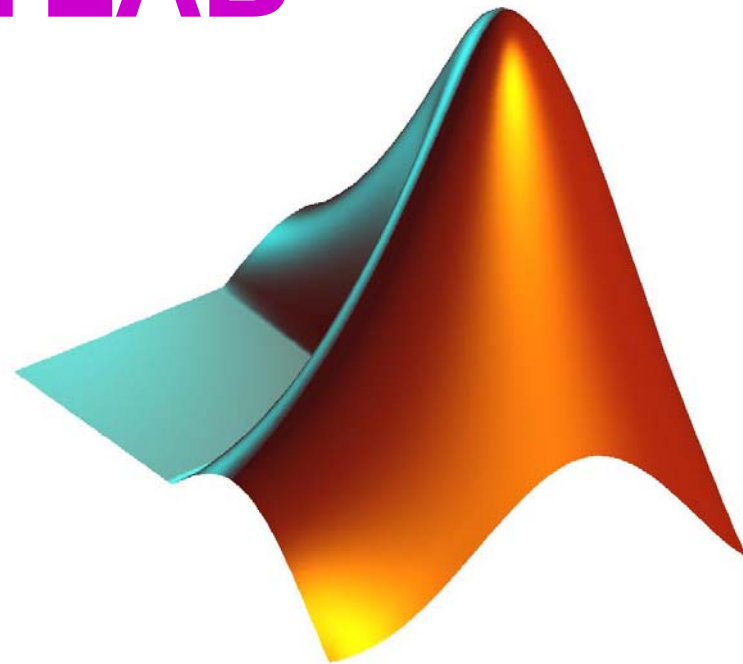
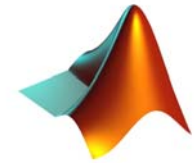


Introduzione all'ambiente MATLAB

Parte I





Tutorial

Sito ufficiale di MATHWORKS:

<http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml>

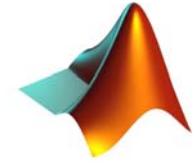
Tutorial in italiano

<http://programmarenin.supereva.it/indmat/matlab.htm?p>

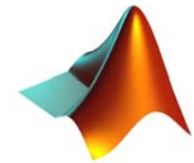
Tutorial in inglese: MATLAB primer

<http://math.ucsd.edu/~driver/21d-s99/matlab-primer.html>

Overview



- MATLAB e le sue potenzialità
 - Funzionalità di MATLAB
 - Ambiente di sviluppo MATLAB
- Il manuale o Help
- Le variabili e lo spazio di lavoro
- Operatori MATLAB
 - Operatori di base, operatori logici, operatori relazionali
 - Vettori e Matrici



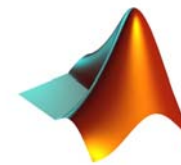
- Operazioni tra matrici
 - Matrici speciali
- Operatori su elementi
- Operazioni aritmetiche su vettori-matrici
 - Variabili complesse
- Grafica in MATLAB



MATLAB (MATrix LABoratory) è un ambiente interattivo ad alto livello che consente di costruire e gestire facilmente matrici e, come casi particolari, vettori e scalari.

La struttura dati di base è la **matrice**: ciò significa che durante l'elaborazione ogni quantità viene trattata dall'ambiente di calcolo come una matrice di dimensione $n \times m$.

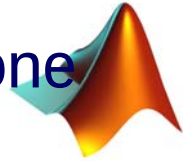
(un **vettore** è una matrice $1 \times n$, mentre uno **scalare** è gestito come una matrice 1×1).



Il pacchetto però non è utilizzato solo per gestire matrici, ma è un'ottima piattaforma di sperimentazione e verifica per il calcolo numerico in genere.

Il calcolo simbolico in MATLAB è basato sul software Maple.

MATLAB fornisce un ambiente di calcolo, visualizzazione e programmazione scientifica, in cui è possibile:



- **calcolare direttamente espressioni matematiche;**

```
>> ((tan(pi/5)+2)*exp(2.3)-0.01)/log(2)
```

```
ans =
```

```
39.2197
```

- **utilizzare il semplice ambiente di programmazione per costruire i propri algoritmi (Parte II);**
- **sfruttare algoritmi di base già implementati**
- built-in function -

```
>> mean([1.5 2.5 3.5])
```

```
ans =
```

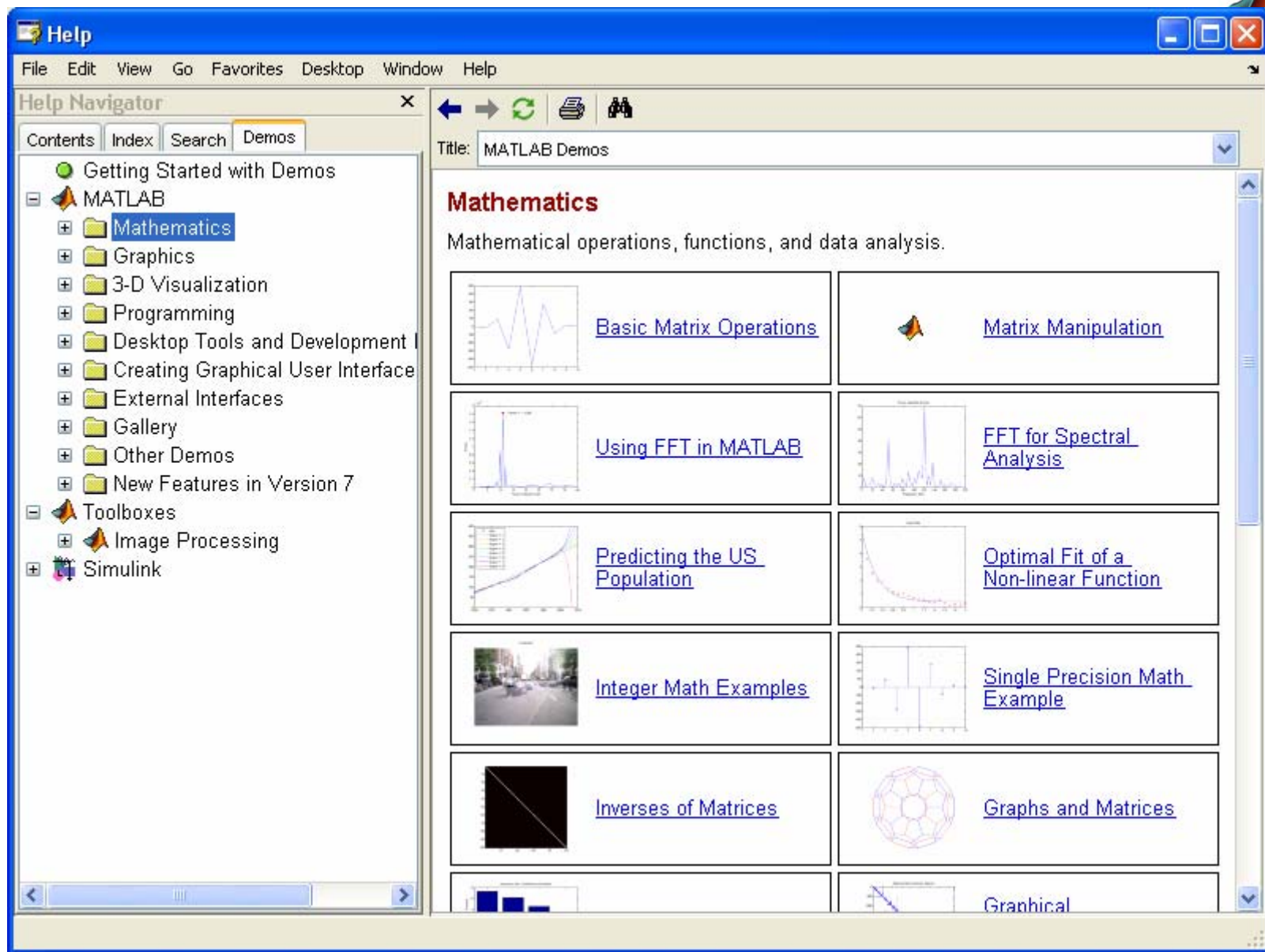
```
2.5000
```



Altre funzionalità di MATLAB includono:

- Matematica e calcolo
- Sviluppo di procedure e applicazioni
- Modellistica, simulazione e prototipizzazione
- Analisi di dati, esplorazione e visualizzazione
- Disegno industriale e scientifico
- Costruzione di interfacce utente
- TOOLBOX vari

Con il comando **demos** è possibile vederne alcuni esempi.





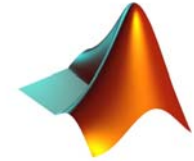
MATLAB ha anche un linguaggio proprio per programmare.

E' un linguaggio interpretato e non compilato:

questo significa che le istruzioni vengono tradotte in linguaggio macchina (il linguaggio “capito” dal processore) e subito eseguite una per volta.

Utilizzando C, Fortran, C++, la traduzione da linguaggio ad alto livello a linguaggio macchina avviene invece nel processo di compilazione, in cui tutto il programma viene tradotto in linguaggio macchina e poi eseguito.

Ambiente di sviluppo MATLAB

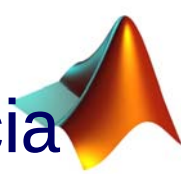


Per **lanciare MATLAB** da ambiente Windows basta cliccare due volte con il mouse sull'icona corrispondente.

La finestra che appare quando si esegue MATLAB viene chiamata desktop.

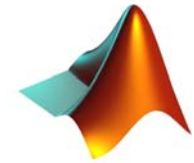
Il simbolo prompt **>>** indica che il calcolatore è pronto a ricevere le istruzioni e ad eseguirle.

