



Laboratorio informatico per l'ingegneria elettrica

Università di Pisa Dipartimento di
Sistemi Elettrici e Automazione

Luca Sani

Dipartimento di Sistemi Elettrici e Automazione

Università di Pisa

tel. 050 2217364

email luca.sani@dsea.unipi.it

Sommario

- Risoluzione di sistemi di equazioni
- I numeri complessi
- I polinomi
- Interpolazione
- Le stringhe
- La grafica 2D
- La grafica 3D
- La programmazione
- Lo studio di funzioni

Risoluzione di sistemi di equazioni

Un sistema di equazioni algebriche può essere scritto in forma matriciale:

$$Ax=b$$

Condizioni affinché il sistema ammetta soluzione:

- se $\text{rango}(A)=\text{rango}(A \ b)$ e $\det(A) \neq 0$ il sistema ammette una sola soluzione;
- se $\text{rango}(A)=\text{rango}(A \ b)$ e $\det(A)=0$ il sistema ha infinite soluzioni;
- se $\text{rango}(A) \neq \text{rango}(A \ b)$ il sistema non ha soluzioni.

Il rango di una matrice si valuta con la funzione **rank**

Sintassi

Rango=rank(var_mat)

Risoluzione di sistemi di equazioni

Sistema con un numero di equazioni pari al numero di incognite:

- $\text{rango}(A)=\text{rango}(A \ b)$ e $\det(A) \neq 0$

Il sistema ammette una sola soluzione che può essere determinata in due modi:

$$x=A \setminus b$$

$$x=\text{inv}(A)*b$$

- $\text{rango}(A)=\text{rango}(A \ b)$ e $\det(A)=0$

Il sistema ammette infinite soluzioni.

Quella in termini di minimi quadrati può essere valutata con la funzione `pinv` che calcola la pseudoinversa

$$x=\text{pinv}(A)*b$$

$$\text{pinv}(A)=(AA^T)^{-1}A$$

Risoluzione di sistemi di equazioni

Sistema con un numero di equazioni maggiore del numero di incognite

- $\text{rango}(A) = \text{rango}(A \ b)$

Il sistema ammette infinite soluzioni.

Quella in termini di minimi quadrati può essere valutata con la funzione `pinv` che calcola la pseudoinversa

$$x = \text{pinv}(A) * b$$

$$\text{pinv}(A) = (A A^T)^{-1} A$$

$$x = A \backslash b$$

Risoluzione di sistemi di equazioni. Esempio

Si considera il seguente sistema di rango pieno:

$$3x - 2y + 8z = 48$$

$$-6x + 5y + z = -12$$

$$9x + 4y + 2z = 24$$

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

```
A =
```

```
     3     -2     8
    -6      5      1
     9      4      2
```

```
>> b=[48; -12; 24]
```

```
b =
```

```
    48
   -12
    24
```

```
>> [rank(A) rank([A b])]
```

```
ans =
```

```
     3     3
```

```
>> x=A\b
```

```
x =
```

```
    2.0000
   -1.0000
    5.0000
```

```
>> x=inv(A)*b
```

```
x =
```

```
    2.0000
   -1.0000
    5.0000
```

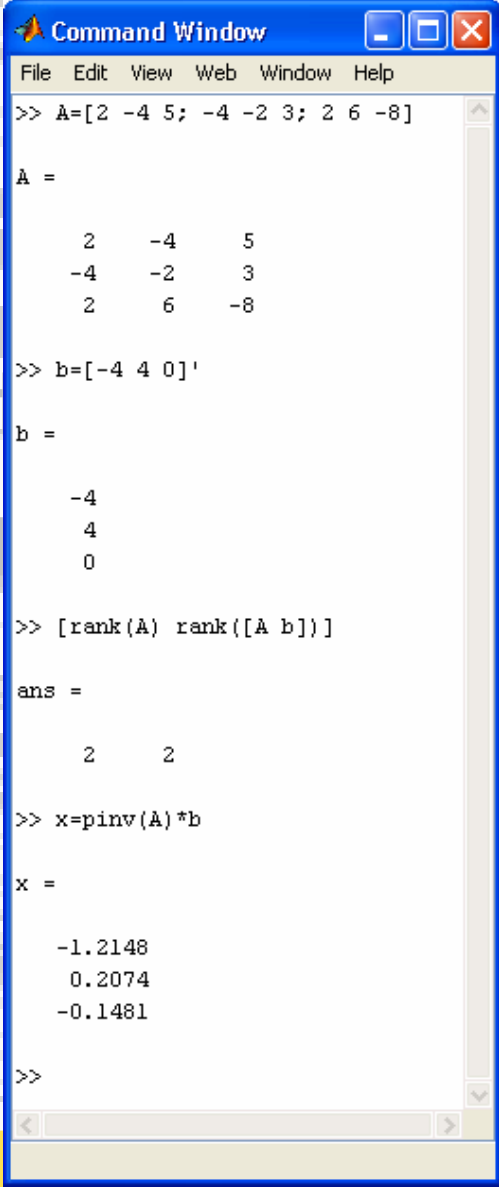
Risoluzione di sistemi di equazioni Esempio

Si considera il seguente sistema con rango minore:

$$2x - 4y + 5z = -4$$

$$-4x - 2y + 3z = 4$$

$$2x + 6y - 8z = 0$$



```
Command Window
File Edit View Web Window Help
>> A=[2 -4 5; -4 -2 3; 2 6 -8]
A =
     2     -4     5
    -4     -2     3
     2      6    -8
>> b=[-4 4 0]
b =
    -4
     4
     0
>> [rank(A) rank([A b])]
ans =
     2     2
>> x=pinv(A)*b
x =
   -1.2148
    0.2074
   -0.1481
>>
```

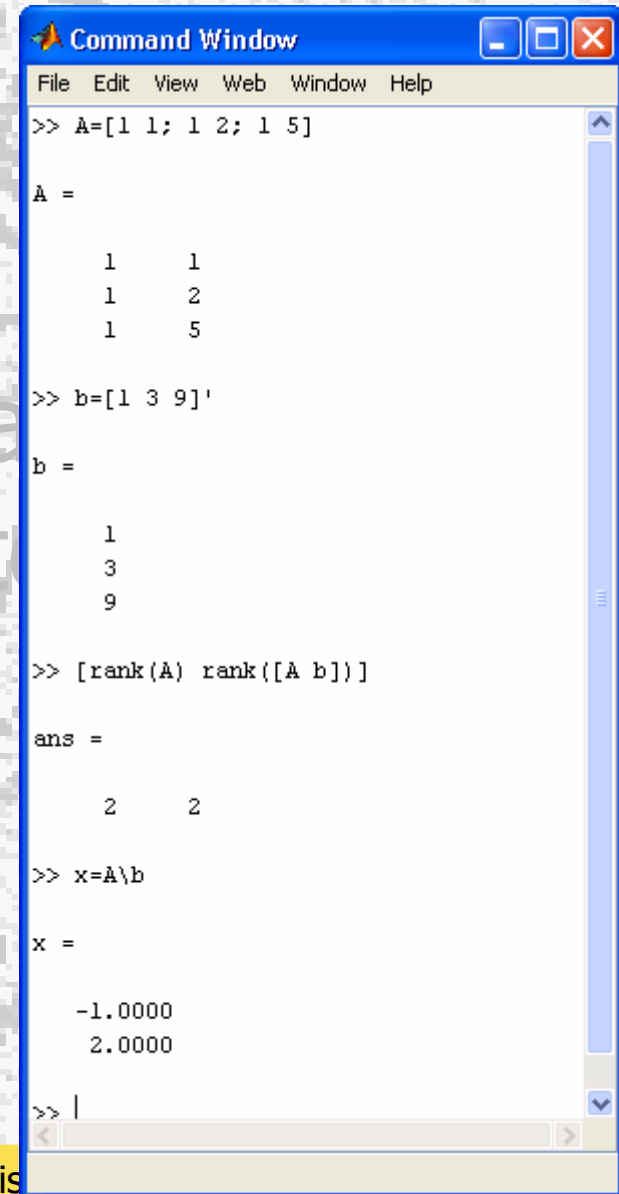
Risoluzione di sistemi di equazioni Esempio

Si considera il seguente sistema:

$$x+y=1$$

$$x+2y=3$$

$$x+5y=9$$



```
Command Window
File Edit View Web Window Help
>> A=[1 1; 1 2; 1 5]
A =
     1     1
     1     2
     1     5
>> b=[1 3 9]
b =
     1
     3
     9
>> [rank(A) rank([A b])]
ans =
     2     2
>> x=A\b
x =
    -1.0000
     2.0000
>> |
```

Numeri Complessi

MATLAB gestisce in modo automatico l'algebra dei numeri complessi.

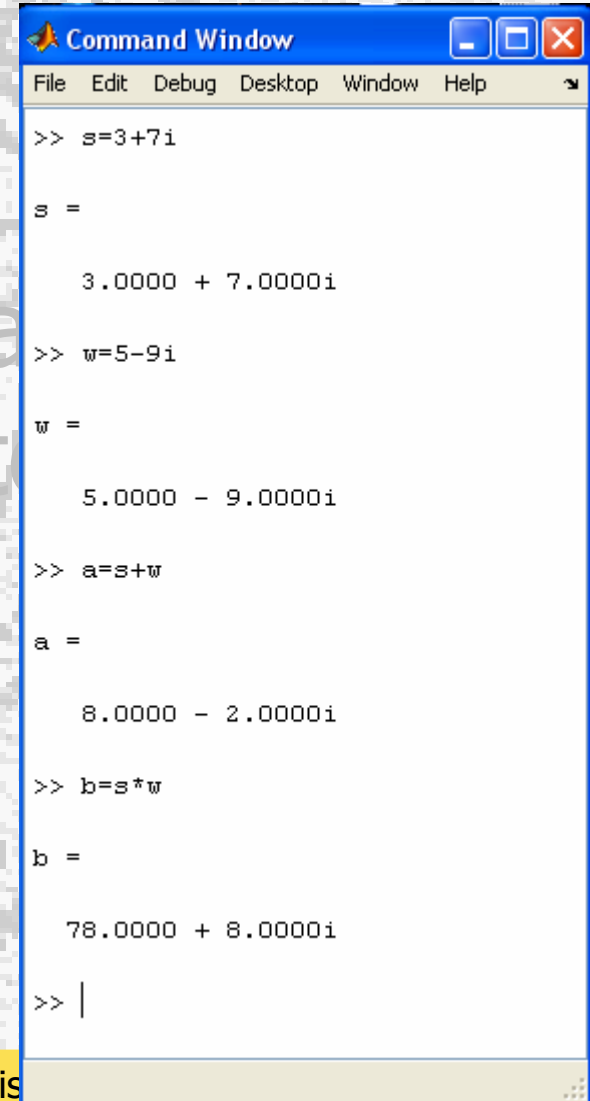
Sono utilizzati i simboli **i** e **j** per specificare la parte immaginaria.

Osservazione

Non serve inserire (*) tra **i** (o **j**) e il numero.

Se **i** (o **j**) è stato utilizzato come variabile, occorre ridefinirlo come operatore complesso:

```
>> i = sqrt(-1)
```



```
Command Window
File Edit Debug Desktop Window Help
>> s=3+7i

s =

    3.0000 + 7.0000i

>> w=5-9i

w =

    5.0000 - 9.0000i

>> a=s+w

a =

    8.0000 - 2.0000i

>> b=s*w

b =

   78.0000 + 8.0000i

>> |
```

Funzioni Numeri Complessi

Le funzioni disponibili sono:

- Coniugato (**conj**)
- Parte reale (**real**)
- Parte immaginaria (**imag**)
- Modulo (**abs**)
- Argomento (**angle**)

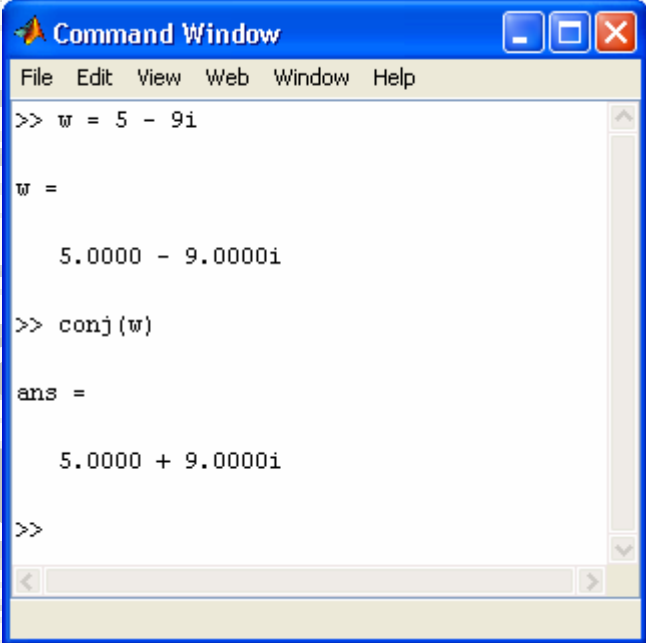
Coniugato

- Funzione **conj**

Calcola il coniugato di un numero complesso.

Sintassi

var_a = conj(var_b)



```
Command Window
File Edit View Web Window Help
>> w = 5 - 9i

w =

    5.0000 - 9.0000i

>> conj(w)

ans =

    5.0000 + 9.0000i

>>
```

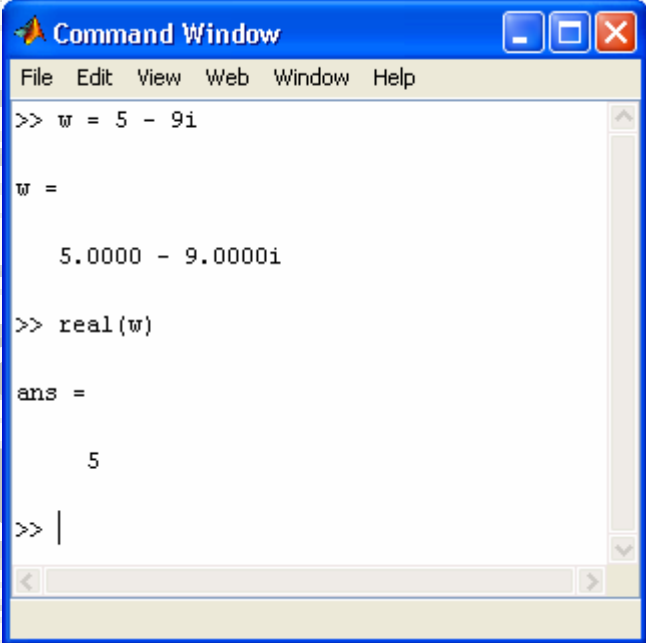
Parte reale

- Funzione **real**

Estrae la parte reale di un numero complesso.

Sintassi

var_a = real(var_b)



```
Command Window
File Edit View Web Window Help
>> w = 5 - 9i

w =

    5.0000 - 9.0000i

>> real(w)

ans =

     5

>> |
```

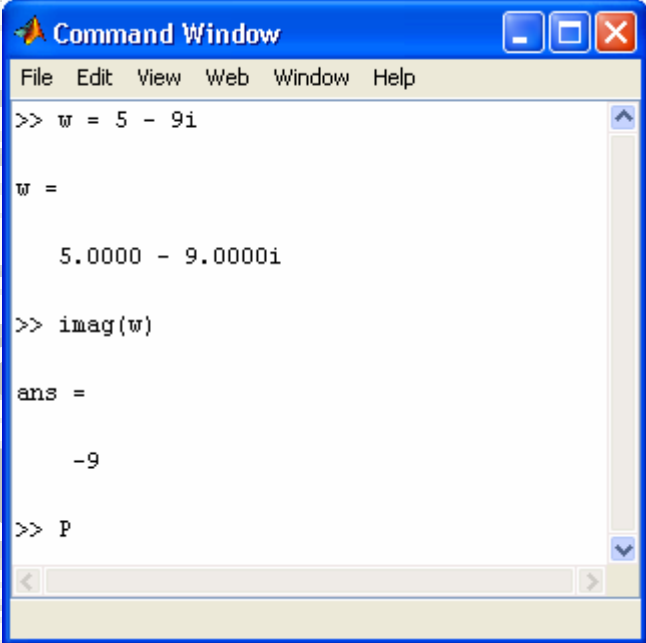
Parte immaginaria

- Funzione **imag**

Estrae la parte immaginaria di un numero complesso.

Sintassi

var_a = imag(var_b)

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area shows the following commands and outputs:

```
>> w = 5 - 9i  
  
w =  
  
    5.0000 - 9.0000i  
  
>> imag(w)  
  
ans =  
  
    -9  
  
>> P
```

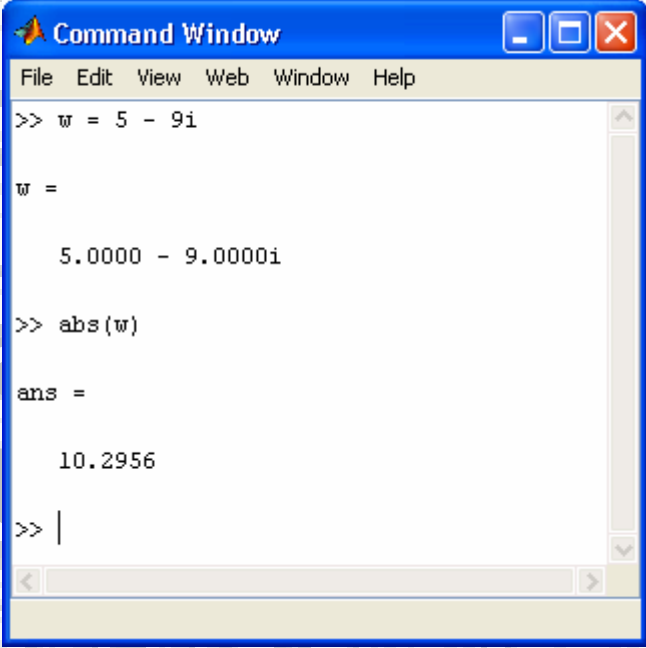
Modulo

- Funzione **abs**

Calcola il modulo di un numero complesso.

Sintassi

var_a = abs(var_b)



```
Command Window
File Edit View Web Window Help
>> w = 5 - 9i

w =

    5.0000 - 9.0000i

>> abs(w)

ans =

    10.2956

>> |
```

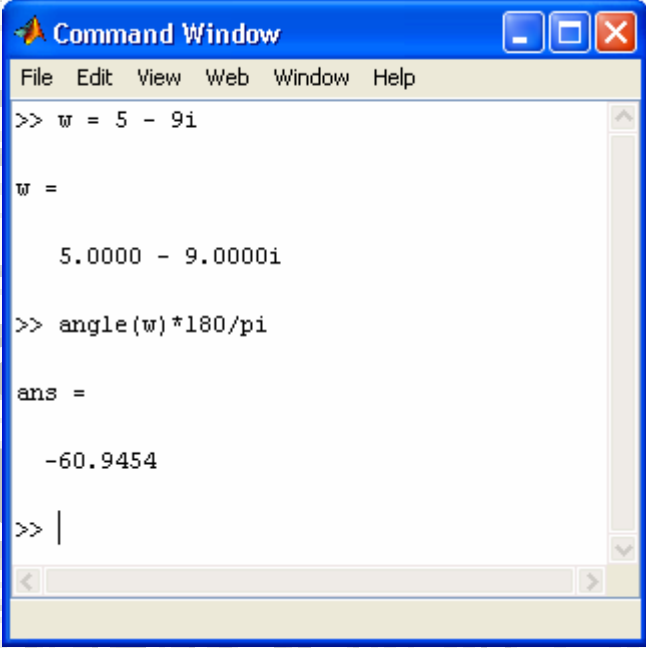
Argomento

- Funzione **angle**

Calcola l'argomento (espresso in radianti) di un numero complesso.

Sintassi

$var_a = angle(var_b)$



```
Command Window
File Edit View Web Window Help
>> w = 5 - 9i

w =

    5.0000 - 9.0000i

>> angle(w)*180/pi

ans =

   -60.9454

>> |
```

Esercizi con i numeri complessi

Se $x = -5 - 8i$, $y = 10 - 5i$ e $z = i^3$ calcolare

xy ;

$z + x/y$;

$|x|$;

$(zy)^2$

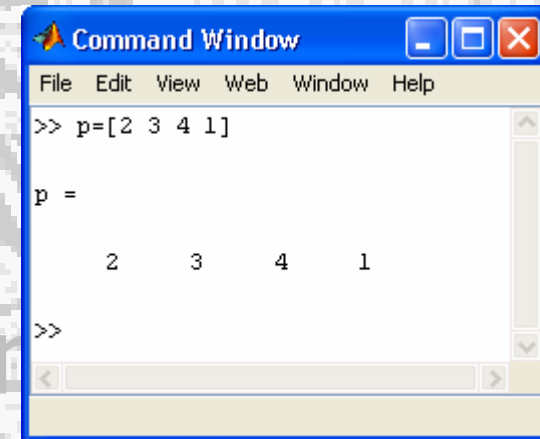
Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

Polinomi

I polinomi in MATLAB si rappresentano come vettori contenenti i coefficienti in ordine decrescente.

