# Programmare con MATLAB©
# Parte 5
# Cicli: `for` e `while`

Lucia Gastaldi

DICATAM - Sezione di Matematica,
http://lucia-gastaldi.unibs.it

# Indice

# La notazione due punti

La **notazione due punti :** serve per creare vettori, sottomatrici e per il ciclo di tipo for

Vettore=Inizio:Passo:Fine

Dati due numeri interi `j` e `k`, il comando

```
>> j:k
```

produce il vettore contenente i numeri interi da `j` a `k` compresi.
Il vettore è vuoto se `j`>`k`.

Esempio

```
>> x = 1:7
x =
     1   2   3   4   5   6   7
```

Dati tre numeri reali `i`, `j` e `k`, il comando

```
>> j:i:k
```

produce il seguente vettore

`[j,j+i,j+2i, ...,j+mi]` essendo `j+mi` $\leq$ `k` e `j+(m+1)i` $>$ `k`

ossia, partendo dal valore `j`, si somma `i` fino a raggiungere un valore tale che sommando `i` si ottiene un valore **maggiore** a `k`.

Il vettore è vuoto se `i` $= 0$, se `i` $> 0$ e `j` $>$ `k`, oppure se `i` $< 0$ e `j` $<$ `k`.

Nel caso in cui `i` $= 1$ allora i due comandi

```
>> j:i:k        >>j:k
```

danno lo stesso risultato.

# Esempi

```
>> x = 1:2:15
x =
     1     3     5     7     9    11    13    15

>> y=1:9
y =
     1     2     3     4     5     6     7     8     9

>> z=10:-2:2
z =
    10     8     6     4     2

>> w=1:0.33:3
w =

  1.0000  1.3300  1.6600  1.9900  2.3200  2.6500  2.9800
```

# Come estrarre righe e colonne di una matrice

Dato un array `A`:

- `A(:,j)` è la $j$-esima colonna di `A`;
- `A(i,:)` è la $i$-esima riga di `A`;
- `A(:,j:k)` è la sottomatrice di `A` che contiene le colonne di `A` dalla $j$-esima alla $k$-esima;
- `A(:)` fornisce tutti gli elementi di `A`, vista come una singola colonna

# Esempi

```
>> A=hilb(5)
A =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111

>> A(2,:)
ans =
    0.5000    0.3333    0.2500    0.2000    0.1667

>> A(:,3)
ans =
    0.3333
    0.2500
    0.2000
    0.1667
    0.1429
```

```
>> A(1:2,1:3)
ans =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
```

# Esercizi

## Esercizio 1

Costruire una matrice di numeri casuali di dimensione $10 \times 10$ con il comando

```
A=rand(10)
```

Estrarre nella matrice $B$ le colonne pari e nella matrice $C$ la sottomatrice principale di dimensione $5 \times 5$ che si ottiene eliminando le ultime 5 righe e colonne da $A$.

## Esercizio 2

Costruire un vettore b di $N > 10$ componenti in modo che valga:

$$b_i = (-1)^{i+1} \quad \text{ossia} \quad b = (1, -1, 1, -1, \ldots, -1^{N+1}).$$

Modificare il vettore $b$ in modo che le componenti multiple di 3 abbiamo valore 0 cioè $b_{3i} = 0$ per $i = 1, \ldots, N/3$.
Modificare il vettore $b$ in modo che valga $b_{10} = 100$.

# for

Il ciclo **for** esegue un gruppo di istruzioni un numero fissato di volte gestito da una variabile di ciclo *indice* che può assumere sia valori interi che reali.

```
for indice = inizio : incremento : fine
    istruzioni
end
```

Incremento di default: 1. Può essere omesso.
All'inizio del ciclo la variabile *indice* assume il valore *inizio*.
Ogni volta che vengono eseguite tutte le istruzioni all'interno del ciclo, la variabile *indice* viene incrementata.
Se *incremento* $> 0$, allora il ciclo termina quando la variabile *indice* è maggiore di *fine*.
Se *incremento* $< 0$, allora il ciclo termina quando la variabile *indice* è minore di *fine*.
In generale, alla fine del ciclo la variabile *indice* ha un valore strettamente maggiore o minore del valore finale.

# for

```
for v = [v1,v2,...,vn]
      istruzioni
end
```

In questo caso, la variabile di ciclo *v* assume di volta in volta i valori delle componenti del vettore a destra dell'uguale.

## Nota Bene

La variabile *v* è una variabile scalare.

