

# INTRODUZIONE A MATLAB

## Descrizione

- ➔ Ambiente/Linguaggio per risolvere problemi di calcolo numerico *MATrix LABORatory*;
- ➔ Marchio registrato da *MathWorks Inc.* (U.S.A.)
- ➔ Può essere ampliato da pacchetti specifici (toolbox)
  - ➔ *Wavelet Toolbox, Signal processing Toolbox*
- ➔ È un interprete in grado di eseguire:
  - ➔ Istruzioni native (built-in)
  - ➔ Istruzioni contenute in files

## IL PROMPT

La linea di comando di MATLAB è indicata dal prompt

>>

MATLAB permette di richiamare dal prompt le ultime righe di comandi inseriti usando le frecce in alto e in basso

Accetta dichiarazioni di variabili, espressioni e chiamate a tutte le funzioni disponibili nel programma.

Tutte le funzioni di MATLAB non sono altro che files di testo, simili a quelli che l'utente può generare con un text editor, e vengono eseguite semplicemente digitandone il nome sulla linea di comando.

---

## HELP DI MATLAB

MATLAB presenta un *help* in linea con informazioni sulla sintassi di tutte le funzioni disponibili.

Per accedere a queste informazioni, basta digitare:

```
help nome_funzione
```

È anche possibile avere un *help* di tutte le funzioni di una certa categoria; ad esempio per sapere quali sono le funzioni specifiche per l'analisi dei segnali, basta digitare:

```
help signal
```

Per sapere quali sono le varie categorie di funzioni disponibili (i *toolbox*), basta digitare:

```
help
```

---

# I FILES DI MATLAB

I files interpretati dal programma sono file di testo ASCII con estensione **.m** ; sono generati con un text editor e sono eseguiti in MATLAB semplicemente digitandone il nome sulla linea di comando (senza estensione!).

È possibile inserire dei commenti al loro interno precedendo ogni linea di commento col simbolo percento **%**

---

## PUNTEGGIATURA E VARIABILI

Le istruzioni (siano esse contenute in un file `.m` lanciato da MATLAB, oppure digitate direttamente dalla linea di comando) vanno terminate con un punto e virgola, in caso contrario viene visualizzato il risultato dell'applicazione dell'istruzione.

```
>> var2=linspace(-10,10,10000);
```

```
>> var1=6
```

```
var1
```

```
6
```

---

# ISTRUZIONI ELEMENTARI: WHO e WHOS

Si supponga di aver definito in memoria una matrice A di dimensione  $2 \times 3$  e una variabile ans.

**who: elenco delle variabili definite in memoria**

```
>> who
your variables are:
A      ans
```

**whos: Informazioni sulle variabili definite in memoria**

```
>> whos
Name Size Bytes Class
A      2x3    48    double array
ans    1x1     8     double array
Grand total is 7 elements using 56 bytes
```

---

## ISTRUZIONI ELEMENTARI: save, clear, load

**save:** salva tutte le variabili in memoria nel file file.m

```
>> save A
```

**clear:** cancella tutte le variabili in memoria o una in particolare se specificata

```
>> clear A
```

```
>> clear
```

**load:** richiama in memoria le variabili salvate nel file specificato

```
>> load file
```

---

# OPERATORI SCALARI

Gli operatori disponibili sono:

- $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$ ,
  - $\sin$ ,  $\cos$ ,  $\tan$ ,
  - $\text{asin}$ ,  $\text{acos}$ ,  $\text{atan}$ ,
  - $\text{exp}$ ,  $\log$  (naturale),  $\log_{10}$  (in base 10),
  - $\text{abs}$ ,  $\text{sqrt}$ ,  $\text{sign}$
-

# ELEMENTI DI BASE DI MATLAB: vettori e matrici

L'inserimento di un vettore o di una matrice in generale viene effettuato tra parentesi quadre, separando gli elementi delle righe con spazi o virgole, e le diverse righe con punti e virgola (oppure andando a capo ad ogni nuova riga).

```
>> x = [1, 2, 3]; % vettore riga
```

```
>> y = [1; 4; 7]; % vettore colonna
```

```
>> A = [1 2 3; 4 5 6; 7 8 9]; % matrice
```

```
>> A = [1 2 3  
        4 5 6  
        7 8 9 ];
```

---

# ELEMENTI DI BASE DI MATLAB: vettori e matrici

Riferimento agli elementi di una matrice A:

- l'elemento  $a_{mn}$  è indirizzato come  $A(m, n)$ ;

```
>> A(2,3)
6
```

- la riga m-esima è indirizzata come  $A(m, :)$ , dove tutte le colonne sono indicate con due punti;

```
>> A(2,:)
4 5 6
```

- la colonna n-esima è indirizzata come  $A(:, n)$ , dove tutte le righe sono indicate con due punti;

```
>> A(:,3)
3
6
9
```

---

# OPERAZIONI SULLE MATRICI

Date due matrici A e B di dimensione opportune, si possono definire le seguenti operazioni:

```
>> S=A+B; % somma di due matrici
```

```
>> P=A*B; % prodotto righe per colonne  
      % di due matrici
```

```
>> At=A'; % trasposta di una matrice
```

```
>> Ai=inv(A); % inversa di una matrice
```

---

# OPERAZIONI SULLE MATRICI

Altre funzioni operanti su *matrici* (e, quindi, su *vettori*, riga o colonna) sono:

max, min,

sort,

sum, prod,

Esistono poi particolari operatori ( $\cdot^*$ ,  $\cdot/$ ,  $\cdot^{\wedge}$ ) che permettono di effettuare operazioni su vettori *elemento per elemento*, senza ricorrere a cicli. Ad esempio, se  $A$  e  $B$  sono due matrici, per sommare elemento per elemento le due matrici basta fare:

```
>> C=A.*B;
```

