Introduzione a Matlab

Carlo Garoni

6 ottobre 2020

Indice

1 Elementi basilari di Matlab

1.1 Matlab come calcolatrice: +, −, *, /, ^

Il modo più semplice di usare Matlab è quello di usarlo come calcolatrice. In particolare, le operazioni aritmetiche +, −, *, / sono tutte definite in Matlab così come l'elevamento a potenza ^.

1.2 Formati di visualizzazione dei numeri: format short/long/short e/long e

Il comando
> format long
fa sì che vengano visualizzate molte cifre decimali dei numeri. Similmente si hanno i comandi
> format short
(poche cifre decimali visualizzate),
> format long e
(notazione esponenziale con molte cifre visualizzate),
> format short e
(notazione esponenziale con poche cifre visualizzate).

1.3 Variabili, assegnamenti, aggiornamenti/sovrascritture: l'uguale =

Ogni stringa di caratteri alfanumerici che inizia con una lettera (non con un numero) è considerata da Matlab una variabile. Ad ogni variabile si può assegnare un valore e gli assegnamenti in Matlab si eseguono semplicemente con il comando =. Ad esempio, il comando
> A=3+4
assegna alla variabile A il valore 7 e il comando
> b5=A*5
assegna alla variabile b5 il valore 35. Il comando
> A=2*A
assegna alla variabile A il valore 14, ottenuto moltiplicando per 2 il vecchio valore di A (cioè 7). Un assegnamento di questo tipo si chiama anche *aggiornamento* o *sovrascrittura*. Matlab è *case-sensitive*: la variabile A è diversa dalla variabile a.

1.4 Spazio di lavoro (workspace): whos, clear

Tutte le variabili definite dall'utente vengono salvate da Matlab nel cosiddetto workspace, che è una parte della memoria del computer che Matlab sceglie per sè. Il comando
> whos
permette di visualizzare i nomi di tutte le variabili presenti nel workspace, insieme con alcune informazioni

relative alle variabili stesse; una di queste è la memoria (in Bytes) occupata da ciascuna variabile. Supponiamo che nel workspace siano presenti 5 variabili chiamate a, b, c, d, e. Il comando
> clear a c d
cancella dal workspace le variabili a, c, d, liberando così della memoria nel workspace stesso. Il comando
> clear
cancella tutte le variabili del workspace.

1.5 Variabile speciale ans

Essa viene creata in automatico da Matlab e contiene il risultato dell'ultimo calcolo effettuato che non è stato assegnato a nessuna variabile. Ad esempio, il comando
> 10+2*3
assegna il valore 16 alla variabile ans. Il comando
> clear ans
cancella la variabile ans dal workspace.

1.6 Variabili predefinite: pi, i, j, eps, Inf, NaN

pi (pi greco), i (unità immaginaria), j (unità immaginaria), eps (precisione di macchina: è un numero molto piccolo che indica "quanto è preciso il calcolatore"), Inf ($+\infty$), NaN (Not a Number: è un oggetto che non può essere considerato un numero; ad esempio, il risultato della divisione $0/0$ è NaN). Per queste variabili predefinite, un assegnamento del tipo
> pi=0
sovrascrive la variabile pi assegnandogli valore 0, ma il comando
> clear pi
fa sì che la variabile pi torni ad essere pi greco.

1.7 Il punto e virgola ;

Il punto e virgola " ; " messo al termine di un comando fa sì che il comando sia eseguito ma il suo risultato non sia visualizzato sullo schermo. Inoltre, il " ; " permette di dare più comandi sulla stessa linea (ciascun comando è separato dal precedente dal " ; ").
Ad esempio, la sequenza di comandi
> A=3+4; b5=A+5; c=b5^2*10; c/2
assegna alla variabile A il valore 7, alla variabile b5 il valore 12, alla variabile c il valore 1440 e alla variabile ans il valore 720. Di questi quattro comandi, viene visualizzato sullo schermo soltanto l'ultimo (perché non è seguito dal " ; ").

1.8 Operazioni booleane: >, <, >=, <=, ==, ~=, and & (&&), or | (||), not ~

Siano x e y due numeri.
- L'espressione x>y è 1 se x>y ed è 0 altrimenti. Discorso analogo vale per x<y, x>=y, x<=y.
- L'espressione x==y è 1 se x è uguale a y ed è 0 altrimenti.
- L'espressione x~=y è 1 se x è diverso da y ed è 0 altrimenti (~= è la versione Matlab di $\neq$).
- x&y è 1 se entrambi x e y sono diversi da 0 ed è 0 se almeno uno fra x e y è uguale a 0.
- x&&y è lo stesso che x&y (con la differenza che se x è 0 allora l'operatore && non va a vedere chi è y e fornisce subito risultato 0).
- x|y è 1 se almeno uno fra x e y è diverso da 0 ed è 0 se entrambi x e y sono 0.

- x||y è lo stesso che x|y (con la differenza che se x è diverso da 0 allora l'operatore || non va a vedere chi è y e fornisce subito risultato 1).
- ∼x è 1 se x è uguale a 0 ed è 0 se x è diverso da 0.

1.9 Altri comandi: freccia in su/giù, %, tic toc, Ctrl+C

- L'uso della freccia in su/giù è molto utile per recuperare dei comandi già dati in precedenza senza doverli riscrivere.
- Ogni riga di comando Matlab che è preceduta dal simbolo % non viene eseguita. Il simbolo % viene spesso usato per inserire commenti e spiegazioni all'interno dei codici Matlab, come ad esempio lo script e la function (si vedano le sezioni 5.4 e 5.5).
- tic toc permette di valutare il tempo che ci mette Matlab a eseguire una data sequenza di istruzioni. L'utilizzo è il seguente:
  > tic; SequenzaDiIstruzioni; toc;
  Dopo aver eseguito la SequenzaDiIstruzioni, Matlab visualizza una stringa che dice quanto tempo è passato tra il "tic" e il "toc".
- Ctrl+C serve per fermare la computazione di Matlab nel caso in cui essa non termina o ci mette un tempo troppo lungo a terminare. Dopo aver cliccato Ctrl+C, occorre però talvolta aspettare un bel po' di tempo prima che Matlab si fermi e/o cliccare nuovamente Ctrl+C.

2 Matrici

2.1 Numeri e vettori sono matrici

Per Matlab un numero è semplicemente una matrice 1×1 e un vettore è semplicemente una matrice in cui una delle due dimensioni è 1. Ad esempio, un vettore riga di 5 componenti è una matrice 1×5; una matrice 7×1 è un vettore colonna di 7 componenti. Pertanto, tutto quello che si dirà per le matrici si applica anche ai vettori.

2.2 Creazione di matrici col metodo base: parentesi quadre [ ], punto e virgola ; e virgola ,

Il comando
> A = [1 2 3 4; –1 –2 –3 –4; 0 –1.2 1.7 pi]
assegna ad A la matrice 3×4 data da

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & -2 & -3 & -4 \\ 0 & -1.2 & 1.7 & \pi \end{bmatrix} \tag{2.1}$$

Il comando
> A = [1, 2, 3, 4; –1, –2, –3, –4; 0, –1.2, 1.7, pi]
fa la stessa cosa del comando precedente. In altri termini, gli elementi di ogni riga della matrice possono essere separati da virgola oppure no, a scelta dell'utente. Invece il punto e virgola al termine di ogni riga è obbligatorio.

2.3 Creazione di matrici speciali: diag, eye, ones, zeros, rand

Se v è un vettore di componenti diciamo $v_1, v_2, \ldots, v_n$, il comando
> D = diag(v)

assegna a D la matrice diagonale $n \times n$ data da

$$D = \begin{bmatrix} v_1 & & & \\ & v_2 & & \\ & & \ddots & \\ & & & v_n \end{bmatrix}$$

Se m ed n sono numeri naturali, il comando
> I = eye(m,n)
assegna ad I la matrice identità m×n, e il comando
> I = eye(n)
assegna ad I la matrice identità n×n. Similmente, si hanno i comandi
> U = ones(m,n)
> U = ones(n)
(per creare matrici di tutti uni),
> Z = zeros(m,n)
> Z = zeros(n)
(per creare matrici di tutti zeri),
> R = rand(m,n)
> R = rand(n)
(per creare matrici di numeri casuali appartenenti all'intervallo $(0, 1)$).

2.4 Creazione di vettori equispaziati (griglie): i due punti : e linspace

Siano a, b due numeri reali con $a < b$ e sia $n \geq 2$ un numero naturale. Allora

$$\text{linspace}(a,b,n) = \left[a,\ a + \frac{b-a}{n-1},\ a + 2\frac{b-a}{n-1},\ \ldots,\ b\right]$$

cioè linspace(a,b,n) è il vettore riga formato dagli n punti equispaziati nell'intervallo $[a, b]$ che cominciano con a e finiscono con b. Ad esempio,

$$\text{linspace}(0,1,6) = \underbrace{[0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1]}_{6 \text{ punti}}$$

Siano a, b due numeri reali con $a < b$ e sia $h > 0$. Allora a:h:b è il vettore riga formato dai punti equispaziati di passo h che cominciano con a e finiscono con il più grande numero b' della forma "a più un multiplo di h" che risulta minore o uguale a b. Ad esempio,

$$1 : 2 : 17 = [1,\ 3,\ 5,\ 7,\ 9,\ 11,\ 13,\ 15,\ 17]$$
$$2 : 1.5 : 10 = [2,\ 3.5,\ 5,\ 6.5,\ 8,\ 9.5]$$
$$3 : 12 : 8 = 3$$

Siano a, b due numeri reali con $a > b$ e sia $h < 0$. Allora a:h:b è il vettore riga formato dai punti equispaziati di passo h che cominciano con a e finiscono con il più piccolo numero b' della forma "a più un multiplo di h" che risulta maggiore o uguale a b. Ad esempio,

$$1 : -2 : -7 = [1,\ -1,\ -3,\ -5,\ -7]$$
$$10 : -1.5 : 2 = [10,\ 8.5,\ 7,\ 5.5,\ 4,\ 2.5]$$
$$3 : -12 : -2 = 3$$

Se non si specifica h, Matlab assume che sia $h = 1$. In altre parole,

$$a : b = a : 1 : b$$

2.5 Lunghezza e dimensioni di una matrice: length, size

Sia A è una matrice $m \times n$.
- size(A) è il vettore riga [m,n].
- length(A) è il massimo tra m e n. In particolare, se A è un vettore v, allora length(v) è la lunghezza di v.

2.6 Operazioni sulle matrici: +, –, *, /, ^

Le operazioni aritmetiche +, –, * e l'elevamento a potenza ^ funzionano non solo sui numeri ma anche sulle matrici. Ricordiamo che, se A e B sono due matrici e n è un numero intero, allora:
- la somma A+B e la sottrazione A–B sono definite quando A e B hanno le stesse dimensioni;
- il prodotto A*B è definito quando la seconda dimensione di A coincide con la prima dimensione di B;
- l'elevamento a potenza A^n è definito quando A è quadrata.

Matlab ammette anche delle eccezioni alle regole matematiche. Infatti, se A è una matrice e x è un numero, allora:
- A+x [A–x] è la matrice ottenuta sommando [sottraendo] a tutte le componenti di A il numero x;
- x*A è la matrice ottenuta moltiplicando tutte le componenti di A per x.
- A*x è uguale a x*A.
- A/x è la matrice ottenuta dividendo tutte le componenti di A per x.

2.7 Operazioni sulle matrici: .*, ./, .^

Se A e B sono matrici delle stesse dimensioni e x è un numero, allora:
- A.*B è il prodotto puntuale di A e B, ovvero la matrice delle stesse dimensioni di A e B ottenuta moltiplicando ogni componente di A per la corrispondente componente di B;
- A./B è la divisione puntuale di A e B, ovvero la matrice delle stesse dimensioni di A e B ottenuta dividendo ogni componente di A per la corrispondente componente di B;
- A.^B è l'elevamento puntuale di A alla B, ovvero la matrice delle stesse dimensioni di A e B ottenuta elevando ogni componente di A alla corrispondente componente di B.
- A.^x è l'elevamento puntuale di A alla x, ovvero la matrice delle stesse dimensioni di A ottenuta elevando alla x ogni componente di A.