Esempio

$$P(s)=2s^3+3s^2+4s+1$$



```
Command Window
File Edit View Web Window Help
>> p=[2 3 4 1]

p =

    2    3    4    1

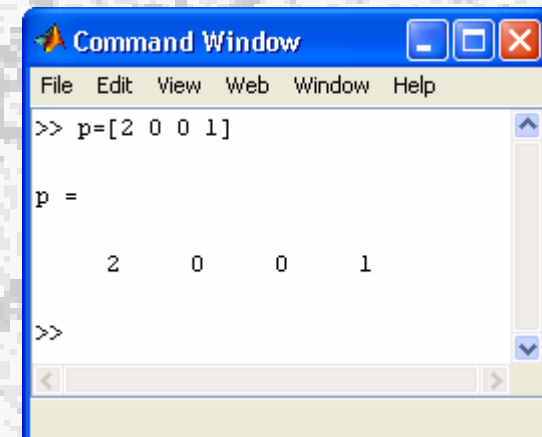
>>
```

Osservazione

Vanno inclusi ovviamente anche i coefficienti nulli.

Esempio

$$P(s)=2s^3+1$$



```
Command Window
File Edit View Web Window Help
>> p=[2 0 0 1]

p =

    2    0    0    1

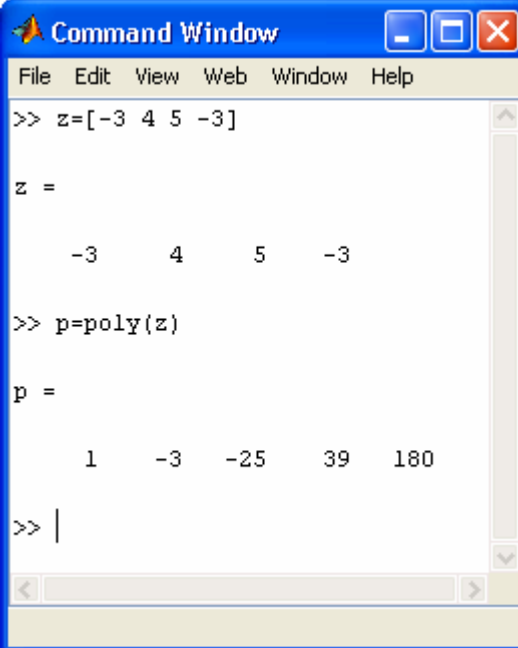
>>
```

Costruzione del polinomio note le radici

Si utilizza la funzione **poly**

Sintassi

var_p = poly([z1, z2, z3..])



```
Command Window
File Edit View Web Window Help
>> z=[-3 4 5 -3]

z =

    -3     4     5    -3

>> p=poly(z)

p =

     1     -3    -25     39    180

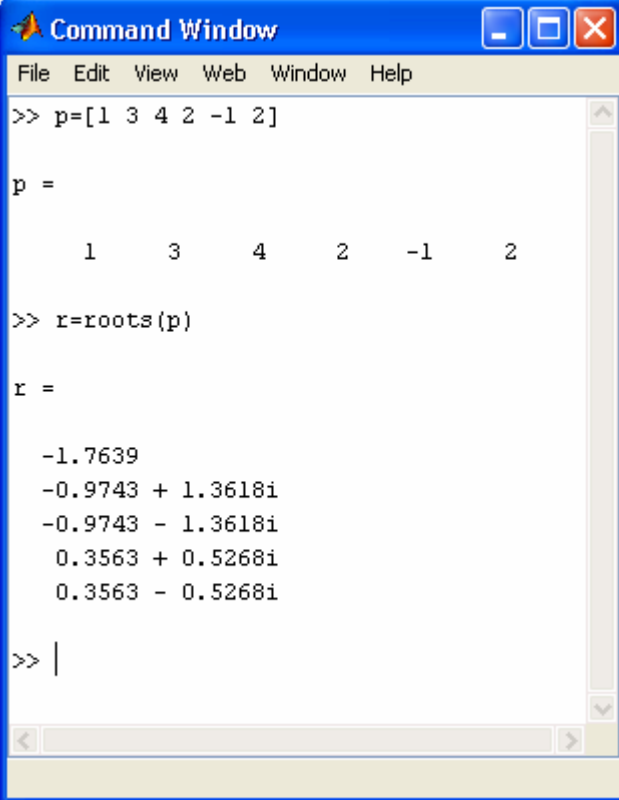
>> |
```

Radici del polinomio

Si utilizza la funzione **roots**

Sintassi

vett_radici = roots(var_pol)



```
Command Window
File Edit View Web Window Help
>> p=[1 3 4 2 -1 2]

p =

    1    3    4    2   -1    2

>> r=roots(p)

r =

-1.7639
-0.9743 + 1.3618i
-0.9743 - 1.3618i
 0.3563 + 0.5268i
 0.3563 - 0.5268i

>> |
```

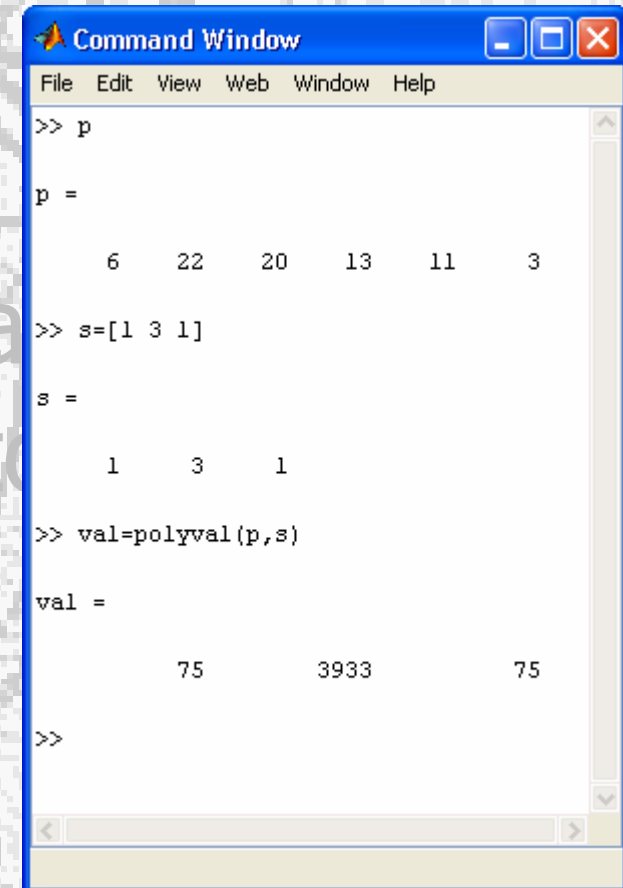
Valore del polinomio in un punto (o in un insieme di punti)

Si utilizza la funzione **polyval**

Sintassi

var_a = polyval(var_pol, s)

dove s è uno scalare o un array.

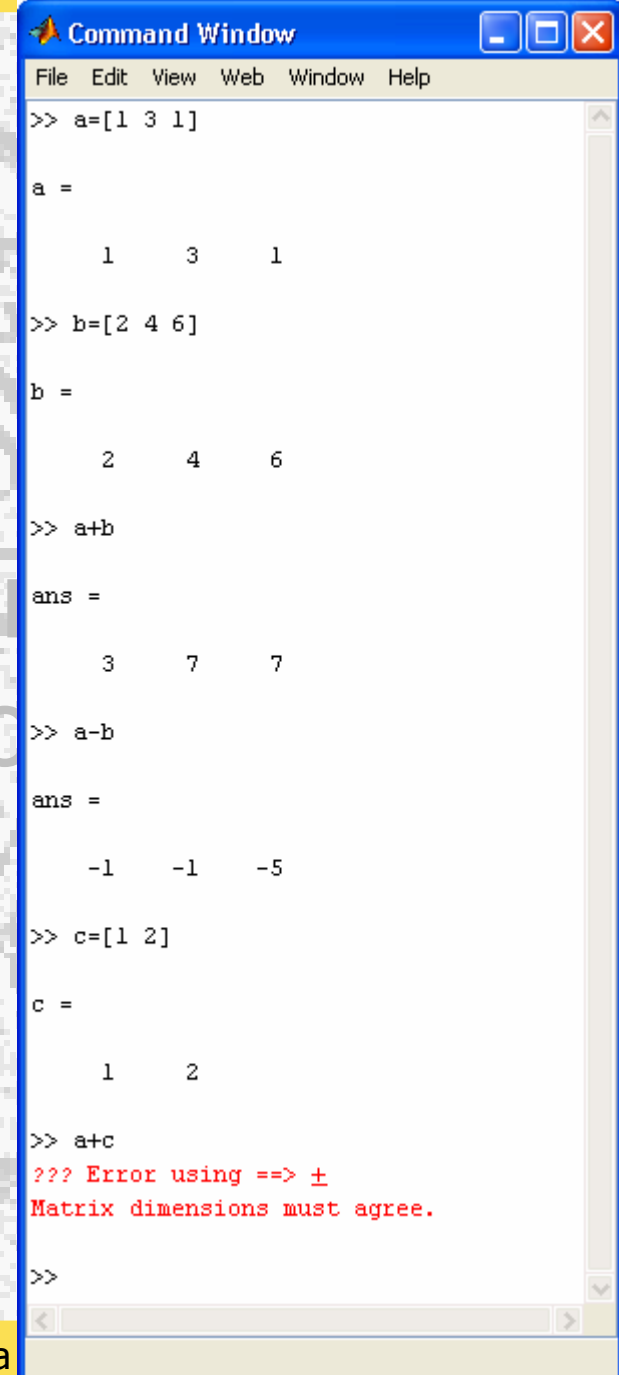


```
Command Window
File Edit View Web Window Help
>> p
p =
    6    22    20    13    11     3
>> s=[1 3 1]
s =
     1     3     1
>> val=polyval(p,s)
val =
    75   3933    75
>>
```

Somma algebrica di polinomi

Si utilizzano gli operatori $+$ e $-$.

E' necessario che i due polinomi abbiano la stessa dimensione.

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. The main area shows the following commands and outputs:

```
>> a=[1 3 1]

a =

     1     3     1

>> b=[2 4 6]

b =

     2     4     6

>> a+b

ans =

     3     7     7

>> a-b

ans =

    -1    -1    -5

>> c=[1 2]

c =

     1     2

>> a+c

??? Error using ==> +
Matrix dimensions must agree.

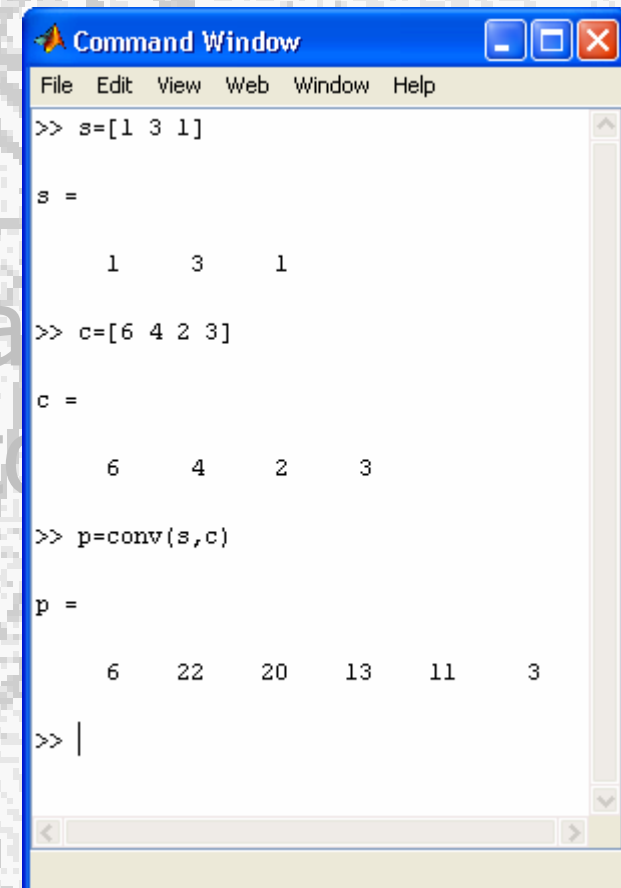
>>
```

Prodotto di polinomi

Si utilizza la funzione **conv**

Sintassi

var_c = conv(p1, p2)



```
Command Window
File Edit View Web Window Help
>> s=[1 3 1]
s =
    1     3     1
>> c=[6 4 2 3]
c =
    6     4     2     3
>> p=conv(s,c)
p =
    6    22    20    13    11     3
>> |
```

Quoziente di polinomi

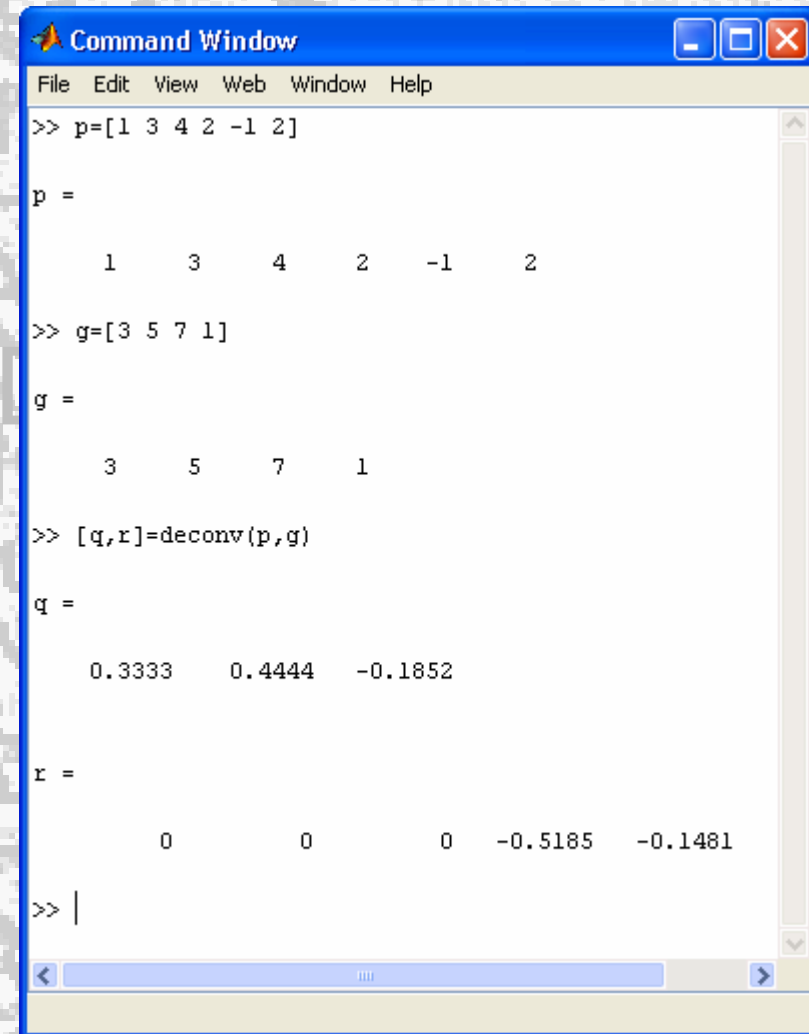
Si utilizza la funzione **deconv**

Sintassi

[quoz, resto] =

deconv(divid., divisore)

Università di Pisa
Sistemi Elettrici e



```
Command Window
File Edit View Web Window Help
>> p=[1 3 4 2 -1 2]

p =

    1    3    4    2   -1    2

>> g=[3 5 7 1]

g =

    3    5    7    1

>> [q,r]=deconv(p,g)

q =

    0.3333    0.4444   -0.1852

r =

    0    0    0   -0.5185   -0.1481

>> |
```

Derivata di un polinomo

Si utilizza la funzione **polyder**

Sintassi

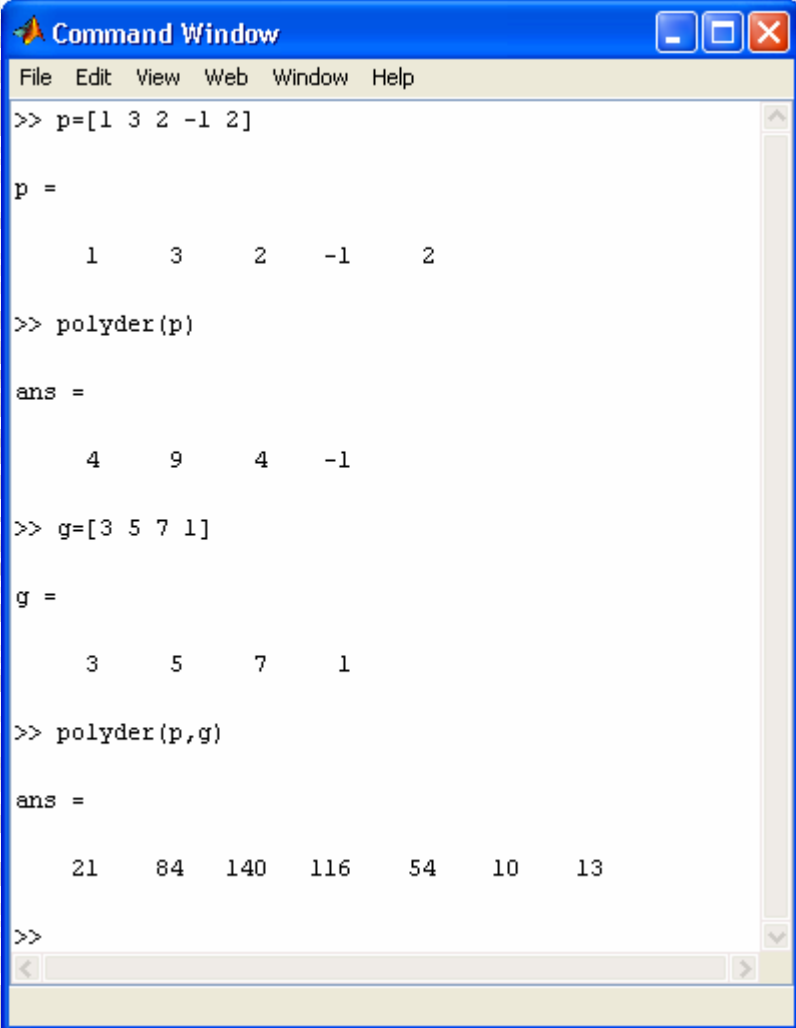
deriv = polyder(p)

derivata di un prodotto

deriv = polyder(p1, p2)

derivata di un quoziente

[n,d] = polyder(p1, p2)



```
Command Window
File Edit View Web Window Help

>> p=[1 3 2 -1 2]

p =

    1    3    2   -1    2

>> polyder(p)

ans =

    4    9    4   -1

>> g=[3 5 7 1]

g =

    3    5    7    1

>> polyder(p,g)

ans =

   21   84  140  116   54   10   13

>>
```

Scomposizione in fratti semplici

Assegnata una funzione razionale fratta $b(s)/a(s)$ con sviluppo in fratti semplici si intende lo sviluppo:

$$\frac{b(s)}{a(s)} = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \dots + \frac{r_n}{s - p_n} + k(s)$$

Si utilizza la funzione **residue**

Sintassi

$[r, p, k] = \text{residue}(b, a)$

Funziona anche in modo inverso, ossia dati i residui e le radici, la funzione residue fornisce i polinomi $a(s)$ e $b(s)$.