# Esercizio

## Algoritmo di sostituzione in avanti

Implementare in una function l'algoritmo di sostituzione in avanti, per risolvere un sistema lineare con matrice dei coefficienti triangolare inferiore.

Usare la seguente riga di dichiarazione della function:

```
function [x]=sost_avanti(L,b)
```

essendo

**Input** `L` matrice triangolare inferiore;
`b` termine noto;

**Output** `x` vettore soluzione.

# Algoritmo di sostituzione in avanti

- $x_1 = b_1/\ell_{11}$
- per $i = 2, \ldots, n$
  - $x_i = \left( b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j \right) \Big/ \ell_{ii}$

## Esempio

Il sistema $Lx = b$ con

$$L = \begin{bmatrix} 3 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 \\ 2 & 2 & -1 & 0 \\ 4 & 2 & 1 & 2 \end{bmatrix} \qquad b = \begin{bmatrix} 6 \\ -5 \\ -1 \\ 9 \end{bmatrix}$$

ha soluzione $x = (2, -1, 3, 0)^\top$.

# Esercizio

## Algoritmo di sostituzione all'indietro

Implementare in una function l'algoritmo di sostituzione all'indietro, per risolvere un sistema lineare con matrice dei coefficienti triangolare superiore.

Usare la seguente riga di dichiarazione della function:

```
function [x]=sost_indietro(U,b)
```

essendo

**Input** `U` matrice triangolare superiore;
`b` termine noto;

**Output** `x` vettore soluzione.

# Algoritmo di sostituzione all'indietro

- $x_n = b_n / u_{nn}$
- per $i = n-1, \ldots, 1$
  - $x_i = \left( b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \Big/ u_{ii}$

## Esempio

Il sistema $Ux = b$ con

$$U = \begin{bmatrix} 1 & 3 & -5 & 2 \\ 0 & 2 & 4 & -1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 10 \\ -6 \\ 2 \\ 8 \end{bmatrix}$$

ha soluzione $x = (1, 0, -1, 2)^\top$.

# Esercizio

## Fattorizzazione LU di Gauss

Implementare in una function l'algoritmo di fattorizzazione LU.

Usare la seguente riga di dichiarazione della function:

```
function [L,U]=miaLU(A)
```

essendo

**Input** `A` matrice da fattorizzare;
**Output** `L` matrice triangolare inferiore;
`U` matrice triangolare superiore.

# Algoritmo di fattorizzazione di Gauss

- per $k = 1, \ldots, n-1$
  - per $i = k+1, \ldots, n$
    $a_{ik} = a_{ik}/a_{kk}$
    - per $j = k+1, \ldots, n$
      $a_{ij} = a_{ij} - a_{ik}a_{kj}$
      fine ciclo in $j$

    fine ciclo in $i$

  fine ciclo in $k$

## Esempio

La fattorizzazione della matrice

$$A = \begin{bmatrix} 3 & 9 & -15 & 6 \\ -1 & 3 & 17 & -5 \\ 2 & 10 & -4 & 0 \\ 4 & 16 & -10 & 16 \end{bmatrix}$$

è data da

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1/3 & 1 & 0 & 0 \\ 2/3 & 2/3 & 1 & 0 \\ 4/3 & 2/3 & -1 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 3 & 9 & -15 & 6 \\ 0 & 6 & 12 & -3 \\ 0 & 0 & -2 & -2 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

# while

Il ciclo **while** esegue un gruppo di istruzioni fintanto che l'**espressione di controllo** rimane vera.

```
while espressione di controllo
      istruzioni
end
```

L'espressione di controllo è una qualunque espressione logica.

# Esercizio

## Problema 3

Sia $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una funzione reale e continua sull'intervallo aperto $I$. Dato $x_0 \in I$, si consideri la successione definita per ricorrenza da

$$x_{n+1} = f(x_n).$$

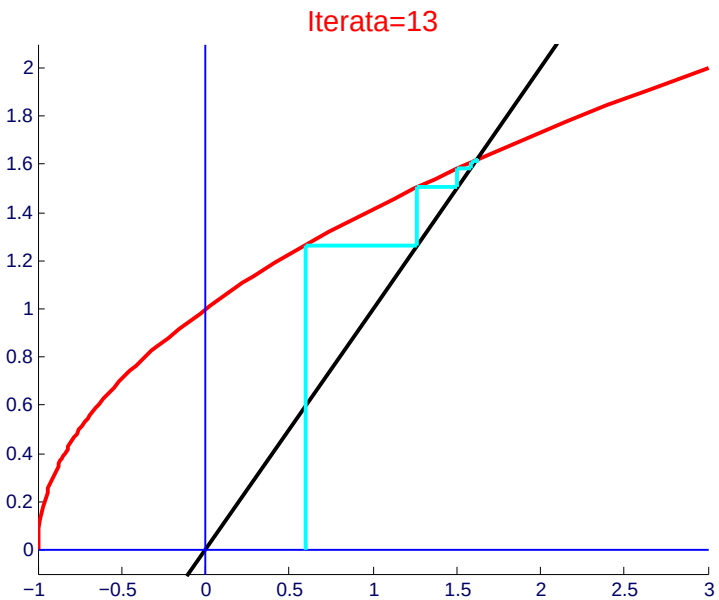
Trovare il limite della successione.

