fl(U+1)=1$.

   Analizzare l'implementazione della definzione operativa e verificare i risultati prodotti nei casi BASIC single e BASIC double.

2. Numeri gradual underflow (ANSI/IEEE std.754). Analizzare lo script `sfiniti.m` che implementa un ciclo `while` sia in versione BASIC single che BASIC double (vedi commenti) e stampa ad ogni iterazione un numero finito.

   (a) Prima di eseguire lo script prevedere cosa verrà prodotto in stampa.

   (b) Eseguire lo script, analizzare i risultati e individuare i numeri finiti **gradual underflow**.

   (c) Modificare la condizione del ciclo `while` da `x>0` all'equivalente `x+1>1`; l'output rimane lo stesso? Spiegare cosa succede.

3. Nello script `sexpression.m` viene effettuato il calcolo della semplice espressione

$$y = ((1 + x) - 1)/x$$

che dovrebbe sempre produrre come risultato il valore esatto 1, invece a seconda del valore assegnato ad x si ottengono risultati inattesi. A lezione si è applicata l'analisi in avanti degli errori per avere una stima dell'**errore algoritmico** di questa espressione e si è ottenuto che:

$$E_{Alg} \leq \left|\frac{1}{x}\right| |\epsilon_1| + 3U, \quad |\epsilon_1| < U,$$

dove ϵ_1 rappresenta l'errore che si commette nell'operazione in aritmetica finita $(1 + x)$.
Sperimentare differenti valori per x (prima finiti, poi anche reali) e dedurre per quali il risultato sarà corretto e per quali no e se i risultati sono coerenti con quanto stimato come E_{Alg}.

4. Approssimazione della derivata. Sia f una funzione continua e derivabile e cerchiamo di approssimare il valore della derivata mediante il limite del rapporto incrementale

$$\lim_{h \to 0} \frac{f(x + h) - f(x)}{h}.$$

L'idea è di valutare il rapporto incrementale per un certo valore di h, ma quanto piccolo? La function `fidiff.m` implementa il calcolo del rapporto incrementale e prova valori di h da 10^0 a 10^{-14} per la funzione `exp()`. La function fa uso di precisione double. Si vede che l'approssimazione migliora al diminuire di h, ma quando h diventa troppo piccolo, l'approssimazione comincia a peggiorare. Che tipo di errore viene commesso?
Successivamente modificare il codice per gestire una generica funzione (fare una function secondaria che ne contiene la definizione) e un differente valore `x`.

B. Trasformazioni Geometriche con il toolbox `anmglib_5.0`
Si esegua lo script `add_path` presente nella cartella per poter utilizzare le funzioni della toolbox `anmglib_5.0`. Ogni esercizio consiste nel realizzare uno script che richiami opportunamente alcune fra le seguenti funzioni per ottenere il disegno richiesto:

`open_figure`	`axis_plot`	`point_plot`	`point_fill`
`point_trans`	`point_trans_plot`	`get_mat2_rot`	`get_mat_trasl`
`get_mat_scale`	`get_mat2_symm`	`rectangle_plot`	`curv2_plot`

1. Disegnare l'oggetto 2D definito dalla poligonale chiusa di Figura 1 a sinistra. Si applichi una rotazione di 90 gradi rispetto al punto `[5,2]`, quindi una simmetria rispetto alla bisettrice del primo quadrante, per ottenere il disegno di Figura 1 a destra.
   Lo script si chiami `sfig_trans2D.m`.

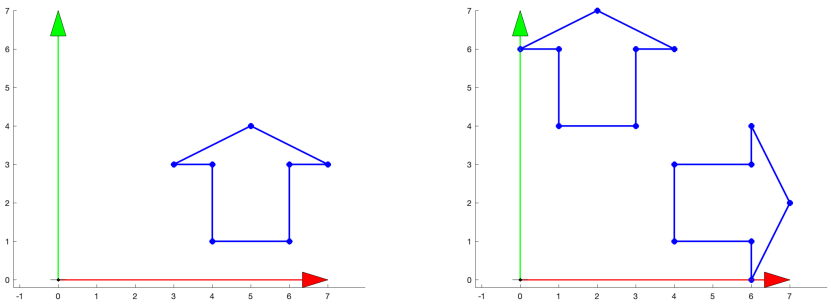

Figura 1: "Freccia 2D" di 7 vertici (sinistra).

2. Si consideri lo script `sload_plot.m` visto nell'Esercitazione 2. Si definisca una matrice di trasformazione per ruotare il disegno/poligonale rispetto al suo baricentro e lo si disegni in una seconda finestra insieme al suo bounding-box. Lo script si chiami `spoligon_rot.m`. Si ripeta lo stesso esercizio, ma per scalare il disegno/poligonale rispetto al suo baricentro.
   Il nuovo script si chiami `spoligon_scale.m`.

3. Si consideri lo script `sload_plot.m` visto nell'Esercitazione 2. Si definisca una matrice di trasformazione per traslare la poligonale e il suo bounding-box del vettore `[1/8,1/8]` e li si disegni; si disegnino poi nella stessa figure le poligonali e i loro bounding-box ottenute per simmetria rispetto agli assi coordinati (vedi Figura 2). Lo script si chiami `spoligon_symm.m`.