Per **uscire dall'ambiente** basta digitare
>> quit

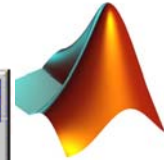


Il desktop contiene strumenti (con interfaccia grafica) per gestire files, variabili e applicazioni legati a MATLAB. In particolare:

- *Command Window* : detta anche finestra dei comandi, è la parte del desktop in cui si scrivono i comandi MATLAB che devono essere eseguiti. Permette pertanto all'utente di comunicare con il programma MATLAB.
- *Command History* : questa finestra mostra tutti i comandi precedentemente immessi nella finestra dei comandi.



- *Launch Pad* : è la finestra in cui sono elencati i toolbox e tutti gli altri programmi di MATLAB che sono installati nel computer.
- *Editor* : è la finestra in cui si scrivono i programmi MATLAB (M-files, cioè file con estensione “.m”). Per richiamarla digitare
>> edit



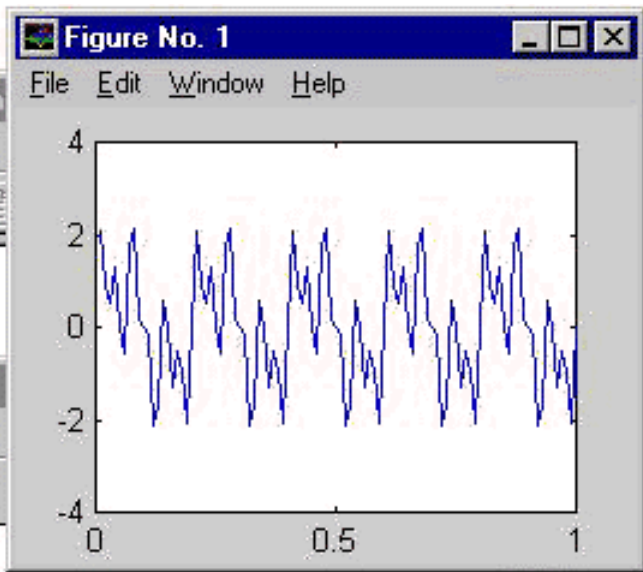
```
MATLAB Editor/Debugger - [D:\MATLAB\toolbox\matlab\fun
File Edit View Debug Window Help
[Icons]
if trace
    lims = [min(x) max(x) min(y) max(y)];
    if ~isreal(v)
        lims
        lims
    end
    ind = f
    if ~isr
        [mi
        lims
    end
    axis(lin
    plot([c
    hold on
    if ~isr
Ready
```

MATLAB Command Window

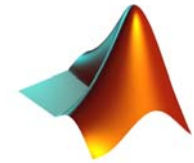
```
>> Fs=100;
>> t = (1:100)/Fs;
>> s = sin(2*pi*t*5)+sin(2*pi*t*15)+sin(2*pi*t*30);
>> whos
Name      Size      Bytes  Class
Fs         1x1         8  double array
s          1x100       800  double array
t          1x100       800  double array

Grand total is 201 elements using 1608 bytes

>> plot(t, s)
>>
```



Il manuale (o help)

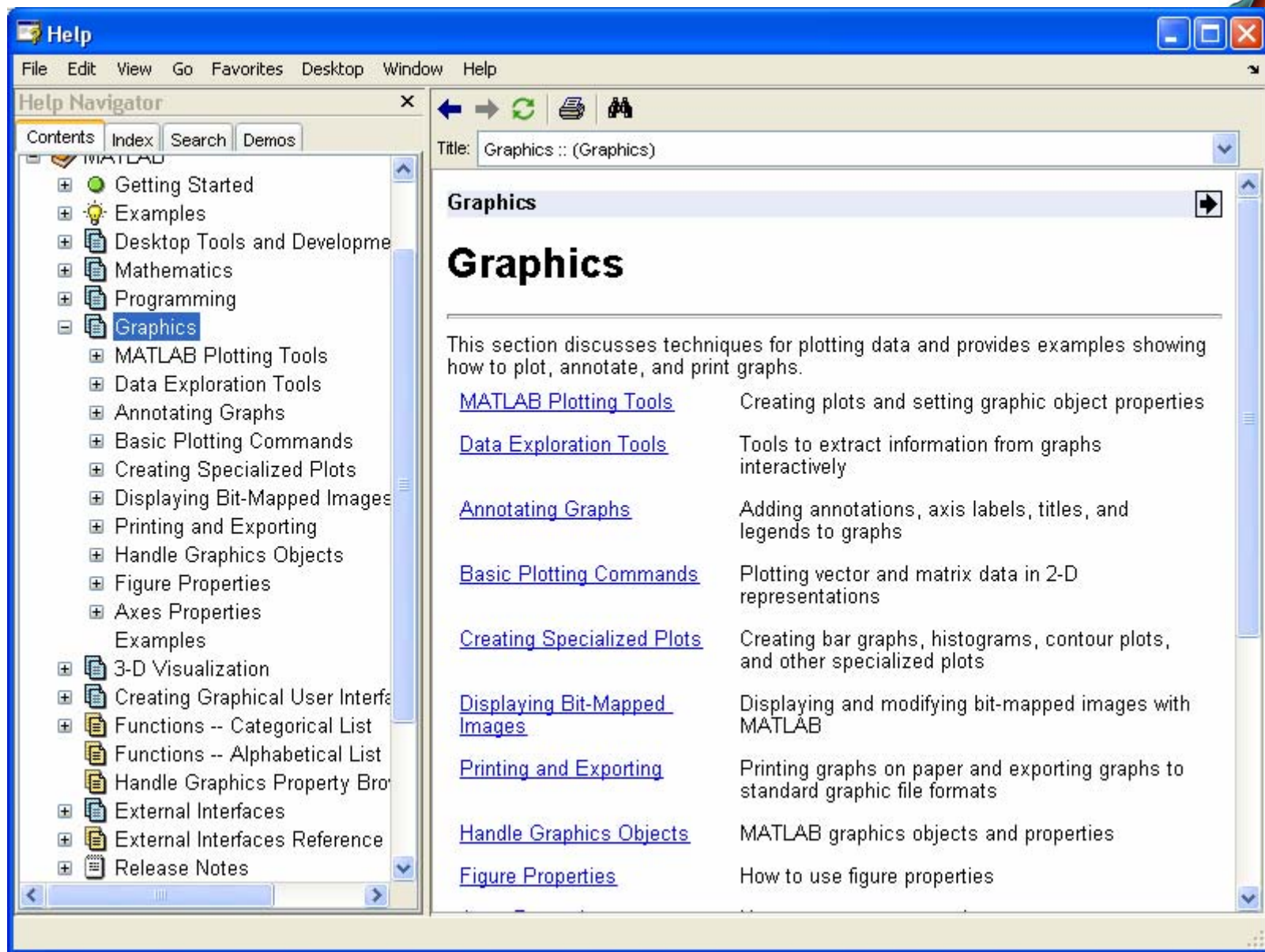


- Si può accedere al manuale o tramite il menu

Help → MATLAB Help (guida in linea)

o digitando **help** dalla Command Window
(help generale di tutte le funzioni).

- Per visualizzare l'help di un singolo comando digitare **help <nomecomando>** dalla Command Window (esempio **help plot**).
- Digitando **lookfor <keyword>** si attiva invece la ricerca di funzioni basate su una parola chiave.
- Attraverso il comando **doc** si accede direttamente alla documentazione online di MATLAB.



Le variabili



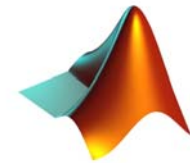
Variable : nome associato ad una entità (scalare, vettore, matrice) che contiene dati.

I nomi scelti (meglio se legati all'entità che rappresentano) devono rispettare le seguenti regole di sintassi:

1. possono contenere solo lettere, cifre e il carattere di sottolineatura (“_”);
2. non possono iniziare con una cifra;
3. non si possono utilizzare parole riservate di MATLAB.

➤ Matlab è un linguaggio case sensitive, ossia distingue fra lettere maiuscole e minuscole: la variabile A è quindi diversa dalla variabile a.

La variabile utilizzata da MATLAB in default è *ans*.



Esempio:

Introdurre 4 variabili e farne la media.

```
>> a=10;
```

```
>> b=20;
```

```
>> c=30;
```

```
>> d=40;
```

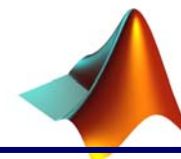
```
>> media=(a+b+c+d)/4
```

```
media =
```

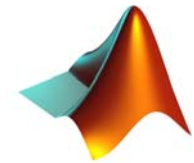
```
25
```

Le variabili **a,b,c,d** contengono
rispettivamente i valori 10,20,30,40

Il valore della media
aritmetica tra 10,20,30,40
è stato memorizzato nella
variabile **media**



Tipo	Tipo dato	Occupazione di memoria
Double	Numeri reali nell'intervallo $[10^{-37}, 10^{37}]$	8 byte
Complex double	Numeri complessi	16 byte
Logical double	Risultato di una operazione logica (1=vero, 0=falso)	8 byte
Char	Carattere	2 byte



Esempi:

a = sqrt(3);



Double

c = -2+i*8.2



Complex double

t = a<1



Logical double

q = 'k'



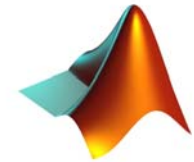
Char

var_testo = 'questa stringa viene assegnata alla
variabile var_testo'



Char array

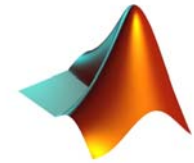
Formato output



L'output può essere visualizzato in diversi modi, pur non influenzando sul formato che MATLAB usa per memorizzare ed elaborare i dati.