2.8 Operazioni sulle matrici: transpose, conj, apostrofo ′, diag, trace

Sia A una matrice $m \times n$.
- transpose(A) è la trasposta di A.
- conj(A) è la coniugata di A, cioè la matrice ottenuta sostituendo ogni elemento di A con il suo complesso coniugato (osserviamo che conj(A)=A se le componenti di A sono tutti numeri reali).
- A′ è la trasposta coniugata di A, cioè A′=transpose(conj(A)) (osserviamo che A′=transpose(A) se le componenti di A sono tutti numeri reali).
- diag(A) è un vettore colonna i cui elementi sono gli elementi diagonali di A.
- trace(A) è la traccia di A, cioè la somma degli elementi diagonali.

2.9 Operazioni sulle matrici: max, min, sum, norm

Sia A una matrice $m \times n$.
- max(A) [min(A)] è un vettore riga $1 \times n$ la cui prima componente è il massimo [minimo] della prima colonna di A, la seconda componente è il massimo [minimo] della seconda colonna di A, eccetera. Nel caso in cui A sia un vettore v (cioè una matrice in cui una delle dimensioni è uguale a 1), allora max(v)

[min(v)] è il massimo [minimo] delle componenti di v. Osserviamo che max(max(A)) [min(min(A))] è il massimo [minimo] di tutti gli elementi di A. [1]

- sum(A) è un vettore riga $1 \times n$ la cui prima componente è la somma degli elementi della prima colonna di A, la seconda componente è la somma degli elementi della seconda colonna di A, eccetera. Nel caso in cui A sia un vettore v (cioè una matrice in cui una delle dimensioni è uguale a 1), allora sum(v) è la somma di tutte le componenti di v. Osserviamo che sum(sum(A)) è la somma di tutti gli elementi di A.
- norm(A,1) è la norma 1 di A, norm(A)=norm(A,2) è la norma 2 di A, norm(A,Inf) è la norma infinito di A, norm(A,'fro') è la norma di Frobenius di A. Se p è un numero compreso tra 1 e $+\infty$ estremi inclusi (ricordiamo che $+\infty$ è una variabile predefinita Inf) e se A è un vettore v (cioè una matrice in cui una delle dimensioni è uguale a 1), allora norm(v,p) è la norma p di v.

2.10 Operazioni sulle matrici: det, inv, ˆ(–1), eig

Sia A una matrice quadrata $n \times n$.

- det(A) è il determinante di A.
- inv(A) è l'inversa di A (che esiste se det(A) è diverso da 0).
- Aˆ(–1) è uguale a inv(A).
- eig(A) è un vettore colonna contenente gli autovalori di A.
- Se si dà il comando
  > [T, Lambda] = eig(A)
  allora T è una matrice $n \times n$ le cui colonne $T^{(1)}$, $T^{(2)}$, ..., $T^{(n)}$ sono gli autovettori di A e Lambda è una matrice diagonale $n \times n$ i cui elementi diagonali λ_1, λ_2, ..., λ_n sono gli autovalori di A. Più precisamente, la prima colonna $T^{(1)}$ è un'autovettore corrispondente al primo autovalore λ_1, la seconda colonna $T^{(2)}$ è un autovettore corrispondente al secondo autovalore λ_2, eccetera; e risulta A*T=T*Lambda.

2.11 Risoluzione di sistemi lineari: backslash \

Sia A una matrice quadrata $n \times n$ invertibile e b un vettore $n \times 1$. La soluzione del sistema lineare Ax=b si ottiene con il comando
> x = A\b
Notiamo che si potrebbe anche usare il comando
> x = inv(A)*b
ma non è consigliato

2.12 Comandi relativi a componenti e sottomatrici/sottovettori: A(i,j), A(I,J), v(i), v(I) e i due punti :

Sia A una matrice $m \times n$.

- A(i,j) è la componente ij di A. Un assegnamento del tipo
  > A(i,j) = 8
  modifica A sostituendo la componente ij con 8.
- Se I è un vettore di numeri in $\{1, \ldots, m\}$ e J è un vettore di numeri in $\{1, \ldots, n\}$, allora A(I,J) è la sottomatrice di A corrispondente alle righe I e alle colonne J. Ad esempio, A([1 3 5],[2 3]) è la sottomatrice di A corrispondente alle righe 1,3,5 e alle colonne 2,3. Similmente, A(2:4,1:2) è la sottomatrice di A corrispondente alle righe 2,3,4 e alle colonne 1,2. Un assegnamento del tipo
  > A([1 3 5],[2 3]) = [0 1; 0 1; 0 1]
  modifica A sostituendo la sottomatrice A([1 3 5],[2 3]) con [0 1; 0 1; 0 1]. Un assegnamento del tipo

[1] I comandi max e min funzionano su matrici A e vettori v a componenti reali (non complessi).

> A([1 3 5],[2 3]) = –1
modifica A sostituendo la sottomatrice A([1 3 5],[2 3]) con tutti –1.

- Se I è un vettore di numeri in $\{1, \ldots, m\}$ allora A(I,:) è la sottomatrice di A corrispondente alle righe I e alle colonne dall prima all'ultima. In altri termini,

$$\mathrm{A}(\mathrm{I},:) = \mathrm{A}(\mathrm{I}, 1{:}n)$$

Similmente, se J è un vettore di numeri in $\{1, \ldots, n\}$ allora A(:,J) è la sottomatrice di A corrispondente alle righe dalla prima all'ultima e alle colonne J. In altri termini,

$$\mathrm{A}(:,\mathrm{J}) = \mathrm{A}(1{:}m, \mathrm{J})$$

Quindi hanno perfettamente senso in Matlab scritture del tipo A(2:4,:), A(:,3:7), A([1 3 5],:), A(1,:), A(:,8), eccetera.

Sia v un vettore di lunghezza n.

- v(i) è la componente i di v. Un assegnamento del tipo
> v(i) = 0
modifica v sostituendo la componente i con 0.
- Se I è un vettore di numeri in $\{1, \ldots, n\}$ allora v(I) è il sottovettore di v corrispondente alle componenti I. Ad esempio, v([1 3 5]) è il sottovettore di v corrispondente alle componenti 1,3,5. Similmente, v(2:5) è il sottovettore di v corrispondente alle componenti 2,3,4,5. Un assegnamento del tipo
> v(2:5) = [–1 0 –1 1]
modifica v sostituendo il sottovettore v(2:5) con [–1 0 –1 1]; si noti che questo comando funziona anche quando v(2:5) è un vettore colonna (e non un vettore riga come [–1 0 –1 1]). Un assegnamento del tipo
> v(2:5) = 7
modifica v sostituendo il sottovettore v(2:5) con tutti 7.

2.13 Comandi relativi a componenti e sottomatrici/sottovettori: A(:), i due punti : e A(i), A(I)

Sia A una matrice $m \times n$.

- A(:) è il vettore colonna di lunghezza mn ottenuto impilando le colonne di A una sotto l'altra. Più precisamente, se le colonne di A sono $\mathrm{A}^{(1)}$, $\mathrm{A}^{(2)}$, …, $\mathrm{A}^{(n)}$, allora

$$\mathrm{A}(:) = \begin{bmatrix} \mathrm{A}^{(1)} \\ \mathrm{A}^{(2)} \\ \vdots \\ \mathrm{A}^{(n)} \end{bmatrix}$$

- A(i) è la componente i di A intesa come la componente i di A(:). Ad esempio, se A è la matrice (2.1) allora A(1)=1, A(2)=–1, A(3)=0, A(4)=2, A(5)=–2, A(6)=–1.2, A(7)=3, A(8)=–3, A(9)=1.7, A(10)=4, A(11)=–4, A(12)=π. L'indicizzazione di A con un unico indice i anziché con due indici i,j si chiama *indicizzazione lineare* di A.
- Se I è un vettore di numeri in $\{1, \ldots, mn\}$ allora A(I) è il sottovettore di A(:) corrispondente alle componenti I. A(I) è vettore riga o colonna a seconda che I sia riga o colonna. Ad esempio, se A è la matrice (2.1) allora A([1 3 10])=[1 0 4] e A([1; 3; 10])=[1; 0; 4].

2.14 Concatenazione di matrici: parentesi quadre [ ], punto e virgola ; e virgola ,

La concatenazione di matrici è molto semplice in Matlab: si seguono le stesse regole della creazione di matrici con il metodo base. Alcuni esempi.

- Se u è un vettore colonna, allora v=[0;u;0] è un vettore colonna uguale ad u con aggiunti due zeri, uno in testa e uno in coda.
- Se B,C sono due matrici con lo stesso numero di righe, allora A=[B C] è una matrice con lo stesso numero di righe di B,C e con le colonne di C accostate a quelle di B. L'assegnamento A=[B,C] fa la stessa cosa di A=[B C].
- Se B,C sono due matrici con lo stesso numero di righe e B,D sono due matrici con lo stesso numero di colonne e C,E sono due matrici con lo stesso numero di colonne, allora possiamo definire la concatenazione A=[B C; D E] che è lo stesso che A=[B,C; D,E].

2.15 Operazioni booleane: >, <, >=, <=, ==, ∼=, and & (&&), or | (||), not ∼

Le operazioni booleane già viste per i numeri agiscono *componente per componente* sulle matrici. Più precisamente, date due matrici A e B delle stesse dimensioni, gli oggetti

A>B, A<B, A>=B, A<=B, A==B, A∼=B, A&B, A&&B, A|B, A||B, ∼A

sono matrici di zeri e uni delle stesse dimensioni di A e B; e l'elemento ij è 0 o 1 a seconda che l'esito dell'operazione booleana tra le componenti ij di A e B sia 0 o 1. Ad esempio,

$$\begin{bmatrix} 1 & -3 \\ 0 & 2 \end{bmatrix} > \begin{bmatrix} 0.5 & 10 \\ 0 & 3 \end{bmatrix} \text{ è uguale a } \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

Se A è una matrice e x è uno scalare, allora gli oggetti

A>x, A<x, A>=x, A<=x, A==x, A∼=x, A&x, A&&x, A|x, A||x

sono matrici di zeri e uni delle stesse dimensioni di A; e l'elemento ij è 0 o 1 a seconda che l'esito dell'operazione booleana tra A(i,j) e x sia 0 o 1. Ad esempio,

$$\begin{bmatrix} 1 & -3 \\ 0 & 2 \end{bmatrix} <= 0.5 \text{ è uguale a } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

2.16 Il comando find

Il comando find permette di trovare la posizione di quegli elementi di una matrice che soddisfano una determinata condizione. Per esprimere una condizione si usano tipicamente gli operatori booleani >, <, >=, <=, ==, ∼=, & (&&), | (||), ∼
Data una matrice A, il comando
> v = find(A)
restituisce un vettore v con gli indici corrispondenti a quelle componenti di A(:) che sono diversi da 0. Di conseguenza, tenendo conto che A>3 è una matrice di zeri e uni in cui gli uni compaiono nelle posizioni degli elementi di A che sono maggiori di 3, il comando
> v = find(A>3)
restituisce un vettore v con gli indici corrispondenti a quelle componenti di A(:) che sono maggiori di 3 (permettendo così di determinare le posizioni degli elementi di A che sono maggiori di 3). Similmente, il comando
> v = find(A==0)
restituisce un vettore v con gli indici corrispondenti a quelle componenti di A(:) che sono uguali a 0 (permettendo così di determinare le posizioni degli elementi di A che sono uguali a 0).

3 Funzioni e grafici

3.1 Funzioni presenti in Matlab e loro azione puntuale su matrici/vettori

Molte funzioni esistono già in Matlab come nelle calcolatrici scientifiche: abs (valore assoluto), exp (esponenziale in base e), log (logaritmo in base e), log2 (logaritmo in base 2), log10 (logaritmo in base 10), sqrt (radice quadrata), sin (seno), cos (coseno), tan (tangente), asin (arcoseno), acos (arcocoseno), atan (arcotangente), sinh (seno iperbolico), cosh (coseno iperbolico), tanh (tangente iperbolica), floor (parte intera inferiore), ceil (parte intera superiore), round (arrotondamento), sign (segno), real (parte reale), imag (parte immaginaria), factorial (fattoriale, solo per argomenti $n \geq 0$ interi). Tutte queste funzioni agiscono su matrici e vettori puntualmente, cioè *componente per componente*.