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

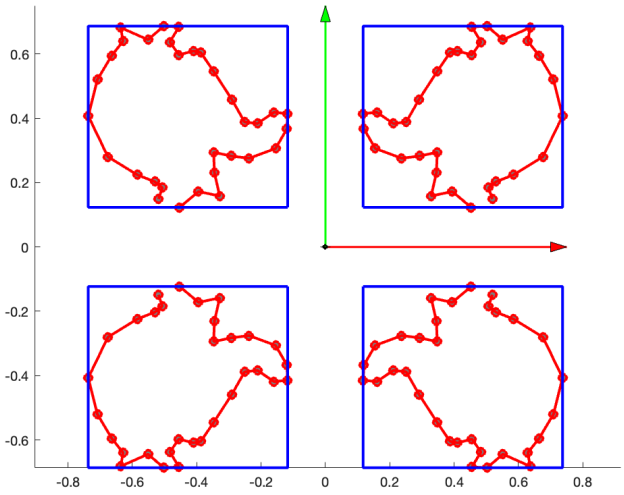

Figura 2: Simmetrie della poligonale/disegno di paperino e del suo bounding-box dopo averli traslati di `[1/8,1/8]`.

Metodi Numerici per il Calcolo

Esercitazione 4: Funzioni Polinomiali e Curve 2D (Curve di Bézier)

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es4.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)**
   Si utilizzi lo script `spoly_eval.m` per valutare la seguente funzione polinomiale sia in precisione single che double:

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

   Viene richiamata la built-in function `polyval` che implementa l'algoritmo di Ruffini-Horner. Considerando il risultato ottenuto in precisione double come esatto, si calcola e rappresenta graficamente l'errore algoritmico ottenuto lavorando in precisione single. Si analizzi il risultato e si individui in corrispondenza di quali ascisse si hanno i maggiori errori; si dia una spiegazione.
   (**Nota.** Se si valuta la funzione polinomiale nell'intervallo indicato in ascisse che siano esattamente rappresentabili in precisione single, poiché i coefficienti del polinomio sono numeri interi, gli eventuali errori numerici saranno solo di tipo algoritmico).
   Successivamente, per vedere quali sono le operazioni floating point che causano gli errori si utilizzi la function `poly_eval` (implementazione dell'algoritmo di Ruffini-Horner con opportune stampe).

2. **Errore Inerente**
   Si utilizzi lo script `spoly_eval_linear.m` che richiama la function `polyval.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 test definiti nella function `def_pol.m`.
   (**Sugg.** si utilizzi la function `bernst_val` del toolbox `anmglib_5.0`)

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 test definiti nella function `def_pol.m`.
   (**Sugg.** si utilizzi la function `decast_val` del toolbox `anmglib_5.0`)

5. **Errore Algoritmico per Alg.2**
   Con riferimento al codice `sdecast.m` dell' esercizio A.4, si vuole determinare l'errore algoritmico nel valutare il polinomio test dell'esercizio A.1 (l'esempio 7 di `def_pol.m`). Si proceda come già fatto nell'esercizio A.1, ossia si consideri il risultato ottenuto in precisione double come esatto e si calcoli e rappresenti graficamente l'errore algoritmico ottenuto lavorando in precisione single. Si completi il codice `sdecast_alg.m`. Cosa possiamo dire sui risultati ottenuti?

6. **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_5.0`)

B. Disegno di curve piane

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

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

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

   Lo script si chiami `scurve2d.m`.

2. 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_5.0`.
   Successivamente si disegni il vettore tangente negli estremi e nel punto centrale, utilizzando la function `curv2_bezier_tan_plot`. (Sugg. si analizzi il file dati `c2_bezier.db` per vedere che informazioni contiene).

3. 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 $i = 1, \ldots, n$ con $n = 7$; ognuna sia disegnata con un differente colore. Si completi lo script `sbezcurv2d_trans.m`.

4. 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 completi lo script per estrarre e disegnare ogni tratto di Bézier con un colore differente.
   (Sugg. si analizzi il file dati `c2_ppbez_esse.db`).

Metodi Numerici per il Calcolo

Esercitazione 5: Interpolazione Polinomiale di dati, funzioni, punti e curve

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es5.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 quello di sperimentare l'interpolazione polinomiale di dati e funzioni, quindi di punti e curve.

A. Interpolazione polinomiale e polinomiale a tratti

1. **Interpolazione polinomiale di dati: Lagrange e Bernstein**
   Si analizzino gli script `spolint_lagr_dati.m` e `spolint_bern_dati.m`, per l'interpolazione polinomiale nelle basi di Lagrange e Bernstein dei dati $(x_i, y_i)_{i=0,\ldots,n}$ (vedi file di dati `dataset1.dat`). Si utilizzano le function `lagrval2.m` (presente nella cartella), `bernst_val.m` e `decast_val.m` entrambe del toolbox `anmglib_5.0`.