Sintassi

$[a, b] = \text{residue}(r, p, k)$

Scomposizione in fratti semplici Esempio

Antitrasformare la seguente trasformata di Laplace:

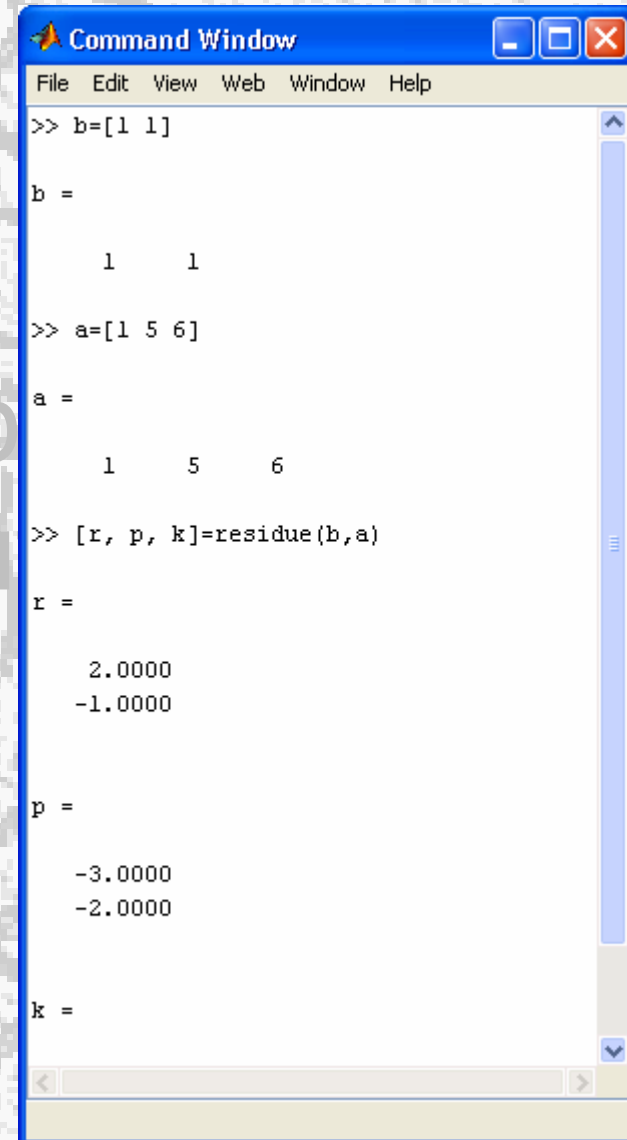
$$F(s) = \frac{s+1}{s^2+5s+6}$$

Scomponendo in fratti semplici si ha:

$$F(s) = \frac{2}{s+3} - \frac{1}{s+2}$$

Da cui:

$$F(t) = 2e^{-3t} - e^{-2t}$$



```
Command Window
File Edit View Web Window Help
>> b=[1 1]
b =
    1    1
>> a=[1 5 6]
a =
    1    5    6
>> [r, p, k]=residue(b,a)
r =
    2.0000
   -1.0000
p =
   -3.0000
   -2.0000
k =
```

Esercizi sui polinomi

Calcolare il seguente prodotto

$$(10x^3 - 9x^2 - 6x + 12)(5x^3 - 4x^2 - 12x + 8)$$

Calcolare il seguente quoziente e resto

$$(14x^3 - 9x^2 + 3x + 9) / (5x^2 + 7x - 4)$$

Calcolare la derivata per $x=5$

$$(8x^2 + 3x + 9) / (5x^3 + 10x^2 - 7x - 4)$$

Antitrasformare la seguente funzione

$$F(s) = \frac{s+1}{s^3 + 5s^2 + 7s + 3}$$

Interpolazione polinomiale

Dato un vettore x contenente i valori della variabile indipendente e un vettore y con i rispettivi valori da interpolare la funzione **polyfit** fornisce un polinomio $p(x)$ di grado n che interpola le coppie (x_i, y_i) con il metodo dei minimi quadrati.

Sintassi

var_p=polyfit(x, y, n)

Interpolazione Esempio

```
Command Window
File Edit View Web Window Help
>> x=1:5

x =

     1     2     3     4     5

>> y=[1 3 2 1.4 1.8]

y =

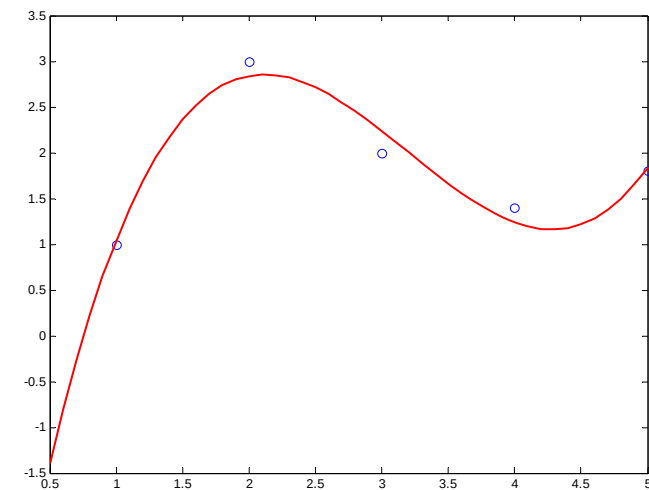
     1.0000     3.0000     2.0000     1.4000

>> p=polyfit(x,y,3)

p =

     0.3333    -3.2000     9.0667    -5.1600

>> |
```



Metodi di interpolazione

Sono assegnati i vettori x e y contenenti rispettivamente i valori della variabile indipendente e di quella dipendente di una funzione $y=f(x)$. Dato un vettore x_i , la funzione **interp1** fornisce un vettore y_i costituito dai valori associati alle ascisse in x_i . I valori y_i sono ottenuti mediante interpolazione.

Sintassi

$y_i = \text{interp1}(x, y, x_i, [\text{metodo}])$

metodo

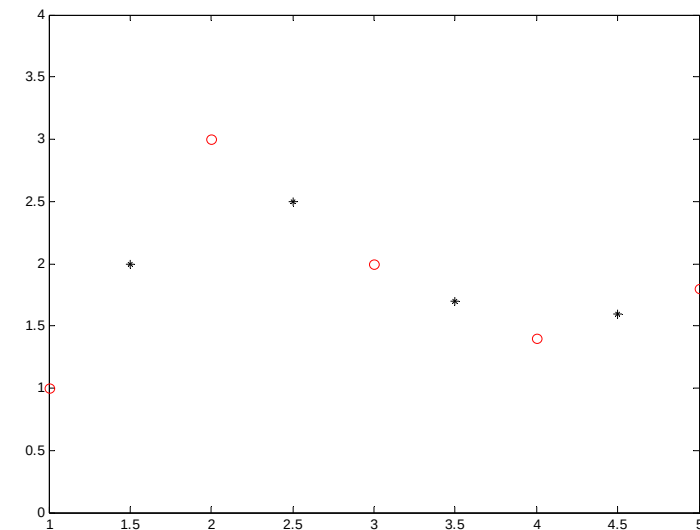
'linear' (default), 'spline', 'cubic'

Osservazione

x e x_i devono essere monotoni crescenti

Interpolazione Esempio

```
Command Window
File Edit View Web Window Help
>> x=1:5
x =
    1    2    3    4    5
>> y=[1 3 2 1.4 1.8]
y =
    1.0000    3.0000    2.0000    1.4000    1.8000
>> xi=1.5:4.5
xi =
    1.5000    2.5000    3.5000    4.5000
>> yi=interp1(x,y,xi)
yi =
    2.0000    2.5000    1.7000    1.6000
>> |
```



Interpolazione bidimensionale

L'istruzione **interp2** funziona come `interp1`, ma su tabelle bidimensionali.

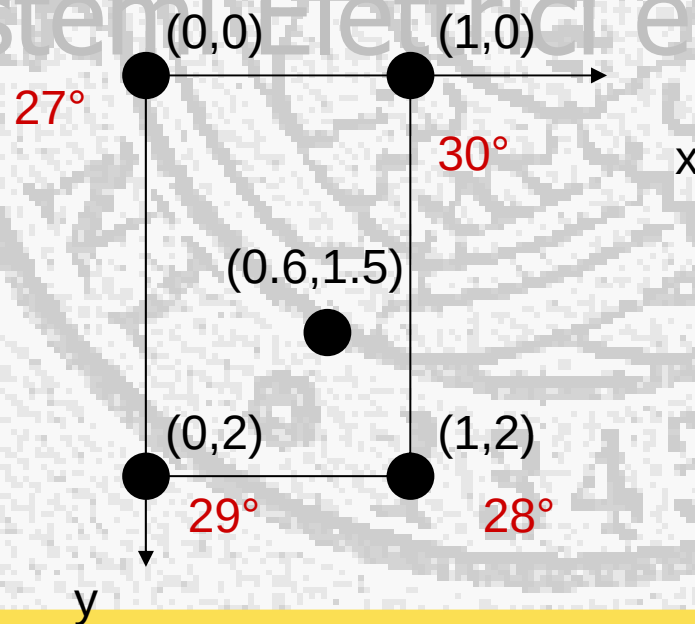
Sintassi

$z_i = \text{interp1}(x, y, z, x_i, y_i, [\text{metodo}])$

Interpolazione bidimensionale Esempio

E' nota la temperatura di quattro località. Queste località si trovano ai vertici di un rettangolo largo 1 km e lungo 2 km. Associato alla prima località l'origine (0,0) di un sistema di coordinate cartesiane, le altre località avranno le coordinate (1,0), (1,2) e (0,2).

Si vuole stimare la temperatura nel punto di coordinate (0.6, 1.5).



```
Command Window
File Edit View Web Window Help
>> x=[0,1];
>> y=[0,2];
>> z=[27 30; 29 28]

z =
    27    30
    29    28

>> zi=interp2(x,y,z,0.6,1.5)

zi =

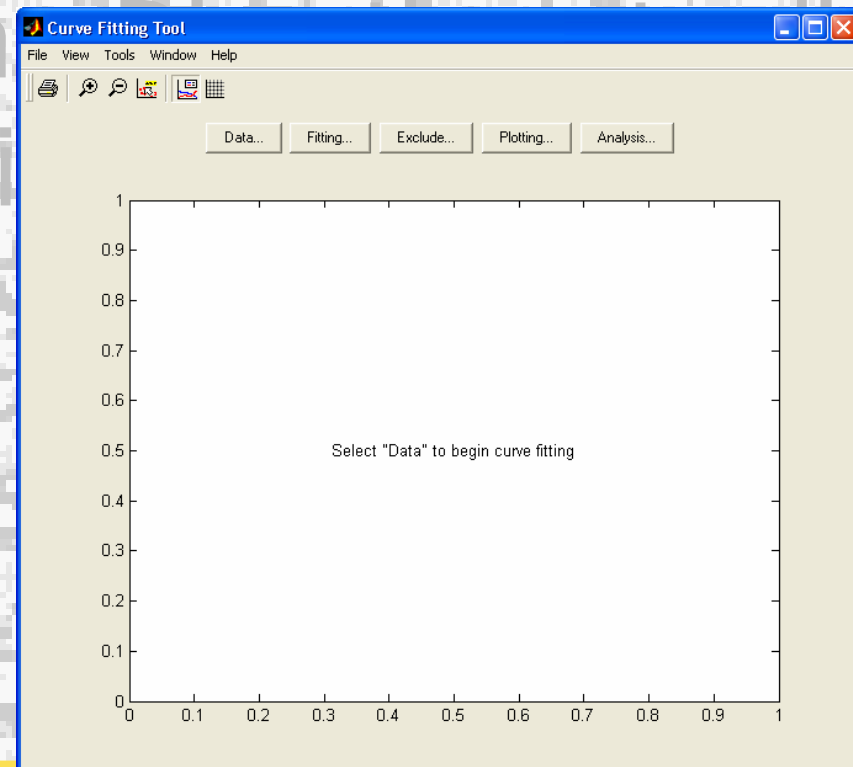
    28.5000

>>
```

Il Curve Fitting Tool

E' una raccolta di funzioni specifiche per l'interpolazione dei dati. Presenta un'interfaccia grafica (GUI) che permette un facile impiego di tali funzioni.

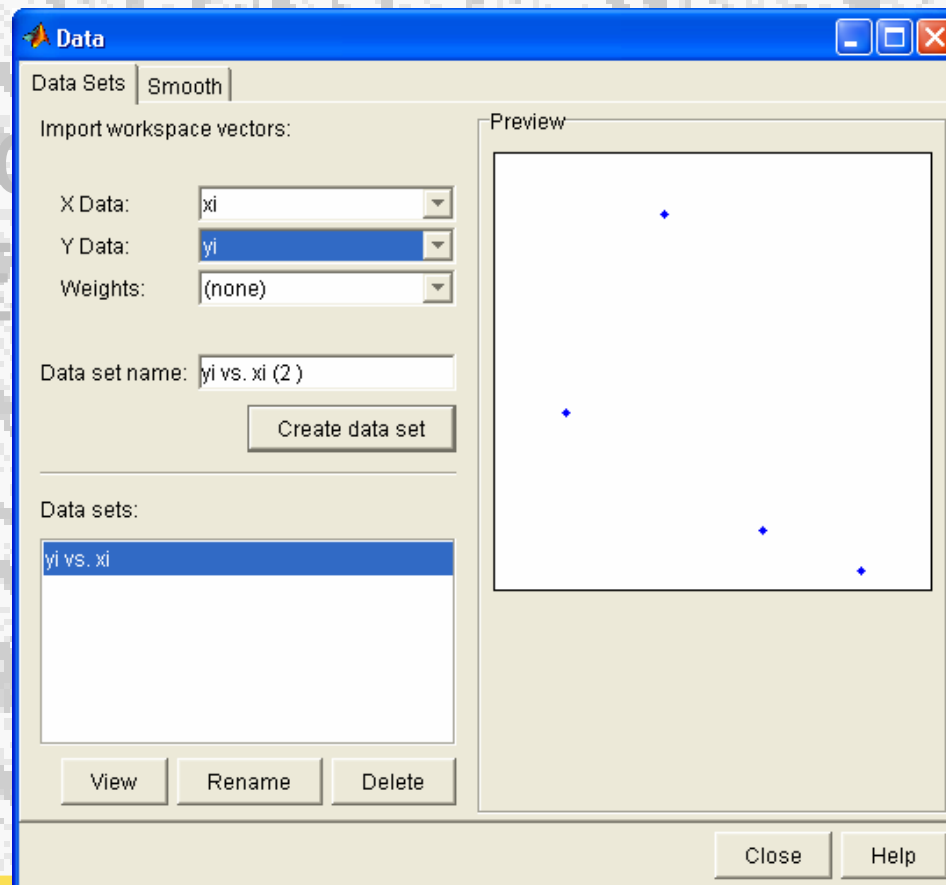
Si avvia dal Workspace con la funzione **cftool**.



Il Curve Fitting Tool

Importazione dati.

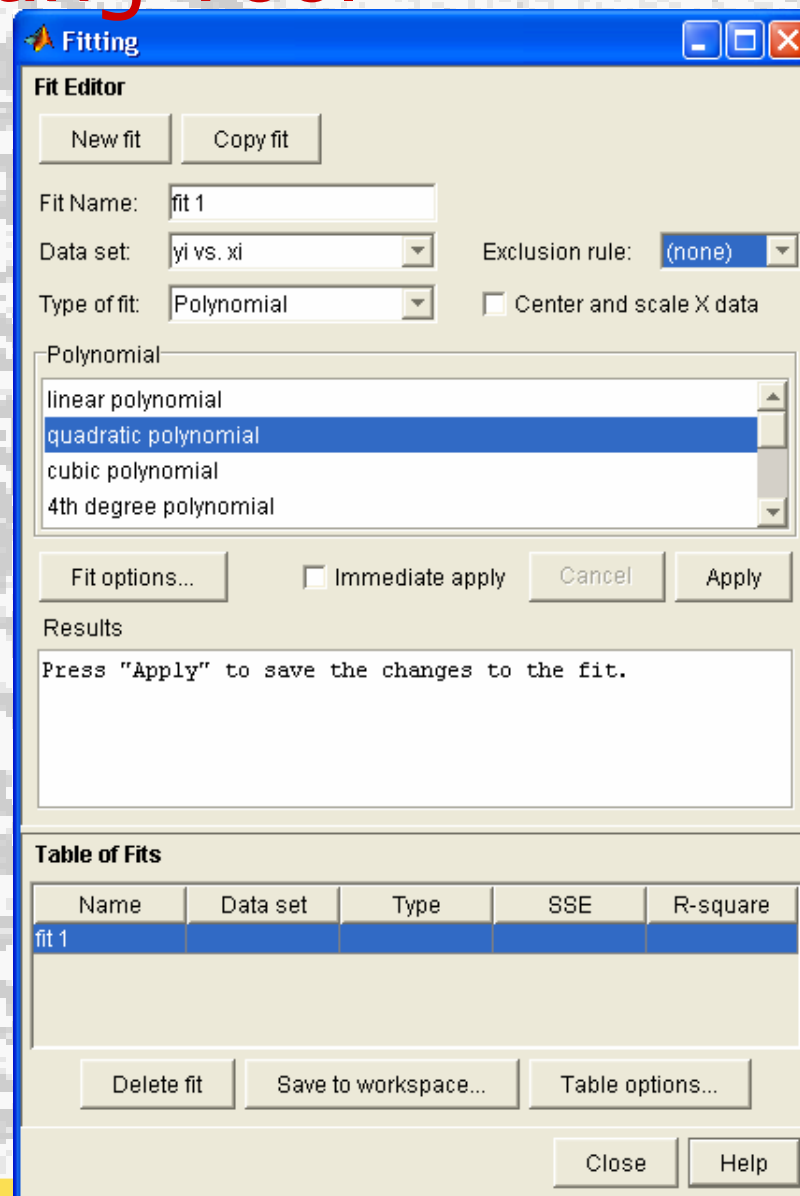
Dal tasto **Data** si accede alla sezione di importazione dati. Questi devono essere già presenti nel Workspace.



Il Curve Fitting Tool

Fitting dati

Dal tasto **Fitting** si accede alla sezione omonima. Si può definire il tipo di modello ed eseguire il fitting, premendo **Apply**.



Il Curve Fitting Tool

Il tool fornisce i risultati in termini dei coefficienti del modello e in forma grafica.

Università di Pisa
Sistemi Elettrici e

Parametri del modello

Fit Editor

New fit Copy fit

Fit Name: fit 1

Data set: yi vs. xi Exclusion rule: (none)

Type of fit: Polynomial ☐ Center and scale X data

Polynomial

linear polynomial
quadratic polynomial
cubic polynomial
4th degree polynomial

Fit options... ☐ Immediate apply Cancel Apply

Results

Linear model Poly2:

$$f(x) = p1 \cdot x^2 + p2 \cdot x + p3$$
 Coefficients (with 95% confidence bounds):
 p1 = -0.15 (-2.991, 2.691)
 p2 = 0.7 (-16.54, 17.94)

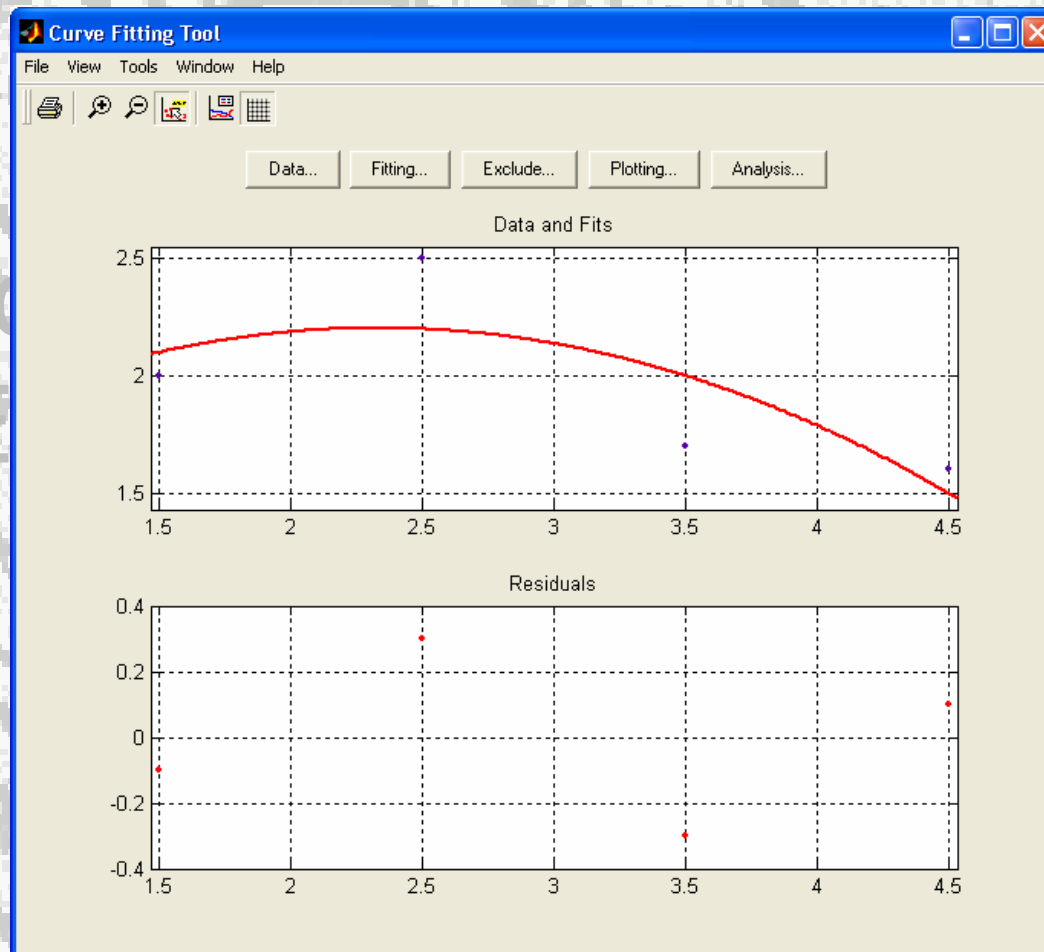
Table of Fits

Name	Data set	Type	SSE	R-square
fit 1	yi vs. xi	Polynomial	0.2	0.59184

Delete fit Save to workspace... Table options... Close Help

Il Curve Fitting Tool

Il tool_fornisce i risultati in termini dei coefficienti del modello e in forma grafica.