3.2 Grafici di funzioni: plot

Siano $x = [x_1, x_2, \ldots, x_n]$ e $y = [y_1, y_2, \ldots, y_n]$ due vettori della stessa lunghezza. Il comando
> plot(x,y)
crea una figura (chiamata "Figure 1") in cui viene rappresentata in un piano cartesiano una linea spezzata che unisce con dei segmenti i punti (x_1, y_1), (x_2, y_2), …, (x_n, y_n) *seguendo questo ordine*: il primo segmento unisce i punti (x_1, y_1) e (x_2, y_2), il secondo segmento unisce i punti (x_2, y_2) e (x_3, y_3), e così via fino all'ultimo segmento che unisce i punti (x_{n-1}, y_{n-1}) e (x_n, y_n). Il comando
> plot(x,y,'COLORE')
fa la stessa cosa di plot(x,y) e in più colora la linea spezzata con il colore COLORE (che può essere b (blue), r (red), g (green), y (yellow), m (magenta), c (cyan), k (black)). Il comando
> plot(x,y,'SIMBOLO')
crea una figura in cui è rappresentato un piano cartesiano insieme con segnati i punti (x_1, y_1), (x_2, y_2), …, (x_n, y_n); tali punti sono segnati con il simbolo SIMBOLO (che può essere o (cerchietto), * (asterisco), + (crocetta), x (crocetta a ics)). Il comando
> plot(x,y,'COLORE SIMBOLO')
fa la stessa cosa di plot(x,y,'SIMBOLO') e in più colora i vari simboli con il colore COLORE. Il comando
> plot(x,y,'COLORE-SIMBOLO')
fa la stesse cose di plot(x,y,'COLORE') e plot(x,y,'COLORE SIMBOLO'), cioè segna i punti (x_1, y_1), (x_2, y_2), …, (x_n, y_n) con il colore COLORE e il simbolo SIMBOLO, e inoltre unisce tali punti con una linea spezzata.

Esempio (*grafico di* $y = \sin(x)$ *sull'intervallo* $[0, \pi]$).
> x = linspace(0,pi,10000);
> y = sin(x);
> plot(x,y)

3.3 Grafici di funzioni: figure, hold on/off, legend, axis, close, close all

Il comando
> figure
apre una nuova figura vuota che potrà essere riempita con il comando plot. Se sono state aperte più figure, diciamo tre figure "Figure 1", "Figure 2" e "Figure 3", il comando
> figure(7)
crea una nuova figura intitolata "Figure 7" mentre il comando
> figure(2)
fa sì che Matlab consideri la "Figure 2" (già creata in precedenza) come la figura corrente sulla quale si

potrà poi lavorare usando il comando
> hold on
Tale comando fa sì che la figura corrente venga "tenuta in sospeso" di modo che su essa si possano tracciare altri grafici senza che questi vadano a rimpiazzare i grafici già tracciati precedentemente. Il comando
> hold off
fa sì che la figura corrente venga "rilasciata" di modo che i nuovi grafici che si tracceranno andranno a rimpiazzare quelli già tracciati precedentemente.
Supponiamo che in una figura siano stati tracciati i grafici di tre funzioni sull'intervallo $[0, 1]$, ad esempio il grafico di $y = \sin(x)$ (primo grafico tracciato), il grafico di $y = \cos(x)$ (secondo grafico tracciato) e il grafico di $y = x^2$ (ultimo grafico tracciato). Il comando
> legend('y=sin(x)','y=cos(x)','y=x^2')
inserisce la legenda. Il comando
> axis([a b c d])
fa sì che sulla figura corrente gli estremi dell'asse x siano a,b e gli estremi dell'asse y siano c,d. Il comando
> axis tight
fa sì che gli assi x e y della figura corrente siano ridotti il più possibile in modo da mettere in risalto il grafico rappresentato nella figura stessa. Il comando
> close
chiude la figura corrente. Il comando
> close all
chiude tutte le figure aperte.

Osservazione. Oltre ai comandi di base presentati in queste sezioni 3.2 e 3.3, nelle versioni recenti di Matlab (già a partire dal 2015) viene messa a disposizione dell'utente un'interfaccia grafica ricca di tavolozze per la manipolazione e l'abbellimento di figure e grafici.

3.4 Manipolazione e creazione di funzioni: function handle @(x), @(x,y)

Consideriamo il comando
> f = @(x)x^2
Tale comando fa sì che la variabile f diventi un oggetto speciale appartenente alla classe delle cosiddette "function handle". In pratica, dopo aver dato il comando precedente, f(1) è uguale a 1^2=1, f(4) è uguale a 4^2=16, f(sqrt(2)) è uguale a sqrt(2)^2=2, eccetera. Più in generale, f(u) *è uguale all'oggetto che si ottiene andando a sostituire* u *al posto di* x *nell'espressione di* f. In breve, f(u) è uguale a u^2, qualunque sia u. Ad esempio, se u è una matrice 2×2 allora f(u) è u^2, ossia la matrice 2×2 ottenuta elevando u al quadrato; se u è un vettore 2×1 allora f(u) dà errore perché l'espressione u^2 non è definita in Matlab per i vettori. Se però diamo il comando
> f =@(x) x.^2
allora f(u) è un'espressione definita anche per i vettori. Ad esempio, f([2 3]) è uguale a [4 9].

Esempio (*grafico di* $y = \sin(x)\cos(8x) + x\mathrm{e}^{-x}\chi_{[0,3]}(x)$ *sull'intervallo* $[0, 5]$).
Ricordiamo innanzitutto che il simbolo $\chi_{[0,3]}(x)$ denota la funzione caratteristica (o indicatrice) dell'intervallo $[0, 3]$, ossia la funzione che vale 1 se x appartiene a $[0, 3]$ e 0 altrimenti. Ecco ora un modo per tracciare il grafico di questa funzione utilizzando il comando @(x).
> y = @(x) sin(x).*cos(8*x)+x.*exp(–x).*(x>=0 & x<=3);
> griglia = linspace(0,5,10000);
> plot(griglia,y(griglia))

Il comando @ funziona anche con più di un argomento x. Ad esempio, se si dà il comando
> f = @(x,y) sin(x+y)
allora f(a,b) *è l'oggetto che si ottiene sostituendo* a *al posto di* x *e* b *al posto di* y *nell'espressione di* f.

4 Strutture di controllo

4.1 if, if/else

La sintassi è la seguente.

```
if condizione
    istruzioni;
end
```

```
if condizione
    istruzioni;
else
    altre istruzioni;
end
```

Nell'if senza else, Matlab verifica se la condizione è soddisfatta: se è soddisfatta esegue le istruzioni all'interno dell'if, altrimenti non fa niente. Nell'if/else, Matlab verifica se la condizione è soddisfatta: se è soddisfatta esegue le istruzioni all'interno dell'if, altrimenti esegue le istruzioni all'interno dell'else.

Per esprimere condizioni si possono usare:
- gli operatori di confronto: >, <, >=, <=, ==, ~=
- gli operatori booleani: and & (&&), or | (||), not ~

Esempio. La seguente sequenza di istruzioni genera un numero casuale c e definisce un nuovo numero d che vale 0 o 1 a seconda che c sia minore di 0.5 oppure maggiore o uguale a 0.5.

```
> c = rand(1);
> if c<0.5
d=0;
else
d=1;
end
```

4.2 for

La sintassi è la seguente.

```
for k = vettore riga
    istruzioni;
end
```

Nel for, Matlab si crea un indice k che scorre lungo le componenti del vettore riga, e per ognuna delle componenti di tale vettore riga esegue le istruzioni (che dipenderanno dal k in esame). Tipicamente, il vettore riga è un vettore del tipo 1:n = [1 2 3 ... n], dove n è un numero naturale. Per uscire dal ciclo for prima che k abbia scorso tutto il vettore riga, si usa il comando
break

Esempio. La seguente sequenza di istruzioni genera un vettore c di numeri casuali di lunghezza n=10 e sostituisce ciascuna componente di c on 0 o 1 a seconda che essa sia minore di 0.5 oppure maggiore o uguale a 0.5.

```
> n=10;
> c=rand(1,n);
> for k=1:n
if c(k)<0.5
c(k)=0;
else
c(k)=1;
end
end
```

4.3 while

La sintassi è la seguente.

```
while condizione
    istruzioni;
end
```

Nel while, Matlab continua ciclicamente a eseguire le istruzioni all'interno del while finché la condizione risulta soddisfatta. Non appena la condizione risulta non più soddisfatta, Matlab si ferma ed esce dal while. Per uscire dal ciclo while prima che la condizione risulti non più soddisfatta, si usa il comando
break

Esempio. La seguente sequenza di istruzioni fa la somma dei quadrati dei primi n=100 numeri e assegna il risultato alla variabile somma.

```
> n = 100;
> somma = 0;
> k = 1;
> while k<=n
somma = somma + k^2;
k = k+1;
end
```

5 Programmi/m-file

5.1 Definizione di m-file

Un m-file è semplicemente un file con estensione `.m`. Questi file sono quelli che possono essere letti ed eseguiti da Matlab, sono quindi dei veri e propri *programmi Matlab*. Affinché però Matlab possa eseguire un determinato m-file, esso *deve trovarsi in una cartella presente nel percorso di ricerca (path) di Matlab*; ad esempio, se l'm-file si trova nella cartella di lavoro (working directory), allora va bene (si veda la sezione 5.2 per i concetti di path e working directory). Le funzioni che abbiamo visto nelle sezioni precedenti (ad esempio, diag, transpose, det, inv, eccetera) sono state tutte scritte in opportuni m-file che si trovano nel path di Matlab (avremo quindi nel path di Matlab gli m-file `diag.m`, `transpose.m`, `det.m`, `inv.m`, eccetera).

5.2 Percorso di ricerca (path) e cartella di lavoro (working directory): path, pwd, cd, addpath, rmpath

Il percorso di ricerca (path) di Matlab è la *sequenza di cartelle (directories) in cui Matlab va a cercare gli m-file* quando essi vengono chiamati dall'utente. Ad esempio, se l'utente dà il comando
> A = rand(4)
allora Matlab va a cercare l'm-file `rand.m` nel suo path e quando lo trova lo esegue, restituendo una matrice casuale 4×4 che viene assegnata alla variabile A. Se Matlab non trovasse un m-file chiamato dall'utente, mostrerebbe un messaggio di errore. Il comando
> path
permette di vedere il path di Matlab ad eccezione di una cartella, la cosiddetta cartella di lavoro (working directory). Quando Matlab va a cercare un m-file, *inizia sempre dalla working directory*, dopodiché, se non trova lì l'm-file, *segue il suo path cominciando dalla cartella posta più in alto* nella lista mostrata dal comando >path. La working directory viene scelta dall'utente e in essa l'utente potrà salvare i propri programmi/m-file per farli eseguire da Matlab. Il comando "print working directory"
> pwd
permette di vedere la working directory corrente. In Matlab R2015b la working directory è visibile anche dall'interfaccia grafica (subito sopra la "Command Window"). Supponiamo di creare una cartella che chiamiamo WorkingDirectoryPreferita e supponiamo di voler cambiare la working directory con la nostra WorkingDirectoryPreferita. Per fare questo si usa il comando "change directory"
> cd 'PercorsoDellaWorkingDirectoryPreferita'
dove PercorsoDellaWorkingDirectoryPreferita è il percorso in cui si trova la WorkingDirectoryPreferita all'interno del computer. Ad esempio, se la WorkingDirectoryPreferita è stata creata direttamente nel disco "C:" allora si userà il comando
> cd 'C:\WorkingDirectoryPreferita'
In Matlab R2015b si può cambiare la working directory anche dall'interfaccia grafica andando a inserire il percorso C:\WorkingDirectoryPreferita subito sopra la "Command Window". Il comando
> addpath('C:\Users\UnAltraCartellaConProgrammiMatlab')
aggiunge al path di Matlab la cartella chiamata UnAltraCartellaConProgrammiMatlab che si trova nel disco "C:\Users". Il comando
> rmpath('C:\Users\UnAltraCartellaConProgrammiMatlab')
rimuove dal path di Matlab la cartella UnAltraCartellaConProgrammiMatlab.

5.3 Editor di Matlab: edit

Nelle prossime sezioni 5.4 e 5.5 vedremo quali sono i due fondamentali tipi di programmi (m-file) Matlab: lo script e la function. In questa sezione sottolineiamo che Matlab mette a disposizione un'editor suo per scrivere m-file. L'editor può essere aperto dando semplicemente il comando
> edit
Il comando
> edit MioPrimoMFile
permette di aprire/modificare il file `MioPrimoMFile.m` creato in precedenza (e collocato nel path di Matlab).