2. **Interpolazione polinomiale di funzione: Lagrange e Bernstein**
   Si analizzi lo script `spolint_lagr_bern_fun.m` per l'interpolazione polinomiale di grado n di una funzione $f(x),\ x \in [a, b]$ a partire da $(x_i, f(x_i))_{i=0,\ldots,n}$ utilizzando sia la base di Lagrange che la base di Bernstein e confrontare i risultati. Eseguire più volte per sperimentare l'interpolazione di funzione all'aumentare del grado n e al variare della distribuzione dei punti (equispaziati e punti di Chebyshev (vedi function `chebyshev2.m` nella cartella)). Si consideri la funzione test di Runge (function `runge.m`):

$$f(x) = 1/(1 + x^2) \qquad x \in [-5, 5].$$

3. **Sulla convergenza dell'interpolante polinomiale**
   Si modifichi lo script dell'esercizio A.2 (lo si chiami `polint_lagr_fun.m`) per trasformarlo in una function per l'interpolazione nella forma di Lagrange delle seguenti funzioni test:

`fun1.m`	$f(x) = \sin(x) - \sin(2x)$	$x \in [-\pi, \pi]$
`fun2.m`	$f(x) = \begin{cases} 0.5 & se \quad x \geq 0 \\ -0.5 & se \quad x < 0 \end{cases}$	$x \in [-2, 2]$
`polfun1.m`	$f(x) = 1 + x/2 + x^2/6 + x^3/24 + x^4/120$	$x \in [-1, 3.5]$.

   Lo script `spolint_lagr_fun_test.m` richiama questa function e per ogni funzione test e distribuzione di punti (equispaziati o di Chebyshev), prevede un grafico dell'errore assoluto (E_{tot}) all'aumentare del grado al fine

di verificare la convergenza o meno dell'interpolante (E_{anal}) alla funzione. (Sugg. si modifichi lo script `spolint_lagr_bern_fun.m` per realizzare una function con la seguente intestazione:

```
function err=polint_lagr_fun(ifun,n,tipo,fig)
```

i cui argomenti nell'ordine sono: indice di una funzione test, grado del polinomio interpolante, tipo di distribuzione punti, flag per i grafici.)

4. **Interpolazione di funzione: cubica a tratti di Hermite nella base di Bernstein e convergenza**
   Lo script `sppbezierCC1_interp_fun.m` calcola l'interpolante cubica a tratti C^1 di Hermite nella base di Bernstein di una funzione su punti equispaziati e di Chebyshev. Sono previsti gli stessi grafici e stampe dello script `spolint_lagr_bern_fun.m`. Confrontare i risultati con l'interpolazione polinomiale di una stessa funzione.
   Lo script richiama la function `ppbezierCC1_interp_der` presente nella cartella di lavoro; analizzarne il codice.

B. Curve di Bézier e di Bézier a tratti di interpolazione

1. Dato un set di punti 2D lo si vuole interpolare con una curva 2D di Bézier. Si completi lo script `sbezierinterp_p2d.m` per realizzare quanto richiesto; si utilizzi la function `curv2_bezier_interp` del toolbox `anmglib_5.0`.

2. Data una curva 2D in forma parametrica (vedi function `c2_curv3_pol`) la si vuole interpolare con una curva di Bézier. Si completi lo script `sbezierinterp_curv2d.m` utilizzando come parametri di interpolazione sia punti equispaziati che punti di Chebishev. Si determini e analizzi l'errore di interpolazione all'aumentare del numero di punti con cui si campiona la curva analitica.

3. Dato un set di punti 2D (vedi file `paperino.txt`) lo si vuole interpolare con una curva di Bézier cubica a tratti C^1 (interpolazione di Hermite). Si consideri la function `curv2_ppbezierCC1_interp` del toolbox `anmglib_5.0` e la si utilizzi per ottenere quanto richiesto. (Sugg. si completi lo script `sppbezierinterp_p2d.m`)

4. Data una curva 2D in forma parametrica la si vuole interpolare con una curva di Bézier cubica a tratti C^1 (interpolazione di Hermite), campionando valori e valori di derivata prima.
   Lo script `sppbezierinterp_curve2d.m` vuole interpolare la curva circle in forma parametrica (vedi function `cp2_circle.m`), utilizzando la seguente function del toolbox `anmglib_5.0`:

```
ppP=curv2_ppbezierCC1_interp_der(Q,Q1,tpar)
```

Come argomenti sono previsti un array di punti 2D (`Q`), un array di vettori tangenti nei punti `Q` (`Q1`) e l'array dei valori parametrici dei punti `Q` (`tpar`); l'output consiste nella struttura della cubica di Bézier a tratti di interpolazione. Si richiede di valutare l'errore fra la curva in forma parametrica e l'interpolante.
Modificare poi lo script per interpolare altre curve in forma parametrica di cui sia nota l'espressione analitica.