SHORT	Virgola fissa 5 cifre
SHORT E	Virgola mobile 5 cifre
SHORT G	Meglio tra virgola fissa e mobile 5 cifre
LONG	Virgola fissa 15 cifre
LONG E	Virgola mobile 15 cifre
LONG G	Meglio tra virgola fissa e mobile 15 cifre
RAT	Approssimazione mediante il rapporto ridotto ai minimi termini

Esempio: **>>y = 8/6**



>>format short

1.3333

>>format short e

1.3333E+000

>>format short g

1.3333

>>format long

1.3333333333333333

>>format long e

1.3333333333333333E+000

>>format long g

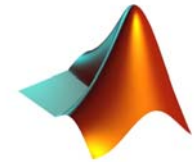
1.3333333333333333

>>format rat

4/3

default : format short

Lo spazio di lavoro

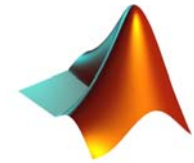


L'insieme delle variabili mantenute in memoria durante la sessione MATLAB viene chiamata *spazio di lavoro* (o **workspace**).

Per aprire il Workspace Browser avete tre possibilità:

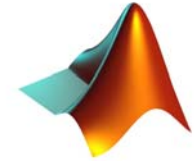
- selezionare l'opzione Workspace dal menu View nel desktop di MATLAB;
- cliccare su Workspace nella scheda Launch Pad;
- digitare Workspace nel prompt della finestra comandi.

Utilizzando tale browser potete:



1. visualizzare e modificare le variabili dell'area di lavoro (eventualmente cambiandone il formato output);
2. cancellare le variabili dell'area di lavoro;
3. rappresentare graficamente le variabili dell'area di lavoro;
4. salvare l'area di lavoro;
5. caricare un'area di lavoro precedentemente salvata.

La sessione di lavoro



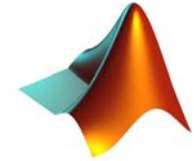
Il comando

diary <nome file>

memorizza nel file ASCII < nomefile > la sessione di lavoro (comandi dati e workspace) da quel punto in poi in modo da poterla poi consultare con un qualsiasi editor (es. WORD).

Non possiamo però utilizzare il file < nomefile > per ricaricare il lavoro fatto e continuare a lavorarci; per questo si devono utilizzare i comandi save e load.

La sessione di lavoro



Ogni sessione di lavoro può essere salvata in un file binario (nomefile.mat) mediante il comando

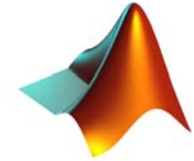
- **save** <nome file>

e ricaricata in ambiente MATLAB mediante

- **load** <nome file>

Le variabili utilizzate in ogni workspace possono essere consultate semplicemente digitando il nome della variabile stessa.

La sessione di lavoro



Per visualizzare la lista delle variabili attive nel corrente spazio di lavoro digitate

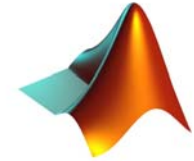
- **who**

Per informazioni più dettagliate utilizzate il comando

- **whos**

Ogni variabile sarà visualizzata insieme allo spazio da essa occupato, al numero di elementi, e al suo tipo.

La sessione di lavoro

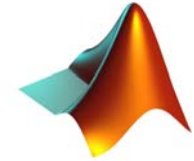


» whos

Name	Size	Bytes	Class
a	1x4	32	double array
ans	3x3	72	double array
f	1x3	24	double array
z	1x1	16	double array(complex)
e	1x4	32	double array(logical)
c	1x4	8	char array

Grand total is 25 elements using 184 bytes

La sessione di lavoro



Per cancellare la variabile *nomevariabile*

- **clear** *<nomevariabile>*

Per cancellare tutte le variabili di una sessione di lavoro

- **clear**

Per pulire il desktop di MATLAB

- **clc**

Per pulire la finestra di una figura

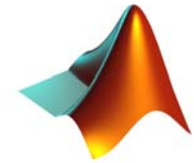
- **clf**

Per avere un elenco degli m-files memorizzati

- **what**
- **why**

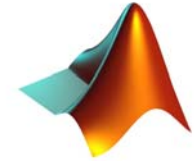
La sessione di lavoro

altri comandi utili



- **pwd** visualizza la directory in cui si sta lavorando
- **dir**, o **ls** elenca i file presenti nella directory
- **cd** modifica la directory attuale
- **path**, o **matlabpath** elenca i possibili percorsi delle directory in MATLAB
- **addpath** aggiunge la directory ai percorsi già esistenti
- **pathtool** permette di accedere alla finestra degli strumenti per la gestione dei percorsi

Operatori MATLAB



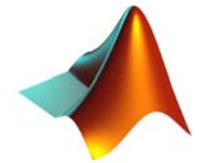
Operatori di base:

+	addizione
-	sottrazione
/	divisione a destra
\	divisione a sinistra
^	elevamento a potenza
*	moltiplicazione

$$1/4=4\backslash 1=0.25$$

Operatori logici:

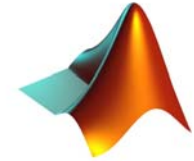
&	and
 	or
~	not



Operatori relazionali:

$\sim =$	diverso
$\leq$	minore uguale
$<$	minore
$\geq$	maggiore uguale
$>$	maggiore
$=$	uguale logico

Scalari, Vettori e matrici



Una matrice ha dimensione $n \times m$ (n righe, m colonne). Un vettore ha dimensione $n \times 1$ (vettore colonna) o $1 \times n$ (vettore riga). Uno scalare ha dimensione 1×1 .

- **$a = 1$** (scalare, ovvero matrice 1×1)
- **$a = [0 \ 1 \ 2 \ 3 \ 4]$, $a = [0,1,2,3,4]$** ed anche **$a = [0:4]$**
- **$b = [0, .5, 1, 1.5, 2, 2.5]$** ed anche **$b = [0:.5:2.5]$**
- **$c = [.1, .1, .1, .1, .1]$** ed anche **$c = \text{ones}(1,5) * 0.1$**

Costruire Vettori



» **a = [1 2 3 4]**

a =

1 2 3 4

Definisce un vettore "a"
(le parentesi quadre indicano
un vettore o matrice)

» **size(a)**

ans =

1 4

fornisce la dimensione di "a"

» **length(a)**

ans =

4

usato per i vettori indica
la loro lunghezza

» **a'**

ans =

1

2

3

4

» **[1;2;3;4]**

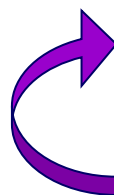
ans =

1

2

3

4



» **size(a')**

ans =

4

1

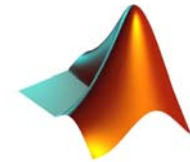
» **length(a')**

ans =

4

trasposta di "a" (ha dimensioni
"invertite" rispetto ad "a")

Costruire Matrici



```
» c = [1 2 3 4 ; 5 6 7 8 ]
```

```
c =
```

```
1  2  3  4  
5  6  7  8
```

Per fare riferimento ad un elemento della matrice “c”:

```
» c(1,1)
```

```
ans =
```

```
1
```

```
» c(2,3)
```

```
ans =
```

```
7
```

Usare “:” per indicare tutte le righe o tutte le colonne, esempio:

c(1,:) indica la prima riga, tutte le colonne

c(:,2) indica tutte le righe, la seconda colonna

c(:,2:4) indica tutte le righe, dalla seconda alla quarta colonna

```
» c(1,:)
```

```
ans =
```

```
1  2  3  4
```

```
» c(:,2)
```

```
ans =
```

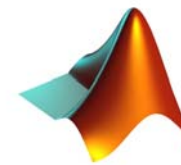
```
2  
6
```

```
» c(:,2:4)
```

```
ans =
```

```
2  3  4  
6  7  8
```

Funzioni di matrici



```
» c = [1 2 3 4 ; 5 6 7 8 ]
```

```
c =
```

```
1 2 3 4  
5 6 7 8
```

```
» sum(c)
```

```
ans =
```

```
6 8 10 12
```

```
» sum(c')
```

```
ans =
```

```
10 26
```

```
» sum(sum(c))
```

```
ans =
```

```
36
```

La funzione **sum** calcola la somma degli elementi di una matrice per colonne; il risultato è un vettore.

Se la matrice è un vettore 1 x m (come sum(c)) , allora la somma è calcolata sugli elementi del vettore.

```
» mean(c)
```

```
ans =
```

```
3 4 5 6
```

```
» max(c)
```

```
ans =
```

```
5 6 7 8
```

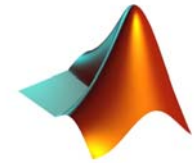
```
» min(c)
```

```
ans =
```

```
1 2 3 4
```

mean fornisce la media per colonne; **max** e **min** il massimo e il minimo ancora per colonne.

Concatenazione di matrici



» **A = [2,0; 0,1; 3,3]**

A =

2	0
0	1
3	3

» **A(1:2,1:2)**

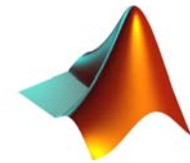
ans =

2	0
0	1

» **A=[A,[1;2;3]]**

A =

2	0	1
0	1	2
3	3	3



```
» a = [1 2 3 4]; b = [5 6 7 8]; c = [a ; b]
```

```
c =
```

```
1 2 3 4  
5 6 7 8
```

Definiamo una matrice “c” come concatenazione dei vettori “a” e “b”.