---

# OPERAZIONI SULLE MATRICI

- Altre funzioni che operano invece essenzialmente su matrici sono:

`det` ← Determinante della matrice

```
>> det(A)
0
```

`size` ← Dimensioni della matrice

```
>> size(A)
3 3
```

`rank` ← Rango della matrice

```
>> rank(A)
2
```

---

# OPERAZIONI SULLE MATRICI:

`eye`, `zeros`, `ones`, `rand`, `diag`

Esistono poi varie funzioni predefinite per la creazione di matrici:

`eye (n)` : matrice identità n righe n colonne

`zeros (m, n)` : matrice di 0 con m righe e n colonne

`ones (m, n)` : matrice di 1 con m righe e n colonne

`rand (m, n)` : matrice casuale di valori tra 0 e 1

`diag (X)` : se  $X$  è un vettore con n elementi, produce una matrice quadrata diagonale di dimensione n per n con gli elementi di  $X$  sulla diagonale. Se invece  $X$  è una matrice quadrata di dimensione n per n, produce un vettore di n elementi pari a quelli sulla diagonale di  $X$ .

---

# RISOLUZIONE DI SISTEMI LINEARI

Calcolare il valore di  $x$ , con  $Ax=B$    $x=A^{-1}B$

```
>> x=A\B;
```

```
>> x=inv(A)*B;
```

Calcolare il valore di  $x$ , con  $xC=D$    $x=DC^{-1}$

```
>> x=D/C;
```

```
>> x=D*inv(C);
```

➤ **slash** / determina la divisione con la matrice posta a destra di  $x$

➤ **backslash** \ determina la divisione con la matrice posta a sinistra di  $x$

---

# GRAFICI IN MATLAB

## La grafica è una delle caratteristiche più sviluppate di MATLAB

- Permette di tracciare più grafici sulla stessa finestra o su più finestre dette "figure"
  - Per default MATLAB traccia grafici sulla finestra 1
  - Volendo aprire più finestre grafiche occorre digitare il comando `figure(n)` dove `n` definisce il numero della finestra
  - Da questo punto in poi MATLAB tratterà grafici sulla finestra `n`-esima fino a quando non si cambierà finestra con un nuovo comando `figure`
  - La chiusura della finestra `n`-esima avviene con il comando `close(n)`
-

# Definizione di Intervalli

Per definire intervalli si utilizza l'operatore colon (:)

Ad esempio, il vettore le cui componenti sono i valori compresi tra 0 e 2 con passo 0.1 è definito come:

```
» a=[0:0.1:2]
```

```
a =
```

```
Columns 1 through 7
```

```
0 0.1000 0.2000 0.3000 0.4000 0.5000  
0.6000
```

```
Columns 8 through 14
```

```
0.7000 0.8000 0.9000 1.0000 1.1000  
1.2000 1.3000
```

```
Columns 15 through 21
```

```
1.4000 1.5000 1.6000 1.7000 1.8000  
1.9000 2.0000
```

---

## LA FUNZIONE *axis*

Per *creare degli assi* cartesiani si usa la funzione *axis*

```
>>axis([x_min,x_max,y_min,y_max]) ↑  
>>axis('string') ↑
```

- Nella prima forma si impongono i limiti inferiore e superiore degli assi cartesiani
- Nella seconda forma, 'string' indica l'aspetto degli assi;

*ad esempio:*

```
String=square per avere i due assi uguale  
String=normal per sfruttare tutto lo schermo  
String=auto   restituisce l'asse in scala default,  
              in maniera automatica
```

---

# LA FUNZIONE *plot*

Se  $x$  è un vettore contenente le ascisse dei punti di un fissato insieme di coppie del piano e  $y$  è il vettore delle corrispondenti ordinate, `plot(x,y)` disegna la spezzata congiungente tali punti