5. Fare uno script per riprodurre il seguente disegno di una farfalla (vedi Figura 1) . Lo script si chiami `sppbezerinterp_p2d_farfalla.m`. Sugg. Le due ali a destra si ottengano effettuando due interpolazione con curve di Bézier dei seguenti due set di punti: (218,385),(231,512),(330,517),(349,404) e (349,404),(490,381),(500,237),(321,241). Si sfrutti poi la simmetria per le altre due ali di sinistra. Infine tutte le curve vengano unite in una curva di Bézier a tratti utilizzando la function `curv2_mdppbezier_join` della libreria `anmglib_5.0`.

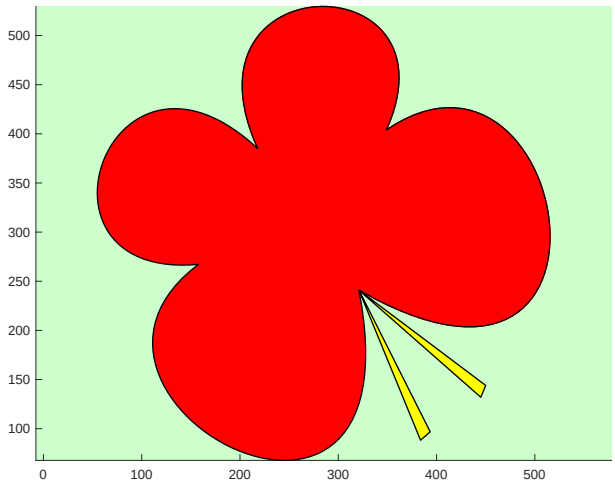

Figura 1: Disegno da riprodurre.

Metodi Numerici per il Calcolo

Esercitazione 6: Integrazione Numerica, Lunghezza ed Area di curve 2D

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es6.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 quello di sperimentare l'integrazione numerica di funzioni e la sua applicazione nel disegno.

A. Integrazione Numerica
Si considerino le seguenti funzioni test di cui si vuole calcolare l'integrale definito sul loro dominio di definizione:

$f_1 = e^{\sqrt{x}}\sin(x) + 2x - 4$	$x \in [0, 12]$	$\int_0^{12} f_1(x)dx$	$= 68.3532891202483$
$f_2 = \frac{32}{1+1024x^2}$	$x \in [0, 4]$	$\int_0^{4} f_2(x)dx$	$= 1.56298398573480$
$f_3 = \frac{e^{-x}}{x}$	$x \in [1, 2]$	$\int_1^{2} f_3(x)dx$	$= 0.170483423687459$
$f_4 = \frac{4}{1+x^2}$	$x \in [0, 1]$	$\int_0^{1} f_4(x)dx$	$= 3.14159265358979$
$f_5 = \sqrt{(1 - x^2)e^x}$	$x \in [0, 1]$	$\int_0^{1} f_5(x)dx$	$= 1.24395050141647$

1. **Formule di Quadrature di Newton-Cotes**
   Le function `trapezi_comp.m` e `simpson_comp.m` (presenti nella cartella) implementano rispettivamente la formula dei trapezi e Simpson composte;

   - si richiamino da Command Window le function `err_trapezi_comp.m` ed `err_simpson_comp.m`. Viene utilizzata la function MATLAB `integral` per simulare un valore "esatto", la function `trapezi_comp.m` o `simpson_comp.m` per avere un valore approssimato e viene stampato l'errore di integrazione. Si usi `help` per saperne di più sulla function `integral`; si esaminino le function `trapezi_comp` e `simpson_comp`.
   - Ricordando le espressioni degli errore di integrazione per le formule dei trapezi e Simpson composte

     $$R_T = -\frac{b-a}{12}h^2 f^{(2)}(\eta), \qquad R_S = -\frac{b-a}{180}h^4 f^{(4)}(\eta));$$

     gli script precedenti sono stati modificati per verificare sperimentalmente che l'errore è di ordine 2 nel caso trapezi ed è di ordine 4 nel caso Simpson, cioè si riducono come h^2 e come h^4. (I due nuovi script/function si chiamano rispettivamente `err2_trapezi_comp.m`,

`err2_simpson_comp.m`). (Sugg. viene applicata la formula composta per valori h e $h/2$ e si stampa il rapporto dei relativi errori; nella cartella sono presenti le derivate seconde e quarte delle funzioni integrande; come possono essere utili?)

2. **Errore di Integrazione ed Estrapolazione di Richardson**
   Si considerino gli script dell'esercizio precedente.

   - A partire da una copia dello script `err2_trapezi_comp.m` la si modifichi per aggiungere l'estrapolazione di Richardson

   $$\int_a^b f(x)dx = \frac{4T(h/2) - T(h)}{3} + O(h^4)$$

   e tre nuove colonne di stampa simili a quelle presenti per verificare sperimentalmente l'ordine dell'errore (il nuovo script si chiami `err2_trapezi_rich.m`).

   - A partire dallo script `err2_simpson_comp.m` si realizzi uno script simile a quanto richiesto al punto precedente per aggiungere l'estrapolazione di Richardson, sapendo che:

   $$\int_a^b f(x)dx = \frac{16S(h/2) - S(h)}{15} + O(h^5)$$

   e quindi per verificare sperimentalmente l'ordine dell'errore. (Il nuovo script si chiami `err2_simpson_rich.m`).