5.4 Script

Uno script è semplicemente un m-file contenente una sequenza di istruzioni/comandi Matlab. Se il nome dello script è `UnoScriptDiMatlab.m` e se lo script si trova nel path di Matlab (ad esempio nella working directory), per lanciare lo script dal prompt dei comandi basta usare il comando

> UnoScriptDiMatlab
Fatto questo, Matlab eseguirà una per una tutte le istruzioni dello script. Sottolineiamo che lanciare uno script è *esattamente lo stesso* che eseguirne le istruzioni direttamente dal prompt dei comandi. Invece di lanciare lo script uno potrebbe quindi copiare le istruzioni nel prompt dei comandi e lanciarle direttamente da lì: è la stessa cosa! In particolare, le variabili create da uno script sono *pubbliche*, vengono create nello spazio di lavoro pubblico (workspace) e sono visibili e utilizzabili dal prompt (il comando whos permette di vederle).

Esempio. Supponiamo di salvare in un script chiamato `Script_SommaDeiCubi.m` la seguente sequenza di istruzioni Matlab.

```
n = 10; Somma = 0;
for k = 1:n
Somma = Somma + k^3;
end
```

Supponiamo inoltre di collocare questo script nella working directory. Allora, dando dal prompt il comando
> Script_SommaDeiCubi
Matlab eseguirà lo script `Script_SommaDeiCubi.m`. Al termine dell'esecuzione, nel workspace ci saranno la variabile n=10 e la variabile Somma che conterrà la Somma dei primi n=10 cubi ovvero Somma = $1^3 + 2^3 + \ldots + 10^3$. Sostituendo n=10 con n=100, salvando nuovamente lo script e rilanciandolo dal prompt, si potrà ottenere la somma dei primi 100 cubi. È chiara la comodità dello script: basta cambiare n per ottenere la somma dei primi n cubi, senza doversi tutte le volte riscrivere la precedente sequenza di istruzioni.

5.5 Function

Una function è un m-file un po' particolare. La prima riga di una function è speciale e deve essere della forma seguente:

function [variabili di output]=NomeDellaFunzione(parametri di input)

oppure

function NomeDellaFunzione(parametri di input)

La seconda forma si usa se non ci sono variabili di output, altrimenti si usa la prima. Le variabili di output vanno scritte tra parentesi quadre e separate da virgole; qualora però ci fosse una sola variabile di output, allora si può anche evitare di racchiuderla tra parentesi quadre. I parametri di input vanno scritti tra parentesi tonde e separati da virgole. Una function deve *sempre* essere chiamata con lo stesso nome dell'm-file in cui essa viene salvata. Nel nostro caso specifico la function sia chiama NomeDellaFunzione e dunque deve essere salvata nell'm-file `NomeDellaFunzione.m`. Dopo la prima riga speciale, tutto il resto della function è una sequenza di istruzioni Matlab standard, come nello script. Al termine dell'esecuzione, una function immagazzina nelle sue variabili di output gli oggetti che essa vuole restituire all'utente. Sottolineiamo che, a differenza dello script, le variabili che una function crea al suo interno sono *private*: esse vengono create in uno spazio di lavoro privato della function (non nel workspace) e non sono visibili né utilizzabili dal prompt.

Esempio. Supponiamo che la prima riga di una function sia
function [x,r,c]=FunzioneDiEsempio(A,b,epsilon)
Allora questa function deve essere salvata in un m-file che si chiama `FunzioneDiEsempio.m` e che deve essere collocato nel path di Matlab (ad esempio, nella working diretory). Se C,y,s sono tre variabili che sono state create dal prompt nello spazio di lavoro pubblico, allora il comando

> FunzioneDiEsempio(C,y,s) fa partire la function FunzioneDiEsempio con parametri di input A=C, b=y, epsilon=s e, al termine dell'esecuzione, immagazzina nella variabile ans l'oggetto che la function ha immagazzinato in x (il suo primo output). Il comando
> v = FunzioneDiEsempio(C,y,s)
fa partire la function FunzioneDiEsempio con parametri di input A=C, b=y, epsilon=s e, al termine dell'esecuzione, immagazzina in v l'oggetto che la function ha immagazzinato in x. Il comando
> [v,r] = FunzioneDiEsempio(C,y,s)
fa partire la function FunzioneDiEsempio con parametri di input A=C, b=y, epsilon=s e, al termine dell'esecuzione, immagazzina in v l'oggetto che la function ha immagazzinato in x e in r l'oggetto che la function ha immagazzinato in r. Il comando
> [v,r,nit] = FunzioneDiEsempio(C,y,s)
fa partire la function FunzioneDiEsempio con parametri di input A=C, b=y, epsilon=s e, al termine dell'esecuzione, immagazzina in v l'oggetto che la function ha immagazzinato in x, in r l'oggetto che la function ha immagazzinato in r e in nit l'oggetto che la function ha immagazzinato in c.

Esempio. Supponiamo di salvare nella working directory la function `Function_SommaDeiCubi.m` così definita.

```
function Somma = Function_SommaDeiCubi(n)
Somma = 0;
for k = 1:n
Somma = Somma + k^3;
end
```

Allora, dando dal prompt il comando
> s10 = Function_SommaDeiCubi(10)
Matlab eseguirà la function `Function_SommaDeiCubi.m` con n=10. Al termine dell'esecuzione della function, la variabile s10 sarà diventata uguale alla variabile di output Somma restituita dalla function. In altri termini, s10 sarà la somma dei primi 10 cubi: $s10 = 1^3 + 2^3 + \ldots + 10^3$. Similmente, il comando
> s100 = Function_SommaDeiCubi(100)
assegna alla variabile s100 la somma dei primi 100 cubi.

6 Esercizi

6.1 Esercizi di base

1. Verificare se la matrice $A = \begin{bmatrix} 1 & 2 & 0 \\ -1 & -1 & 4 \\ 5 & 3 & 2 \end{bmatrix}$ è invertibile e, se lo è, risolvere il sistema lineare $A\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

2. Verificare che la matrice $B = \begin{bmatrix} 1 & -1 & 4 \\ -1 & 3 & -3 \\ -1 & 5 & -2 \end{bmatrix}$ non è invertibile e trovare un vettore $\mathbf{v} \in \mathbb{R}^3$ tale che $C = B + 3\mathbf{v}\mathbf{v}^T$ sia invertibile.

3. Una matrice quadrata A si dice *nilpotente* se esiste $n \geq 1$ tale che $A^n = O =$ matrice nulla. Verificare che le matrici

$$U = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad N = \frac{1}{8} \begin{bmatrix} 2 & -1 \\ 4 & -2 \end{bmatrix}$$

sono nilpotenti e per ciascuna di esse trovare il primo intero n per il quale la potenza n-esima è uguale alla matrice nulla. Verificare successivamente che tutti gli autovalori di U e di N sono nulli (in generale, vale il seguente risultato: una matrice quadrata è nilpotente se e solo se tutti i suoi autovalori sono nulli).

4. Costruire per $n = 5, 10, 100$ la matrice $n \times n$

$$M_n = \begin{bmatrix} 3 & -1 & \cdots & \cdots & -1 \\ -1 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -1 \\ -1 & \cdots & \cdots & -1 & 3 \end{bmatrix}, \qquad (M_n)_{ij} = \begin{cases} 3 & \text{se } i = j, \\ -1 & \text{altrimenti.} \end{cases}$$

Dire se M_{100} è invertibile, calcolarne l'autovalore minimo e dire se M_{100} è definita positiva (ricordiamo che una matrice simmetrica è definita positiva se e solo se tutti i suoi autovalori sono positivi).

5. Si consideri la matrice

$$B = \begin{bmatrix} 1 & 2 & 4 \\ 4 & 5 & 7 \\ 1 & 0 & -2 \end{bmatrix}.$$

Dire quali dei seguenti vettori appartengono al nucleo di B:

$$\mathbf{u} = [1, 2, 3]^T, \qquad \mathbf{v} = [2, -3, 1]^T, \qquad \mathbf{w} = [0, -2, 1]^T.$$

6. Costruire per $n = 5, 10, 100$ la matrice simmetrica $n \times n$

$$A_n = \begin{bmatrix} 4 & 1 & \cdots & \cdots & 1 \\ 1 & \ddots & & \bigcirc & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & \bigcirc & & \ddots & 1 \\ 1 & \cdots & \cdots & 1 & 4 \end{bmatrix}, \qquad (A_n)_{ij} = \begin{cases} 4 & \text{se } i = j; \\ 1 & \text{se } i = 1 \text{ e } j > 1, \text{ oppure se } i = n \text{ e } j < n, \\ & \text{oppure se } i > 1 \text{ e } j = 1, \text{ oppure se } i < n \text{ e } j = n; \\ 0 & \text{altrimenti.} \end{cases}$$

Calcolare $\|A_{100}\|_2$; calcolare poi l'autovalore massimo e minimo di A_{100} e verificare che entrambi sono in modulo minori o uguali di $\|A_{100}\|_2$ e che almeno uno di essi è in modulo uguale a $\|A_{100}\|_2$.

7. Costruire per $n = 2, 3, 4, 100$ la matrice $3n \times 3n$

$$C_n = \left[\begin{array}{c|c|c} \mathbf{1}_n & \mathbf{2}_n & \mathbf{3}_n \\ \hline \mathbf{4}_n & \mathbf{5}_n & \mathbf{6}_n \\ \hline \mathbf{7}_n & \mathbf{8}_n & \mathbf{9}_n \end{array}\right], \tag{6.1}$$

dove per ogni $k = 1, \ldots, 9$ il simbolo $\mathbf{k}_n$ denota la matrice $n \times n$ i cui elementi sono tutti uguali a k. Costruire poi la matrice D_{100} identica a C_{100} con l'unica differenza che il blocco centrale $\mathbf{5}_{100}$ è sostituito dalla matrice A_{100} dell'esercizio precedente.

8. Costruire la matrice a freccia

$$F_n = \begin{bmatrix} 1 & & & & & & 1 \\ & 2 & & & \mathbf{O} & & 2 \\ & & 3 & & & & 3 \\ & & & 4 & & & 4 \\ & \mathbf{O} & & & \ddots & & \vdots \\ & & & & & \ddots & \vdots \\ 1 & 2 & 3 & 4 & \cdots & \cdots & n \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{6.2}$$

per $n = 5, 10, 100$.

9. Si costruisca il vettore riga $\mathbf{v} = [1 \;\; 2 \;\; 3 \;\; 4 \;\; \cdots \;\; 100]$. Con un'unica riga di comando dal prompt di Matlab, si calcoli il risultato del seguente prodotto:

$$[1 \;\; 8 \;\; 27 \;\; 64 \;\; \cdots \;\; 1000000] \begin{bmatrix} -1 & -1 & \cdots & \cdots & \cdots & -1 \\ -1 & -1 & \cdots & \cdots & \cdots & -1 \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & & & \vdots \\ -1 & -1 & \cdots & \cdots & \cdots & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1/4 \\ 1/9 \\ 1/16 \\ \vdots \\ 1/10000 \end{bmatrix}.$$

10. Con un'unica riga di comando dal prompt di Matlab, calcolare la somma S delle prime 100 radici cubiche,

$$S = \sqrt[3]{1} + \sqrt[3]{2} + \sqrt[3]{3} + \ldots + \sqrt[3]{100}.$$

11. Calcolare il raggio spettrale della matrice $A = \begin{bmatrix} 1 & -1 \\ 2 & 4 \end{bmatrix}$ con un'unica riga di comando dal prompt di Matlab.

12. Tracciare il grafico delle seguenti funzioni sull'intervallo $[0.1, 2.7]$:
    - e^x
    - $|x|\mathrm{e}^{-2x}$
    - $\sqrt{x}\,\cos(x^3) + 1.3$
    - $\cos(8x)\sin^2(x) + 1$
    - $|x|^3 \log^2(x|\cos(4x)| + 1/2)$
    - $x^2\mathrm{e}^{-x} + \cos(x^2 - 1)$
    - $\dfrac{x^2\cos(x^3 - 1)}{1 + x^{\log(x)}}$
    - $\dfrac{\mathrm{e}^x|x^2 - 1 - \log(x)|}{\arctan(x) + \pi + \sqrt[3]{x}}$
    - $\dfrac{2^x \arccos(x^2/10)}{1 + x^2\mathrm{e}^{-|x^2\sin(x) - \cos(3x)|}}$

13. Tracciare il grafico delle seguenti funzioni:

- $\mathrm{e}^{-x^2}\chi_{(-\infty,0]}(x)$ sull'intervallo $[-1,2]$
- $\cos(\sqrt[3]{x^4+1})\chi_{[0,3]}(x)+\chi_{(3,\infty)}(x)$ sull'intervallo $[-1,4]$
- $(2x-x^2)\chi_{[0,1]}(x)+\frac{1}{x}\chi_{(1,2]}(x)+\frac{1}{2}\chi_{(2,\infty)}(x)$ sull'intervallo $[-1,4]$

14. Tracciare in un unico grafico di Matlab la quinta e la sesta funzione dell'esercizio 12, una in colore rosso e l'altra in colore verde; aggiungere poi una legenda che ci dica qual'è la funzione rossa e qual'è quella verde.