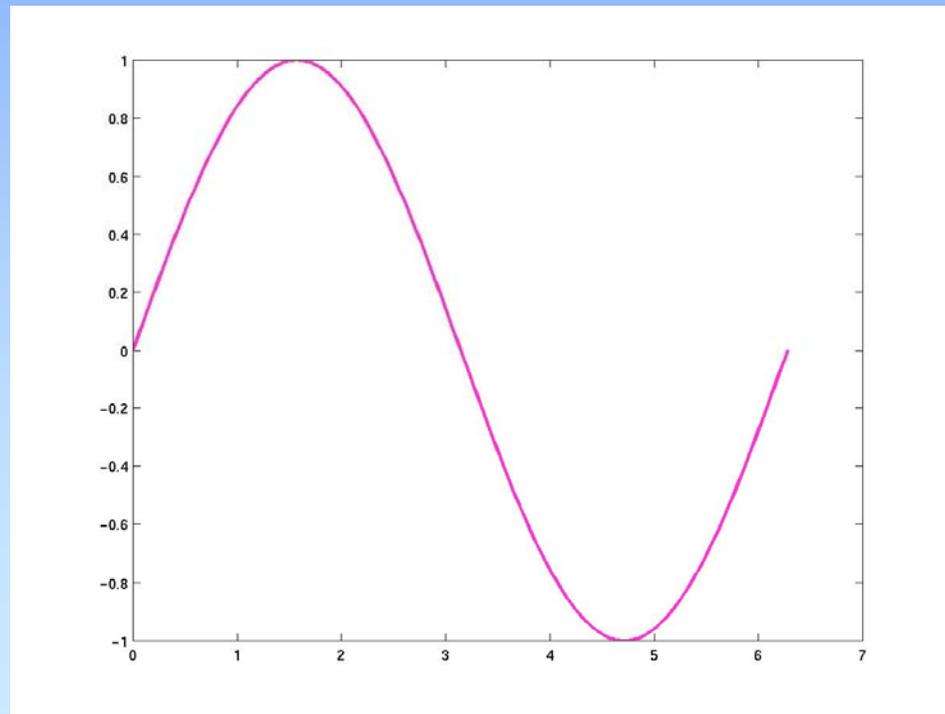
## *Esempio 1:*

Disegnare la funzione  $\sin(x)$  da zero a  $2\pi$

```
» x=[0:pi/100:2*pi];
```

```
» y = sin(x);
```

```
» plot(x,y) ↵
```

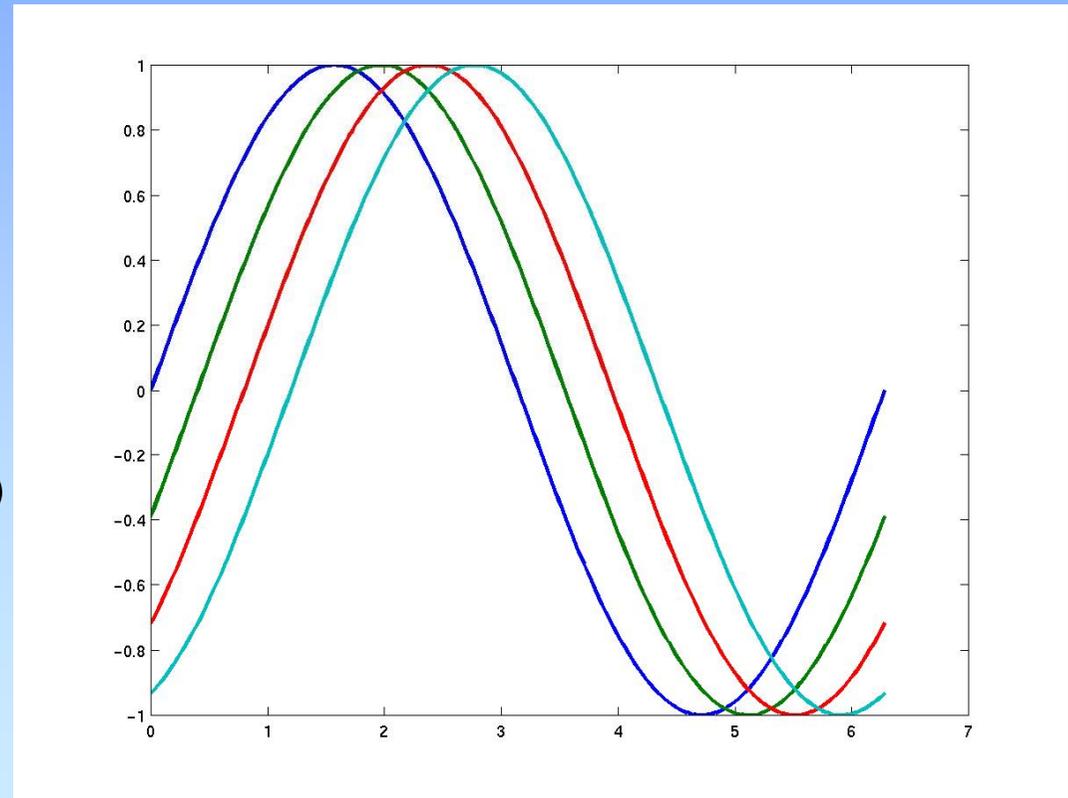


# LA FUNZIONE *plot*

E' possibile riportare sulla stessa figura più funzioni rappresentate da diverse coppie (x,y)

## Esempio 2:

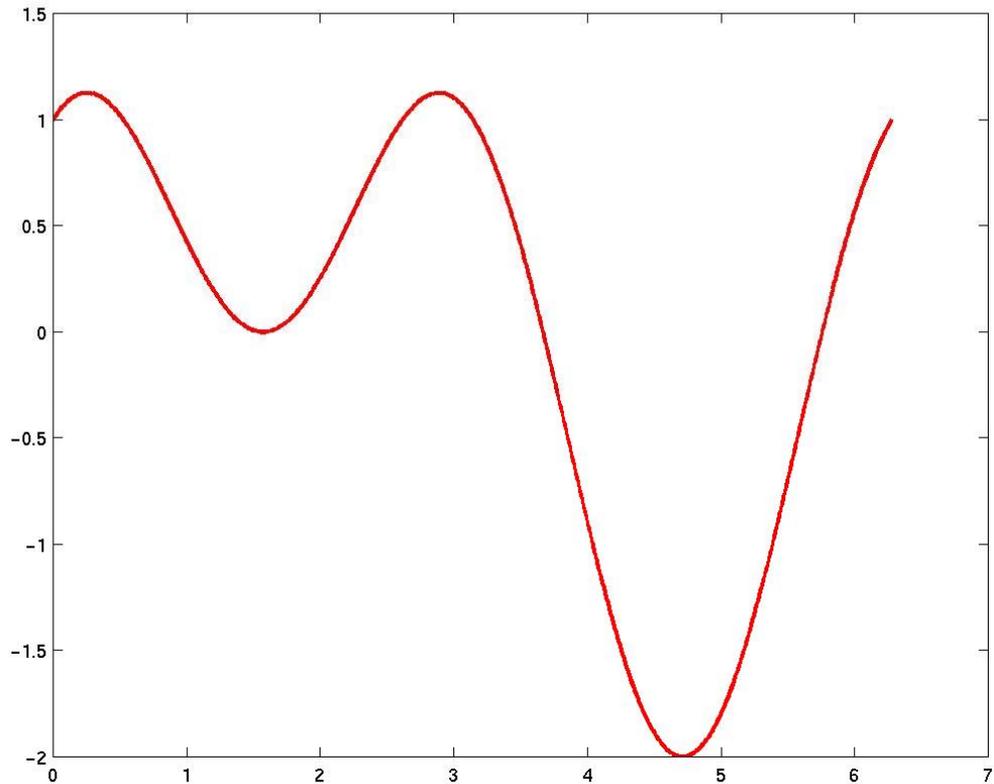
```
» x = [0:pi/100:2*pi];  
» y = sin(x);  
» y2 = sin(x - .40);  
» y3 = sin(x - .8);  
» y4 = sin(x - 1.2);  
» plot(x,y,x,y2,x,y3,x,y4)
```