Nota: l’uso di “;” entro [....] indica la fine della riga.

```
» size(c)
```

```
ans =
```

```
2 4
```

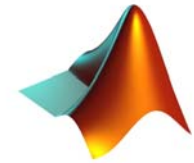
Definiamo la matrice “d” ponendo “a” e “b” a fianco: notare il risultato diverso da “c”

```
» d = [a b]
```

```
d =
```

```
1 2 3 4 5 6 7 8
```

Condizioni logiche su matrici



```
» d = c(1,:)
```

```
d =
```

```
1    2    3    4
```

Possiamo definire un vettore “e” che è una funzione logica di d

```
» e = d>2
```

```
e =
```

```
0    0    1    1
```

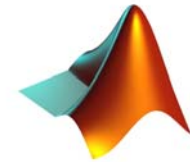
Possiamo ora usare “e” per trovare gli elementi di d>2

```
» d(e)
```

```
ans =
```

```
3    4
```

Matrici speciali



» **ones(2,3)**

ans =

1 1 1

1 1 1

» **zeros(1,4)**

ans =

0 0 0 0

» **rand(3,3)**

ans =

0.2176 0.4909 0.8985

0.4054 0.1294 0.5943

0.5699 0.5909 0.3020

» **ones(2)**

ans =

1 1

1 1

» **zeros(2,1)**

ans =

0

0

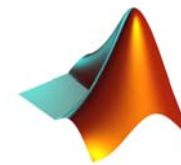
» **eye(2)**

ans =

1 0

0 1

Operazioni aritmetiche su vettori-matrici



```
» a = [1 2 3]
```

```
a =  
    1    2    3
```

```
» b = [4 5 6]
```

```
b =  
    4    5    6
```

```
» a + b
```

```
ans =  
    5    7    9
```

```
» a*b'
```

```
ans =  
    32
```

Operatori:

+ - * /

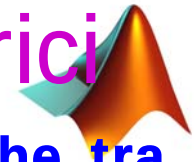
Somma/sottrazione purchè le dimensioni siano compatibili

Moltiplicazione tra matrici:

moltiplicare una matrice $n \times m$ con una matrice $m \times p$ per ottenere una matrice $n \times p$

qui moltiplichiamo una matrice 1×3 per una 3×1 per ottenere una 1×1 (scalare)

Operatori su elementi di vettori-matrici



Gli operatori su elementi indicano operazioni aritmetiche tra elementi corrispondenti: `.*` `./` `.\` `.^`

» `f = [1 2 3]; g = [4 5 6];` Definiamo i vettori 1x3 `f` e `g`.

Nota: usando `“;”` alla fine della linea si elimina la stampa del risultato.

» `h = f.*g`

`h = 4 10 18`

» `h = f.\g`

`h = 4.0000 2.5000 2.0000`

» `h = f./g`

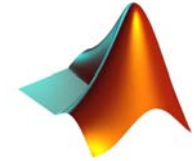
`h = 0.2500 0.4000 0.5000`

» `h = f.^2`

`h = 1 4 9`

Valido per vettori e matrici.

Variabili complesse



Un complex double array è visto come la somma di due double array.

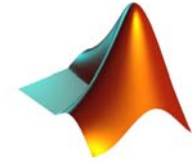
Con le variabili 'i' o 'j' si indica l'unità complessa

$$i = \sqrt{-1}$$

- Numeri complessi

$$z = 3 + 4 * i \quad (o \quad z = 3 + 4 * j)$$

Variabili complesse



- Array di variabili complesse

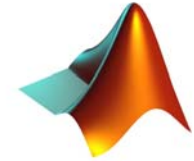
$$\mathbf{B} = [\mathbf{1} \ \mathbf{2} ; \mathbf{3} \ \mathbf{4}] + i * [\mathbf{5} \ \mathbf{6} ; \mathbf{7} \ \mathbf{8}]$$

o equivalentemente

$$\mathbf{B} = [\mathbf{1+5*j} \ \mathbf{2+6*j} ; \mathbf{3+7*j} \ \mathbf{4+8*j}]$$

Non ci devono essere spazi bianchi
nell'espressione del numero complesso.

Grafici



plot

plot x-y lineare

loglog

plot log-log x-y

semilogx

semi-log x-y plot(logaritmico in x)

semilogy

semi-log x-y plot (logaritmico in y)

polar

plot in coords polari

mesh

mesh di superficie 3D

contour

plot a linee di livello

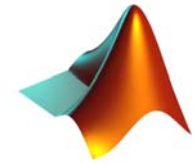
bar

plot a barre

stairs

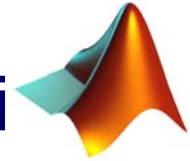
plot a gradini

Prodotto il plot, i seguenti comandi possono essere utili per arricchirlo:



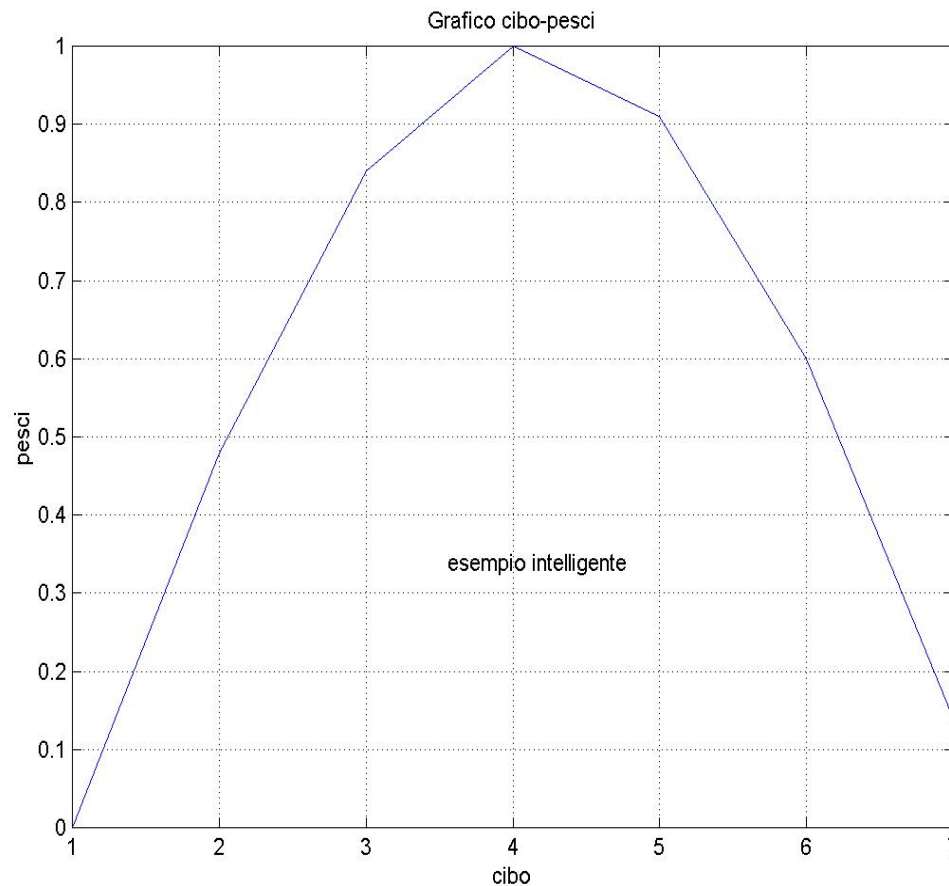
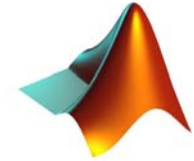
title	titolo del plot
xlabel	etichetta associata all'asse x
ylabel	etichetta associata all'asse y
text	testo posizionato in modo arbitrario
gtext	testo posizionato mediante mouse
grid	linee griglia

I seguenti comandi vengono invece utilizzati per gestire il grafico:



axis	scala degli assi
hold	mantiene il plot corrente sullo schermo
shg	mostra il grafico sullo schermo
clf	pulisce la figura corrente
subplot	suddivide un plot

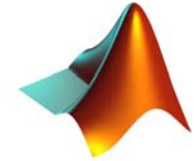
Esempio



- » `y = [0 .48 .84 1 .91 .6 .14];`
- » `plot(y)`
- » `title('Grafico cibo-pesci')`
- » `xlabel('cibo')`
- » `ylabel('pesci')`
- » `grid`
- » `gtext('esempio intelligente')`

N.B. Ogni stringa testo
deve essere racchiusa tra
apostrofi

Più grafici



Per mantenere il grafico attivo nella finestra si dovrà utilizzare il comando **hold on**, altrimenti verrà perso quando si chiede di visualizzare il nuovo plot.

```
>> t=0:5/200:5;
```

```
>> hold on           % mantine il plot precedente
```