Le stringhe

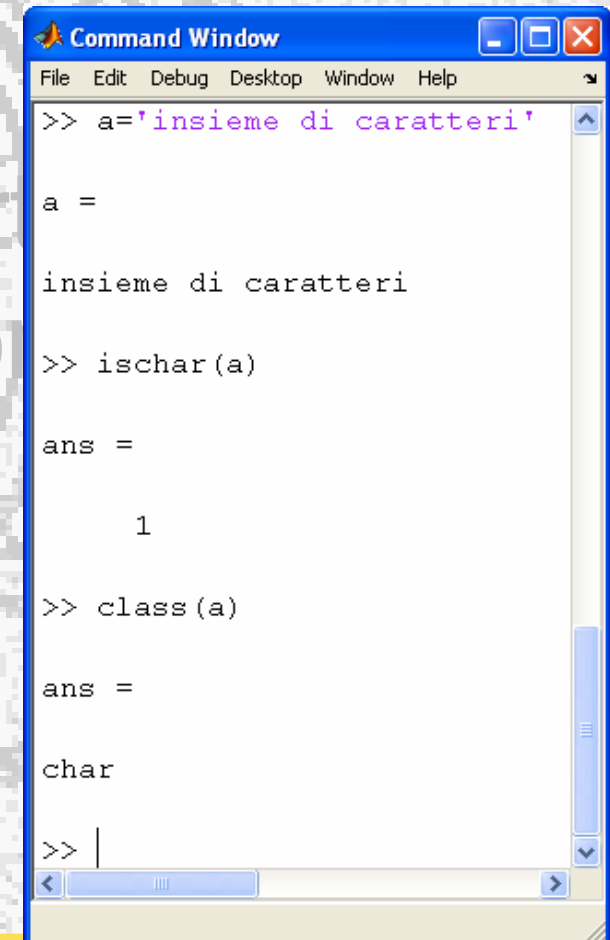
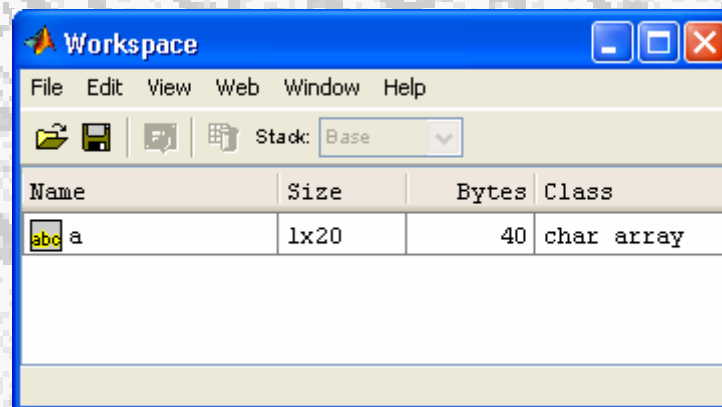
In Matlab con **stringa** si intende un array di caratteri.

Si crea con un'operazione di assegnamento, raccogliendo i caratteri tra apici.

Sintassi

Nome_var = 'stringa'

Con le funzioni **class** e **ischar** è possibile verificare il tipo di dato memorizzato.

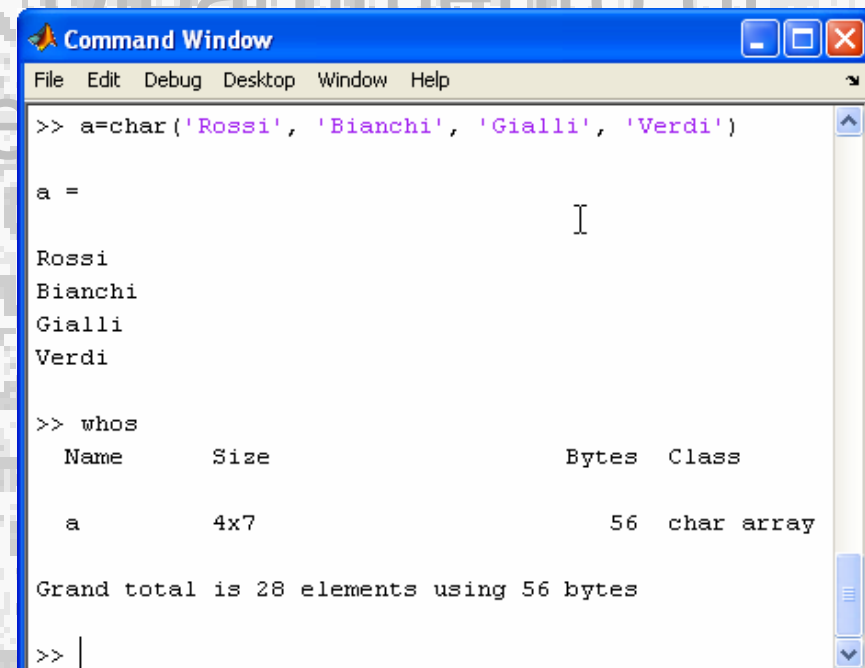


Matrici di stringhe

E' possibile creare matrici di stringhe. Si ha allora una stringa per riga. E' necessario che tutte le stringhe abbiano la stessa lunghezza. Si usa la funzione **char**.

Sintassi

Nome_var = char('stringa 1', 'stringa 2', ...)



```
>> a=char('Rossi', 'Bianchi', 'Gialli', 'Verdi')

a =

Rossi
Bianchi
Gialli
Verdi

>> whos

Name      Size      Bytes  Class

a         4x7       56     char array

Grand total is 28 elements using 56 bytes

>> |
```

Funzioni per le stringhe

strcat unione di stringhe

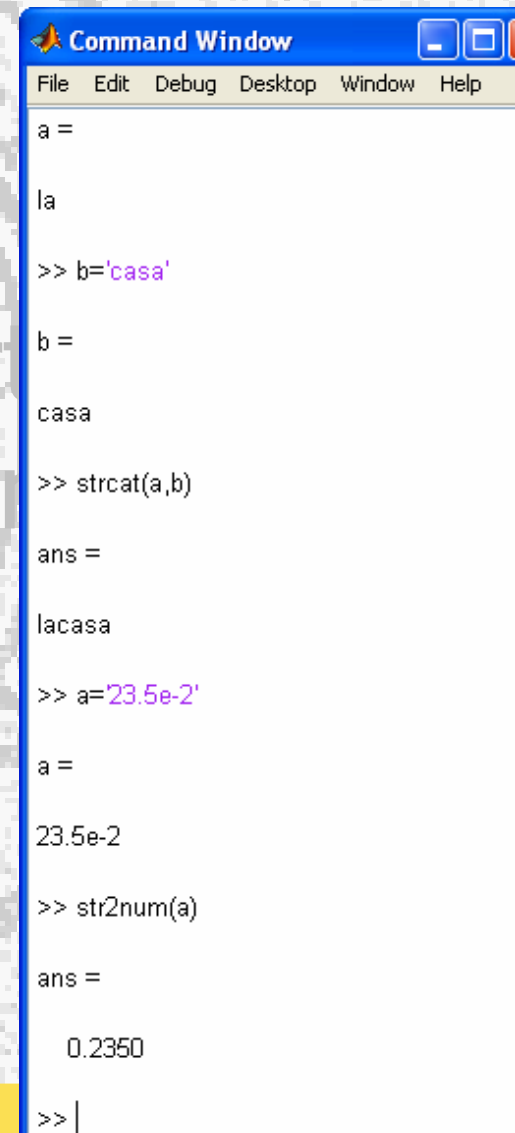
Sintassi

Var = strcat(s1, s2, ...)

str2num converte il contenuto della stringa nel numero corrispondente

Sintassi

var = str2num(var_s)



```
Command Window
File Edit Debug Desktop Window Help

a =

la

>> b='casa'

b =

casa

>> strcat(a,b)

ans =

lacasa

>> a='23.5e-2'

a =

23.5e-2

>> str2num(a)

ans =

0.2350

>> |
```

Funzioni per le stringhe

strcmp confronto tra due stringhe

Sintassi

Var = strcmp(s1, s2)

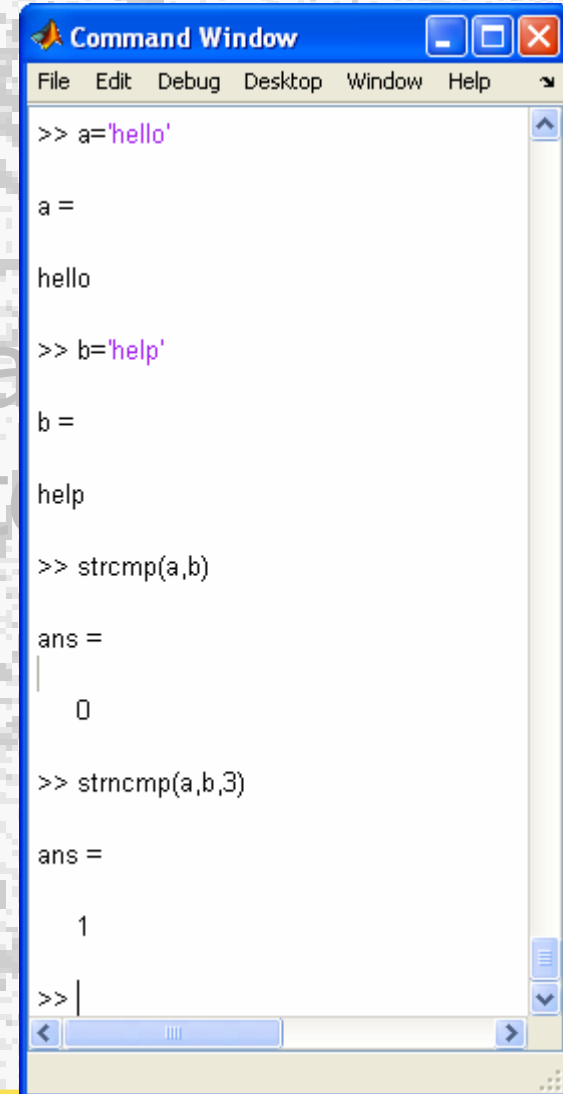
il risultato è 1 se le due stringhe sono uguali, 0 se falso

strncmp confronto tra i primi n caratteri di due stringhe

Sintassi

Var = strncmp(s1, s2, n)

il risultato è 1 se vero, 0 se falso

A screenshot of a MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. The main area of the window contains the following text:

```
>> a='hello'

a =

hello

>> b='help'

b =

help

>> strcmp(a,b)

ans =

    0

>> strncmp(a,b,3)

ans =

    1

>> |
```

Funzioni per le stringhe

== confronta i singoli caratteri delle due stringhe (devono essere della stessa lunghezza)

Sintassi

s1 == *s2*

il risultato è un vettore numerico della stessa lunghezza i cui elementi sono 1 (se i caratteri corrispondenti sono uguali) e 0 (se sono diversi)

isletter indica quali sono i caratteri all'interno di una stringa



```

Command Window
File Edit Debug Desktop Window Help
>> a='castello'
a =
castello
>> b='cestello'
b =
cestello
>> a==b
ans =
     1     0     1     1     1     1     1     1
>> a='stanza 342'
a =
stanza 342
>> isletter(a)
ans =
     1     1     1     1     1     1     0     0     0     0
>>
  
```

Funzioni per le stringhe

findstr individua la posizione di una porzione di stringa (s2) all'interno di una stringa (s1)

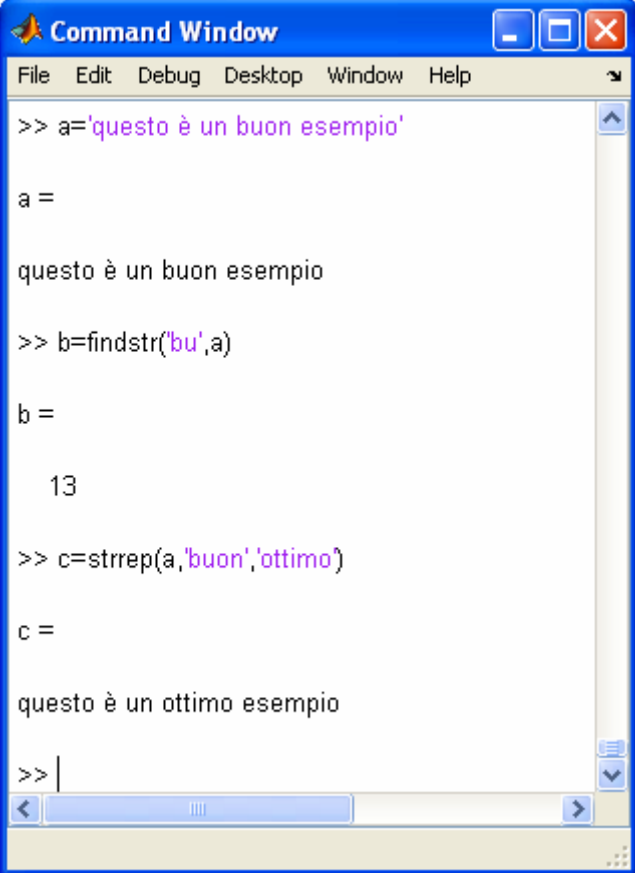
Sintassi

Var = findstr(s2, s1)

strrep in una stringa (s1) ne sostituisce una parte (s2) con una nuova stringa (s3)

Sintassi

var = strrep(s1, s2, s3)

A screenshot of the MATLAB Command Window. The window has a blue title bar with the text 'Command Window' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. The main area shows the following commands and outputs:

```
>> a='questo è un buon esempio'

a =

questo è un buon esempio

>> b=findstr('bu',a)

b =

    13

>> c=strrep(a,'buon','ottimo')

c =

questo è un ottimo esempio

>> |
```

Funzioni per le stringhe

int2str converte un intero in una stringa

Sintassi

Var = int2str(var_n)

num2str converte un numero in una stringa.

E' possibile specificare il numero di cifre (n) della stringa di uscita

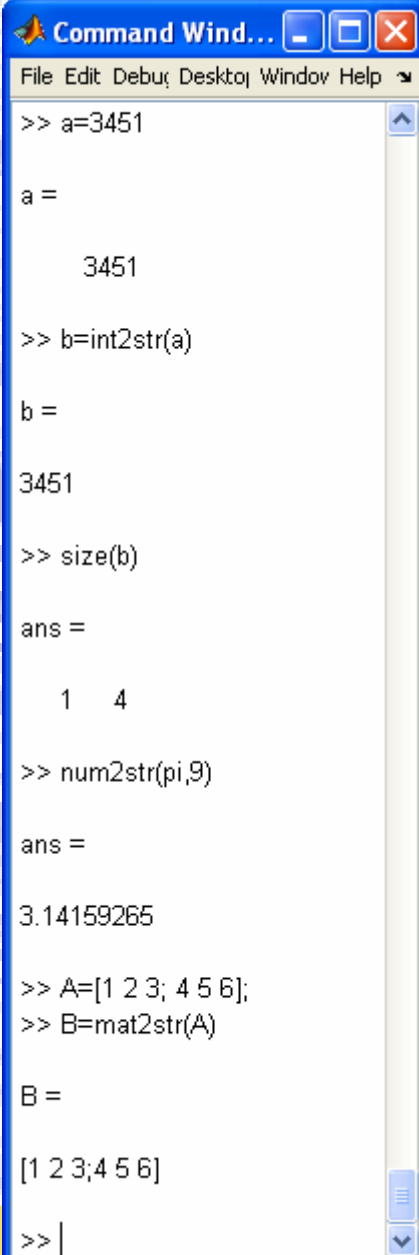
Sintassi

var = num2str(s, n)

mat2str converte una matrice di numeri in una stringa

Sintassi

Var = mat2str(var_n)

A screenshot of the MATLAB Command Window. The window title is 'Command Wind...'. The menu bar includes 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. The command history shows the following interactions: 1. User enters '>> a=3451'. The response is 'a =' followed by '3451' on the next line. 2. User enters '>> b=int2str(a)'. The response is 'b =' followed by '3451' on the next line. 3. User enters '>> size(b)'. The response is 'ans =' followed by '1 4' on the next line. 4. User enters '>> num2str(pi,9)'. The response is 'ans =' followed by '3.14159265' on the next line. 5. User enters '>> A=[1 2 3; 4 5 6];'. 6. User enters '>> B=mat2str(A)'. The response is 'B =' followed by '[1 2 3;4 5 6]' on the next line. The prompt '>> |' is visible at the bottom of the window.

```
>> a=3451
a =
    3451

>> b=int2str(a)
b =
    3451

>> size(b)
ans =
     1     4

>> num2str(pi,9)
ans =
3.14159265

>> A=[1 2 3; 4 5 6];
>> B=mat2str(A)
B =
[1 2 3;4 5 6]

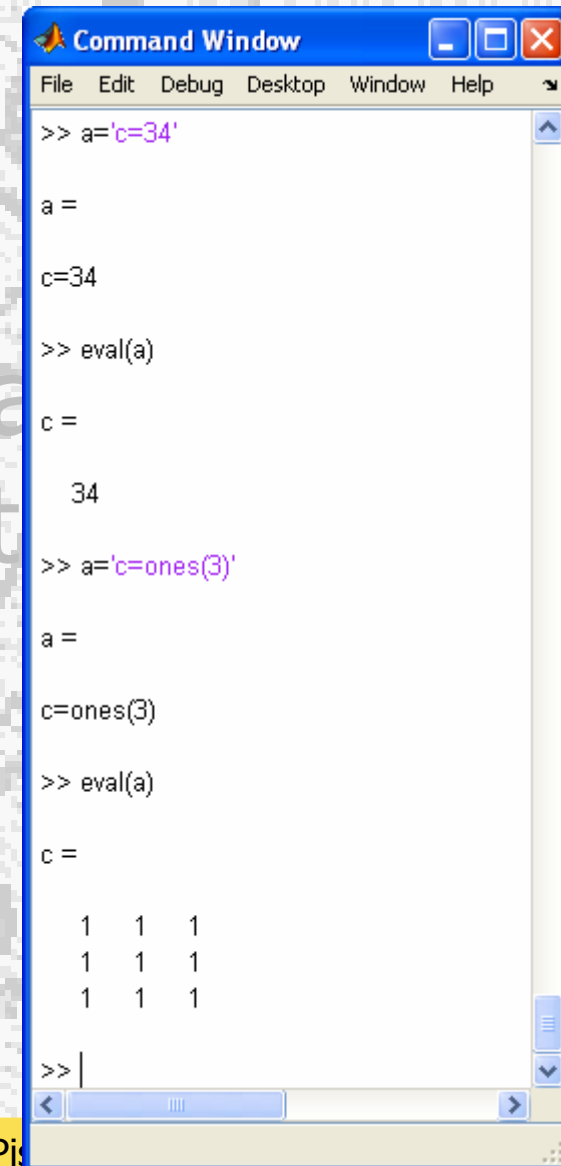
>> |
```

Funzioni per le stringhe

eval esegue il comando Matlab contenuto in una stringa

Sintassi

eval(s)



```
Command Window
File Edit Debug Desktop Window Help
>> a='c=34'
a =
c=34
>> eval(a)
c =
34
>> a='c=ones(3)'
a =
c=ones(3)
>> eval(a)
c =
1 1 1
1 1 1
1 1 1
>> |
```

Grafica 2D

La grafica è una delle caratteristiche più sviluppate di Matlab.

Matlab traccia i grafici su finestre separate rispetto al desktop e chiamate **figure**.

Per default Matlab traccia i grafici nella figura 1.

Volendo disegnare il grafico in una diversa figura occorre per prima cosa aprirla con il comando `figure(n)`, dove `n` è il numero della finestra.

Con il comando **close**(n) si chiude la n-esima finestra.

Il comando principale per tracciare i grafici bidimensionali è **plot**.