# LA FUNZIONE *plot*

## *Esempio 3:*

```
» x=[0:0.01:2*pi];  
» y=sin(x)+cos(2*x);  
» plot(x,y) ↑
```

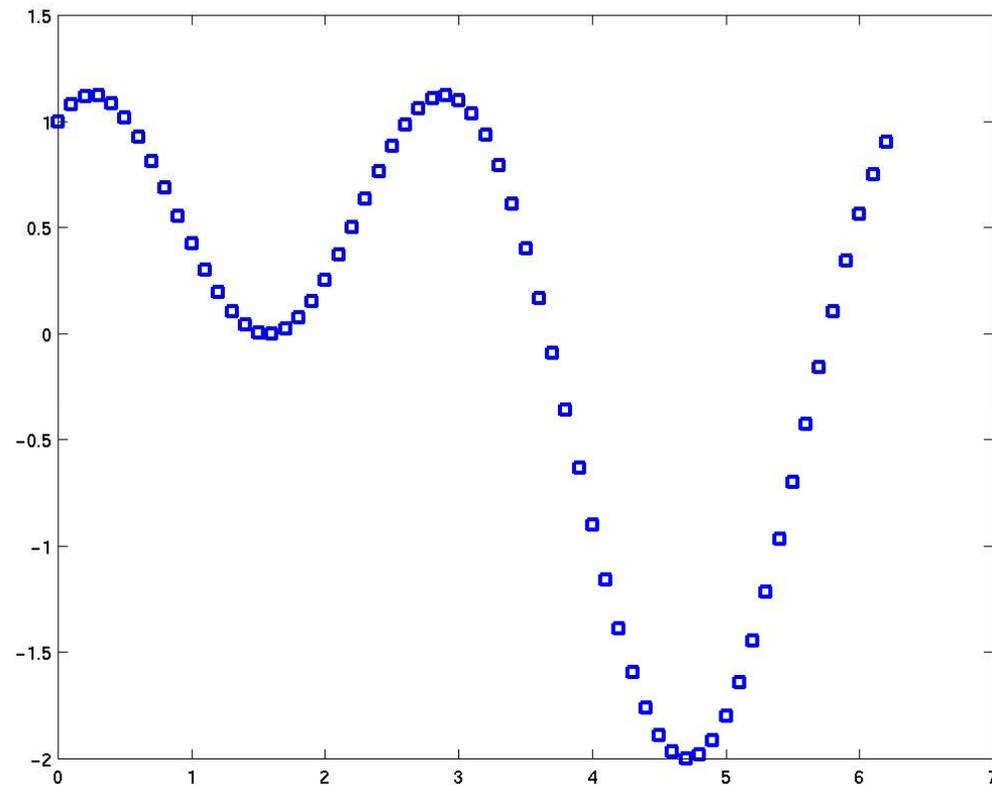


# LA FUNZIONE *plot*

La funzione *plot* è in grado di tracciare le curve impiegando diversi stili:

## *Esempio 3:*

```
» x=[0:0.1:2*pi];  
» y=sin(x)+cos(2*x);  
» plot(x,y,'s')
```



La sintassi di *plot* nel caso in cui si vogliono utilizzare simboli, colori o stili diversi di linee è la seguente:

```
plot(x,y,'stile') ↴
```

Linea continua	—	Punto	.	Colore rosso	<b>r</b>
Linea tratteggiata	--	Più	+	Colore verde	<b>g</b>
Linea punteggiata	:	Cerchio	<b>o</b>	Colore blu	<b>b</b>
Linea tratto punto	-. .	Stella	*	Colore bianco	<b>w</b>
		Croce	<b>x</b>	Colore invisibile	<b>i</b>
		Quadrato	<b>s</b>		

#### **Esempio 4:**

```
>> plot(x,y,'b+:') ↴
```

traccia una linea blu e punteggiata, con il simbolo + in corrispondenza di ogni valore