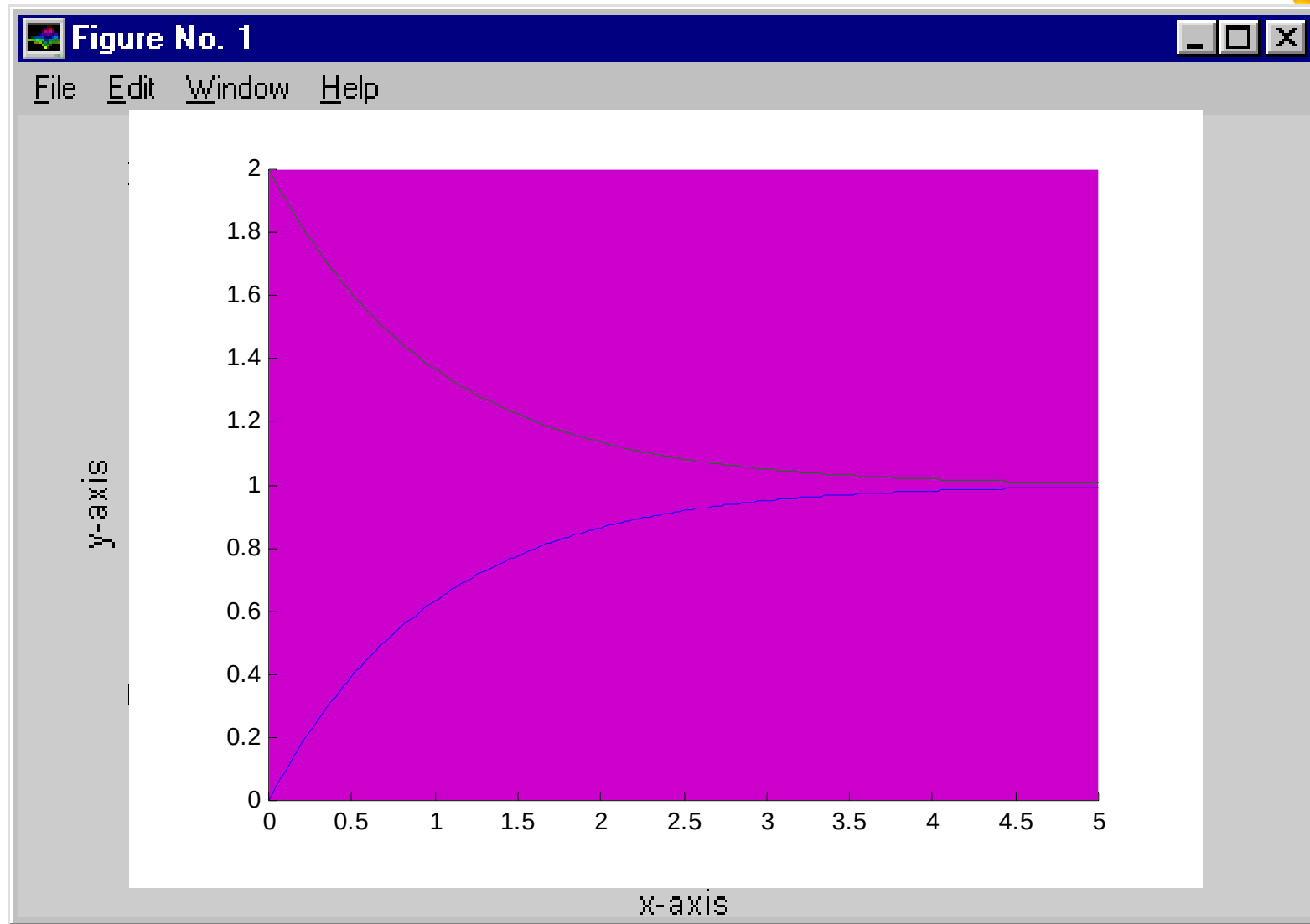
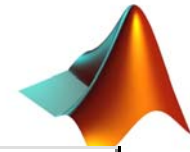
```
>> y1=1-exp(-t);
```

```
>> y2=1+exp(-t);
```

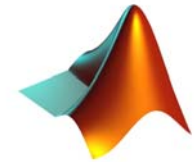
```
>> plot(t,y1,t,y2)
```

```
>> hold off         % disattiva l'hold on
```

Più grafici



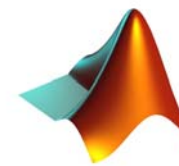
Stile di linee e punti



Specialmente quando si visualizzano più grafici in una stessa finestra, si consiglia di assegnare stili differenti ad ogni linea grafica per una migliore leggibilità. Questo è possibile aggiungendo un altro argomento al comando plot:

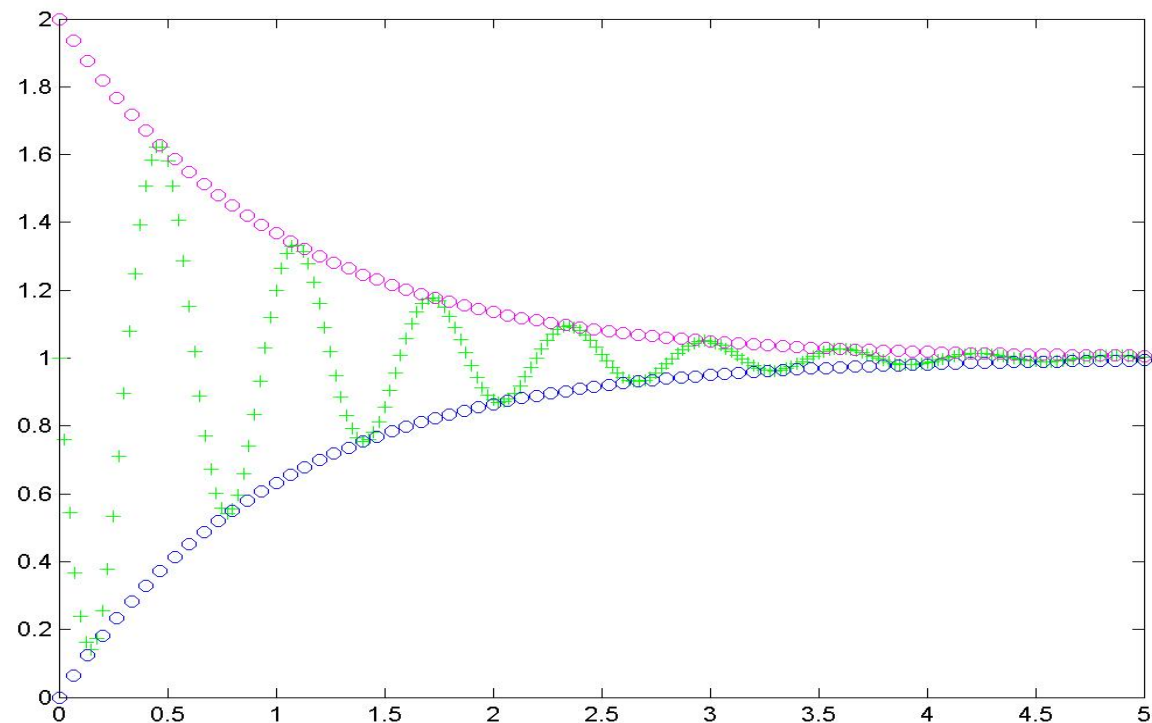
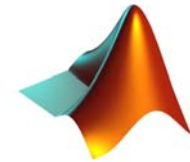
> >plot(t,y,':')

Automaticamente MATLAB unisce i punti con delle linee, se si vogliono visualizzare solo i punti si può specificare il tipo di punto desiderato.

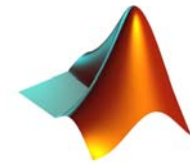


TIPO LINEA		COLORI		TIPO PUNTI	
solid	-	blu	b	point	.
tratteggio	--	nero	k	più	+
a punti	:	verde	g	star	*
tratto-punti	-.	rosso	r	cerchio	o
		giallo	y	segno x	x
		bianco	w		
		ciano	c		
		magenta	m		

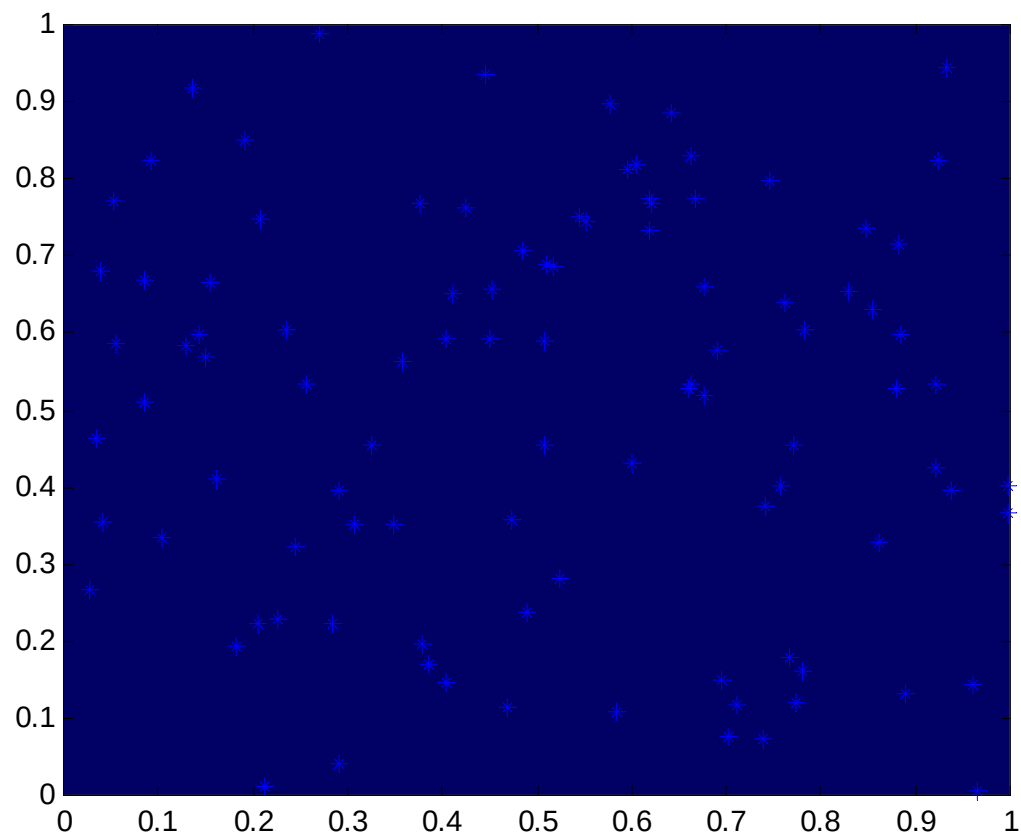
```
>>t=0:1/40:5; t1=0:1/15:5;  
>>y=1-exp(-t).*sin(10*t);  
>>y1=1-exp(-t1);  
>>y2=1+exp(-t1);  
>>plot(t,y, 'g+', t1,y1, 'bo', t1,y2, 'mo')
```



Grafici



```
» x = rand(1,100);  
» y = rand(1,100);  
» plot(x,y,'*')
```



Gestione interattiva della finestra di una figura



MATLAB permette di gestire interattivamente la finestra della figura utilizzando i menu e le icone presenti. E' possibile:

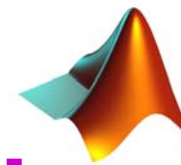
- inserire testi, legende, linee e frecce;
- zoom, ruotare, cambiare la posizione dell'osservatore;
- cambiare colore al grafico, allo sfondo, alla finestra;

Per caricare una figura da MATLAB è necessario salvarla in formato **.fig**:

File → **Save As** → **< figura1.fig >**

Per stamparla o includerla in documenti word si deve salvare in formato **.jpg**, **.eps**,...:

File → **Export** → **< figura1.eps >**

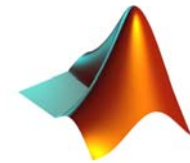


Gestione di più finestre grafiche

Se si desidera avere più finestre grafiche distinte, si utilizza il comando

subplot(mnp)

La finestra grafica si divide in una matrice $m \times n$ di finestre grafiche e la p -esima è selezionata dal plot corrente.