15. Tracciare il grafico delle seguenti curve in $\mathbb{R}^2$:
    - (t,t^2) per $t\in[-1.5,1.5]$
    - $(t,\mathrm{e}^{-t}\cos t)$ per $t\in[0,2\pi]$
    - $(\cos t,3\sin t)$ per $t\in[0,2\pi]$
    - $(t-\sin t,1-\cos t)$ per $t\in[0,5\pi]$ [(2)]
    - $(1+\cos(t^2),\mathrm{e}^{\sin t})$ per $t\in[0,2\pi]$
    - $(\log(t+1),\sin(5t)\cos(5t))$ per $t\in[0,2\pi]$
    - $(\tan\frac{1+\mathrm{e}^{-t}}{1+t^2},t^{\cos(t^{2/3})})$ per $t\in[0.5,5]$

16. Costruire per $n=5,10,100$ la matrice di Hilbert $n\times n$, la cui componente (i,j) è

$$(H_n)_{ij}=\frac{1}{i+j-1},\qquad i,j=1,\ldots,n. \tag{6.3}$$

    Ad esempio,

$$H_4=\begin{bmatrix}1 & 1/2 & 1/3 & 1/4\\ 1/2 & 1/3 & 1/4 & 1/5\\ 1/3 & 1/4 & 1/5 & 1/6\\ 1/4 & 1/5 & 1/6 & 1/7\end{bmatrix}.$$

    Si può dimostrare che H_n è una matrice simmetrica definita positiva per tutti gli n, quindi in particolare H_n è non singolare per tutti gli n. Ciononostante, H_n risulta malcondizionata, nel senso che il suo numero di condizionamento $\mu_2(H_n)$ è molto grande già per valori piccoli di n e aumenta al crescere di n. Per verificare questo fatto, calcolare il numero di condizionamento $\mu_2(H_n)=\|H_n\|_2\|H_n^{-1}\|_2$ per $n=5,7,10$ e verificare che è molto grande. Calcolare poi anche $\det(H_n)$ per $n=5,7,10$ e verificare che è estremamente piccolo (H_n è "quasi singolare" e questo è uno dei motivi per cui è malcondizionata).

17. Costruire un vettore di numeri casuali $\mathbf{v}$ di dimensione $100\,000$ [(3)] e calcolare il numero ξ così definito: se $\mathbf{v}$ contiene almeno un numero $<10^{-4}$ allora ξ è il primo indice $j\in\{1,\ldots,100\,000\}$ tale che $\mathbf{v}(j)<10^{-4}$, altrimenti ξ è il minimo di $\mathbf{v}$. (Domanda: qual'è la probabilità che il vettore $\mathbf{v}$ contenga almeno un numero $<10^{-4}$?)

18. Costruire un vettore di numeri casuali $\mathbf{w}$ di dimensione $10\,000$ e sostituire tutte le componenti di $\mathbf{w}$ in $[0,1/5)$ con 0, tutte le componenti di $\mathbf{w}$ in $[1/5,2/5)$ con $1/5$, tutte le componenti di $\mathbf{w}$ in $[2/5,3/5)$ con $2/5$, tutte le componenti di $\mathbf{w}$ in $[3/5,4/5)$ con $3/5$ e tutte le componenti di $\mathbf{w}$ in $[4/5,1)$ con $4/5$ (in questo modo è stata "compressa" l'informazione contenuta in $\mathbf{w}$ consentendo solo i 5 valori $0,1/5,2/5,3/5,4/5$).

---

(2) Questa curva si chiama "cicloide" ed è la traiettoria di un punto su una circonferenza di raggio 1 che rotola su una retta.
(3) Ricordare il ; per non visualizzare il vettore...

19. Costruire per $n = 5, 10, 100$ il "laplaciano discreto", cioè la matrice A_n così definita:

$$A_n = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{6.4}$$

Calcolare poi gli autovalori di A_{10} e verificare che sono dati da

$$\lambda_j(A_{10}) = 2 - 2\cos\left(\frac{j\pi}{11}\right), \qquad j = 1, \ldots, 10.$$

20. Calcolare i vettori $\mathbf{x}^{(5)}, \mathbf{x}^{(10)}, \mathbf{x}^{(100)}$, dove $\{\mathbf{x}^{(n)}\}_{n=0,1,2,\ldots}$ è la successione di vettori in $\mathbb{R}^2$ definita ricorsivamente nel modo seguente:

$$\begin{cases} \mathbf{x}^{(0)} = [1, -1], \\ \mathbf{x}^{(n+1)} = [x_1^{(n)} + \sin(x_2^{(n)}), (x_2^{(n)})^2 + 1], \qquad n \geq 0. \end{cases}$$

Costruire inoltre la matrice $(n+1) \times 2$ data da

$$X_n = \begin{bmatrix} \mathbf{x}^{(0)} \\ \hline \mathbf{x}^{(1)} \\ \hline \vdots \\ \hline \mathbf{x}^{(n)} \end{bmatrix}$$

per $n = 5, 10, 100$.

21. Creare una function che prende in input due interi $k, h \in \mathbb{Z}$ e restituisce

$$\delta_{hk} = \begin{cases} 1 & \text{se } h = k \\ 0 & \text{altrimenti} \end{cases}$$

22. Creare una function che prende in input un vettore $\mathbf{v}$ di dimensione ≥ 2 (chiamiamo n la dimensione di $\mathbf{v}$) e costruisce il vettore $\mathbf{w}$ di dimensione $n-1$ la cui componente i-esima è

$$\mathbf{w}(i) = \frac{\mathbf{v}(i) + \mathbf{v}(i+1)}{2}, \qquad i = 1, \ldots, n-1.$$

23. Creare uno script che all'inizio si definisce un intero $n \geq 1$ e poi costruisce la matrice a freccia F_n data dalla (6.2); utilizzare lo script per costruire F_5, F_{10}, F_{100} come nell'esercizio 8 e verificare tramite il comando whos che le matrici F_5, F_{10}, F_{100} vengono create nello spazio di lavoro pubblico (come se F_5, F_{10}, F_{100} fossero state create dal prompt).

24. Creare una function che prende in input un intero $n \geq 1$ e costruisce la matrice a freccia F_n. Utilizzare la function per costruire F_5, F_{10}, F_{100}.

25. Creare uno script che all'inizio si definisce un intero $n \geq 1$ e poi costruisce la matrice di Hilbert H_n data dalla (6.3); utilizzare lo script per costruire H_5, H_7, H_{10} come nell'esercizio 16 e verificare tramite il comando whos che le matrici H_5, H_7, H_{10} vengono create nello spazio di lavoro pubblico. Calcolare dal prompt anche gli autovalori di H_{10} per verificare che H_{10}, che è stata creata nello spazio di lavoro pubblico, è utilizzabile liberamente dal prompt come se fosse stata creata dal prompt stesso invece che dallo script. Usare format long per visualizzare un maggior numero di cifre di precisione negli autovalori di H_{10}.

26. Creare una function che prende in input un intero $n \geq 1$ e costruisce la matrice di Hilbert H_n. Utilizzare la function per costruire H_5, H_7, H_{10}.

27. Creare uno script che all'inizio si definisce un intero $n \geq 1$ e poi costruisce la matrice C_n data dalla (6.1). Verificare la correttezza dello script per $n = 3$.

28. Creare una function che prende in input un intero $n \geq 1$ e costruisce la matrice C_n data dalla (6.1). Utilizzare la function per costruire C_3.

29. Creare una function che prende in input tre numeri reali $a, b, c \in \mathbb{R}$ e un intero $n \geq 2$ e costruisce la matrice tridiagonale $n \times n$ denotata con $T_n(a, b, c)$ e data da

$$T_n(a,b,c) = \begin{bmatrix} b & c & & & \\ a & b & c & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & c \\ & & & a & b \end{bmatrix}.$$

    Utilizzare la function per costruire il laplaciano discreto A_n dato dalla (6.4) per $n = 5, 10, 30$.

30. Costruire una function che prende in input due numeri $a, b > 0$ e restituisce il più piccolo intero $n \geq 1$ tale che $na > b$.

31. Costruire una function che prende in input un intero $n \geq 1$ e un reale $\alpha > 0$ e calcola la somma

$$\sum_{k=1}^{n} k^\alpha = 1^\alpha + 2^\alpha + \ldots + n^\alpha$$

    e il prodotto

$$\prod_{k=1}^{n} k^\alpha = 1^\alpha \, 2^\alpha \, \cdots \, n^\alpha.$$

    La function deve restituire entrambi i risultati e, nel caso in cui l'utente ne richieda uno solo, deve restituire la somma.

32. Costruire una function che prende in input un numero reale $\rho > 0$ e calcola il primo intero $r \geq 0$ tale che $\dfrac{\rho}{2^r} < 1$. La function deve restituire sia r sia il rapporto $\dfrac{\rho}{2^r}$.

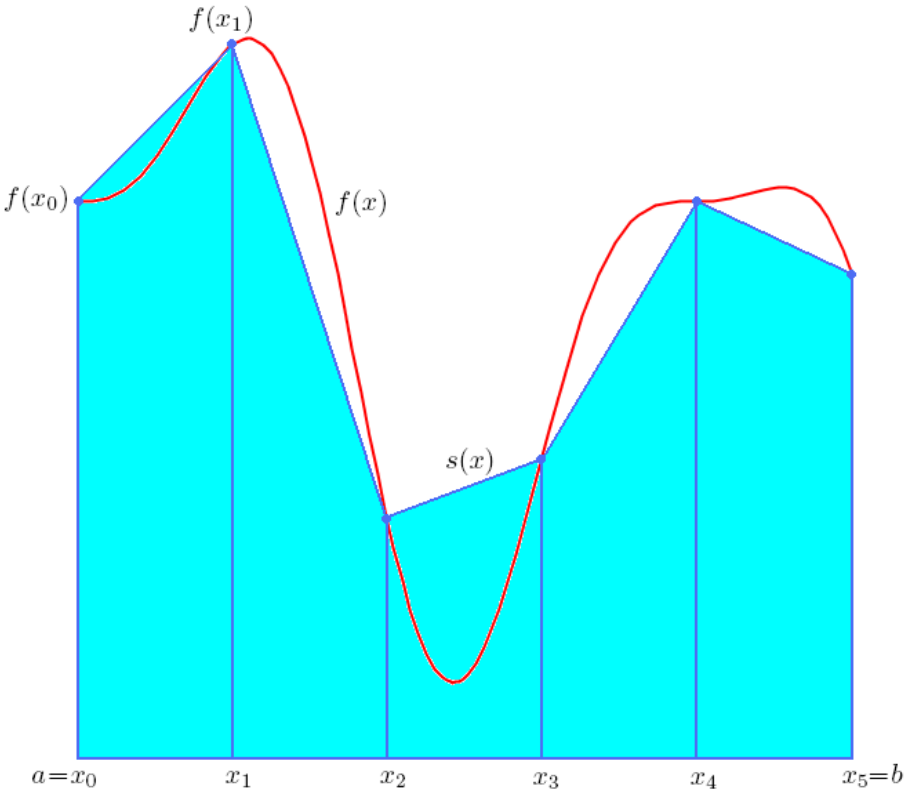

Figura 6.1: Formula dei trapezi per $n = 5$.