---

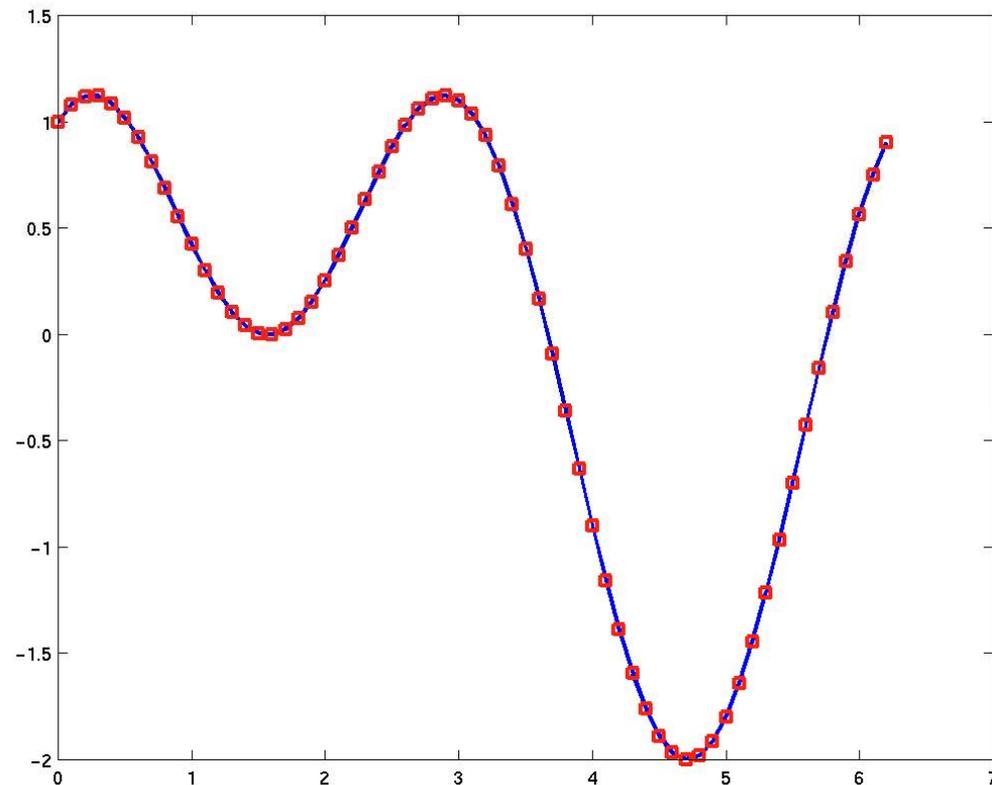
# LA FUNZIONE *hold*

La funzione *hold* conserva il riferimento di assi cartesiani e il corrispondente grafico già esistente; il grafico successivo verrà sovrapposto al precedente.

Si usa nella forma  
*hold on*  
*hold off*

## Esempio 4:

```
» x=[0:0.1:2*pi];  
» y=sin(x)+cos(2*x);  
» plot(x,y);  
» hold on  
» plot(x,y,'s') ↑
```

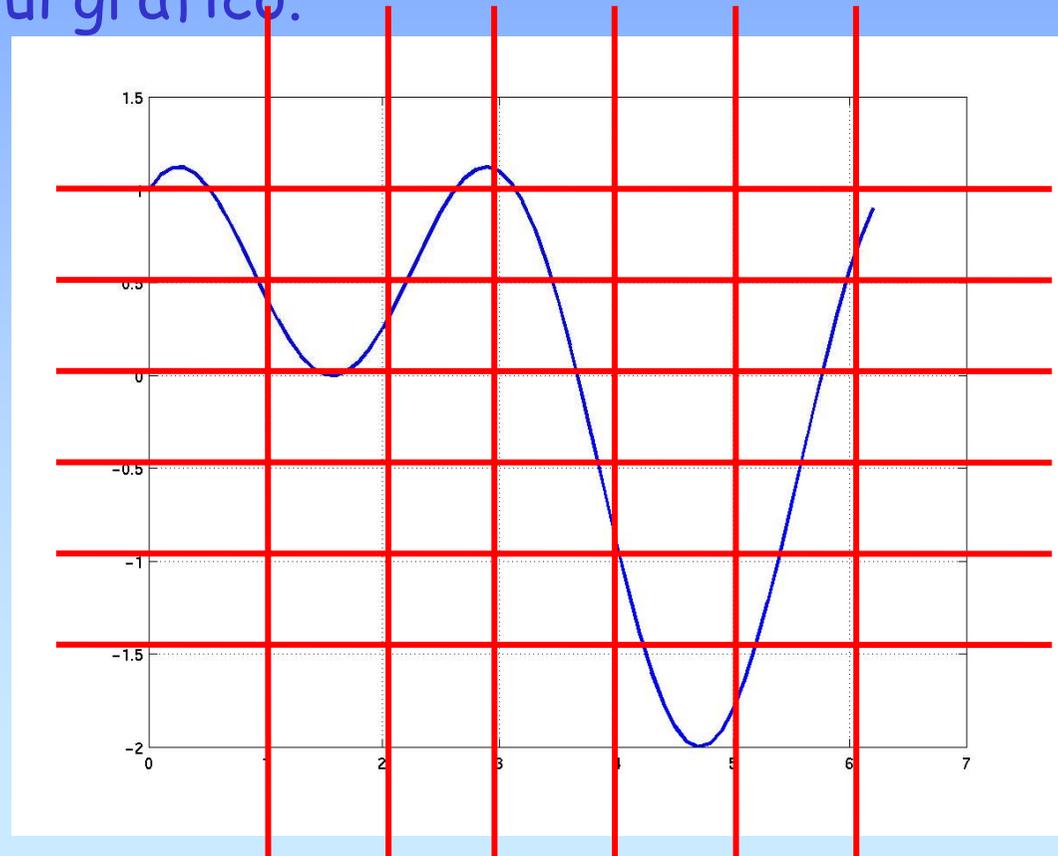


## Le FUNZIONI: *zoom*, *grid*

- La funzione *zoom* permette l'ingrandimento di regioni del grafico.
- L'attivazione della funzione *grid* traccia un reticolato sul grafico.