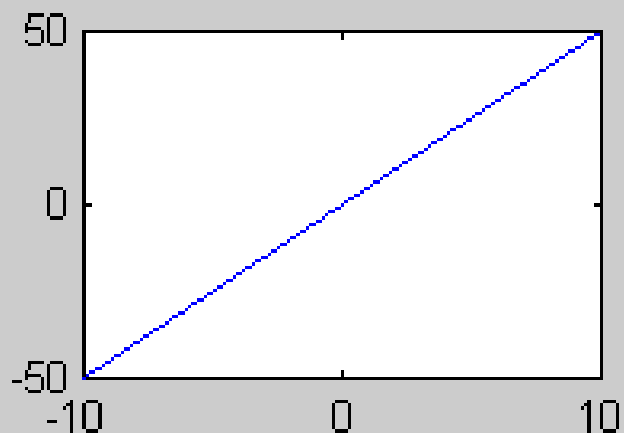
```
» t=-10:.01:10;  
» y1=t;  
» y2=t.^2;  
» y3=exp(t);  
» y4=abs(t);  
» subplot(221)  
» plot(t,y1), title('Here is the line')  
» subplot(222)  
» plot(t,y2), title('Here is the parabola')  
» subplot(223)  
» plot(t,y3), title('Here is the exponential')  
» subplot(224)  
» plot(t,y4), title('Here is the absolute value')
```



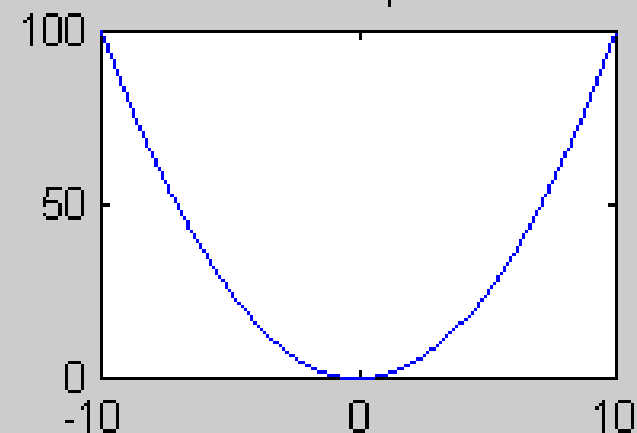
Figure No. 1

File Edit Window Help

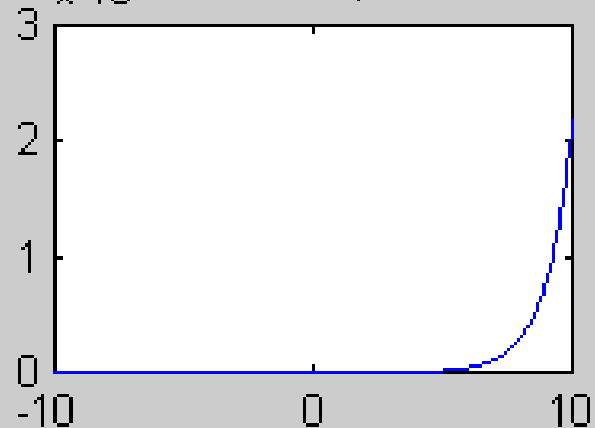
Here is the line



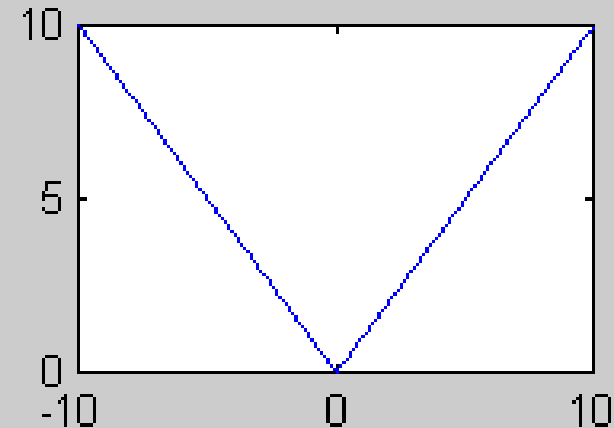
Here is the parabola



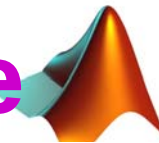
Here is the exponential



Here is the absolute value



Gestione di più finestre grafiche

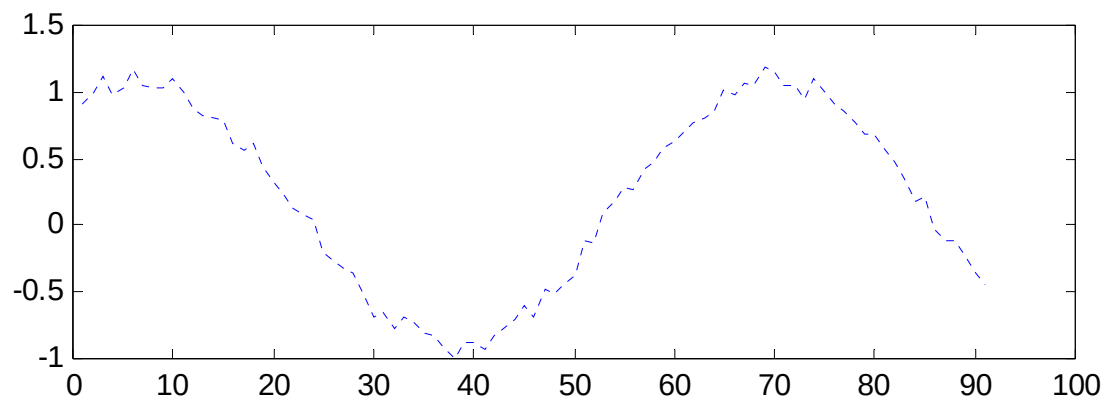
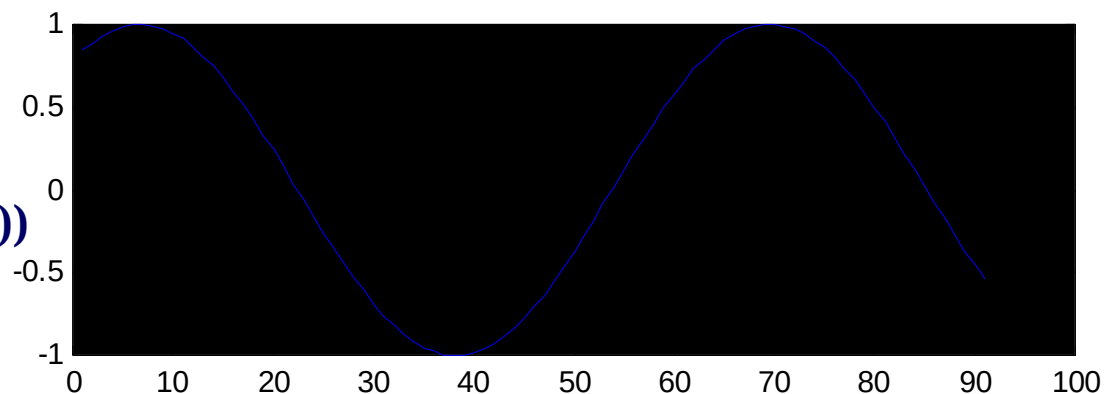


```
» clear  
» a = 1:0.1:10;  
» subplot(2,1,1); plot(sin(a))  
» size(a)
```

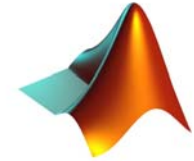
ans =

1 91

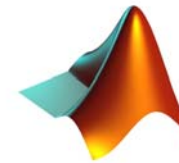
```
» r = rand(1,91)*0.2;  
» subplot(2,1,2); plot(sin(a)+r)  
» subplot(2,1,2); plot(sin(a)+r,':')
```