B. Lunghezza ed area di una curva di Bézier

1. Data una curva 2D di Bézier lo script `slung_bezier.m` calcola la sua lunghezza utilizzando la function `curv2_bezier_len` del toolbox. (Sugg. si analizzi questa funzione e la `gc_norm_c1_val.m` che viene passata come argomento alla builtin function `integral` di MATLAB).
   Si modifichi poi lo script per calcolare la lunghezza di una curva 2D di Bézier a tratti (lo script si chiami `slung_ppbez.m` e calcoli la lunghezza della curva definita nel file `ppbez_esse.db`); analogamente per una curva di Bézier a tratti multi-grado.

2. Data una curva 2D di Bézier lo script `sarea_bezier.m` calcola la sua area utilizzando la function `curv2_bezier_area` del toolbox. (Sugg. si analizzi questa function e la `gc_cxc1_val.m` che viene passata come argomento alla builtin function `integral` di MATLAB).
   Si modifichi poi lo script per calcolare l'area di una curva 2D di Bézier a tratti (lo script si chiami `sarea_ppbez.m` e calcoli l'area della curva definita nel file `ppbez_esse.db`); analogamente per una curva di Bézier a tratti multi-grado.

3. Si analizzi e completi il codice `smdppbezmodel_curve2d_square_smooth.m` che vuol essere un esempio di modellazione procedurale con una curva di Bézier a tratti multi-grado. Si salvi la curva nel file `mdppbez_square_smooth.db`. Si realizzi quindi uno script che legga la curva definita in questo file e la scali affinché abbia area/lunghezza predefinita, per esempio unitaria. La scala della curva sia fatta rispetto al suo baricentro. (Gli script si chiamino `sarea_mdppbez_unitaria.m` ed `slung_mdppbez_unitaria.m`).

4. Lo script `smdppbezmodel_calice_interp_svg.m` è un esempio di modellazione procedurale di un disegno a partire dall'interpolazione con curve di Bézier dei due seguenti set di punti (vedi Figura 1).

$$[5.0, 4.0; 3.4, 4.4; 2.9, 5.6; 4.0, 8.0]$$

$$[3.0, 1.0; 4.5, 1.5; 5.0, 2.0]$$

Dopo aver eseguito lo script, si esami il codice che lo ha generato. Alla fine del codice ci sono delle istruzioni commentate che si riferiscono al salvataggio del disegno in formato SVG, al suo caricamento e al suo plotting sfruttando alcune function del toolbox `anmglib_5.0` (`svg_struct_add`, `svg_save`, `svg_load` ed `svg_plot`). Si apra con "Open as text" il file `calice.svg` nell Editor di MATLAB per analizzarne la struttura e poi lo si visualizzi su un browser (per esempio Chrome).

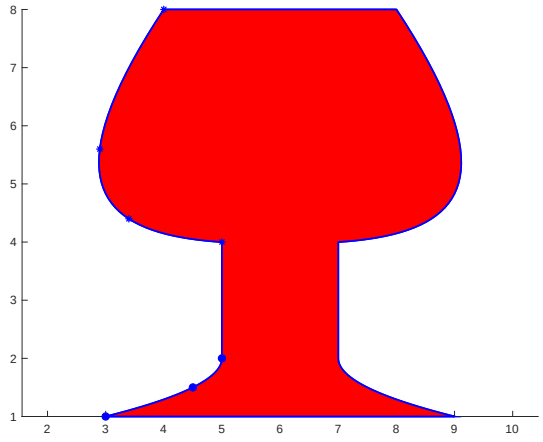

Figura 1: Disegno di un calice da riprodurre; punti da interpolare evidenziati con marker o e *.

Metodi Numerici per il Calcolo

Esercitazione 7: Equazioni non Lineari, Intersezione di curve 2D

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es7.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 e la loro applicazione nel disegno.

A. Radici di Equazioni non Lineari

Si considerino le seguenti funzioni test di cui si vogliono determinare gli zeri o radici dell'equazione associata. Nella cartella è presente la function `def_fun.m` che le implementa:

`zfunf01`	$f(x) = (4x-7)/(x-2) \quad x \in [1, 1.9]$
`zfunf02`	$f(x) = 1 + 3/x^2 - 4/x^3 \quad x \in [0.5, 6]$
`zfunf03`	$f(x) = 1 - 2.5x + 3x^2 - 3x^3 + 2x^4 - 0.5x^5 \quad x \in [0, 3]$
`zfunf04`	$f(x) = x^n - 2 \quad x \in [0, 2] \quad n = 2, 3, 4$
`zfunf05`	$f(x) = x^3 - 3x + 2 \quad x \in [-2.5, 2]$
`zfunf06`	$f(x) = 1/x - 2 \quad x \in (0, 4]$
`zfunf07`	$f(x) = \tanh(x-1) \quad x \in [-1, 3]$
`zfunf08`	$f(x) = (1-x)^3 - 13/3(1-x)^2x + 25/4(1-x)x^2 - 3x^3 \quad x \in [0, 1]$

Si noti che la function `def_fun.m` contiene anche le derivate prime, nella forma `zfunp0x`.