Il comando Plot

Sintassi

`plot(x,y,stile)`

`x` vettore delle ascisse `y` vettore delle ordinate

`stile` è una stringa racchiusa tra apici, che definisce il tipo di linea, di marcatore e il colore. Vale la seguente tabella:

TIPO DI LINEA	TIPO DI PUNTO	COLORE
Continua <code>-</code>	Punto <code>.</code>	Blu <code>b</code>
Tratteggiata <code>--</code>	Più <code>+</code>	Verde <code>g</code>
Punteggiata <code>:</code>	Asterisco <code>*</code>	Rosso <code>r</code>
A tratto punto <code>-.</code>	Cerchio <code>o</code>	Ciano <code>c</code>
	Croce <code>x</code>	Magenta <code>m</code>
	Quadrato <code>s</code>	Giallo <code>y</code>
	Rombo <code>d</code>	Bianco <code>w</code>
	Triangoli <code>v, ^, >, <</code>	Nero <code>k</code>
	Stelle <code>p, h</code>	

Il comando Plot

```
Command Window
File Edit Debug Desktop Window Help

>> x=1:10

x =

     1     2     3     4     5     6     7     8     9    10

>> y=rand(1,10)

y =

Columns 1 through 6

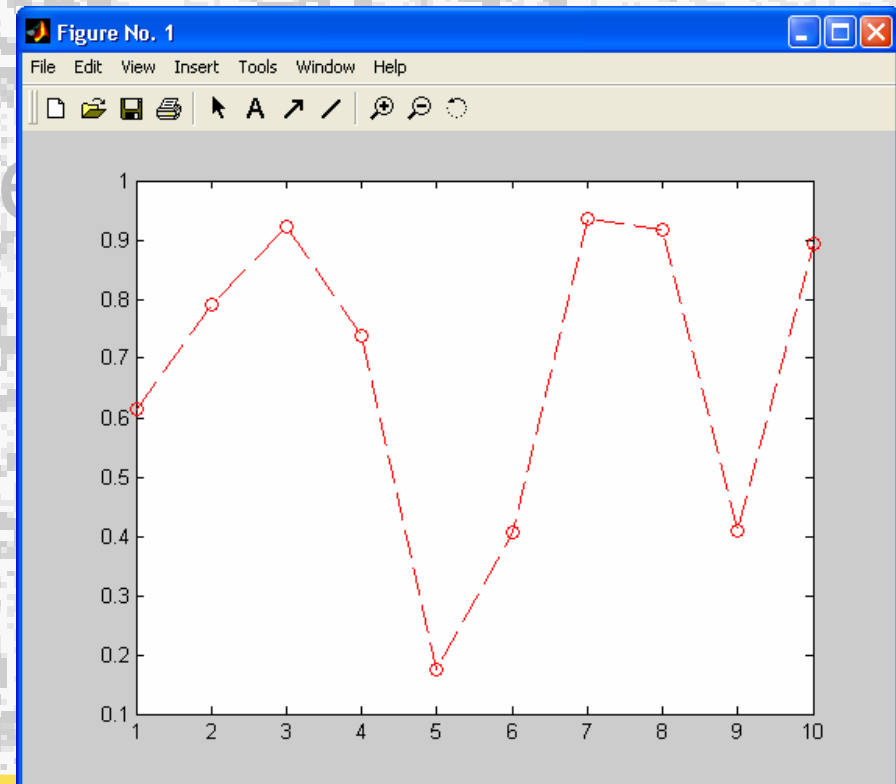
    0.0579    0.3529    0.8132    0.0099    0.1389    0.2028

Columns 7 through 10

    0.1987    0.6038    0.2722    0.1988

>> plot(x,y,'--or')
```

Il comando Plot adegua in modo automatico le scale degli assi in base ai valori dei due vettori.



Il comando Plot

Se la stringa di stile è omessa, i valori di default sono:

colore blu

linea continua

marcatore assente

Nella stringa di stile non è detto che siano indicati tutti e tre i parametri. Quando sono omessi vengono assunti quelli di default.

Esempio

plot(x,y,'r')

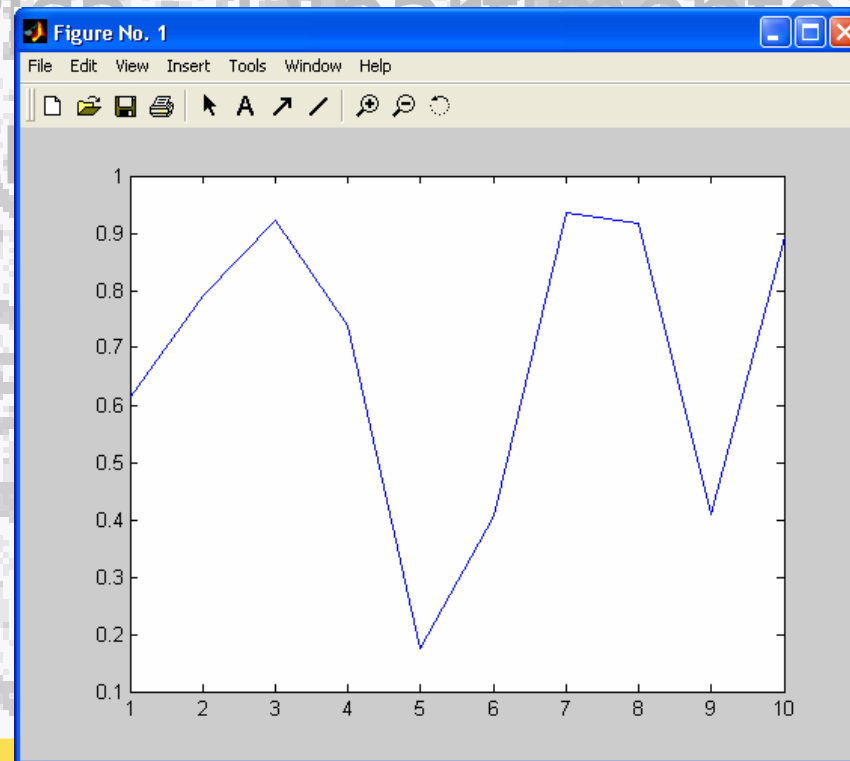
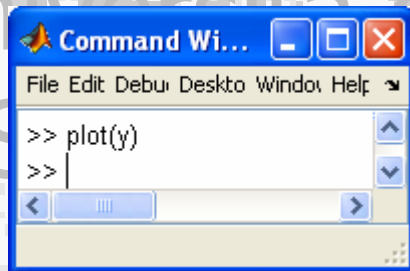
traccia la curva con una linea continua di colore rosso, senza marcatori di punto.

Il comando Plot

Sintassi

plot(x, stile)

Matlab traccia la curva che ha per ascisse gli indici del vettore x e per ordinata il vettore x stesso. Questo se x è un vettore di numeri reali.

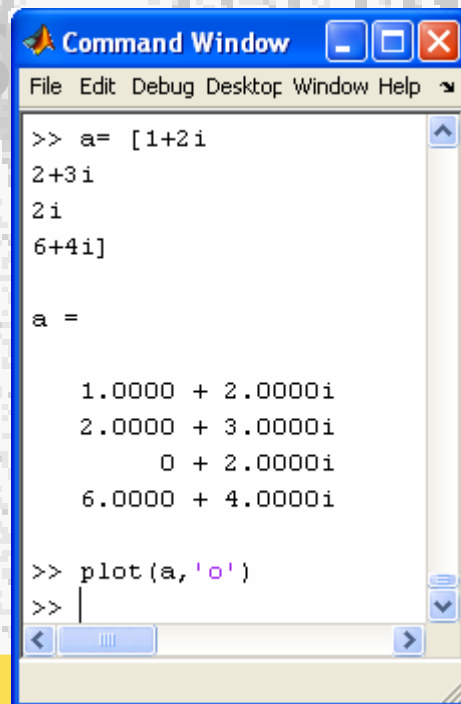


Plot di numeri complessi

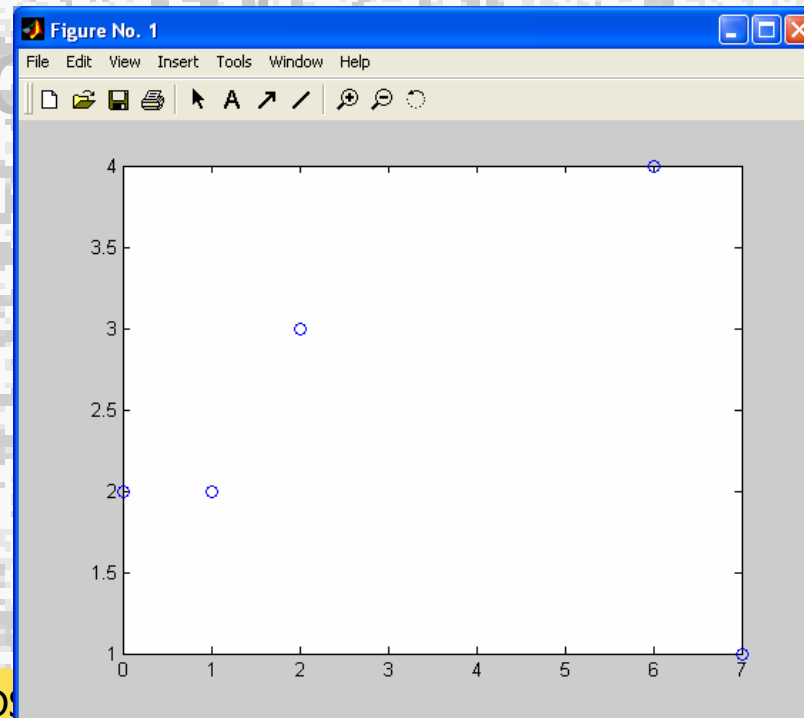
Sintassi

plot(x, stile)

Se x è un vettore di numeri complessi, Matlab traccia la parte immaginaria in funzione di quella reale. Nel caso $\text{plot}(x,y)$ con x o y complessi la parte immaginaria è ignorata



```
>> a= [1+2i  
2+3i  
2i  
6+4i]  
  
a =  
  
    1.0000 + 2.0000i  
    2.0000 + 3.0000i  
         0 + 2.0000i  
    6.0000 + 4.0000i  
  
>> plot(a, 'o')  
>>
```



Grafici Multipli

Per confrontare più curve il modo migliore è tracciarle sulla stessa finestra. Ci sono tre modi per tracciare grafici sovrapposti.

Caso 1

Le varie funzioni hanno lo stesso vettore di ascissa.

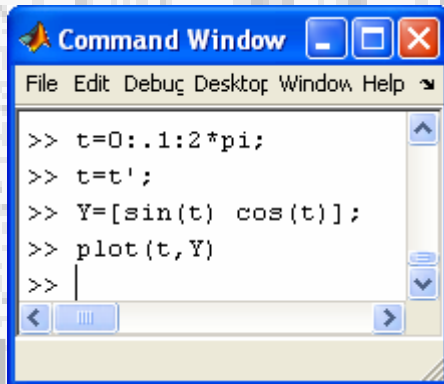
Si può usare plot con argomenti matriciali

`plot(x,Y)`

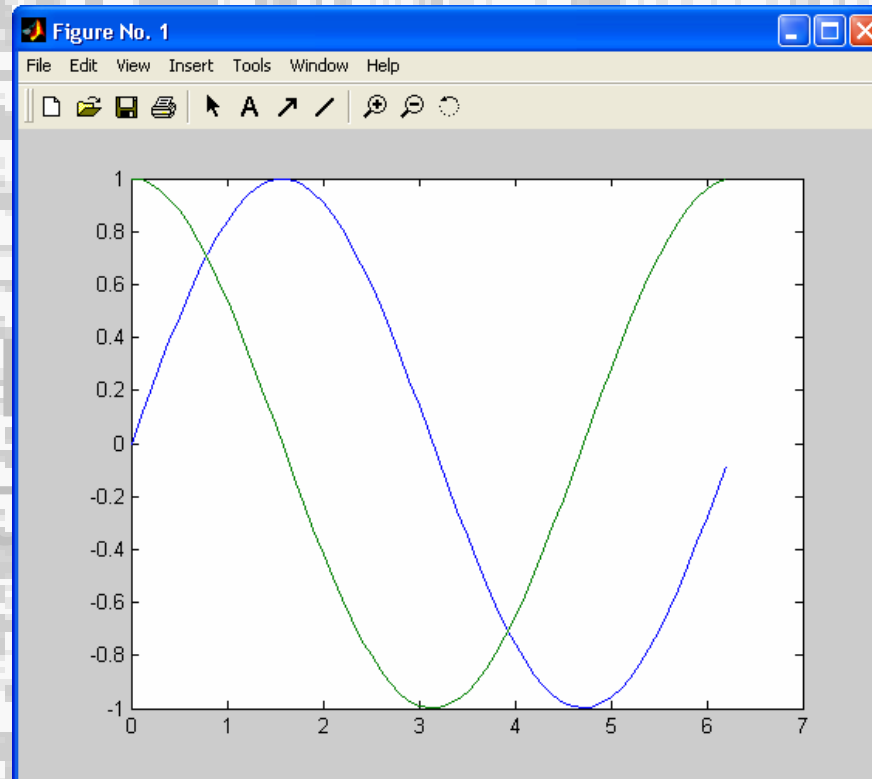
`x` vettore delle ascisse

`Y` matrice in cui ogni colonna rappresenta un'ordinata.

Grafici Multipli



```
>> t=0:.1:2*pi;  
>> t=t';  
>> Y=[sin(t) cos(t)];  
>> plot(t,Y)  
>>
```



Anche in questo caso la scalatura degli assi è automatica in base ai massimi valori di ascissa e di ordinata. Ogni curva è tracciata con un colore diverso secondo l'ordine della tabella.

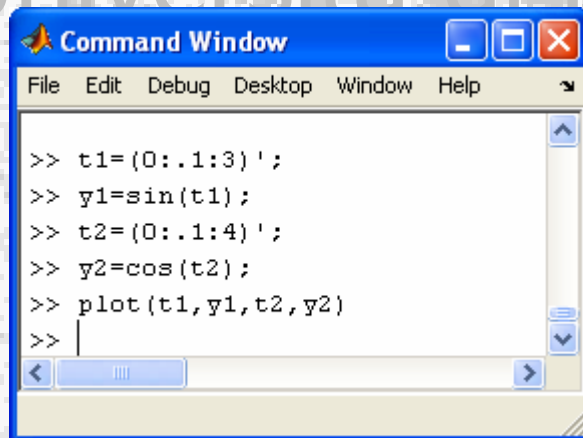
Grafici Multipli

Caso 2

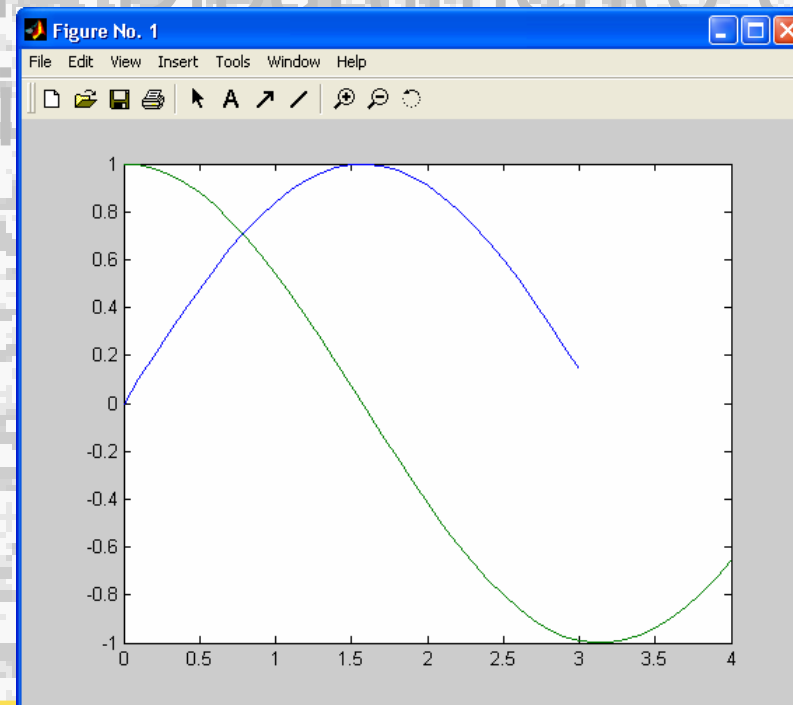
Le varie funzioni hanno vettori di ascissa diversi.

Si può usare plot con più argomenti

plot(x1,y1,stile1,x2,y2,stile2,x3,y3.....)



```
>> t1=(0:.1:3)';  
>> y1=sin(t1);  
>> t2=(0:.1:4)';  
>> y2=cos(t2);  
>> plot(t1,y1,t2,y2)  
>>
```



Grafici Multipli

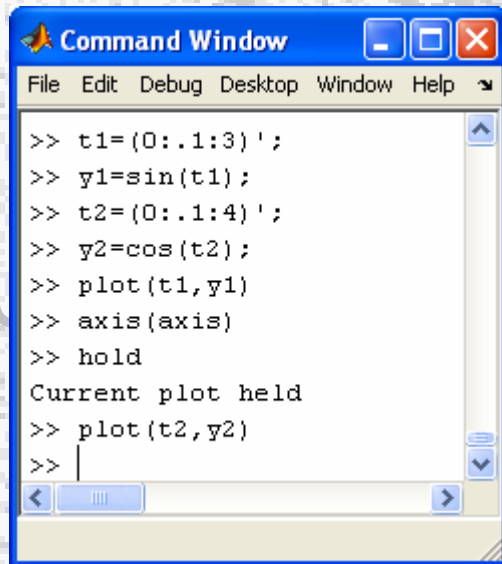
Caso 3

Le varie funzioni hanno vettori di ascissa diversi, però non si vuole che la scalatura degli assi sia automatica. Si vuol cioè privilegiare il grafico di una curva rispetto agli altri.

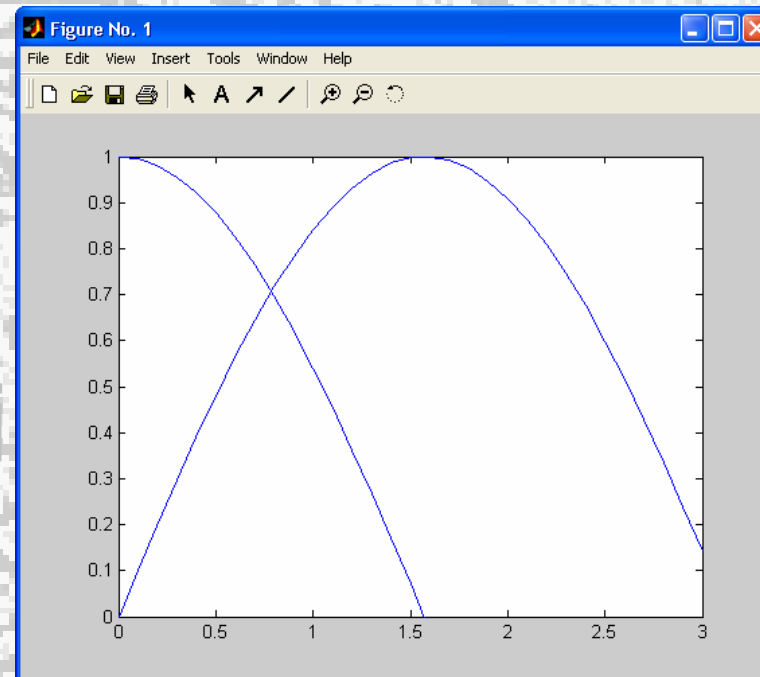
Procedimento

- Si traccia il grafico della curva principale.
- Si congela la finestra con i comand **axis(axis)** e **hold**.
- Si tracciano le altre curve.

Grafici Multipli



```
>> t1=(0:.1:3)';  
>> y1=sin(t1);  
>> t2=(0:.1:4)';  
>> y2=cos(t2);  
>> plot(t1,y1)  
>> axis(axis)  
>> hold  
Current plot held  
>> plot(t2,y2)  
>>
```



Se si usa il solo comando hold, gli assi vengono ancora scalati in modo automatico come nel caso 2, mentre i colori non sono modificati.

Il comando hold è attivo finché non viene ripetuto.

Tracciamento di più grafici nella stessa finestra

Quando è necessario confrontare più curve che però hanno scale molto diverse tra loro è necessario tracciarle su grafici diversi, ma presenti sulla stessa finestra. Si usa allora l'istruzione **subplot**.

Sintassi

subplot(m,n,i)

suddivide la finestra in m righe e n colonne e seleziona la i-esima sottofinestra (sono numerate da sinistra a destra e dall'alto in basso).