Grafica 3D

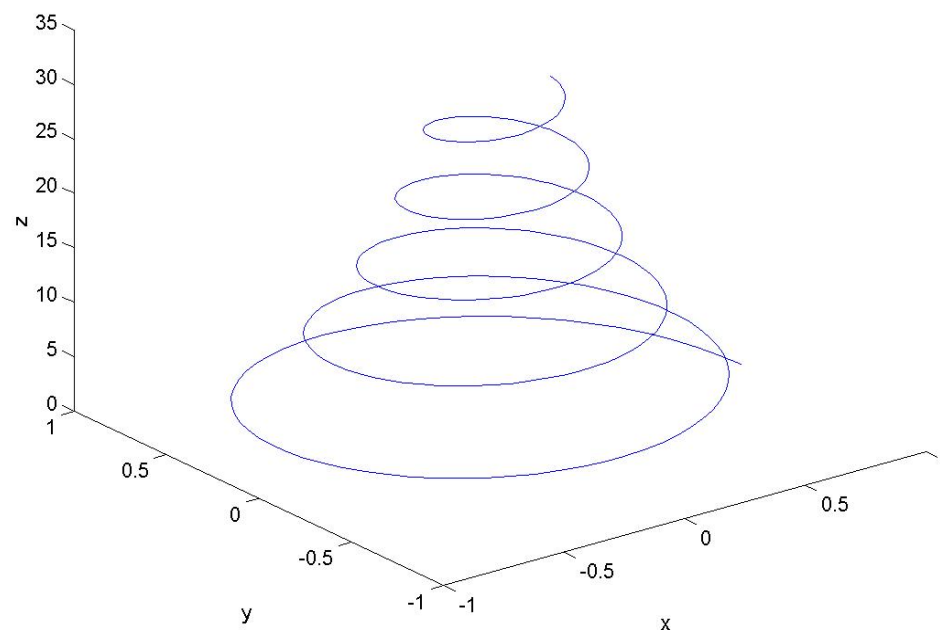


- **plot3** permette di disegnare una curva $(x(t), y(t), z(t))$, $t \in [a, b]$ nello spazio 3D
- **mesh** visualizza superfici in 3 dimensioni, produce superfici wireframe colorando solamente le linee che connettono i punti definiti
- **surf** visualizza superfici in 3 dimensioni, espone in colore le linee che connettono e le facce della superficie

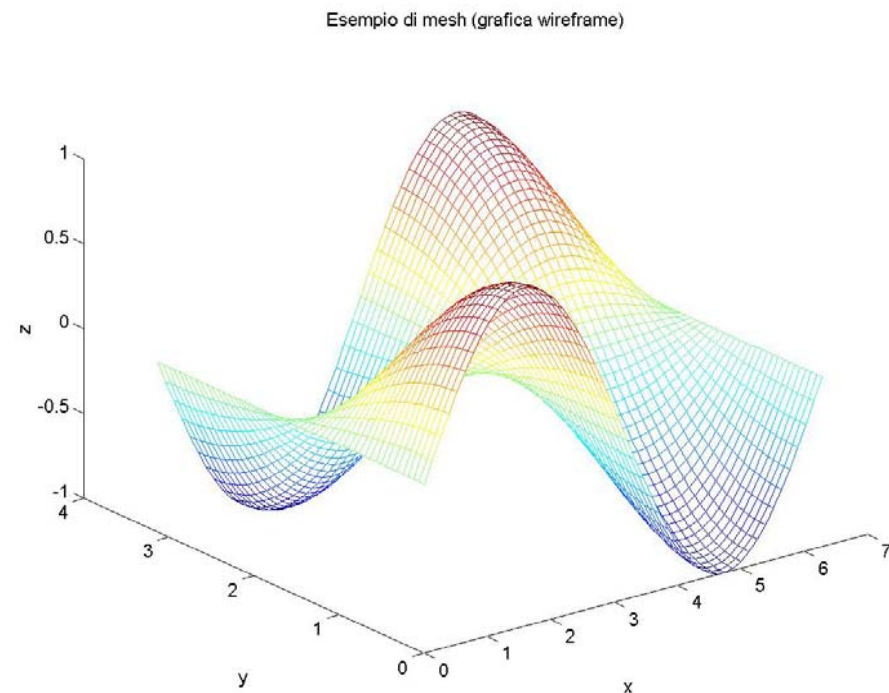
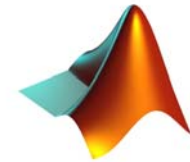


```
>>t=0:0.1:10*pi;  
>>x=exp(-t/20).*cos(t);  
>>y=exp(-t/20).*sin(t);  
>>z=t;  
>>plot3(x,y,z);  
>>title('Esempio di plot 3D');  
>>xlabel('x');  
>>ylabel('y');  
>>zlabel('z');
```

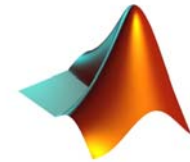
Esempio di plot 3D



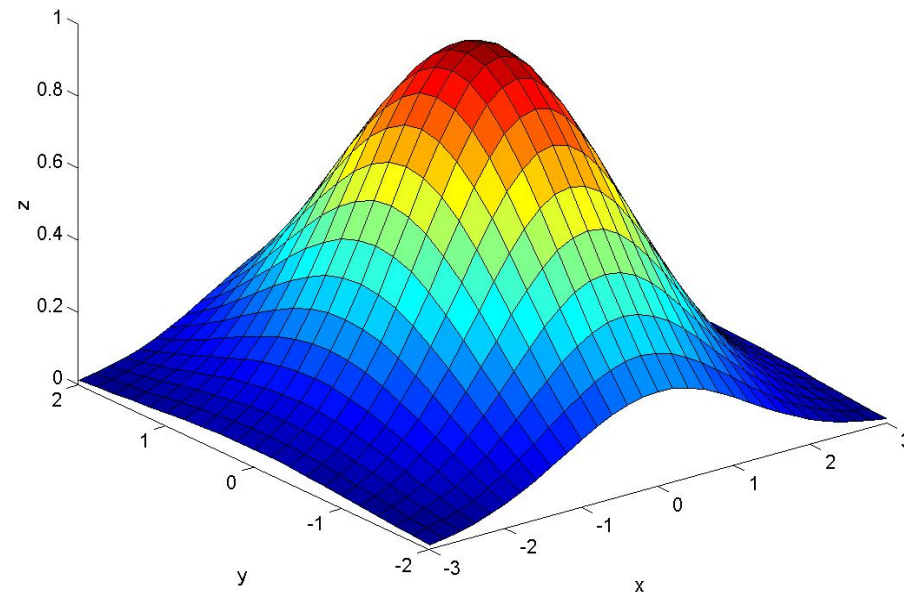
```
>>x=linspace(0,2*pi,50);  
>>y=linspace(0,pi,50);  
>>[X,Y]=meshgrid(x,y);  
>>Z=sin(X).*cos(Y);  
>>mesh(X,Y,Z);  
>>title('Esempio di mesh (grafica wireframe)');  
>>xlabel('x');  
>>ylabel('y');  
>>zlabel('z');
```



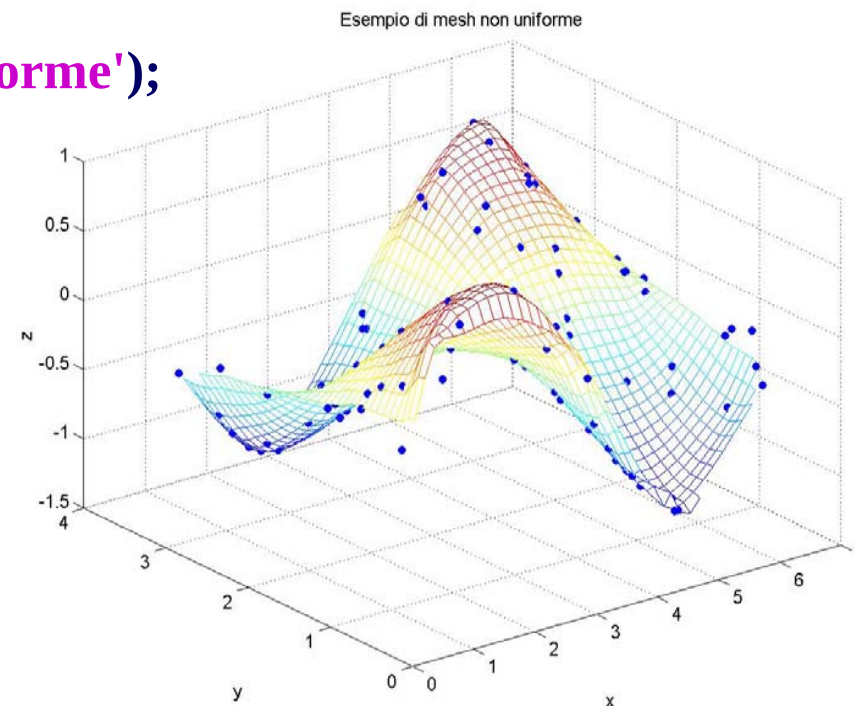
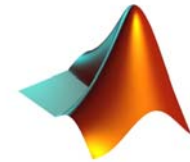
```
>>[X,Y]=meshgrid(-3:.2:3,-2:.2:2);  
>>Z=exp(-(X.^2+Y.^2)/3);  
>>surf(X,Y,Z);  
>>title('Esempio di surf (grafica a faccette)');  
>>xlabel('x');  
>>ylabel('y');  
>>zlabel('z');
```



Esempio di surf (grafica a faccette)



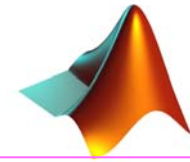
```
>>x=rand(100,1)*2*pi;  
>>y=rand(100,1)*pi;  
>>z=sin(x).*cos(y);  
>>xlin=linspace(min(x),max(x),40);  
>>ylin=linspace(min(y),max(y),40);  
>>[X,Y]=meshgrid(xlin,ylin);  
>>Z=griddata(x,y,z,X,Y,'cubic');  
>>mesh(X,Y,Z);  
>>title('Esempio di mesh non uniforme');  
>>hold on;  
>>plot3(x,y,z,'.','MarkerSize',15);  
>>xlabel('x');  
>>ylabel('y');  
>>zlabel('z');  
>>grid on;
```