- Fermare il ciclo quando la differenza fra due iterate successive $|x_{n+1} - x_n|$ è inferiore a `toll=1.e-4`.
- Fissare un numero massimo di iterazioni (`nmax=100`).
- $f(x) = \dfrac{x^2 + 2}{2x}$ per $1/2 < x < 2$, $x_0 = 1$.
- $f(x) = x - x^2 + 2$ per $-1 < x < 3$, $x_0 = 0.6$.
- $f(x) = \sqrt{x + 1}$ per $-1 < x < 3$, $x_0 = 0.6$.

# Metodo delle approssimazioni successive

**Osservazione** Se la successione $x_n$ è convergente, cioè $\lim_{n\to\infty} x_n = x$, allora vale la seguente equazione $x = f(x)$ e si dice che $x$ è punto fisso di $f$.

Scrivere un programma di tipo function che calcola la soluzione del problema di punto fisso mediante il **metodo delle approssimazioni successive**;



# Function **ricorrenza**

```
[xf,iter]=ricorrenza(f,a,b,x0,tol,nmax)
```

**Input**

| | |
|---|---|
| `f` | nome della funzione; |
| `a, b` | estremi dell'intervallo; |
| `x0` | punto iniziale; |
| `tol` | tolleranza desiderata; |
| `nmax` | numero massimo di iterazioni. |

**Output**

| | |
|---|---|
| `xf` | valore del limite; |
| `iter` | iterazioni eseguite. |

# Traccia dell'esercizio

```
function [xf,iter]=ricorrenza(f,a,b,x0,tol,nmax)
```

1. uso il comando `fplot` per il grafico della funzione;
2. inizializzo le variabili `delta=1` e `i=0` per il controllo;
3. inizio ciclo while: `while delta>tol&i<nmax`
   - 3.1 salvo il valore vecchio: `x=x0`;
   - 3.2 calcolo il nuovo valore: `x0=f(x0)`;
   - 3.3 calcolo la differenza: `delta=abs(x-x0)`;
   - 3.4 incremento il contatore: `i=i+1`.
4. Assegno i risultati alle variabili di output: `xf=x0` e `iter=i`.