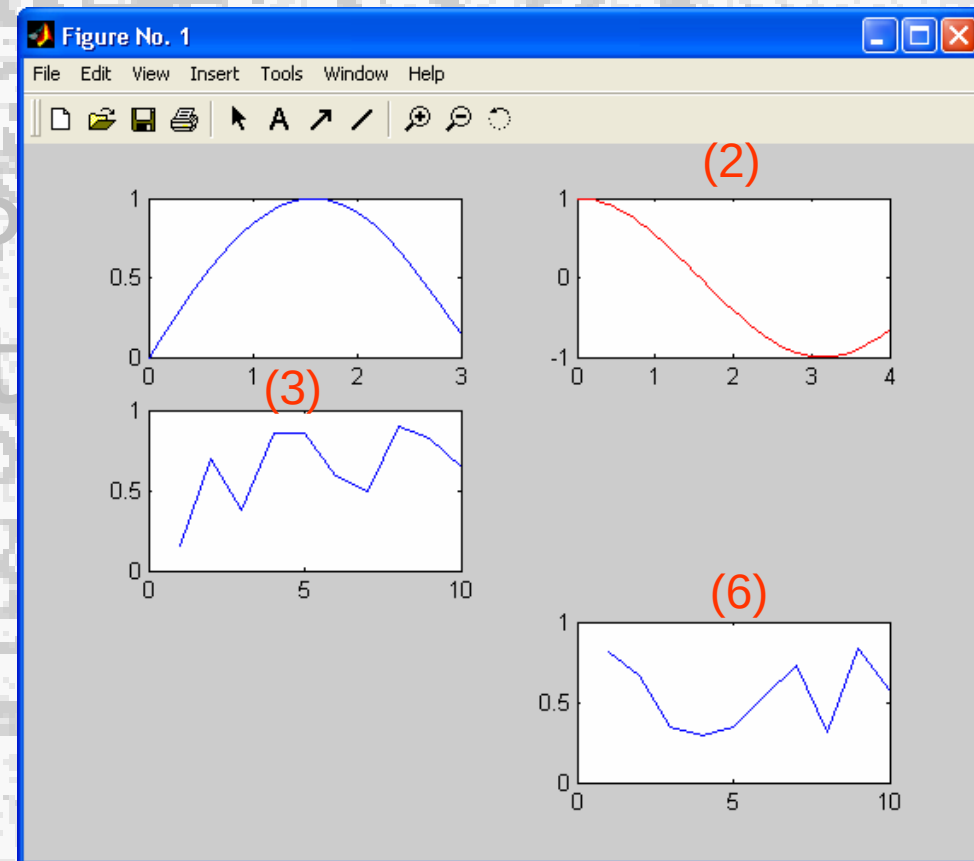
Tutti i successivi comandi grafici agiranno in questa sottofinestra, per passare ad un'altra finestra occorre ripetere il comando subplot.

Per eliminare la partizione in sottofinestre si digita

subplot(1,1,1)

Tracciamento di più grafici nella stessa finestra

```
Command Window
File Edit Debug Desktop Window Help
>> subplot(3,2,1)
>> t1=(0:.1:3)';
>> y1=sin(t1);
>> plot(t1,y1)
>> subplot(3,2,2)
>> t2=(0:.1:4)';
>> y2=cos(t2);
>> plot(t2,y2,'r')
>> subplot(3,2,3)
>> y=rand(1,10);
>> plot(y)
>> subplot(3,2,6)
>> y=rand(1,10);
>> plot(y)
```



Grafici in scale logaritmiche

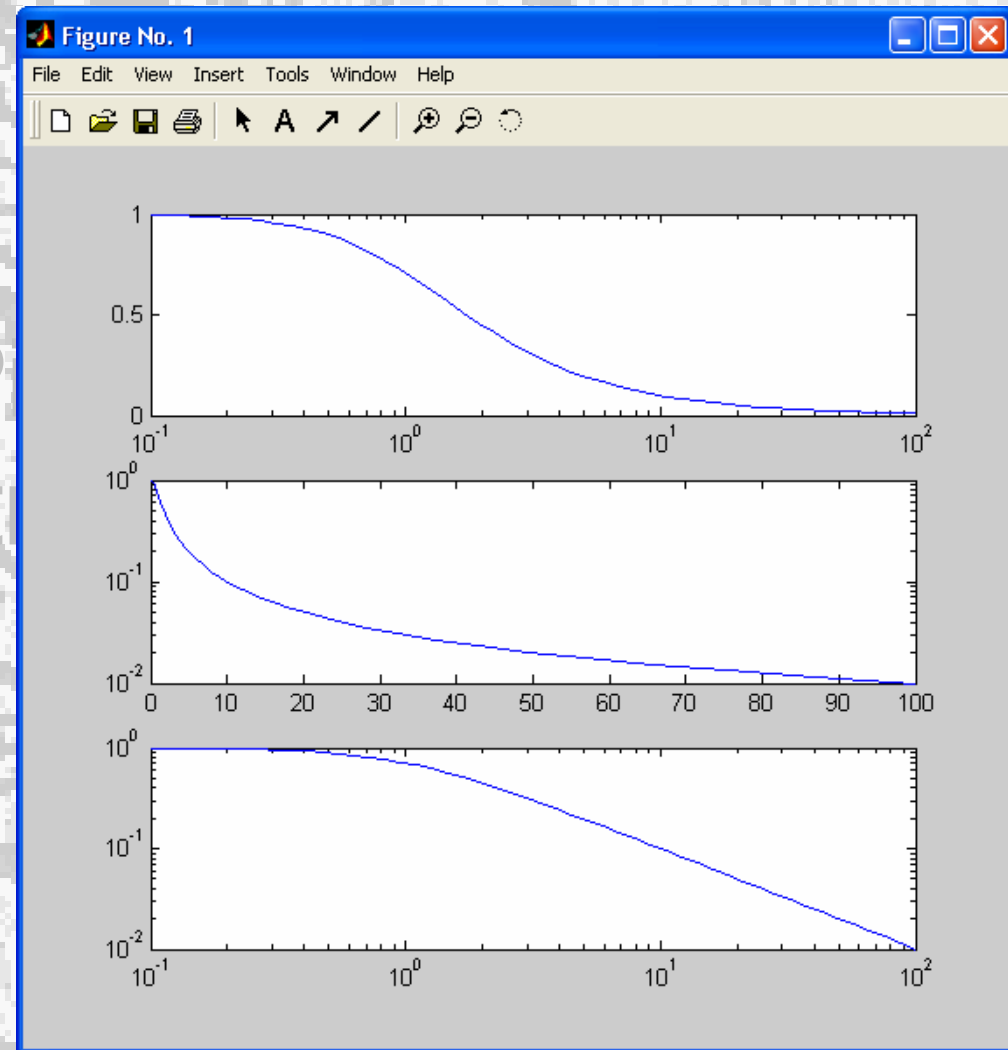
semilogx traccia l'asse x in scala logaritmica (in base 10). I parametri sono identici a plot.

semilogy traccia l'asse y in scala logaritmica (in base 10). I parametri sono identici a plot.

loglog traccia entrambi gli assi in scala logaritmica (in base 10). I parametri sono identici a plot.

Grafici in scale logaritmiche

```
Command Window
File Edit Debug Desktop Window Help
>> x=logspace(-1,2);
>> y=1./sqrt(x.^2+1);
>> subplot(3,1,1)
>> semilogx(x,y)
>> subplot(3,1,2)
>> semilogy(x,y)
>> subplot(3,1,3)
>> loglog(x,y)
>>
```



Grafici in coordinate polari

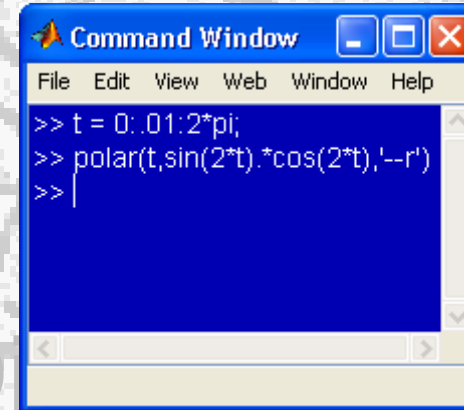
polar traccia i grafici in coordinate polari.

Sintassi

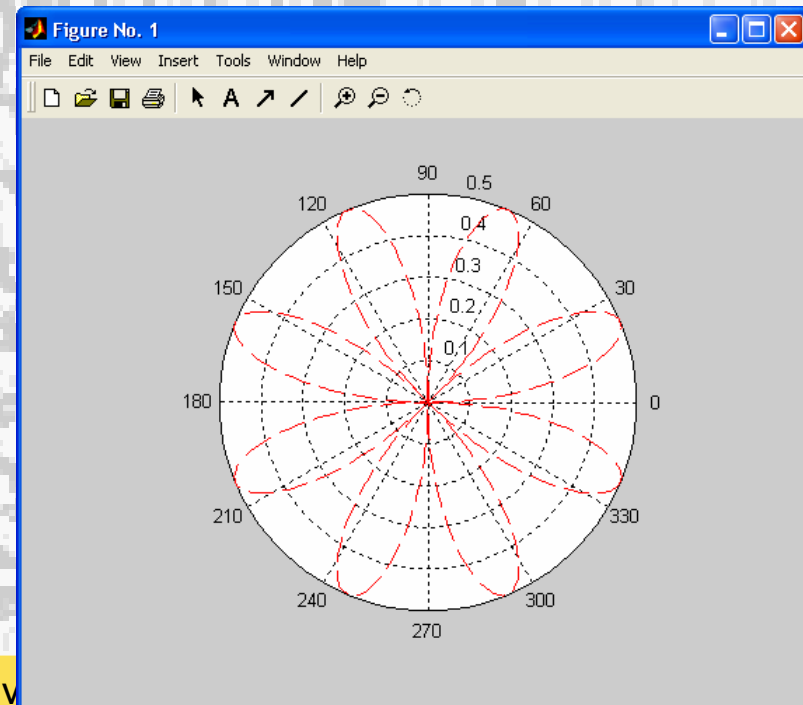
polar(theta,ro,stile)

L'angolo theta è quello misurato rispetto all'asse x e va immesso in radianti, mentre gli assi sono tracciati in gradi.

Stile è identico a plot.



```
>> t = 0:0.01:2*pi;  
>> polar(t,sin(2*t).*cos(2*t),'-r')  
>>
```



Grafici a barre

bar traccia i grafici a barre verticali

barh traccia i grafici a barre orizzontali

Sintassi (uguale per entrambi)

bar(x,Y, stile)

x è un vettore crescente di dimensione $n \times 1$. Può essere omesso.

Y è una matrice $n \times m$

se stile = **'groups'** [default] il comando traccia un grafico costituito da n gruppi di barre, ciascuno formato da m barre.

Gli elementi di un gruppo sono quelli di una riga di Y. Il gruppo i-esimo è posizionato in corrispondenza dell'ascissa rappresentata dall'elemento i-esimo di Y.

se stile = **'stacked'** si ha una barra per ogni riga di Y. L'altezza è la somma dei valori di quella riga e viene suddivisa con i colori in aree proporzionali ai singoli elementi della riga.

Grafici a barre

```

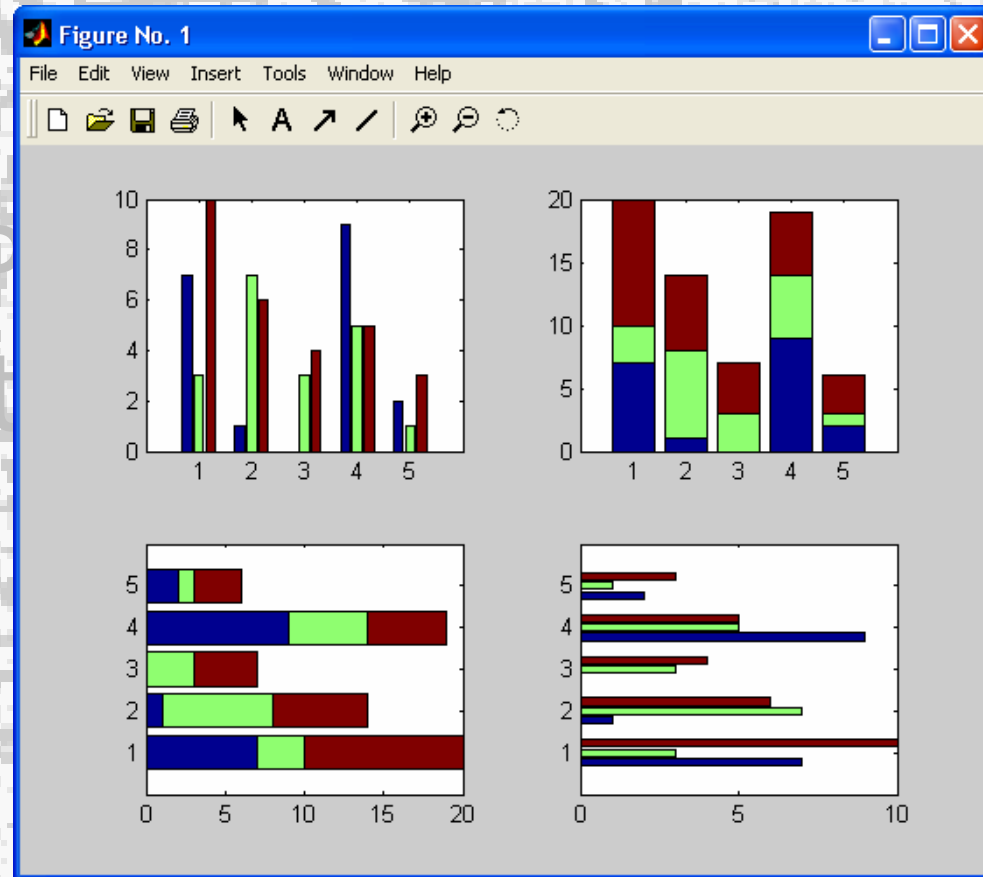
Command Window
File Edit Debug Desktop Window Help
>> Y = round(rand(5,3)*10)

Y =

     5     7     7
     9     3     3
     8     3     8
     6     3     6
     8     5     4

>> subplot(2,2,1)
>> bar(Y,'group')
>> subplot(2,2,2)
>> bar(Y,'stack')
>> subplot(2,2,3)
>> barh(Y,'stack')
>> subplot(2,2,4)
>> barh(Y)
>>
>>

```



Grafici a torta

pie traccia i grafici a torta

Sintassi

pie(x, explode)

x è un vettore di dimensione n

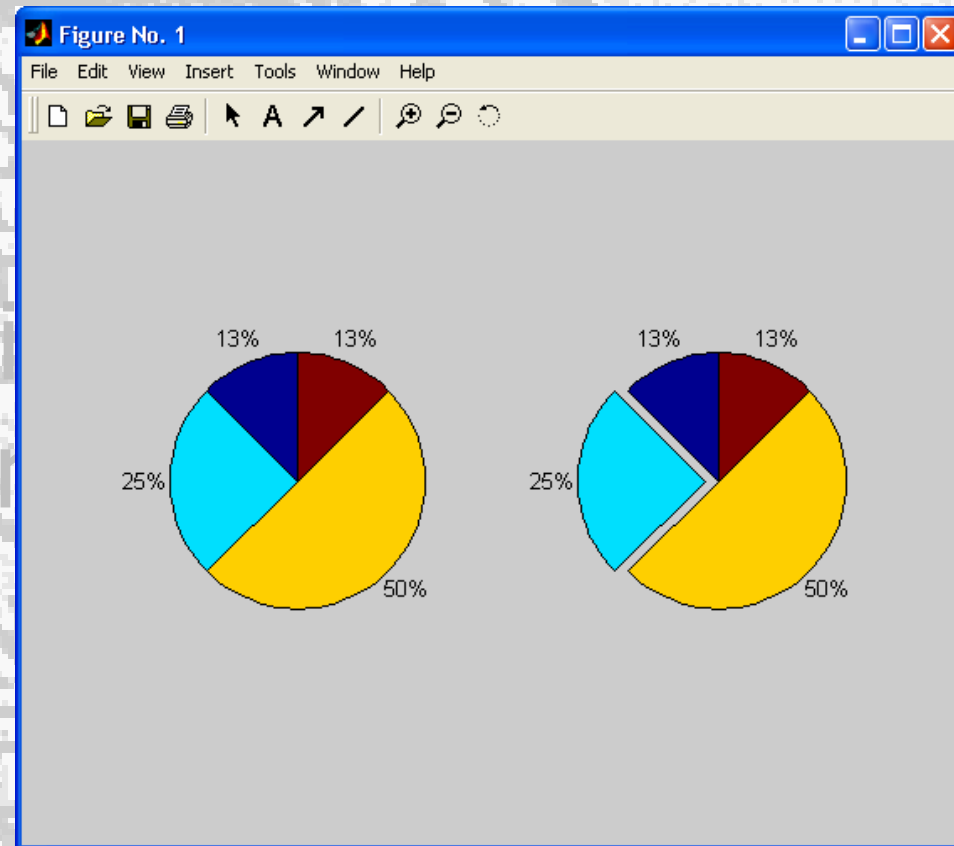
Il comando pie traccia una torta di n fette.

L'ampiezza della i-esima fetta è proporzionale alla percentuale del corrispondente elemento nel vettore x rispetto alla somma di tutti gli elementi di x.

Il parametro explode (opzionale) è un vettore di n elementi ciascuno dei quali indica di quanto deve essere separata la fetta corrispondente dal resto della torta.

Grafici a torta

```
Command Window
File Edit Debug Desktop Window Help
>> a=[10 20 40 10]
a =
    10    20    40    10
>> subplot(1,2,1)
>> pie(a)
>> b=[0 1 0 0]
b =
     0     1     0     0
>> subplot(1,2,2)
>> pie(a,b)
>>
```



Istogrammi

hist traccia l'istogramma della serie di campioni in coordinate cartesiane.

rose traccia l'istogramma della serie di campioni in coordinate polari.

Sintassi

hist(y, x) e rose(y,x)

Se x è un vettore di dimensione n allora la serie di dati y viene suddivisa in n gruppi.

Se x è uno scalare intero è assunto $n=x$

Se x è omissso viene assunto $n=10$

Per ciascun gruppo viene calcolato il numero di elementi e il grafico tracciato è un grafico a barre, in cui la i -esima barra è il numero di elementi dell' i -esimo gruppo ed è posizionata in corrispondenza dell' i -esimo elemento di x (se x è un vettore).

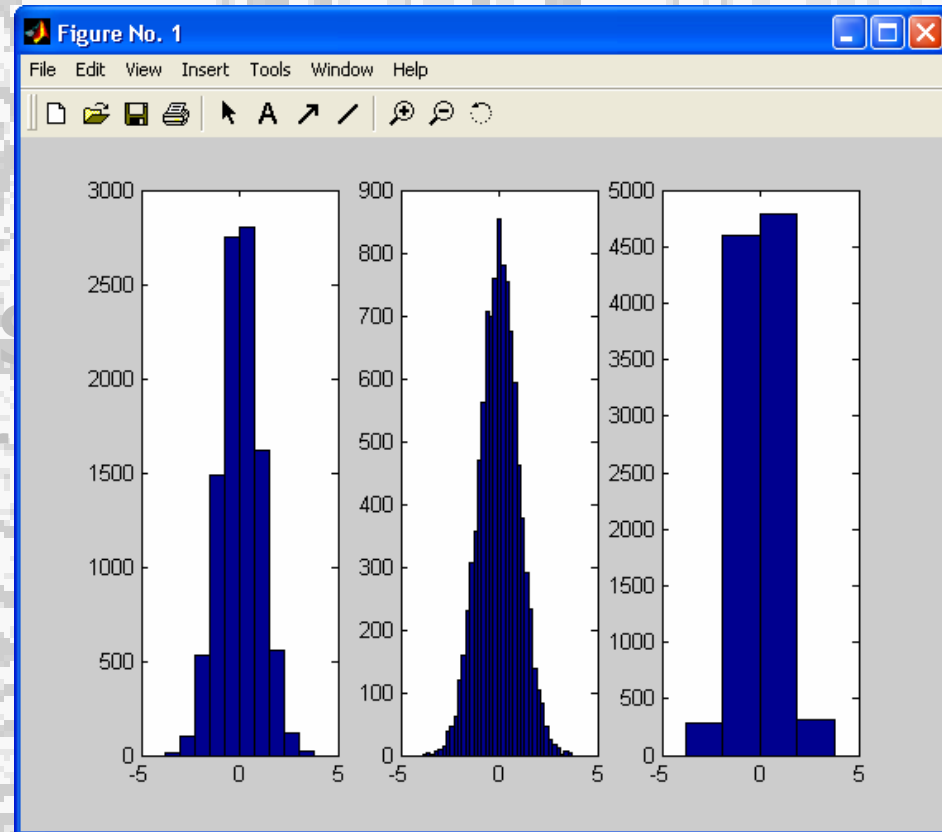
Istogrammi

```
Command Window
File Edit Debug Desktop Window Help
>> yn = randn(10000,1);
>> subplot(1,3,1)
>> hist(yn)
>> x = min(yn):.2:max(yn);
>> subplot(1,3,2)
>> hist(yn,x)
>> n=4

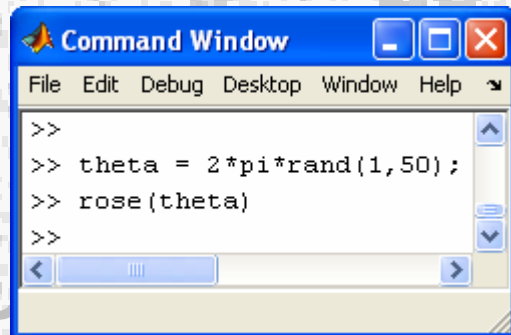
n =

    4

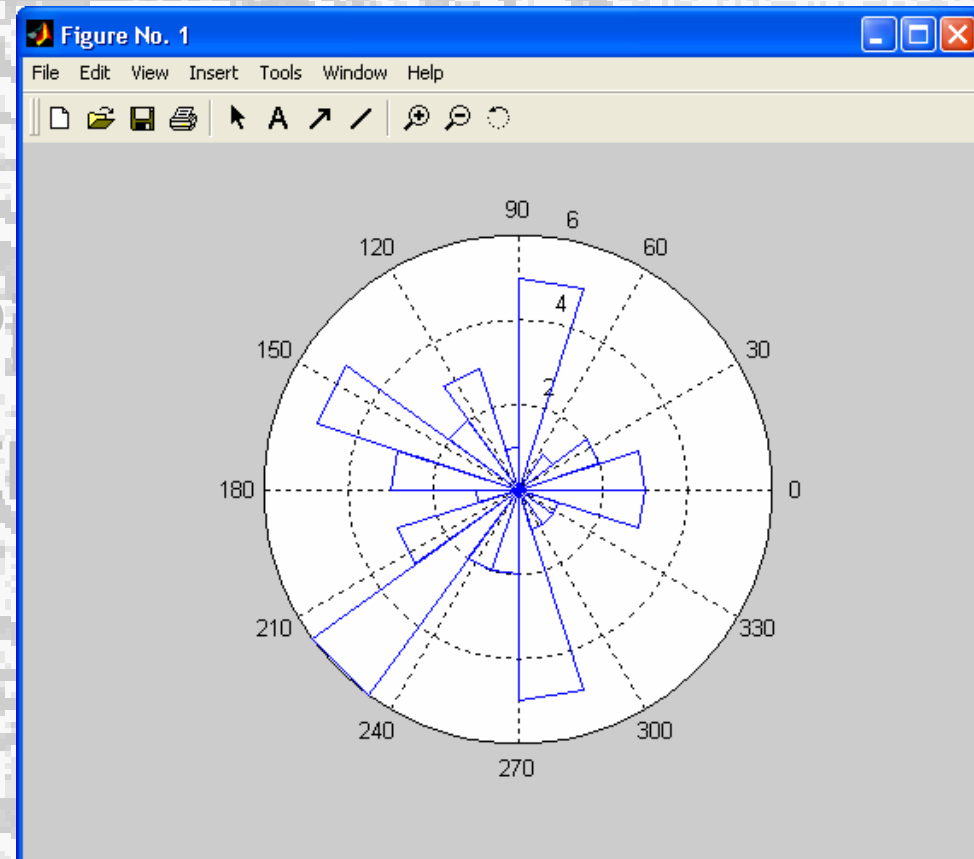
>> subplot(1,3,3)
>> hist(yn,n)
>>
```