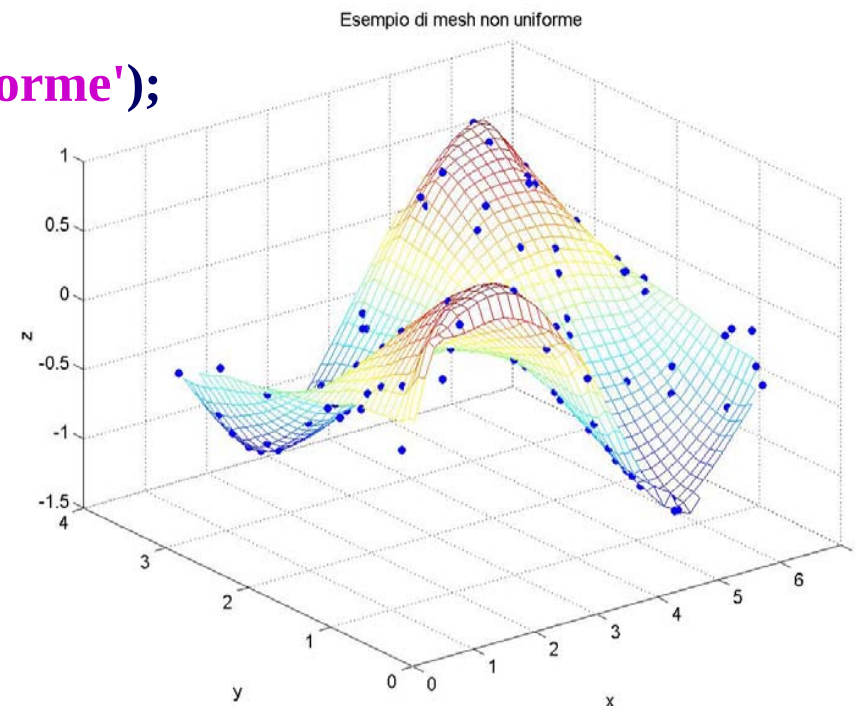
```

>>x=rand(100,1)*2*pi;
>>y=rand(100,1)*pi;
>>z=sin(x).*cos(y);
>>xlin=linspace(min(x),max(x),40);
>>ylin=linspace(min(y),max(y),40);
>>[X,Y]=meshgrid(xlin,ylin);
>>Z=griddata(x,y,z,X,Y,'cubic');
>>mesh(X,Y,Z);
>>title('Esempio di mesh non uniforme');
>>hold on;
>>plot3(x,y,z,','MarkerSize',15);
>>xlabel('x');
>>ylabel('y');
>>zlabel('z');
>>grid on;

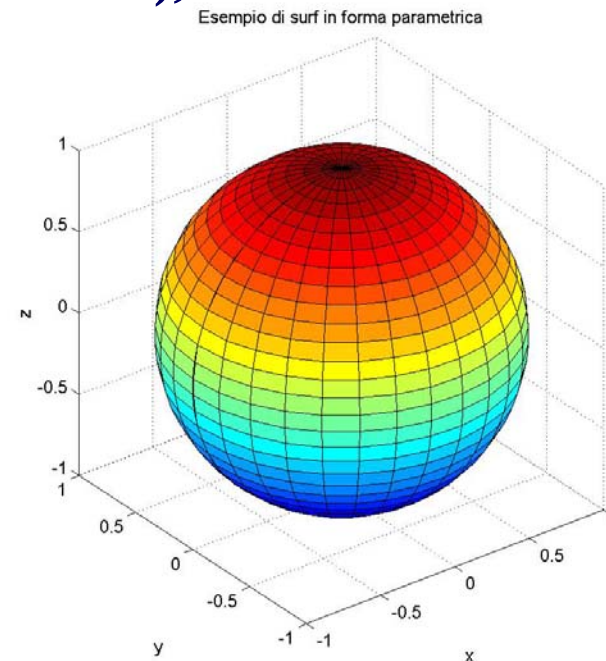
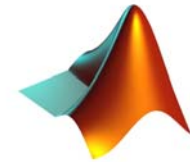
```



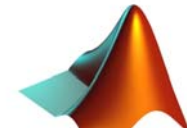
Interpolazione di dati scattered (x,y,z),
Visualizzazione funzione interpolante
valutata nella griglia XY



```
>>n=31;  
>>theta=pi*(-n:2:n)/n;  
>>phi=(pi/2)*(-n:2:n)'/n;  
>>X=cos(phi)*cos(theta);  
>>Y=cos(phi)*sin(theta);  
>>Z=sin(phi)*ones(size(theta));  
>>surf(X,Y,Z);  
>>title('Esempio di surf in forma parametrica');  
>>xlabel('x');  
>>ylabel('y');  
>>zlabel('z');  
>>grid on;  
>>axis square;
```



Immagini MATLAB



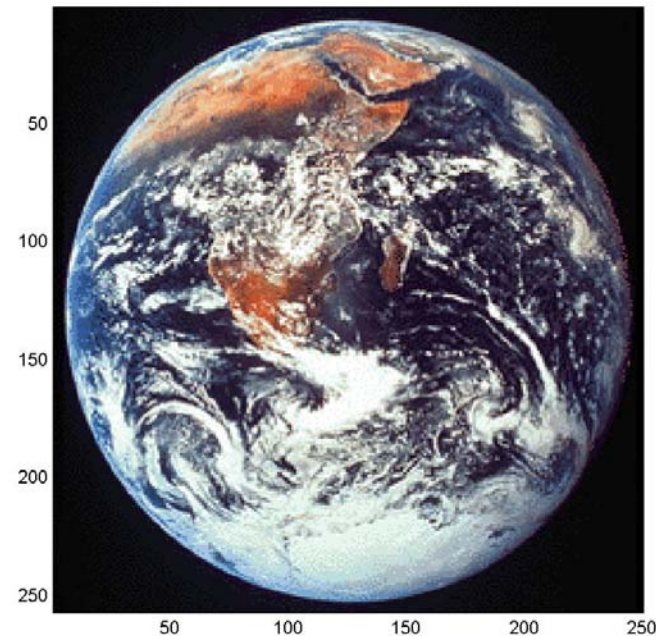
- » `clear`
- » `load earth`
- » `image(X)`
- » `colormap(map)`
- » `axis image`

Carica immagine
di dimensioni
257x250

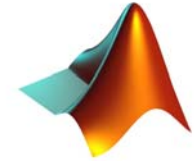
visualizza
un'immagine

associa una
colormap all'immagine

rende la proporzione 1:1 dell'immagine

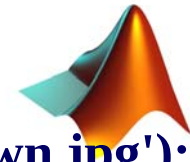


Tipi di immagini



- **Indicizzata** array $m \times n$ di interi tra $[1,p]$ che rappresentano gli indici della colormap
- **Intensità** array $m \times n$ di floating point, tra $[0,1]$ tipicamente, che rappresentano le intensità della colormap
- **True color** array $m \times n \times 3$ di floating point tra $[0,1]$ che rappresentano le intensità delle tre componenti dei colori **rgb** (solo con hardware true color)

Immagini MATLAB



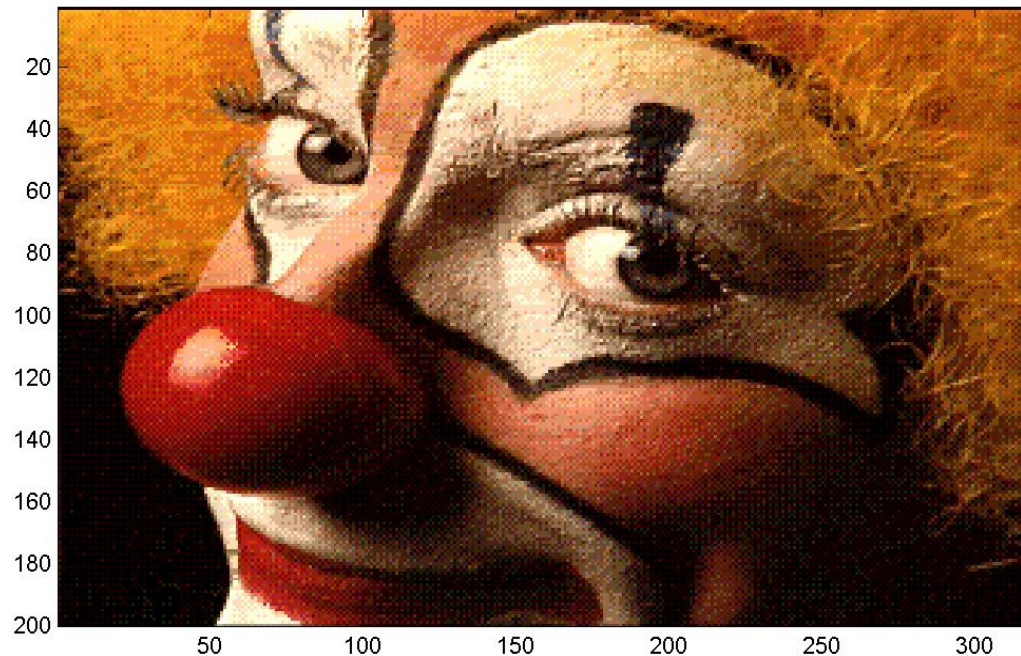
```
» clear
```

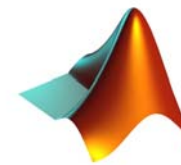
```
» i = imread('c:\immagini\clown.jpg');
```

```
» image(i)
```

```
» whos i
```

Name	Size	Bytes	Class
i	200x320	512000	double array





**Il miglior modo per
imparare MATLAB
è usare MATLAB**