Viene usata nella forma  
**grid on**  
**grid off**

» `grid on`

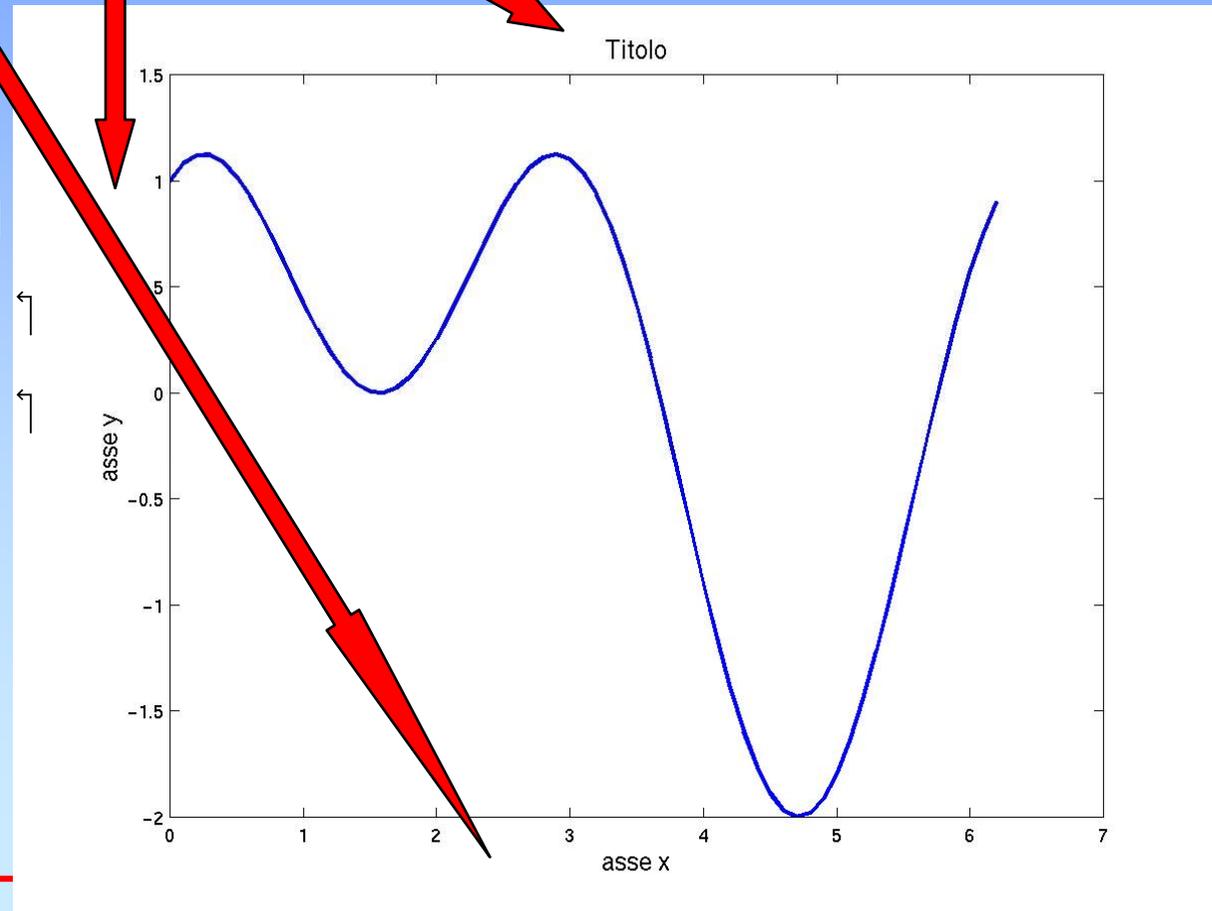


## Le FUNZIONI: *zoom*, *grid*

Il comando *clf* pulisce la finestra corrente, mentre *figure* ne apre una nuova.

Le istruzioni *xlabel*, *ylabel* e *title* etichettano gli assi e la figura:

- » `title('Titolo')` ↴
- » `xlabel('asse x')` ↴
- » `ylabel('asse y')` ↴



# LA FUNZIONE *text*

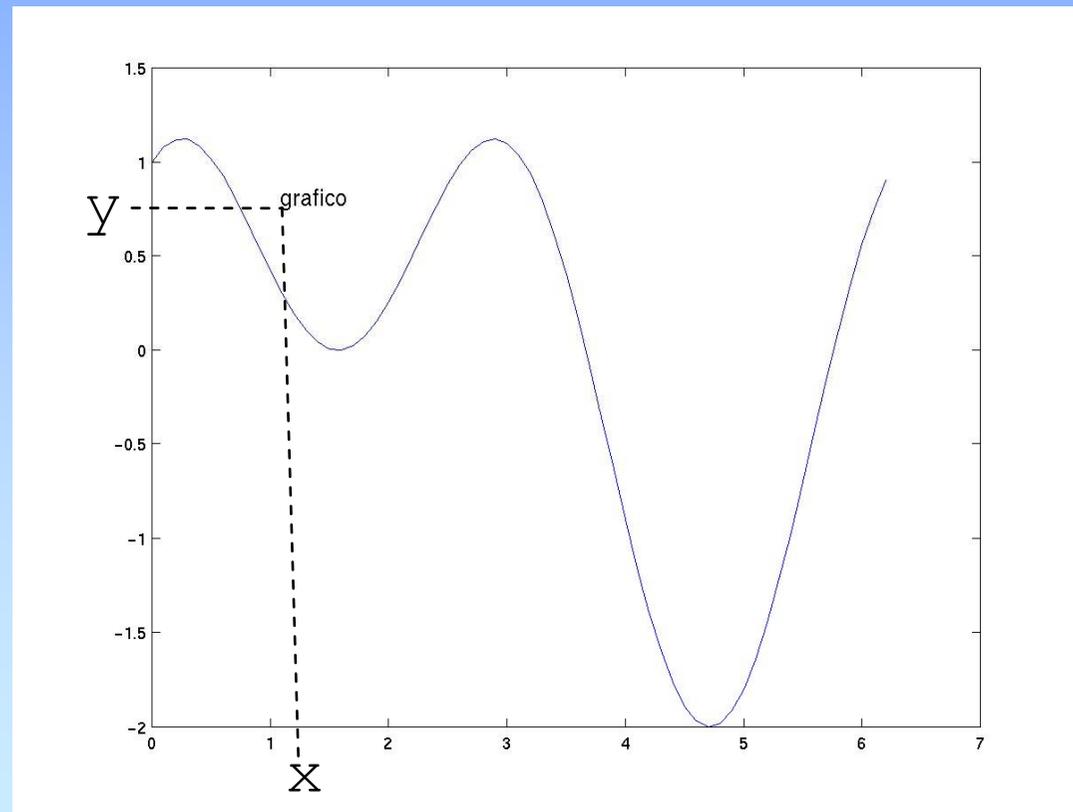
La funzione *text* permette di scrivere una didascalia sul grafico.

```
text(x,y,'testo')
```