Istogrammi



```
>>  
>> theta = 2*pi*rand(1,50);  
>> rose(theta)  
>>
```

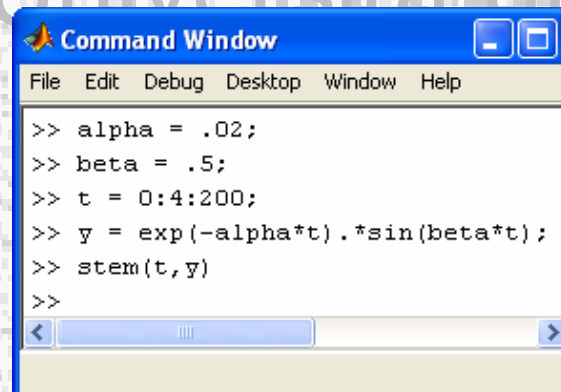


Plot per dati discreti

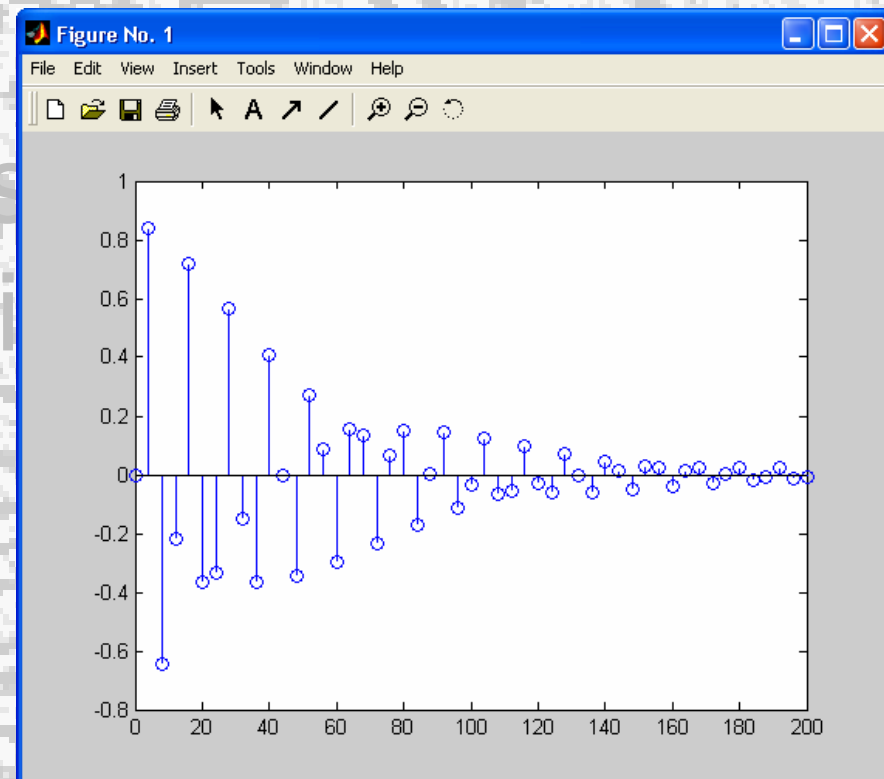
stem traccia i dati di un vettore come segmenti

Sintassi

stem(x, y)



```
>> alpha = .02;  
>> beta = .5;  
>> t = 0:4:200;  
>> y = exp(-alpha*t).*sin(beta*t);  
>> stem(t,y)  
>>
```

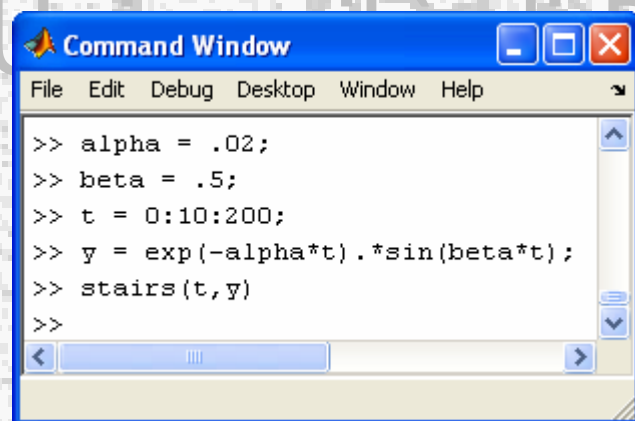


Plot per dati discreti

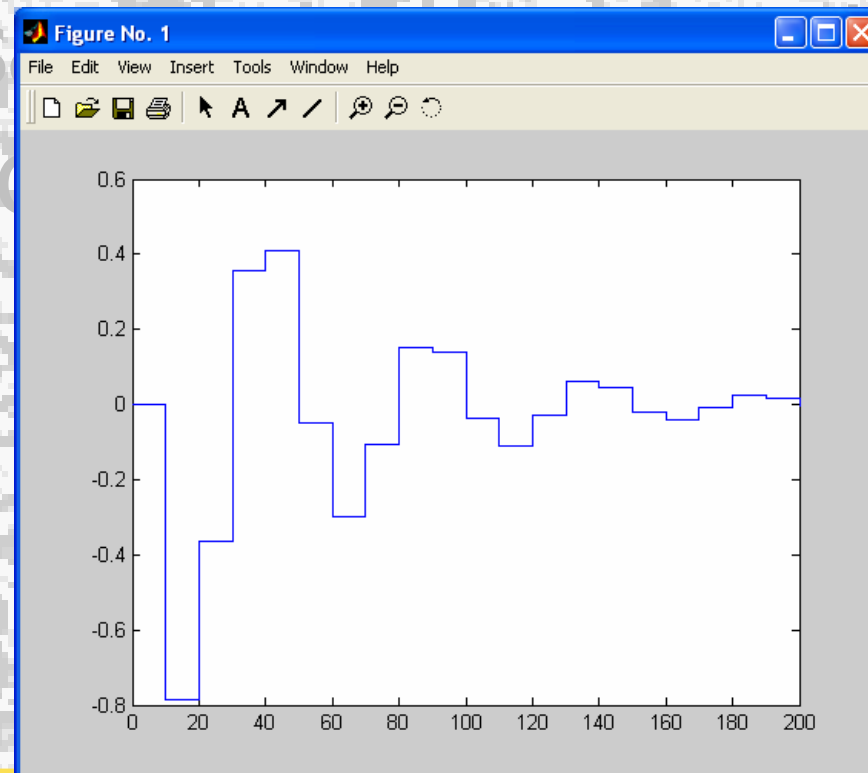
stairs traccia i dati di un vettore con funzioni a gradino

Sintassi

stairs(x, y)



```
>> alpha = .02;  
>> beta = .5;  
>> t = 0:10:200;  
>> y = exp(-alpha*t).*sin(beta*t);  
>> stairs(t,y)  
>>
```



Plot con più assi

plotyy traccia i grafici di due curve utilizzando entrambi gli assi delle ordinate

Sintassi

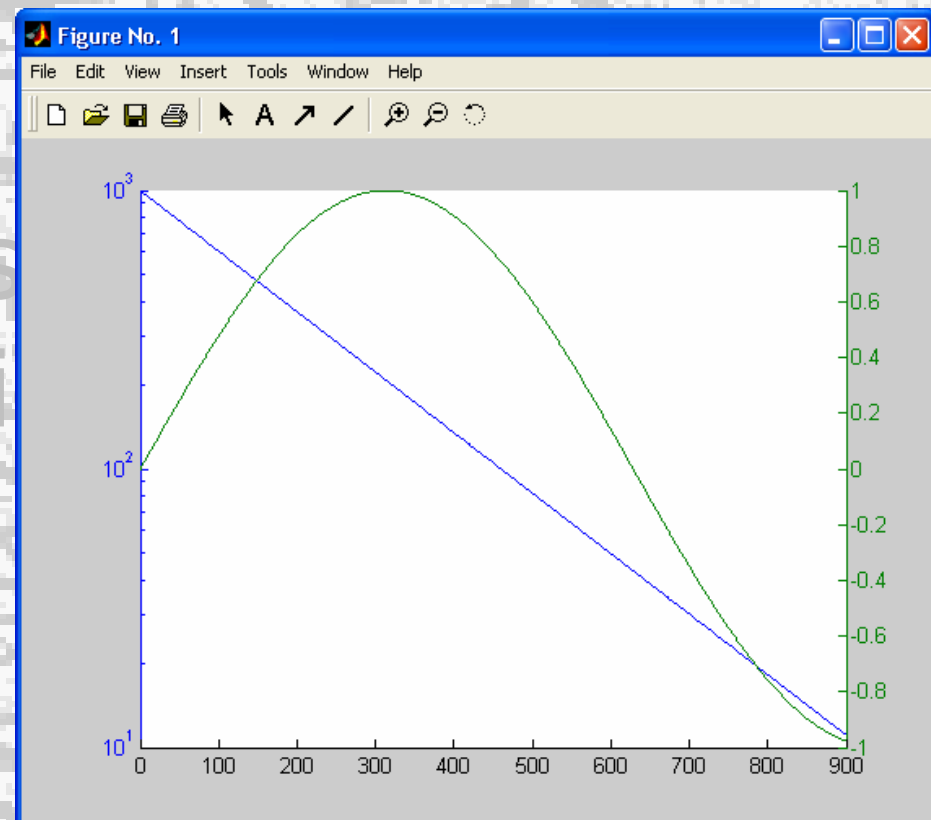
plotyy(x1, y1, x2, y2, 'tipo1', 'tipo2')

dove **tipo** può essere plot, semilogx, semilogy, loglog, stem

Il valore di default (se omissso) è plot

Plot con più assi

```
Command Window
File Edit Debug Desktop Window Help
>>
>> t = 0:900; A = 1000;
>> a = 0.005; b = 0.005;
>> z1 = A*exp(-a*t);
>> z2 = sin(b*t);
>> plotyy(t,z1,t,z2,'semilogy','plot')
>>
```



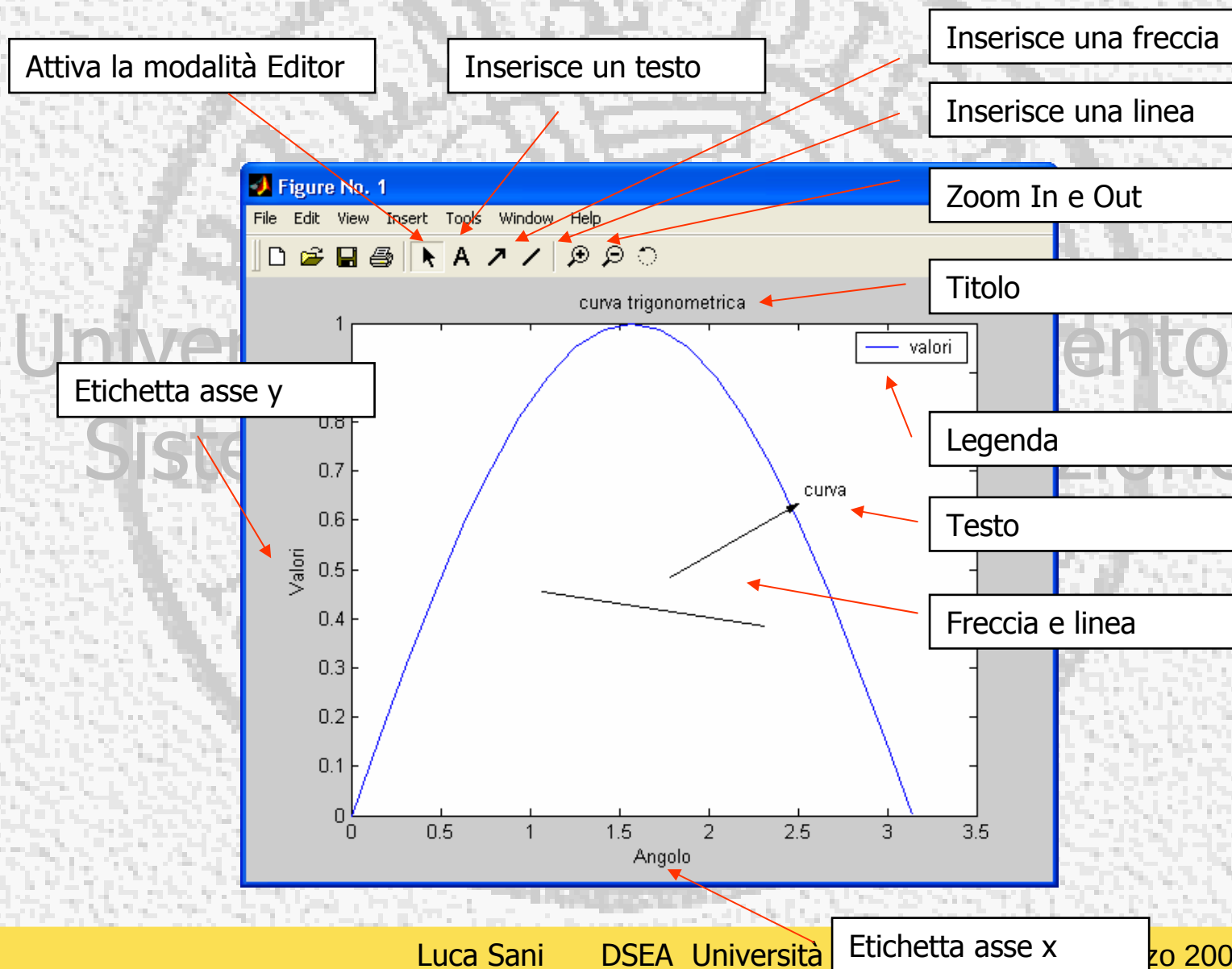
Formattazione dei grafici

E' possibile completare un grafico (aggiungere la griglia, le etichette sugli assi, il titolo) in due modi:

- mediante righe di comando
- utilizzando il plot edit

Università di Pisa - Dipartimento di
Sistemi Elettrici e Automazione

La finestra di Plot



La finestra di Plot

Dalla finestra di plot è possibile

- attivare la modalità editor
- aggiungere un testo
- aggiungere le etichette agli assi
- ingrandire o ridurre l'area di plot
- stampare la figura
- salvare la figura in formato mat. con estensione fig.
- esportare la figura in svariati formati grafici
- individuare la curva di interpolazione
- eseguire un'analisi statistica dei dati

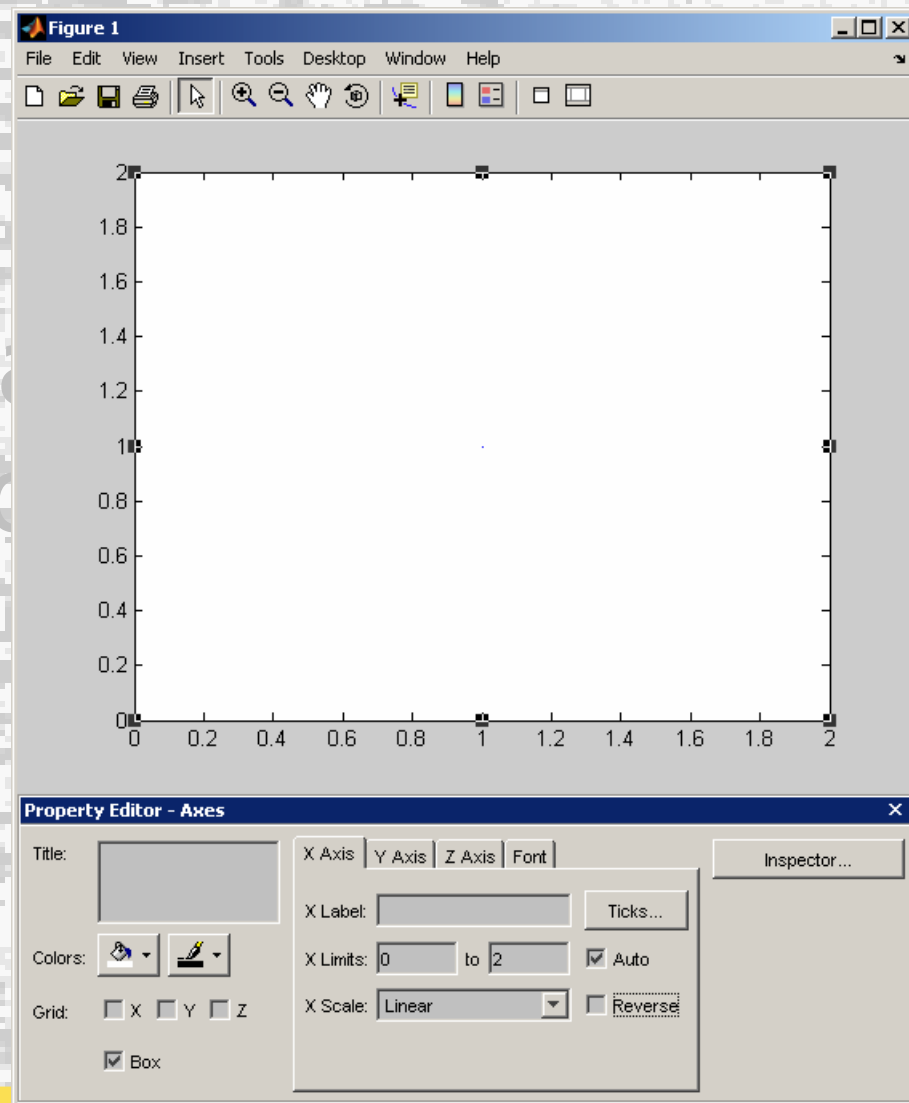
Finestra di Plot Modalità editor

Si attiva da menu **tool** ->
edit Plot oppure dal tasto



Una volta attivo, cliccando con il tasto sinistro in un qualunque punto del grafico si apre la finestra di **Property Editor**.

Tutti gli elementi del grafico sono oggetti di cui è possibile modificare le proprietà.



Aggiunta di un titolo

E' possibile aggiungere un titolo

- Dal menu della figura **Insert -> Title**
- Dal Property Editor nel pannello **Style -> Title**
- Da linea di comando con la funzione **Title**

Sintassi

title(stringa)

Nella stringa sono accettate anche sequenze di caratteri stile LATEX.
Queste iniziano con il segno \

Aggiunta di un titolo all'asse delle ascisse

E' possibile aggiungere un'etichetta all'asse delle x

- Dal menu della figura **Insert -> X Label**
- Dal Property Editor nel pannello **X -> Label**
- Da linea di comando con la funzione **xlabel**

Sintassi

xlabel(stringa)

Nella stringa sono accettate anche sequenze di caratteri stile LATEX.
Queste iniziano con il segno \

Aggiunta di un titolo all'asse delle ordinate

E' possibile aggiungere un'etichetta all'asse delle y

- Dal menu della figura **Insert -> Y Label**
- Dal Property Editor nel pannello **Y -> Label**
- Da linea di comando con la funzione **ylabel**

Sintassi

ylabel(stringa)

Nella stringa sono accettate anche sequenze di caratteri stile LATEX.
Queste iniziano con il segno \

Aggiunta della griglia

E' possibile aggiungere la griglia

- Dal Property Editor nel pannello X -> Grid e nel pannello Y -> Grid
- Da linea di comando con la funzione **grid**

Sintassi

grid commuta lo stato della griglia

grid on inserisce la griglia

grid off rimuove la griglia

Aggiunta della legenda

E' possibile aggiungere la legenda dei dati disegnati

- Dal menu della figura **Insert -> Legend**
- Da linea di comando con la funzione **legend**

Sintassi

legend(s1, s2, ...)