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 le iterazioni effettuate (successione degli intervalli). Si può stimare il numero di iterazioni che vengono effettuate per ottenere la tolleranza 10^{-15}?

2. **Metodi di Newton e delle secanti**
   La function `main_tangmet.m` utilizza la function `tangmet.m` che implementa il metodo di Newton. Viene effettuato il grafico della funzione $f(x)$ così da poter localizzare visivamente gli zeri della funzione nell'intervallo e poter definire l'iterato iniziale del metodo.

   - Dopo aver analizzato il grafico di una funzione test stimare possibili iterati iniziali.
   - Analizzare il codice, e sperimentare con alcune funzioni test la ricerca degli zeri modificando la tolleranza di arresto.

   Ripetere l'esercizio per il metodo delle secanti (vedi la function `main_secmet.m` che utilizza la function `secmet.m`).

3. **Metodo di Newton e ordine di convergenza**
   Richiamando la function `tangmet.m` con `ftrace` ad 1 si hanno tutti gli iterati calcolati dal metodo di Newton; si modifichi la function `main_tangmet.m` (la 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 multipli o semplici della funzione test.

B. Interrogazione e intersezione 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 utilizzando la function `lane_riesenfeld` della libreria `anmglib_5.0`). Dopo aver realizzato quanto richiesto si modifichi lo script affinché determini il bounding-box della curva (vedi Figura 1). (Sugg. il bounding-box sarà il più piccolo rettangolo che contiene i punti estremi e il primo ed ultimo punto della curva). Infine si modifichi la curva, si riesegua lo script e si osservino i nuovi punti estremi e il bounding-box.

2. 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 (vedi Figura 2). Lo script si chiami `main_bezier_intersect.m`. (Sugg. si utilizzino le function `curv2_intersect`, `decast_subdiv` e `curv2_mdppbezier_join`).

   Si osservi che le function del toolbox `anmglib_5.0` che gestiscono curve di Bézier a tratti (suffisso `ppbezier`) possono essere utilizzate anche per curve di Bézier, tranne la function `curv2_ppbezier_load`.

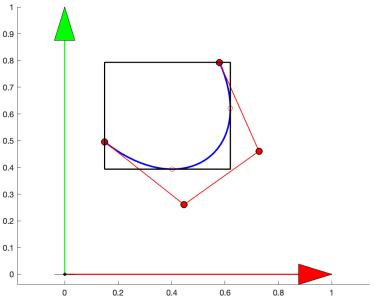

Figura 1: Bounding box

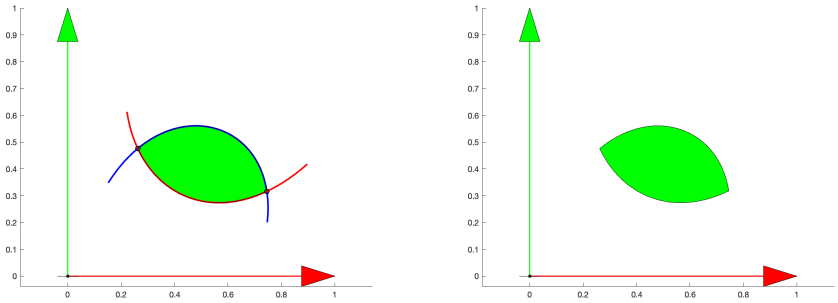

Figura 2: Curve da intersecare (a sinistra); regione chiusa (a destra)

3. Fare uno script per riprodurre il disegno di Figura 3 (destra). Si intepolino le due seguenti funzioni:

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

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

(Figura 3 sinistra) con due curve di Bézier in modo che il max. dell'errore di interpolazione sia minore di 10^{-2} in entrambe le interpolazioni. Quindi si intersechino e si determini la curva di Bézier a tratti, base del disegno. Lo script si chiami `sppbezinterp_curv2d_bicorn.m`. (Sugg. si utilizzino le function `curv2_intersect`, `decast_subdiv`, `curv2_mdppbezier_join` e `curv2_mdppbezier_close`); nella cartella è presente il file `girandola.fig` che si può aprire con il comando `openfig` e interrogare con il mouse.

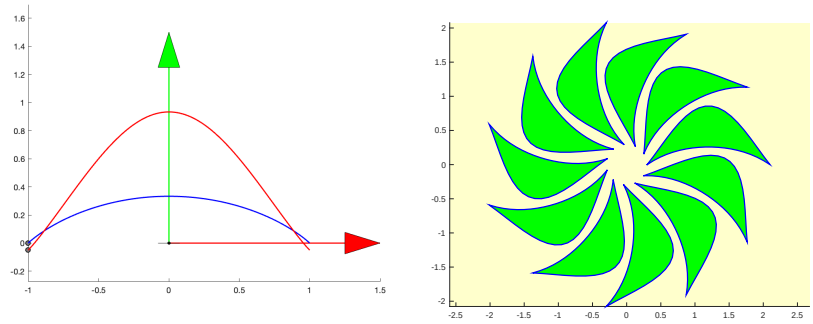

Figura 3: Funzioni da interpolare (a sinistra); disegno da riprodurre (a destra)

Metodi Numerici per il Calcolo

Esercitazione 8: $A\mathbf{x} = \mathbf{b}$, Fattorizzazione LU, riproduzione di disegni vettoriali

A.A.2025/26

Scaricare dalla pagina web del corso l'archivio mnc2526_es8.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 sperimentare la fattorizzazione LU di una matrice e la soluzione di sistemi lineari.

