

# Introduzione al MATLAB<sup>©</sup>

## Parte 2

## Funzioni

Lucia Gastaldi

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

# Indice

- 1 Funzioni matematiche
  - Assegnazione di funzioni
  
- 2 Grafici
  - Grafico di funzione
  - Grafici in 3D
  - Operazioni punto

# Funzioni matematiche predefinite

```
>> y=cos(pi/4)+sin(pi/4)
```

```
y = 1.4142
```

```
>> y=exp(1)
```

```
y = 2.7183
```

---

Funzione	Significato
sin, cos, tan	seno, coseno, tangente
asin, acos, atan	arcoseno, arcocoseno, arcotangente
exp	esponenziale
sinh, cosh	seno iperbolico, coseno iperbolico
tanh	tangente iperbolica
log, log2, log10	logaritmo in base $e$ , in base 2 e in base 10
sqrt	radice quadrata
abs	valore assoluto
sign	funzione segno

---

## Come assegnare una funzione: @

```
f=@(arglist) espressione
f=@(arglist) [espressione]
```

dichiara una **function** di nome **f**:

la stringa **espressione** contiene l'espressione di **f**  
**arglist** è la lista dei nomi delle variabili da cui dipende **f**.

### Esempio

```
>> f=@(x) x^2*atan(x)    produce
```

```
f=
    @(x) x^2*atan(x)
```

```
>> g=@(x,y) sqrt(x^2+y^2)    produce
g=
    @(x,y) sqrt(x^2+y^2)    Da cui g(3,4) = 5
```

```
>> p=@(x,y) [x^3 y^2]    produce
p=
    @(x,y) [x^3 y^2]    Da cui p(3,4) = 27 16
```

## Come valutare una funzione

Per valutare  $f$  nel punto  $x$ :

```
>> x=1.718;  
>> y=f(x);
```

Per valutare  $g$  nel punto  $(a, b)$ :

```
>> a=1.71; b=2.23;  
>> z=g(a,b);
```

Il nome della variabile in cui si calcola il valore di una funzione assegnata come  $@$  non deve essere necessariamente uguale al nome delle variabili usate nella assegnazione della funzione

Si può valutare la funzione anche in un vettore di punti:

```
>> x=[0 1 3];  
>> f=@(x) 3*sin(x);  
>> f(x)  
ans=
```

```
0          2.5244      2.7279
```

Se  $x$  è un array  $f(x)$  è un array della stessa dimensione di  $x$ .

# Esercizio

## Problema 1:

valutare  $f(x) = x^2 \cos(x)$  sull'intervallo  $I = [-1, 2]$  e rappresentarla graficamente.

Due possibili modi:

- usando il comando **fplot**
- usando il comando **plot**

## fplot

Per fare il grafico di una funzione  $f$  su un intervallo  $[a, b]$  si può usare il comando `fplot` con la seguente sintassi

```
fplot(f, [a,b])
```

oppure

```
fplot('stringa', [a,b])
```

dove `stringa` contiene l'espressione della funzione.

Se la funzione è assegnata con un `M-file` di tipo `function` la sintassi è:

```
fplot(@f, [a,b])
```

### Due funzioni sullo stesso grafico

```
f=@(x) x^2;
```

```
g=@(x) 2*x*sin(4*x)
```

```
fplot(f, [-1 2])
```

```
hold on
```

```
fplot(g, [-1 2], 'r--')
```

## Esempio

Per  $n = 1, \dots, 6$  fare il grafico della funzione  $f(x) = x^n$  nell'intervallo  $[-2, 2]$ .

**Svolgimento** Possiamo definire la funzione dipendente dalla variabile  $x$  e dal parametro  $n$  come segue:

$$f=@(x,n) x^n$$

e poi usare il seguente comando per fare il grafico, ad esempio per  $n = 2$ :

$$\text{fplot}(@(\text{x}) f(\text{x},2), [-2 \ 2])$$

Per generare i grafici delle 4 funzioni e vederli in 4 grafici separati si può usare la seguente procedura:

```
for n=1:6
    subplot(2,3,n)
    fplot(@(x) f(x,n), [-2 2])
    title(['n= ', num2str(n)])
end
```

## plot

Dati due vettori delle stesse dimensioni  $x$  e  $y$ , il comando **plot** genera una spezzata che congiunge a due a due i punti di coordinate  $(x_i, y_i)$ .

```
plot(x,y, 'color_linestyle_marker')
```

```
>> plot(x,y, 'm-*')
```