dove  $x,y$  sono le coordinate da cui inizierà il testo, sapendo che  $(0,0)$  è l'angolo in basso a sinistra e  $(1,1)$  quello in alto a destra

**Esempio:**

```
» text(x,y,'grafico');
```



# GRAFICI TRIDIMENSIONALI

Grafici tridimensionali sono tipicamente tracciati per mezzo delle funzioni *plot3*, *mesh* e *surf*.

*plot3* consente di tracciare una curva nello spazio a partire dalle sue equazioni parametriche.

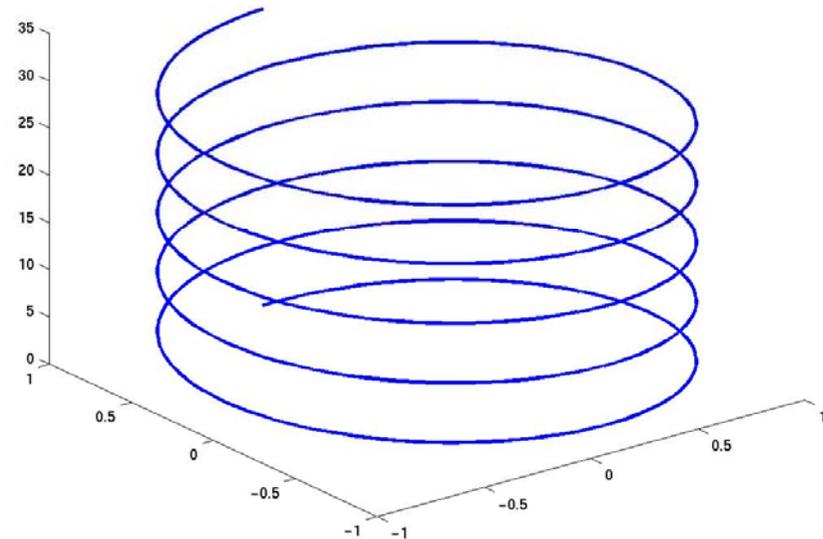
## Esempio:

Dalle equazioni parametriche dell'elicoide

$$\begin{cases} x = \sin t \\ y = \cos t \\ z = t \end{cases}$$

```
» t=[0:pi/50:10*pi];
```

```
» plot3(sin(t),cos(t),t);
```



# GRAFICI TRIDIMENSIONALI

Grafici tridimensionali sono tipicamente tracciati per mezzo delle funzioni *plot3*, *mesh* e *surf*.

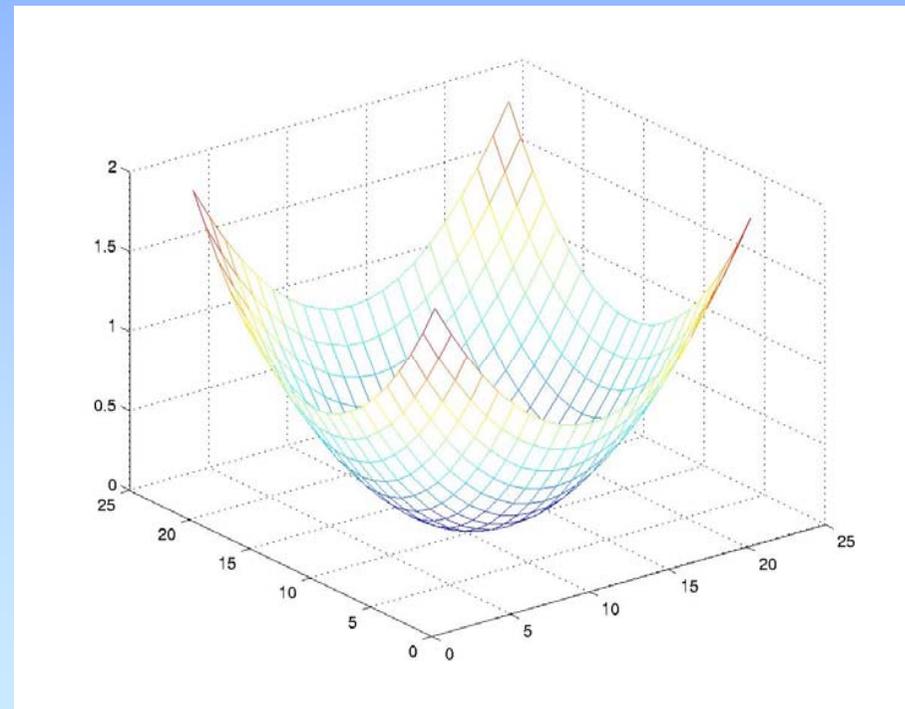
*mesh* consente rappresentare una superficie nello spazio a partire dalle sue equazioni parametriche.