A. Soluzione di Sistemi Lineari

Nella cartella sono presenti alcune function che se richiamate restituiscono una matrice $n \times n$ non singolare che può essere utilizzata come matrice test per un sistema lineare $A\mathbf{x} = \mathbf{b}$. Come metodologia di lavoro si proceda scegliendo un vettore $\mathbf{x} \in \mathbb{R}^n$ (per esempio $\mathbf{x} = (1, 1, \ldots, 1)^T$) e si determini $\mathbf{b}$ moltiplicando A per $\mathbf{x}$ affinché la soluzione del sistema sia il vettore $\mathbf{x}$, così da conoscerne la soluzione esatta.

1. **Function `lu` di MATLAB/Octave**
   Completare lo script `main_linsys.m` che definito un sistema lineare $A\mathbf{x} = \mathbf{b}$, lo risolve nei due seguenti modi:

   - utilizzando l'operatore "left-division" di MATLAB/Octave;
   - utilizzando la function di MATLAB/Octave `lu` che implementa la fattorizzazione di Gauss con scambio delle righe e perno massimo. Più precisamente siamo interessati alla chiamata:

     ```
     [L,U,P]=lu(A)
     ```

     (vedere `help lu`); quindi si usino le function `lsolve.m` e `usolve.m` presenti nella cartella per risolvere i sistemi

     $$L\mathbf{y} = P\mathbf{b}$$
     $$U\mathbf{x} = \mathbf{y}.$$

2. **Sulla stabilità della fattorizzazione LU**
   Si completi lo script `main_lufact.m` in modo che richiami la function `lu` di MATLAB come nell'esercizio A.1 (fattorizzazione LU di una matrice con scambio delle righe e perno massimo) e verifichi che:

   $$\max |\ell_{i,j}| \leq 1, \qquad \max |u_{i,j}| \leq 2^{n-1} \max |a_{i,j}|$$

dove $\ell_{i,j}$ e $u_{i,j}$ sono gli elementi delle matrici L e U determinate. Si stampi una tabella con i seguenti valori per le matrici di esempio `mat_k, k=2,3,4,5` di dimensioni $n = 5, 10, 50$

mat_k	$n \times n$	$\max \lvert\ell_{i,j}\rvert$	$\max \lvert u_{i,j}\rvert$	$2^{n-1} \max \lvert a_{i,j}\rvert$

Si realizzi uno script `main_qrfact.m` che richiami la built-in function MATLAB `qr`, del tutto simile allo script `main_lufact.m`, per confrontare i valori degli elementi delle matrici Q ed R ottenuti nella fattorizzazione QR; si verifichi inoltre che

$$\max |q_{i,j}| \leq 1, \qquad \max |r_{i,j}| \leq \sqrt{n} \max |a_{i,j}|.$$

3. **Condizionamento di un sistema lineare**
Si consideri la matrice H di Hilbert $n \times n$ (function MATLAB `hilb`, vedi l'`help`) Si calcoli $\mathbf{b}$ in modo che $H\mathbf{x} = \mathbf{b}$, dove $\mathbf{x} = (1, 1, \ldots, 1)^T$. Si aggiunga a $\mathbf{b}$ una perturbazione

$$\delta \mathbf{b}_p = 10^{-p} \texttt{rand}(n, 1).$$

Si risolva il sistema $H\tilde{\mathbf{x}}_p = \mathbf{b} + \delta\mathbf{b}_p$, per $p = 1, \ldots, 5$ e si stampi per ogni valore di p la seguente quantità:

$$K_p = \frac{\|\mathbf{x} - \tilde{\mathbf{x}}_p\|}{\|\mathbf{x}\|} \frac{\|\mathbf{b}\|}{\|\delta\mathbf{b}_p\|}.$$

I valori K_p così ottenuti sono il valore effettivo sperimentale del numero di condizione della matrice H. Verificare che per ogni p sia

$$K_p \leq \texttt{cond}(H).$$

Lo script `main_hilb.m` implementa già quanto detto; eseguire per differenti dimensioni $n \times n$ e analizzare i risultati.

B. Riproduzione di disegni

1. Si vuole riprodurre il disegno di Figura 1 a destra, utilizzando curve di Bézier a tratti multi-grado, ottenute come join di curve di Bézier di interpolazione e/o modellazione. Si colorino quindi le regioni delimitate da tali curve per replicare, come richiesto, il disegno. Lo script si chiami `smdppbez_interp_italy.m` .
(Sugg. La curva di bordo esterna, Figura 1 a sinistra, si ottenga per interpolazione con una curva di Bézier cubica a tratti di Hermite della curva `cp2_circle.m` vista nell'Esercitazione 4. La curva interna si ottenga per scala uniforme con fattore 0.97. Dopo avere riprodotto i segmenti retti si proceda al join di tali curve per ottenere curve di Bézier a tratti multi-grado che definiscono il bordo delle regioni colorate).