6.2 Esercizi più difficili

i. Costruire una function che prende in input un vettore $\mathbf{w}$ con tutte le componenti in $(0, 1)$ e un intero $n \geq 1$ e "comprime" l'informazione contenuta in $\mathbf{w}$ andando a calcolare un nuovo vettore $\widetilde{\mathbf{w}}$ che ha la stessa lunghezza di $\mathbf{w}$ e che è fatto nel modo seguente: ogni componente $\widetilde{w}_i$ di $\widetilde{\mathbf{w}}$ si ottiene sostituendo la corrispondente componente w_i di $\mathbf{w}$ con k/n, dove k è l'unico intero in $\{0, \ldots, n-1\}$ tale che l'intervallo $[k/n,\ (k+1)/n)$ contiene w_i. Ad esempio, se $\mathbf{w} = [0.1,\ 0.43,\ 0.89,\ 0.23,\ 0.5,\ 0.63,\ 0.21,\ 0.29]$ e $n = 5$ allora $\widetilde{\mathbf{w}} = [0,\ 0.4,\ 0.8,\ 0.2,\ 0.4,\ 0.6,\ 0.2,\ 0.2]$.

ii. Data una funzione continua $f : [a, b] \to \mathbb{R}$, un modo per approssimare $\int_a^b f(x)\mathrm{d}x$ è il seguente: si suddivide l'intervallo $[a, b]$ in n sottointervalli tutti della stessa ampiezza $h = \frac{b-a}{n}$, si pone $x_j = a + jh$ per ogni $j = 0, 1, \ldots, n$, e si prende come approssimazione di $\int_a^b f(x)\mathrm{d}x$ il valore $\int_a^b s(x)\mathrm{d}x$, dove

$$s : [a, b] \to \mathbb{R}, \quad \begin{cases} s(x) = f(x_j) + \dfrac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j}(x - x_j) = f(x_j) + \dfrac{f(x_{j+1}) - f(x_j)}{h}(x - x_j), \\ \text{se } x \in [x_j, x_{j+1}], \quad j = 0, \ldots, n-1, \end{cases}$$

è la funzione lineare a tratti mostrata in Figura 6.1. Quindi il valore che si prende come approssimazione di $\int_a^b f(x)\mathrm{d}x$ è

$$I_n = \int_a^b s(x)\mathrm{d}x = \sum_{j=0}^{n-1} \int_{x_j}^{x_{j+1}} s(x)\mathrm{d}x = \sum_{j=0}^{n-1} \frac{f(x_j) + f(x_{j+1})}{2}\, h = \frac{h}{2} \sum_{j=0}^{n-1} [f(x_j) + f(x_{j+1})]$$
$$= h \left[\frac{f(a) + f(b)}{2} + \sum_{j=1}^{n-1} f(x_j) \right]. \tag{6.5}$$

La (6.5) prende il nome di *formula dei trapezi di ordine n.*

Scrivere una function in Matlab che prende in input gli estremi a, b di un intervallo, un numero naturale $n \geq 1$ e una funzione $f(x)$ definita su $[a, b]$, e calcola I_n (l'approssimazione di $\int_a^b f(x)\mathrm{d}x$ data dalla formula dei trapezi di ordine n).

Una volta creata la function, digitare "format long" dal prompt di Matlab (in modo da vedere molte cifre decimali nei numeri) e sperimentare la function nel modo seguente.

- Calcolare l'approssimazione di $\int_0^2 x\mathrm{e}^x\mathrm{d}x$ ottenuta con I_{25}, I_{50}, I_{75}, I_{100}, I_{200} e confrontarle con il valore esatto $\int_0^2 x\mathrm{e}^x\mathrm{d}x = 1 + \mathrm{e}^2$.
- Calcolare l'approssimazione di $\int_0^1 \sqrt{\cos x}\,\mathrm{d}x$ ottenuta con

$$I_5,\ I_{10},\ I_{20},\ I_{40},\ I_{80},\ I_{160},\ I_{320},\ I_{640},\ I_{1280},\ I_{2560},\ I_{5120},\ I_{10240},\ I_{20480}.$$

  Verificare se i valori calcolati si stabilizzano intorno ad uno stesso valore (dovrebbero in teoria stabilizzarsi intorno al valore esatto di $\int_0^1 \sqrt{\cos x}\,\mathrm{d}x$, verso il quale la successione I_n converge per $n \to \infty$; notiamo che $\int_0^1 \sqrt{\cos x}\,\mathrm{d}x$ non può essere calcolato esattamente perché non si conosce una primitiva elementare di $\sqrt{\cos x}$).

iii. Siano $(x_0, y_0), \ldots, (x_n, y_n) \in \mathbb{R}^2$ con $x_0, \ldots, x_n$ tutti distinti. Per un teorema dell'analisi numerica, esiste un unico polinomio $p(x) \in \mathbb{R}_n[x]$ tale che $p(x_i) = y_i$ per ogni $i = 0, \ldots, n$; esso si chiama *polinomio d'interpolazione dei valori $y_0, \ldots, y_n$ sui nodi $x_0, \ldots, x_n$.*[(4)] Se per ogni $j = 0, \ldots, n$ definiamo il polinomio

$$L_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)},$$

che si chiama *j-esimo polinomio di Lagrange relativo ai nodi $x_0, \ldots, x_n$,* allora per ogni $i, j = 0, \ldots, n$ risulta

$$L_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{se } i = j, \\ 0 & \text{altrimenti.} \end{cases}$$

Da questo segue che il polinomio d'interpolazione dei valori $y_0, \ldots, y_n$ sui nodi $x_0, \ldots, x_n$ è

$$p(x) = \sum_{j=0}^{n} y_j L_j(x). \tag{6.6}$$

Infatti il polinomio $p(x)$ definito in (6.6) appartiene a $\mathbb{R}_n[x]$ come combinazione lineare dei polinomi $L_0(x), \ldots, L_n(x) \in \mathbb{R}_n[x]$ e soddisfa $p(x_i) = \sum_{j=0}^n y_j L_j(x_i) = \sum_{j=0}^n y_j \delta_{ij} = y_i$ per ogni $i = 0, \ldots, n$.

Utilizzando la (6.6), che si chiama *forma di Lagrange* del polinomio d'interpolazione $p(x)$, scrivere una function in Matlab che prende in input due vettori $[x_0, \ldots, x_n]$, $[y_0, \ldots, y_n]$ e un punto $t \in \mathbb{R}$ e calcola $p(t)$, dove $p(x)$ è il polinomio d'interpolazione dei valori $y_0, \ldots, y_n$ sui nodi $x_0, \ldots, x_n$.

iv. Utilizzando la function costruita risolvendo l'esercizio iii., scrivere una nuova function che prende in input tre vettori $[x_0, \ldots, x_n]$, $[y_0, \ldots, y_n]$, $[t_1, \ldots, t_m]$ e restituisce il vettore $[p(t_1), \ldots, p(t_m)]$, dove $p(x)$ è il polinomio d'interpolazione dei valori $y_0, \ldots, y_n$ sui nodi $x_0, \ldots, x_n$.

[(4)] Se $y_0, \ldots, y_n$ sono i valori di una funzione $f(x)$ in $x_0, \ldots, x_n$, cioè se risulta $f(x_i) = y_i$ per ogni $i = 0, \ldots, n$, allora $p(x)$ si chiama anche *polinomio d'interpolazione di $f(x)$ sui nodi $x_0, \ldots, x_n$.*

v. Utilizzando la function costruita risolvendo l'esercizio iv., disegnare in un'unica figura di Matlab i seguenti grafici sull'intervallo $[0, 5]$:

- il grafico della funzione $\sqrt{x}$ (con linea continua blu);
- il grafico del polinomio d'interpolazione $p(x)$ di $\sqrt{x}$ relativo ai nodi

$$1.69,\ 1.7689,\ 1.8769,\ 1.96,\ 2.0449,\ 2.1609,\ 2.25$$

(con cerchietti rossi, senza linea continua).

Aggiungere alla figura una legenda che ci dica qual'è la funzione $\sqrt{x}$ e qual'è $p(x)$. Calcolare inoltre il vettore (colonna)

$$\left[p(\zeta_1) - \sqrt{\zeta_1} \qquad p(\zeta_2) - \sqrt{\zeta_2} \qquad \cdots \qquad p(\zeta_{101}) - \sqrt{\zeta_{101}}\right]^T,$$

dove $\zeta_i = \frac{i-1}{20}$ per ogni $i = 1, \ldots, 101$, e osservare il modo in cui varia la differenza $p(\zeta_i) - \sqrt{\zeta_i}$ al variare di i da 1 a 101.

vi. Costruire una function che prende in input un intero $n \geq 1$ e un numero $p \in (0, 1)$ e agisce nel modo seguente.

- Esegue n esperimenti: all'i-esimo esperimento ($1 \leq i \leq n$), "estrae" uno dopo l'altro dei numeri casuali in $(0, 1)$ e si ferma all'istante T_i del primo successo (successo = estrazione di un numero $\leq p$, insuccesso = estrazione di un numero $> p$).
- Calcola la media $\frac{1}{n}\sum_{i=1}^{n} T_i$.

Siccome in ciascuno degli n esperimenti ogni estrazione ha probablità p di essere un successo, $T_1, \ldots, T_n$ sono variabili aleatorie geometriche indipendenti di parametro p. Per la legge forte dei grandi numeri, $\frac{1}{n}\sum_{i=1}^{n} T_i$ tende quasi certamente a $\mathbb{E}(T_1) = 1/p$, e questo significa che se eseguiamo la function per valori di n crescenti, siamo quasi certi (con probabilità 1 di indovinare) che il risultato $\frac{1}{n}\sum_{i=1}^{n} T_i$ della function converge a $1/p$. Dopo aver costruito la function, utilizzarla per $p = 1/2$ (il caso del lancio di una moneta), scegliendo

$$\begin{aligned} n = 5,\ 10,\ 20,\ 40,\ 80,\ 160,\ 320,\ 640,\ 1280,\ 2560,\ 5120,\\ 10240,\ 20480,\ 40960,\ 81920,\ 163840,\ 327680, 655360, \end{aligned} \tag{6.7}$$

e guardare se i risultati convergono a 2 al crescere di n. Utilizzarla poi per $p = 1/6$ (il caso del lancio di un dado) e $p = 1/12$ (il caso del lancio di un dodecaedro regolare), scegliendo ancora i valori di n in (6.7), e guardare se i risultati convergono rispettivamente a 6 e a 12.

vii. Sia $f : [a, b] \to \mathbb{R}$ una funzione continua con $f(a)f(b) < 0$ e con un unico zero $\zeta \in (a, b)$. Un metodo per determinare un'approssimazione ξ di ζ è il metodo di bisezione: dato un $\epsilon > 0$, il metodo costruisce la successione di intervalli

$$[\alpha_k, \beta_k], \qquad k = 0, 1, \ldots$$

in cui $[\alpha_0, \beta_0] = [a, b]$ e, per $k \geq 1$,

$$[\alpha_k, \beta_k] = \begin{cases} \left[\alpha_{k-1}, \dfrac{\alpha_{k-1} + \beta_{k-1}}{2}\right], & \text{se } \zeta \in \left[\alpha_{k-1}, \dfrac{\alpha_{k-1} + \beta_{k-1}}{2}\right] \text{ cioè } f(\alpha_{k-1}) \cdot f\left(\dfrac{\alpha_{k-1} + \beta_{k-1}}{2}\right) \leq 0 \\ \left[\dfrac{\alpha_{k-1} + \beta_{k-1}}{2}, \beta_{k-1}\right], & \text{altrimenti.} \end{cases}$$

La successione di intervalli così costruita gode delle seguenti proprietà:

- $\zeta \in [\alpha_k, \beta_k]$ per tutti i $k \geq 0$;
- ogni intervallo è metà del precedente e la lunghezza di $[\alpha_k, \beta_k]$ è $\beta_k - \alpha_k = \frac{b-a}{2^k}$ per ogni $k \geq 0$;
- $\alpha_k \to \zeta$ e $\beta_k \to \zeta$ per $k \to \infty$.

Il metodo si arresta al primo indice K tale che $\beta_K - \alpha_K \leq \epsilon$ e restituisce come risultato il numero $\xi = \frac{\alpha_K + \beta_K}{2}$ (in questo modo, siccome $\zeta \in [\alpha_K, \beta_K]$, si ha $|\zeta - \xi| \leq \frac{\epsilon}{2}$). Osserviamo che K è il più piccolo intero ≥ 0 tale che $\frac{b-a}{2^K} \leq \epsilon$, ovvero $K = \left\lceil \log_2 \left(\frac{b-a}{\epsilon} \right) \right\rceil$.