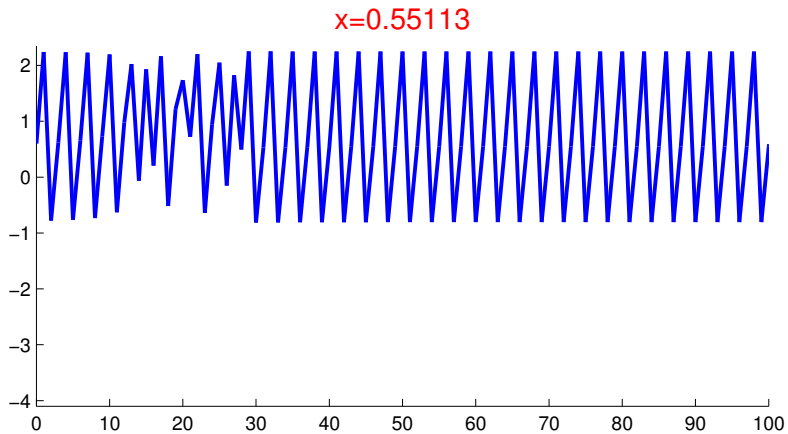
# Soluzione dell'esercizio

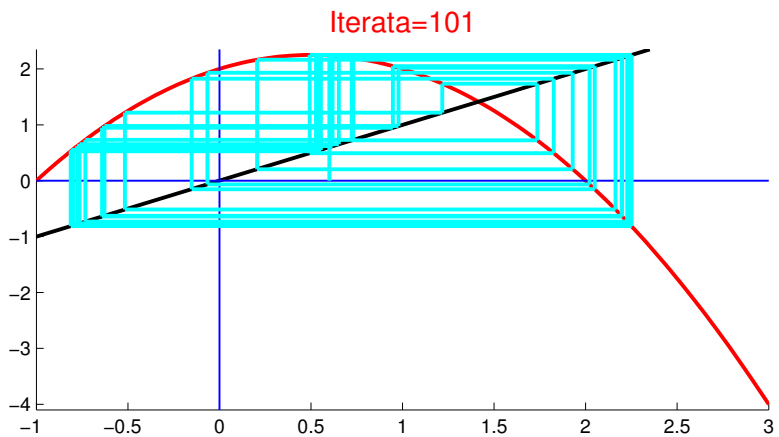
**Prima funzione** Dopo 4 iterazioni si raggiunge la soluzione $x_4 = 1.414215686274510$. Essendo il punto fisso dato da $x = \sqrt{2}$ l'errore relativo è `1.501825092935183e-06`.

**Seconda funzione** La successione è oscillante fra valori compresi nell'intervallo $(-1, 2.5)$. Il comportamento della successione è riportato nella slide successiva.

**Terza funzione** Dopo 9 iterazioni si raggiunge la soluzione $x_9 = 1.617933956351018$. Essendo il punto fisso dato da $x = (1 + \sqrt{5})/2$ l'errore relativo è `6.182342248197602e-05`.

# Risultato per la seconda funzione

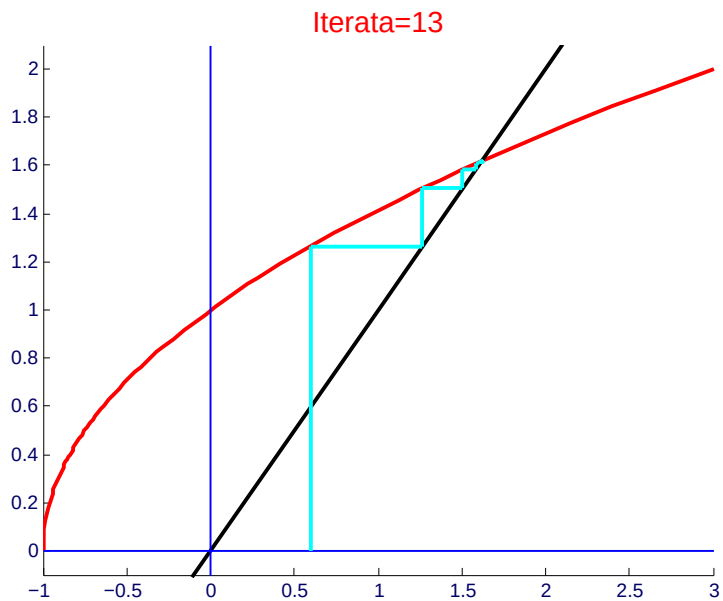


# Rappresentazione della successione

Scrivere un programma di tipo function che esegua le seguenti operazioni:

- riporta il grafico della funzione di iterazione e della bisettrice del primo e terzo quadrante;
- calcola la soluzione del problema di punto fisso mediante il **metodo delle approssimazioni successive**;
- ad ogni iterata riporta sul grafico della funzione l'evoluzione della successione (vedi figura).



# Successione per ricorrenza: caso lineare

Si consideri la seguente successione per ricorrenza di tipo **lineare**:

$$x_{n+1} = ax_n + b.$$

L'equazione di punto fisso corrispondente è $x = ax + b$ quindi la soluzione è $x = b/(1-a)$ per $a \neq 1$.
Usare la function `ricorrenza` per studiare il comportamento della successione con i seguenti dati:

1. $a = 2$, $b = -1$; $x_0 = 1.1$, intervallo $[-0.2, 5]$
2. $a = 1$, $b = 3$; $x_0 = 0.1$, intervallo $[0, 20]$
3. $a = -1$, $b = 5$; $x_0 = 1$, intervallo $[0, 5]$
4. $a = 1/2$, $b = 1$; $x_0 = 4.5$, intervallo $[1, 5]$
5. $a = -1/2$, $b = 5$; $x_0 = 4.5$, intervallo $[-5, 5]$
6. $a = -2$, $b = 1$; $x_0 = 0.5$, intervallo $[-20, 20]$
7. $a = 4$, $b = -1$; $x_0 = 0.4$, intervallo $[0, 10]$

Ripetere la prova negli ultimi due casi scegliendo come dato iniziale $x_0 = 1/3$. Osservare il comportamento della successione in 100 iterazioni se si toglie il controllo su $|x_{n+1} - x_n|$.