*color*: c,m,y,r,b,g,w,k

*linestyle*: -,--, :, -., none

*marker*: +,o,\*,.,x,square

Per disegnare 2 o piú coppie di vettori sullo stesso grafico:

```
plot(x1,y1, 'b:', x2,y2, 'r-'); oppure
```

```
plot(x1,y1,x2,y2)
```

```
legend('primo grafico', 'sec. grafico')
```

## Grafico di una funzione con il comando `plot`

- Definire una griglia sull'intervallo  $I = [-1, 2]$ , ovvero scegliere un insieme discreto di punti rappresentativo per  $I$ :

```
x=linspace(-1,2,50);
```

► linspace

*Crea un vettore riga di 50 elementi, contenente i valori di 50 punti equispaziati in  $I$*

- Definire la funzione e valutarla:

*$x$  è un vettore, si vuole calcolare  $y_i = x_i^2 \cos(x_i)$  per ogni  $i$ , quindi si devono usare le operazioni " . "*

► " . "

```
>> f=@(x) x.^2.*cos(x); y=f(x);
```

- Rappresentare i punti  $(x_i, y_i)$  su di un piano cartesiano:

```
>> plot(x,y)
```

## Esempio di grafico di una superficie

Sia data la funzione

$$f(x, y) = \sin(\sqrt{x^2 + y^2}) \quad (x, y) \in [-3\pi/2, 3\pi/2] \times [-3\pi/2, 3\pi/2]$$

Per costruire il grafico serve la griglia dei valori  $(x, y)$  che si ottiene dai vettori  $x$  e  $y$ , mediante la function **meshgrid**.

```
>> m=20;n=25;  
>> x=linspace(-pi,pi,n);  
>> y=linspace(-pi,pi,m);  
>> [X,Y]=meshgrid(x,y);
```

In questo modo si ottengono due matrici che contengono rispettivamente le ascisse e le ordinate dei punti della griglia. Per disegnare la funzione:

```
>> Z=sin(X.*Y);  
>> mesh(X,Y,Z)
```

# Grafici di superfici

<b>Funzione</b>	<b>Significato</b>
<code>view</code>	cambia l'orientamento del grafico.
<code>colormap</code>	cambia il colore al grafico.
<code>shading</code>	cambia l'ombreggiatura al grafico.
<code>mesh</code>	disegna un grafico a griglia.
<code>surf</code>	disegna un grafico di superficie.
<code>surf1</code>	disegna un grafico di superficie con ombreggiatura.
<code>contour</code>	disegna un grafico a curve di livello.
<code>contourf</code>	disegna un grafico a curve di livello riempite.
<code>pcolor</code>	disegna una scacchiera colorando le caselle.
<code>meshgrid</code>	genera i punti di una griglia.

## contour

Il comando `contour` permette di disegnare le linee di livello di una superficie.

I vettori `X,Y,Z` contengono i punti della mesh e i valori della funzione rispettivamente.

```
>> contour(X,Y,Z)           disegna 9 linee di livello
>> contour(X,Y,Z,N)        disegna N linee di livello
>> contour(X,Y,Z,[v v])    disegna la linea di livello v
>> contour(X,Y,Z,...       disegna le linee di livello
    [v1 v2 v3 v4 ... vn])  v1 v2 v3 v4 ... vn
```

Il comando `contour3` disegna le linee di livello in un grafico tridimensionale.

## linspace

Se il passo non è intero, può essere preferibile il comando **linspace** per creare un vettore di punti equispaziati in un intervallo:

**linspace** (*Inizio, Fine, Numero di Punti*)

```
>> a=0; b=1; n=8;
>> x=linspace(a,b,n)
x =
Columns 1 through 7
0 0.1429 0.2857 0.4286 0.5714 0.7143 0.8571
Column 8
1.0000
```

Il vettore ha componenti:

$$x(i) = a + (i - 1) \frac{b - a}{n - 1} \quad \text{per } i = 1, \dots, n.$$

## Operazioni “punto”

Le operazioni **punto** agiscono su array che abbiano le stesse dimensioni:

- . \* prodotto elemento per elemento
- . / divisione elemento per elemento
- . ^ potenza elemento per elemento

```
>> a1b=a1.*b
```

```
a1b =
```

```
1  
4  
15  
16
```

$$(a1b)_i = (a1)_i * b_i$$

$$\text{con } a1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \text{ e } b = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 4 \end{bmatrix}$$

◀ Return