- Costruire una function che prende in input gli estremi a, b di un intervallo, una funzione continua $f : [a, b] \to \mathbb{R}$, con $f(a)f(b) < 0$ e con un unico zero $\zeta \in (a, b)$, e un $\epsilon > 0$, e calcola l'approssimazione ξ di ζ ottenuta con il metodo di bisezione sopra descritto.
- Si consideri la funzione $f(x) = x^3 + 3x - 1 - \mathrm{e}^{-x^2}$ e se ne disegni (con Matlab) il grafico sull'intervallo $[0, 1]$. Dopo aver verificato dal grafico che $f(0)f(1) < 0$ e che $f(x)$ ha un unico zero $\zeta \in (0, 1)$, calcolare con la function costruita al punto precedente un'approssimazione ξ di ζ tale che

$$|\xi - \zeta| \leq 10^{-12}$$

  e calcolare successivamente $f(\xi)$ (che dovrebbe essere circa 0).

viii. Siano $A \in \mathbb{C}^{n \times n}$ invertibile e $\mathbf{b} \in \mathbb{C}^n$. Consideriamo il sistema lineare

$$A\mathbf{x} = \mathbf{b}, \tag{6.8}$$

il quale ha un'unica soluzione $\mathbf{x} = A^{-1}\mathbf{b} \in \mathbb{C}^n$. Per risolvere (6.8), si può usare un metodo iterativo costruito nel modo seguente.

- Si considera una decomposizione di A della forma

$$A = M - (M - A)$$

  con $M \in \mathbb{C}^{n \times n}$ invertibile.
- Si sceglie un vettore iniziale $\mathbf{x}^{(0)} \in \mathbb{C}^n$ e si calcolano i termini della successione

$$\mathbf{x}^{(k)} = (I - M^{-1}A)\mathbf{x}^{(k-1)} + M^{-1}\mathbf{b}, \qquad k = 1, 2, \ldots \tag{6.9}$$

Per un teorma dell'analisi numerica, se il raggio spettrale $\rho(I - M^{-1}A) < 1$ allora, *qualunque sia il vettore iniziale* $\mathbf{x}^{(0)}$, la successione (6.9) converge a $\mathbf{x} = A^{-1}\mathbf{b}$ (componente per componente e anche rispetto ad ogni norma di $\mathbb{C}^n$).

Sia ora $A \in \mathbb{C}^{n \times n}$ una matrice invertibile con elementi diagonali non nulli e sia $\mathbf{b} \in \mathbb{C}^n$. In tal caso possiamo costruire il metodo di Jacobi per risolvere il sistema (6.8), il quale non è altro che il metodo appena descritto con

$$M = D = \begin{bmatrix} a_{11} & & & \mathrm{O} \\ & a_{22} & & \\ & & \ddots & \\ \mathrm{O} & & & a_{nn} \end{bmatrix}$$

(notiamo che D è invertibile perché abbiamo supposto $a_{ii} \neq 0$ per ogni $i = 1, \ldots, n$). Scrivere una function che prende in input una matrice invertibile $A \in \mathbb{C}^{n \times n}$ con elementi diagonali non nulli, un vettore $\mathbf{b} \in \mathbb{C}^n$, una tolleranza $\epsilon > 0$ e un vettore iniziale $\mathbf{x}^{(0)} \in \mathbb{C}^n$, e fa le cose seguenti.

(a) Calcola i termini della successione

$$\mathbf{x}^{(k)} = (I - D^{-1}A)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}, \qquad k = 1, 2, \ldots \tag{6.10}$$

associata al metodo di Jacobi.

(b) Si arresta al primo termine $\mathbf{x}^{(K)}$ tale che

$$\|\mathbf{b} - A\mathbf{x}^{(K)}\|_\infty \leq \epsilon. \tag{6.11}$$

Per ogni $k \geq 0$, il vettore $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ si chiama residuo k-esimo del sistema (6.8) o anche residuo del sistema (6.8) corrispondente a $\mathbf{x}^{(k)}$. Quindi la function deve arrestarsi al primo vettore $\mathbf{x}^{(K)}$ tale che il corrispondente residuo $\mathbf{r}^{(K)} = \mathbf{b} - A\mathbf{x}^{(K)}$ soddisfa la (6.11). Il criterio di arresto (6.11) si chiama *criterio di arresto del residuo*.

(c) Restituisce nell'ordine i seguenti output: il vettore $\mathbf{x}^{(K)}$ (=l'approssimazione calcolata per la soluzione $\mathbf{x} = A^{-1}\mathbf{b}$), l'indice K (=il numero di iterazioni fatte) e la norma ∞ dell'ultimo residuo ovvero $\|\mathbf{r}^{(K)}\|_\infty = \|\mathbf{b} - A\mathbf{x}^{(K)}\|_\infty$ (che dovrebbe essere $\leq$ della tolleranza ϵ passata come parametro).

Due osservazioni: primo, la (6.10) può essere riscritta come

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + D^{-1}(\mathbf{b} - A\mathbf{x}^{(k-1)}) = \mathbf{x}^{(k-1)} + D^{-1}\mathbf{r}^{(k-1)}$$

(questa osservazione può facilitare il compito di costruire la function richiesta); secondo, la function richiesta non è detto che termini in generale perché, se la successione (6.10) non converge o se ϵ è troppo piccola (inferiore alla precisione di macchina), la condizione di arresto (6.11) potrebbe non essere mai raggiunta; ricordiamo allora il comando Ctrl+C per arrestare la function nel caso in cui essa non termina.

Dopo aver costruito la function, digitare "format long" dal prompt e sperimentare la function nel modo seguente:

- costruire

$$A = \begin{bmatrix} 3 & 1 & 0 \\ 2 & 5 & 1 \\ 1 & 0 & -3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix};$$

A è a dominanza diagonale in senso stretto per righe, per cui un teorema dell'analisi numerica ci assicura che il metodo di Jacobi è convergente, cioè $\rho(I - D^{-1}A) < 1$;
- calcolare un'approssimazione $\widehat{\mathbf{x}}$ della soluzione di $A\mathbf{x} = \mathbf{b}$ con il metodo di Jacobi, scegliendo come tolleranza $\epsilon = 10^{-9}$ e facendosi restituire dalla function anche il numero di iterazioni eseguite e la norma ∞ dell'ultimo residuo;
- calcolare la soluzione $\mathbf{x} = A^{-1}\mathbf{b}$ con il comando \ di Matlab e confrontarne le cifre di precisione con l'approssimazione $\widehat{\mathbf{x}}$ ottenuta con Jacobi;
- ripetere i due punti precedenti scegliendo però prima $\epsilon = 10^{-14}$ e poi $\epsilon =$eps (la precisione di macchina).

ix. Si consideri la seguente equazione differenziale con condizioni di Dirichlet omogenee al contorno:

$$\begin{cases} -u''(x) = f(x), \quad x \in (0,1), \\ u(0) = u(1) = 0, \end{cases} \tag{6.12}$$

dove $f : [0,1] \to \mathbb{R}$ è una funzione continua, $f \in C[0,1]$. In base a teoremi di analisi matematica, esiste un'unica funzione $u : [0,1] \to \mathbb{R}$, continua su $[0,1]$ e derivabile due volte su $(0,1)$, che soddisfa (6.12); tale funzione u risulta poi essere di fatto una funzione di classe $C^2[0,1]$.

Un modo per approssimare numericamente u è il seguente: si sceglie $n \geq 3$, si pone $h = \frac{1}{n+1}$ e $x_j = jh,\ j = 0, \ldots, n+1$, e si cerca un vettore $\mathbf{u} = [u_1, u_2, \ldots, u_n]^T \in \mathbb{R}^n$ tale che

$$\begin{cases} -\dfrac{u_{j+1} - 2u_j + u_{j-1}}{h^2} = f(x_j), \quad j = 1, \ldots, n, \\ \text{(nelle precedenti } n \text{ equazioni abbiamo posto } u_0 = u_{n+1} = 0) \end{cases} \iff A_n\mathbf{u} = h^2\mathbf{f}, \tag{6.13}$$

dove A_n è il laplaciano discreto dato dalla (6.4) e $\mathbf{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$. Il sistema (6.13) ha un'unica soluzione $\mathbf{u} \in \mathbb{R}^n$ perché il laplaciano discreto A_n è una matrice invertibile (infatti è addirittura definita positiva con autovalori (tutti positivi) dati da $\lambda_j(A_n) = 2 - 2\cos\frac{j\pi}{n+1}$, $j = 1, \ldots, n$). Le componenti $u_1, u_2, \ldots, u_n$ del vettore $\mathbf{u}$ vengono prese come approssimazioni dei valori $u(x_1), u(x_2), \ldots, u(x_n)$ della soluzione u. Quanto più la griglia $x_1, x_2, \ldots, x_n$ è fitta (cioè quanto più n è grande), tanto migliori saranno le approssimazioni $u_1, u_2, \ldots, u_n$.

Creare uno script in Matlab che si definisce all'inizio una funzione $f \in C[0,1]$ e un intero $n \geq 3$ e fa le cose seguenti.

(a) Calcola il vettore $\mathbf{u} = [u_1, u_2, \ldots, u_n]^T$ soluzione di (6.13).
(b) Disegna (con asterischi rossi, senza linea continua) il grafico della soluzione numerica, cioè la funzione lineare a tratti definita su $[0,1]$ che vale u_j in x_j per $j = 0, \ldots, n+1$ (con $u_0 = u_{n+1} = 0$).

Sperimentare lo script nel modo seguente: per ogni fissato $n \in \{5, 10, 15, 100, 200\}$, lanciare lo script con i dati n e $f(x) = 1$ e, sulla figura che lo script fa comparire (il grafico della soluzione numerica), disegnare in linea continua blu il grafico della soluzione esatta, che per il dato $f(x) = 1$ è semplicemente la funzione $u(x) = \frac{x(1-x)}{2}$. Cosa si nota? In effetti si può dimostrare che per il dato iniziale $f(x) = 1$, qualunque sia l'indice $n \geq 3$ che si sceglie, la soluzione numerica coincide con la soluzione esatta nei nodi x_j, $j = 0, \ldots, n+1$.

Sperimentare ancora lo script nel modo seguente: per ogni fissato $n \in \{5, 10, 15, 100, 200\}$, lanciare lo script con i dati n e $f(x) = 10\sin(\pi x)$ e, sulla figura che lo script fa comparire (il grafico della soluzione numerica), disegnare in linea continua blu il grafico della soluzione esatta, che per il dato $f(x) = 10\sin(\pi x)$ è la funzione $u(x) = \frac{10}{\pi^2}\sin(\pi x)$.

x. (Generalizzazione dell'esercizio ix.) Si consideri la seguente equazione differenziale con condizioni di Dirichlet al contorno:

$$\begin{cases} -u''(x) + b(x)u'(x) + c(x)u(x) = f(x) \quad x \in (\alpha, \beta), \\ u(\alpha) = A, \qquad u(\beta) = B, \end{cases} \tag{6.14}$$

dove $A, B \in \mathbb{R}$ e $b, c, f : [\alpha, \beta] \to \mathbb{R}$ sono funzioni continue su $[\alpha, \beta]$ con $c \geq 0$ su $[\alpha, \beta]$. In base a teoremi di analisi matematica, esiste un'unica funzione $u : [\alpha, \beta] \to \mathbb{R}$, continua su $[\alpha, \beta]$ e derivabile due volte su (α, β), che soddisfa (6.14); tale funzione u risulta poi essere di fatto una funzione di classe $C^2[\alpha, \beta]$.

Un modo per approssimare numericamente u è il seguente: si sceglie $n \geq 3$, si pone $h = \frac{\beta-\alpha}{n+1}$ e $x_j = \alpha + jh$, $j = 0, \ldots, n+1$, e si cerca un vettore $\mathbf{u} = [u_1, u_2, \ldots, u_n]^T \in \mathbb{R}^n$ tale che

$$\begin{cases} -\dfrac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + b(x_j)\dfrac{u_{j+1} - u_{j-1}}{2h} + c(x_j)u_j = f(x_j), \quad j = 1, \ldots, n \\ \text{(nelle precedenti } n \text{ equazioni abbiamo posto } u_0 = A \text{ e } u_{n+1} = B) \end{cases} \iff A_n(b,c)\mathbf{u} = h^2\mathbf{f}, \tag{6.15}$$

dove

$$A_n(b,c) = \begin{bmatrix} 2+c(x_1)h^2 & -1+b(x_1)\frac{h}{2} & & & \\ -1-b(x_2)\frac{h}{2} & 2+c(x_2)h^2 & -1+b(x_2)\frac{h}{2} & & \\ \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & -1-b(x_{n-1})\frac{h}{2} & 2+c(x_{n-1})h^2 & -1+b(x_{n-1})\frac{h}{2} & \\ & & -1-b(x_n)\frac{h}{2} & 2+c(x_n)h^2 & \end{bmatrix},$$

$$\mathbf{f} = \begin{bmatrix} f(x_1)+\dfrac{A}{h^2}+b(x_1)\dfrac{A}{2h} \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n)+\dfrac{B}{h^2}-b(x_n)\dfrac{B}{2h} \end{bmatrix}.$$

Il sistema (6.15) ha un'unica soluzione $\mathbf{u} \in \mathbb{R}^n$ per ogni $n \geq 3$ tale che $\frac{h}{2}\|b\|_\infty < 1$, dove $\|b\|_\infty = \max_{x\in[\alpha,\beta]} |b(x)|$, perché il tal caso la matrice $A_n(b,c)$ è definita positiva per i teoremi di Gershgorin e quindi è invertibile. Le componenti $u_1, u_2, \ldots, u_n$ del vettore $\mathbf{u}$ vengono prese come approssimazioni dei valori $u(x_1), u(x_2), \ldots, u(x_n)$ della soluzione u.