## Esempio :

Dall'equazione del paraboloide ellittico

$$z = x^2 + y^2$$

- » `x = [-1:0.1:1];`
- » `y = [-1:0.1:1];`
- » `[X,Y] = meshgrid(x,y);`
- » `mesh(Z) ↑`



# GRAFICI TRIDIMENSIONALI

Grafici tridimensionali sono tipicamente tracciati per mezzo delle funzioni *plot3*, *mesh* e *surf*.

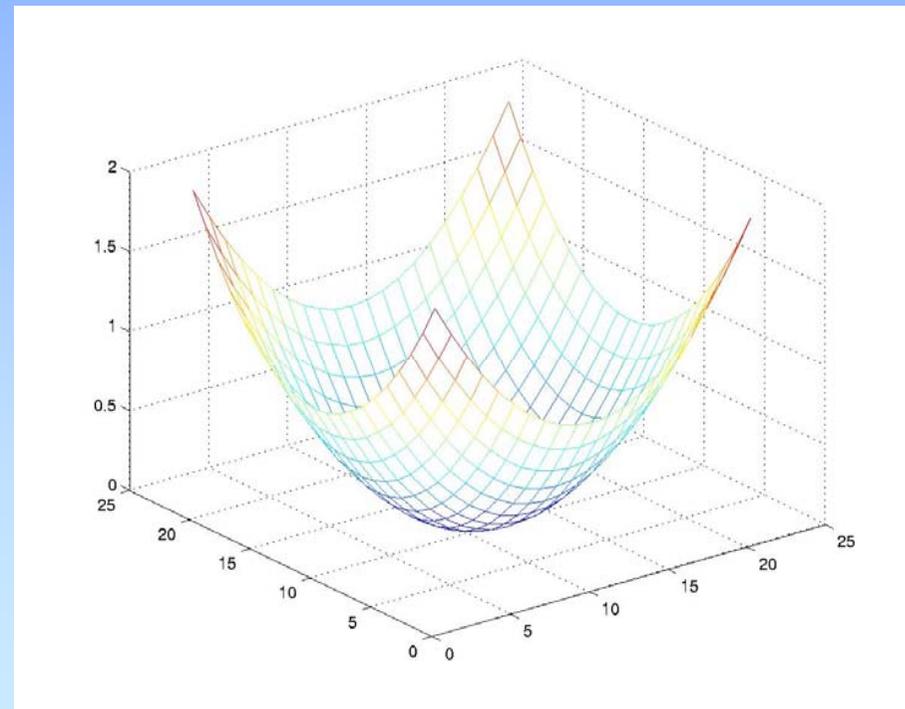
*surf* consente rappresentare una superficie nello spazio a partire dalle sue equazioni parametriche.

## Esempio :

Dall'equazione del paraboloide ellittico

$$z = x^2 + y^2$$

- » `x = [-1:0.1:1];`
- » `y = [-1:0.1:1];`
- » `[X,Y] = meshgrid(x,y);`
- » `surf(Z) ^`



## TRACCIAMENTO DI PIU' GRAFICI NELLA STESSA FINESTRA

Il comando *subplot(m,n,p)* divide la finestra corrente in m righe, n colonne e seleziona la finestra p-esima,

ad esempio

```
subplot(2,2,3)
```

spezza la finestra in quattro sottofinestre e seleziona quella in basso a sinistra (riga 2, colonna 1)

il comando *subplot(m,n,p)* deve essere, dunque, usato prima del *plot*, per fornire informazioni sulla sottofinestra in cui visualizzare il grafico desiderato

---

**ALCUNI ESERCIZI  
con MATLAB**

# OPERAZIONI MATRICIALI

*Scrivere un file .m contenente le istruzioni relative alla risoluzione dei seguenti quesiti:*

*1. Risolvere il seguente sistema lineare:*

$$\left\{ \begin{array}{l} 2x_1 - 4x_2 - 7x_3 - 4x_4 = 5 \\ 9x_1 - 3x_2 - 2x_3 - 7x_4 = -1 \\ 5x_1 - 2x_2 - 3x_3 + x_4 = -3 \\ 6x_1 - 5x_2 - 4x_3 - 3x_4 = 2 \end{array} \right.$$

- calcolare il prodotto scalare  $s = u \cdot v^T$ , della seguente coppia di punti:*

$$u = [5, 3, -2, -4, -1] \quad v = [2, -1, 0, -7, 2]$$

---

1. Data la matrice

$$A = \begin{bmatrix} 5 & 3 & -6 \\ 7 & 2 & 0 \\ -4 & 8 & 1 \end{bmatrix} \left[ \text{calcolare:} \right.$$

```
A1=A*A;
```

```
A2=A'*A;
```

```
A3=A.*A;
```

```
d1=diag(A);
```

```
d2=diag(A,1);
```

```
e=exp(A);
```

```
sq=sqrt(A);
```

```
e1=exp(log(A+7));
```

```
m=max(A);
```

```
sn=sign(A);
```

# Fine Esercitazione