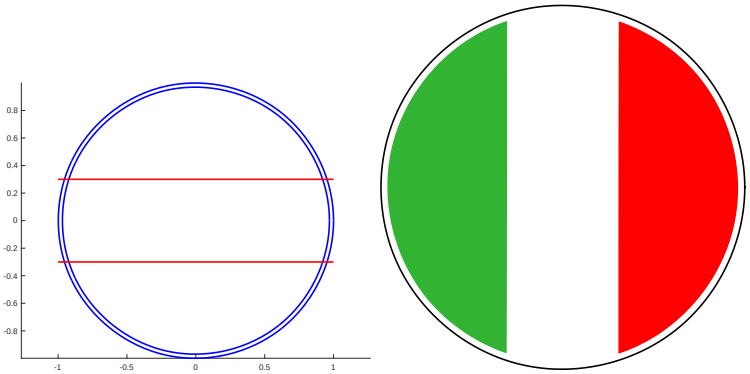

Figura 1: geometria del disegno (sinistra), disegno da riprodurre (destra)

2. Si vuole riprodurre il disegno di Figura 2 a destra, utilizzando curve di Bézier a tratti multi-grado, ottenute come join di curve di Bézier di interpolazione e/o modellazione. Si colorino quindi le regioni delimitate da tali curve per replicare, come richiesto, il disegno. Lo script si chiami `smdppbez_interp_segnale.m` .
   (Sugg. La curva di bordo esterna, Figura 2 a sinistra, si ottenga per interpolazione con una curva di Bézier cubica a tratti di di Hermite della curva `cp2_circle.m` vista nell'Esercitazione 4. La curva interna si ottenga per scala uniforme con fattore 0.97. Dopo avere riprodotto i segmenti retti si proceda al join di tali curve per ottenere curve di Bézier a tratti multi-grado che definiscono il bordo delle regioni colorate).

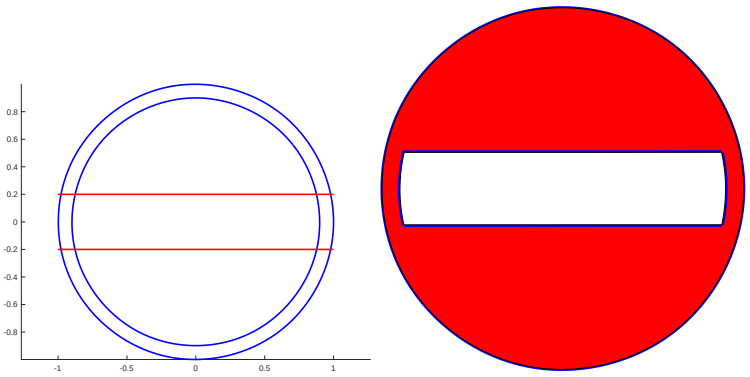

Figura 2: geometria del disegno (sinistra), disegno da riprodurre (destra)

3. Si vuole riprodurre il disegno di Figura 3 a destra, utilizzando curve di Bézier a tratti multi-grado, ottenute come join di curve di Bézier di in-

terpolazione e/o modellazione. Si colorino quindi le regioni delimitate da tali curve per replicare, come richiesto, il disegno. Lo script si chiami `smdppbez_cuore_france.m` .
(Sugg. La curva di bordo esterna in Figura 3 a sinistra si ottenga caricando la curva di Bézier `c2_bezier_heart.db` utilizzata nell'Esercitazione 4. Si interpoli la metà destra (parte verde) con una curva di Bézier di grado 6 e si ottenga la curva offset (parte magenta) con il comando `curv2_bezier_offset` dando come distanza d il valore 0.009. Si proceda a generare la parte simmetrica (parte in blu) e a determinare le curve di Bézier a tratti multi-grado che definiscono il bordo delle regioni colorate).

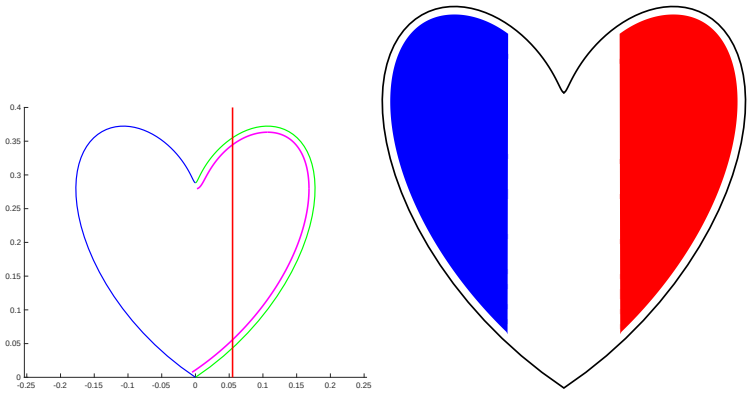

Figura 3: geometria del disegno (sinistra), disegno da riprodurre (destra)