Creare uno script in Matlab che si definisce all'inizio gli estremi α, β di un intervallo, tre funzioni $b, c, f \in C[\alpha,\beta]$ con $c \geq 0$, le condizioni al bordo $A, B \in \mathbb{R}$ e un intero $n \geq 3$ tale che $\frac{h}{2}\|b\|_\infty < 1$ ($h = \frac{\beta-\alpha}{n+1}$) e fa le cose seguenti.

(a) Calcola il vettore $\mathbf{u} = [u_1, u_2, \ldots, u_n]^T$ soluzione di (6.15).
(b) Disegna (con asterischi e linea continua rossi) il grafico della soluzione numerica, cioè la funzione lineare a tratti su $[\alpha,\beta]$ che vale u_j in x_j per ogni $j = 0, \ldots, n+1$ (con $u_0 = A$ e $u_{n+1} = B$).

xi. Sia $f : \mathbb{R}^2 \to \mathbb{R}$ una funzione continua e localmente lipschitziana [5] e sia $(x_0, y_0) \in \mathbb{R}^2$ un punto fissato. In questa situazione, grazie a teoremi di analisi matematica, per il problema di Cauchy

$$\begin{cases} y'(x) = f(x, y(x)) \\ y(x_0) = y_0 \end{cases} \tag{6.16}$$

esiste un'unica funzione $y(x)$ con le seguenti proprietà:

- $y(x)$ è soluzione di (6.16), cioè $y : I \to \mathbb{R}$ è una funzione definita su un certo intervallo I contenente x_0 che soddisfa $y(x_0) = y_0$ e $y'(x) = f(x, y(x))$ per ogni $x \in I$;
- per ogni altra soluzione $u : J \to \mathbb{R}$ di (6.16) si ha $J \subseteq I$ e $u(x) = y(x)$ per ogni $x \in J$.

La funzione $y(x)$ si chiama *soluzione massimale* del problema di Cauchy (6.16) e l'intervallo I su cui essa è definita si chiama *intervallo massimale*. Si può dimostrare che l'intervallo massimale I è sempre un'intervallo aperto e $y \in C^1(I)$.

[5] $f : \mathbb{R}^2 \to \mathbb{R}$ si dice localmente lipschitziana se per ogni $(x_0, y_0) \in \mathbb{R}^2$ esiste un intorno U di (x_0, y_0) e una costante L tali che $|f(x,y) - f(x,u)| \leq L|y-u|$ per $(x,y), (x,u) \in U$. Questa proprietà è sicuramente soddisfatta se $f \in C^1(\mathbb{R}^2)$.

Fissato $T > x_0$ in modo che $[x_0, T]$ sia contenuto nell'intervallo massimale I, un modo per approssimare la soluzione $y(x)$ sull'intervallo $[x_0, T]$ è il seguente. Si sceglie $n \geq 1$ e si pone $h = \frac{T-x_0}{n}$ e $x_j = x_0 + jh$ per $j = 0, \ldots, n$. Si osserva che per ogni $j = 0, \ldots, n-1$ risulta

$$\begin{aligned} y(x_{j+1}) &= y(x_j) + \int_{x_j}^{x_{j+1}} y'(x)\mathrm{d}x = y(x_j) + \int_{x_j}^{x_{j+1}} f(x, y(x))\mathrm{d}x \\ &\approx y(x_j) + f(x_j, y(x_j))(x_{j+1} - x_j) = y(x_j) + f(x_j, y(x_j))h, \end{aligned}$$

per cui i valori $y(x_1), \ldots, y(x_n)$ della soluzione esatta $y(x)$ soddisfano *approssimativamente*

$$\begin{cases} y(x_{j+1}) = y(x_j) + f(x_j, y(x_j))h, \quad j = 0, \ldots, n-1, \\ y(x_0) = y_0. \end{cases} \tag{6.17}$$

Si approssimano i valori $y(x_1), \ldots, y(x_n)$ con i valori $y_1, \ldots, y_n$ che soddisfano *esattamente* la (6.17):

$$y_{j+1} = y_j + f(x_j, y_j)h, \quad j = 0, \ldots, n-1. \tag{6.18}$$

Partendo da y_0 che è dato, la (6.18) permette di calcolare uno dopo l'altro tutti i valori y_j, $j = 1, \ldots, n$. Il metodo (6.18) si chiama *metodo di Eulero esplicito*.

Creare uno script che si definisce all'inizio una funzione $f : \mathbb{R}^2 \to \mathbb{R}$ di classe $C^1(\mathbb{R}^2)$ (quindi sicuramente continua e localmente lipshitziana), un punto x_0, un dato iniziale y_0, un numero $T > x_0$ e un $n \geq 1$; lo script assume senza fare verifiche che $[x_0, T]$ sia contenuto nell'intervallo massimale e fa le cose seguenti.

(a) Calcola $y_1, \ldots, y_n$ mediante la (6.18).
(b) Disegna il grafico della funzione lineare a tratti $\widetilde{y} : [x_0, T] \to \mathbb{R}$ tale che $\widetilde{y}(x_j) = y_j$ per ogni $j = 0, \ldots, n$.

Sperimentare lo script prendendo

$$f(x, y) = \frac{1 + 10\cos x}{\sqrt[3]{1 + y^2}} \tag{6.19}$$

e x_0, y_0, T, n a scelta. [6]

xii. Costruire una function che prende in input un intero $n \geq 2$ e tre matrici quadrate A, B, C della stessa dimensione (diciamo r), e costruisce la matrice tridiagonale a blocchi data da

$$\begin{bmatrix} B & C & & & \\ A & B & C & & \\ & \ddots & \ddots & \ddots & \\ & & A & B & C \\ & & & A & B \end{bmatrix} \in \mathbb{C}^{rn \times rn}.$$

Si consideri poi la matrice

$$T_n = \begin{bmatrix} B_0 & B_1^T & & & \\ B_1 & B_0 & B_1^T & & \\ & \ddots & \ddots & \ddots & \\ & & B_1 & B_0 & B_1^T \\ & & & B_1 & B_0 \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$$

[6] Per il teorema di esistenza in grande, siccome $f(x, y)$ è a crescita al più lineare in y, il problema di Cauchy (6.16) con la funzione (6.19) e con un qualsiasi $(x_0, y_0) \in \mathbb{R}^2$ ha soluzione massimale $y(x)$ definita su tutto $\mathbb{R}$, per cui l'intervallo massimale è $I = \mathbb{R}$ e dunque $[x_0, T]$ è sempre contenuto nell'intervallo massimale qualunque sia $T > x_0$.

in cui $B_0 = \frac{1}{3}\begin{bmatrix} 16 & -8 \\ -8 & 14 \end{bmatrix}$ e $B_1 = \frac{1}{3}\begin{bmatrix} 0 & -8 \\ 0 & 1 \end{bmatrix}$, e sia K_n la sottomatrice principale di testa di T_n di ordine $2n-1$. Si può dimostrare che per ogni $n \geq 2$ la matrice K_n è simmetrica definita positiva e quindi i suoi autovalori sono tutti reali positivi. Usando la function costruita precedentemente, verificare sperimentalmente che per $n \to \infty$

$$2n^2\lambda_{\min}(K_n) \to \pi^2, \qquad \lambda_{\max}(K_n) \to \frac{32}{3}.$$

xiii. Sia $f : \mathbb{R} \to \mathbb{R}$ di classe $C^1(\mathbb{R})$, sia $u_0 : \mathbb{R} \to \mathbb{R}$ e si consideri il problema di Cauchy (legge di conservazione)

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) + \dfrac{\partial}{\partial x} f(u(x,t)) = 0, \\ u(x,0) = u_0(x). \end{cases} \tag{6.20}$$

Un metodo per risolvere (6.20) su un certo dominio $[a,b] \times [0,T]$ è il metodo di Lax-Friedrichs che ora descriviamo. Si scelgono due interi $n_x, n_t \geq 1$ in modo che, posto

$$h = \frac{b-a}{n_x}, \qquad \Delta t = \frac{T}{n_t}, \qquad \lambda = \frac{\Delta t}{h},$$

sia soddisfatta la condizione CFL

$$\lambda \|f'(u_0)\|_\infty \leq 1,$$

dove $\|f'(u_0)\|_\infty = \sup_{x\in\mathbb{R}} |f'(u_0(x))|$. Si pone

$$\begin{aligned} t^n &= n\Delta t, && n = 0, \ldots, n_t, \\ x_j &= a + \Big(j - \frac{1}{2}\Big)h, && j = 0, \ldots, n_x + 1; \end{aligned}$$

$x_0, \ldots, x_{n_x+1}$ sono definiti così affinché i punti x_0 e x_{n_x+1} (i punti "fantasma") si trovino al centro delle rispettive celle "fantasma" $[a-h, a]$ e $[b, b+h]$, mentre i punti x_j, $j = 1, \ldots, n_x$, si trovino al centro delle rispettive celle $[a + (j-1)h, a + jh]$, $j = 1, \ldots, n_x$. Infine, si calcolano i valori U_j^n per $0 \leq j \leq n_x + 1$ e $0 \leq t \leq n$ secondo lo schema seguente:

$$\begin{cases} U_j^0 = u_0(x_j), \quad j = 0, \ldots, n_x + 1, \\ \left\{ \begin{array}{l} U_j^{n+1} = \frac{1}{2}\left(U_{j+1}^n + U_{j-1}^n\right) - \frac{\lambda}{2}\left(f(U_{j+1}^n) - f(U_{j-1}^n)\right), \quad j = 1, \ldots, n_x, \\ U_0^{n+1} = U_1^{n+1}, \qquad U_{n_x+1}^{n+1} = U_{n_x}^{n+1} \qquad \text{(condizioni al bordo "free flow")} \end{array} \right\} \quad n = 0, \ldots, n_t - 1. \end{cases}$$

U_j^n viene preso come approssimazione di $u(x_j, t^n)$, dove $u(x,t)$ è la soluzione esatta di (6.20), cioè la cosiddetta (unica) soluzione entropica di (6.20).

Creare uno script in Matlab che si definisce all'inizio una funzione $f : \mathbb{R} \to \mathbb{R}$ di classe $C^1(\mathbb{R})$, un dato iniziale $u_0 : \mathbb{R} \to \mathbb{R}$, due numeri a, b con $a < b$, un $T > 0$ e due interi $n_x, n_t \geq 1$ in modo che la condizione CFL sia soddisfatta, e fa le cose seguenti.

(a) Calcola U_j^n per $j = 0, \ldots, n_x + 1$ e $n = 0, \ldots, n_t$;
(b) Traccia il grafico della funzione lineare a tratti $U^{n_t} : \left[a - \frac{h}{2}, b + \frac{h}{2}\right] \to \mathbb{R}$ tale che $U^{n_t}(x_j) = U_j^{n_t}$ per ogni $j = 0, \ldots, n_x + 1$.

Sperimentare lo script scegliendo $f(u) = u^2/2$ (il flusso dell'equazione di Burgers), $u_0(x) = \chi_{(-\infty,1]}(x)$ (una dato iniziale a gradino), $a = -1$, $b = 9$, $T = 8$, $n_x = 200$, $n_t = 200$ (quindi $\lambda = 1$ e la CFL è soddisfatta perché $\|f'(u_0)\|_\infty = \|u_0\|_\infty = 1$).