E' possibile modificare l'aspetto della legenda e le sue proprietà nel Property Editor della figura all'oggetto Legend

Nella stringa sono accettate anche sequenze di caratteri stile LATEX. Queste iniziano con il segno \.

Aggiunta di un testo

E' possibile aggiungere un testo nella figura

- Dal menu della figura **Insert -> Text** o dal relativo tasto
- Da linea di comando con la funzione **text** o **gtext**

Sintassi

text(x,y, stringa)

x, y sono le coordinate dove deve essere posizionata la stringa

gtext(stringa)

è la versione tramite mouse

Nella stringa sono accettate anche sequenze di caratteri stile LATEX.
Queste iniziano con il segno \

E' possibile modificare l'aspetto della legenda e le sue proprietà nel Property Editor della figura all'oggetto Text.



Aggiunta di freccia

E' possibile aggiungere una freccia nella figura

- Dal menu della figura **Insert -> Arrow** o dal relativo tasto

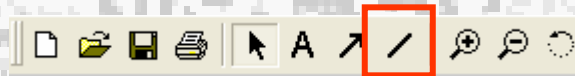


E' possibile modificare l'aspetto della legenda e le sue proprietà nel Property Editor della figura all'oggetto Arrow.

Aggiunta di linea

E' possibile aggiungere una linea nella figura

- Dal menu della figura **Insert -> Line** o dal relativo tasto



E' possibile modificare l'aspetto della legenda e le sue proprietà nel Property Editor della figura, all'oggetto Line.

La funzione zoom

E' possibile attivare la funzione zoom

- Dal menu della figura **Tool -> Zoom In** (rimpicciolire) o **Tool -> Zoom Out** (ingrandire) oppure dai relativi tasti



- Da linea di comando con la funzione **zoom**

Sintassi

zoom commuta lo stato dello zoom

zoom on abilita lo zoom

zoom off disabilita lo zoom

zoom reset annulla lo zoom

la regione da ingrandire viene selezionata mediante il mouse.

Scalatura degli assi

E' possibile modificare le scale degli assi, e le rispettive etichette

- Nel Property Editor all'oggetto Axes e poi nelle sottocartelle X, Y, Z
- Da linea di comando con la funzione **axis**

Sintassi

axis([xmin xmax ymin ymax]) definisce i valori estremi di ascissa a ordinata. Il comando va eseguito dopo il plot

axis(axis) congela gli assi

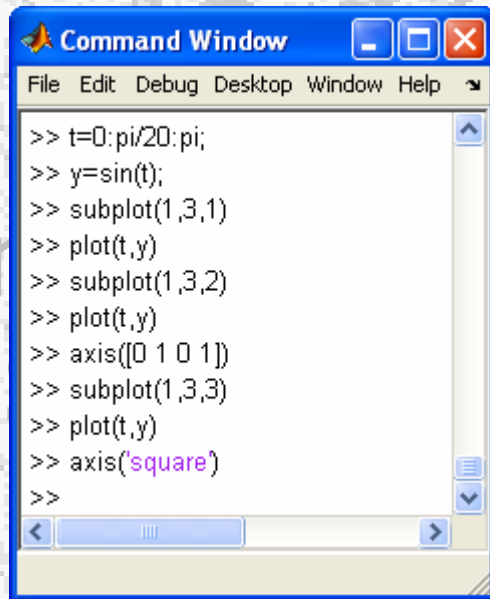
axis('auto') ripristina la scalatura automatica

axis('square') definisce una regione grafica quadrata

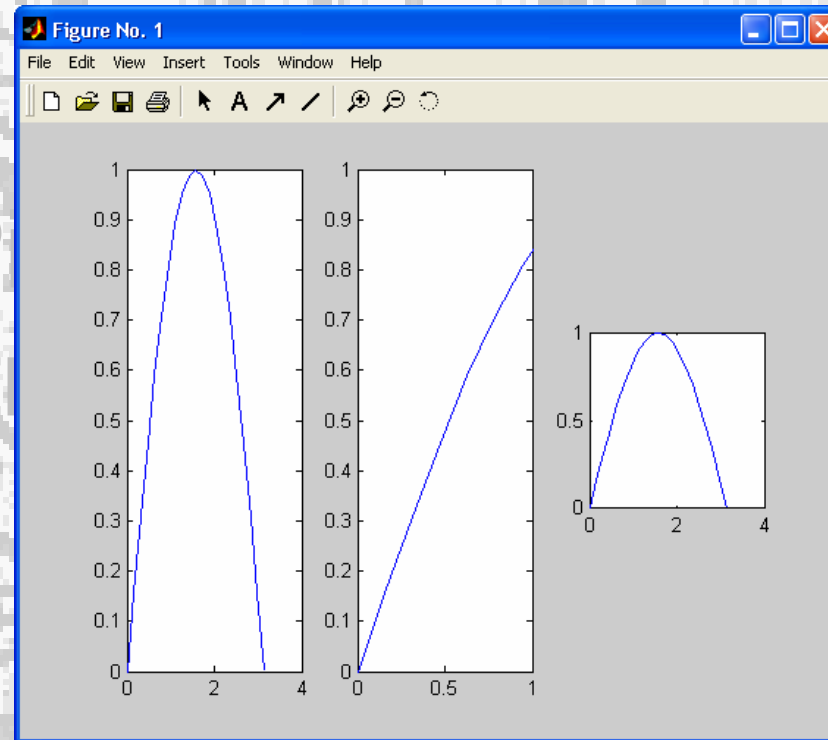
axis('on') attiva il tracciamento assi

axis('off') disattiva il tracciamento assi

Scalatura degli assi



```
>> t=0:pi/20:pi;  
>> y=sin(t);  
>> subplot(1,3,1)  
>> plot(t,y)  
>> subplot(1,3,2)  
>> plot(t,y)  
>> axis([0 1 0 1])  
>> subplot(1,3,3)  
>> plot(t,y)  
>> axis('square')  
>>
```



Finestra di plot Interpolazione della curva

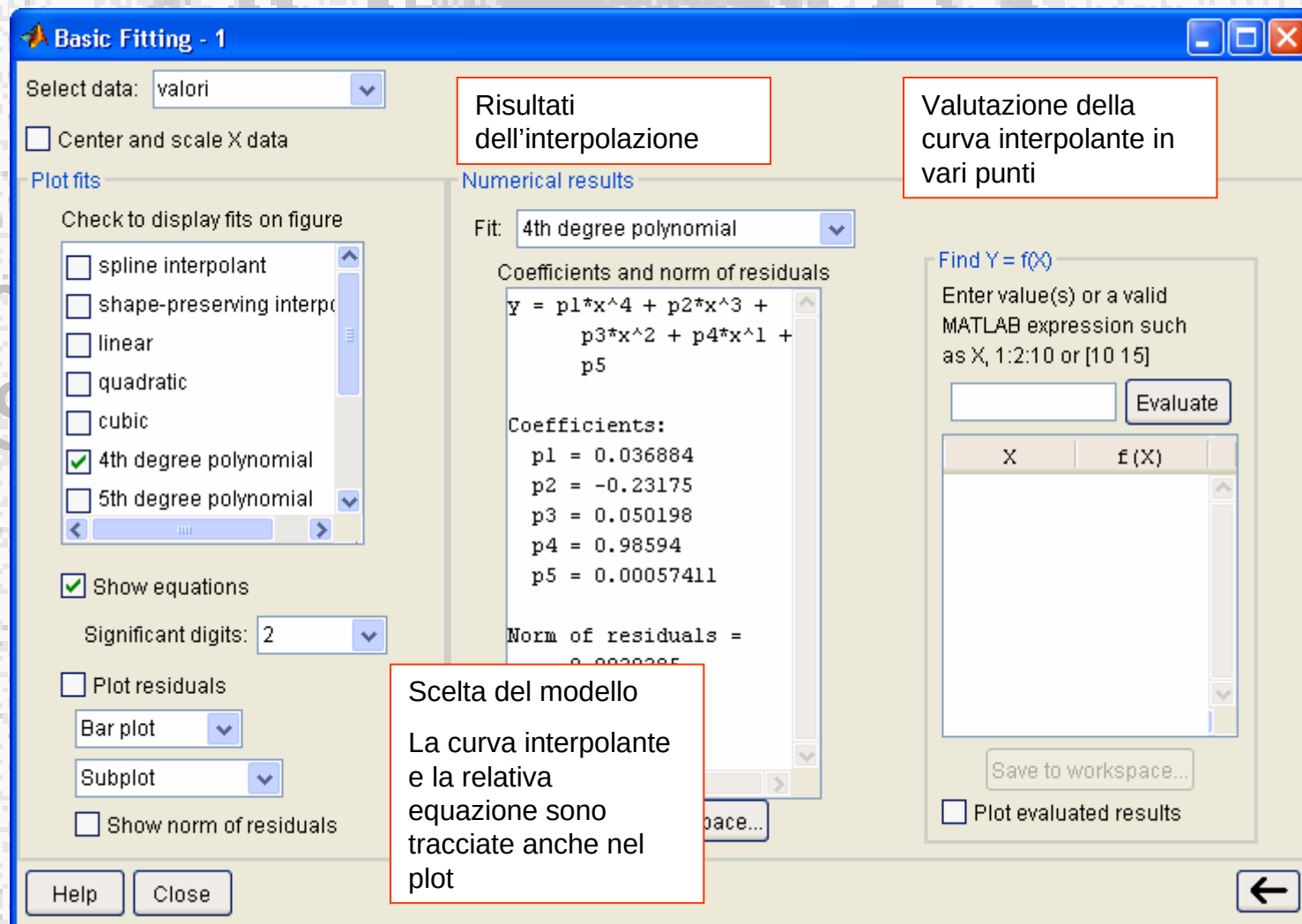
Si attiva da menu **tool -> Basic fitting**

Si compone di tre sottofinestre

- scelta del modello
- risultato dell'interpolazione
- fitting dei dati in un certo intervallo

I risultati possono essere salvati in una variabile nel workspace.

Finestra di plot Interpolazione della curva



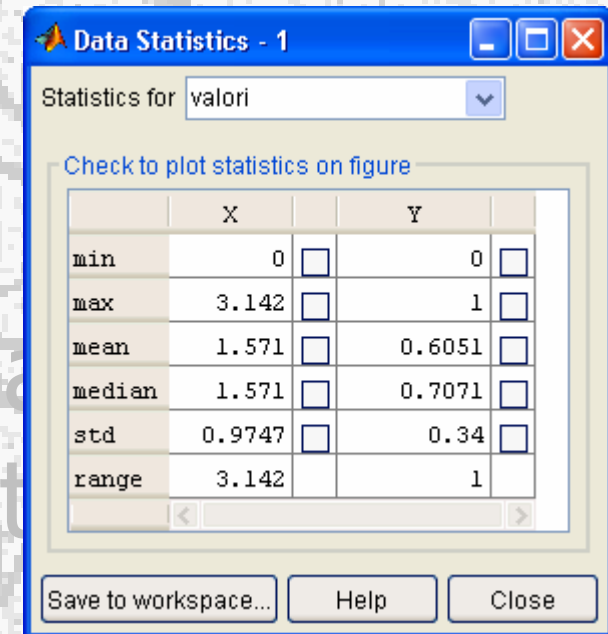
Finestra di Plot Analisi statistica dei risultati

Si attiva da menu **tool -> Data statistics**

Fornisce i parametri statistici del vettore selezionato

- massimo
- minimo
- mediana
- deviazione standard
- range

Se selezionati tali valori vengono individuati sul grafico, oppure possono essere salvati in una variabile nel workspace.



Grafici 3D

La grafica di oggetti tridimensionali è decisamente più complicata.

Le curve tridimensionali si possono classificare in tre categorie:

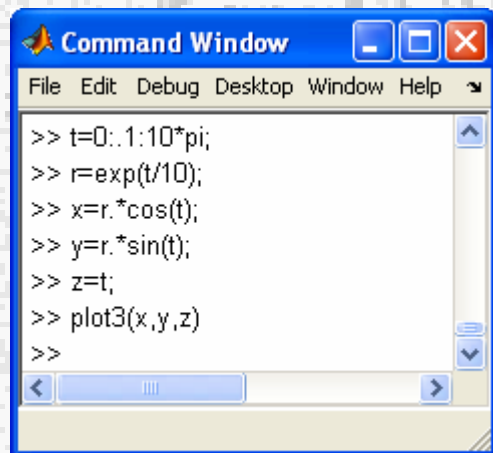
- Curve o superfici di cui è nota una descrizione parametrica $x=x(t)$, $y=y(t)$ e $z=z(t)$.
- Superfici generate da funzioni matematiche di due variabili $z=f(x,y)$.
- Superfici generate dalla rotazione di una curva intorno ad un asse fisso.

Grafici 3D Curve parametriche

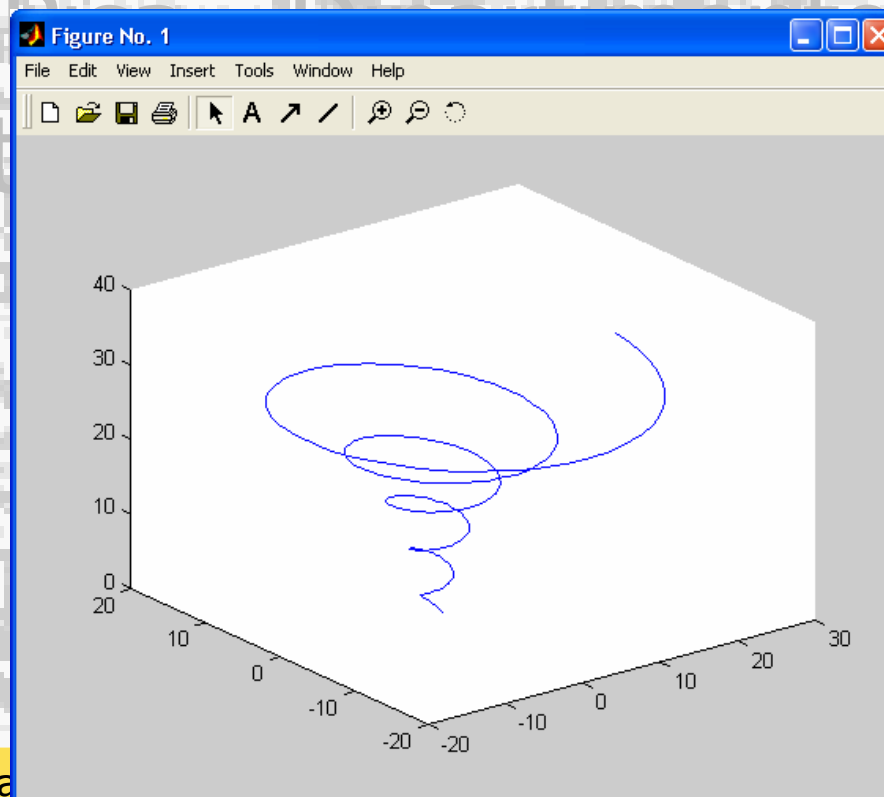
Si utilizza la funzione **plot3**

Sintassi

plot3(x,y,z)



```
>> t=0:1:10*pi;  
>> r=exp(t/10);  
>> x=r.*cos(t);  
>> y=r.*sin(t);  
>> z=t;  
>> plot3(x,y,z)  
>>
```



Grafici 3D Funzioni di due variabili

Si tratta del caso in cui $z=f(x,y)$

La procedura è la seguente:

- si creano i vettori x e y che definiscono gli intervalli per i rispettivi assi. Gli elementi di x e y devono essere equispaziati;
- si crea il dominio rettangolare della funzione con il comando **meshgrid**.

Sintassi

$[X, Y]=meshgrid(x,y)$

X è una matrice che avrà tante righe, tutte uguali al vettore x , quanti sono gli elementi di y e dualmente Y ha tante colonne, tutte uguali a y , quanti sono gli elementi di x . In questo modo il punto di coordinate (x_i, y_j) potrà essere ricavato dalle matrici X e Y come $[X(i,j), Y(i,j)]$

Grafici 3D Funzioni di due variabili

- Si determinano i punti della superficie $Z=f(X,Y)$.
- Si rappresenta la curva con le funzioni mesh, surf meshc, surfc, meshz

Sintassi

mesh(X,Y,Z) congiunge i valori di z con segmenti

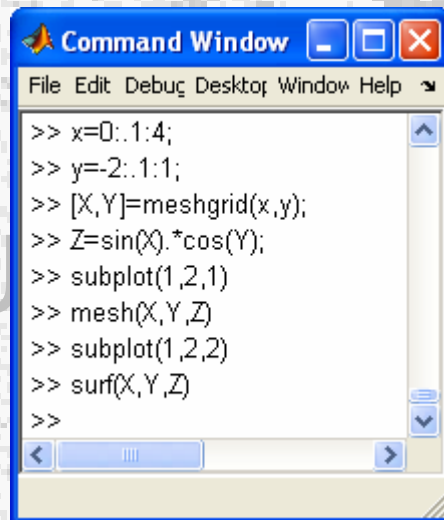
surf(X,Y,Z) colora le facce delimitate dai segmenti che congiungono i valori di Z.

meshc(X,Y,Z) come mesh con l'aggiunta nel piano x-y delle curve di livello.

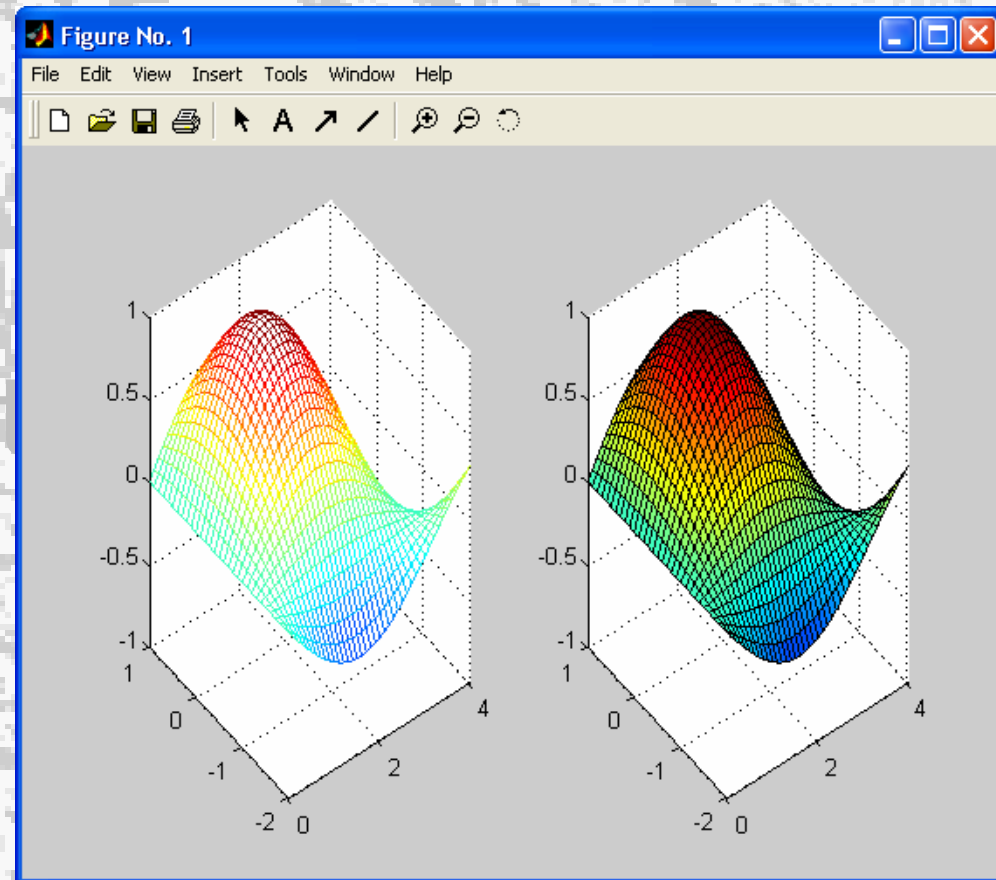
surfc(X,Y,Z) come surf con l'aggiunta nel piano x-y delle curve di livello.

meshz(X,Y,Z) come mesh con l'aggiunta di una base che congiunge i punti a quota zero con il piano x-y.

Grafici 3D Funzioni di due variabili

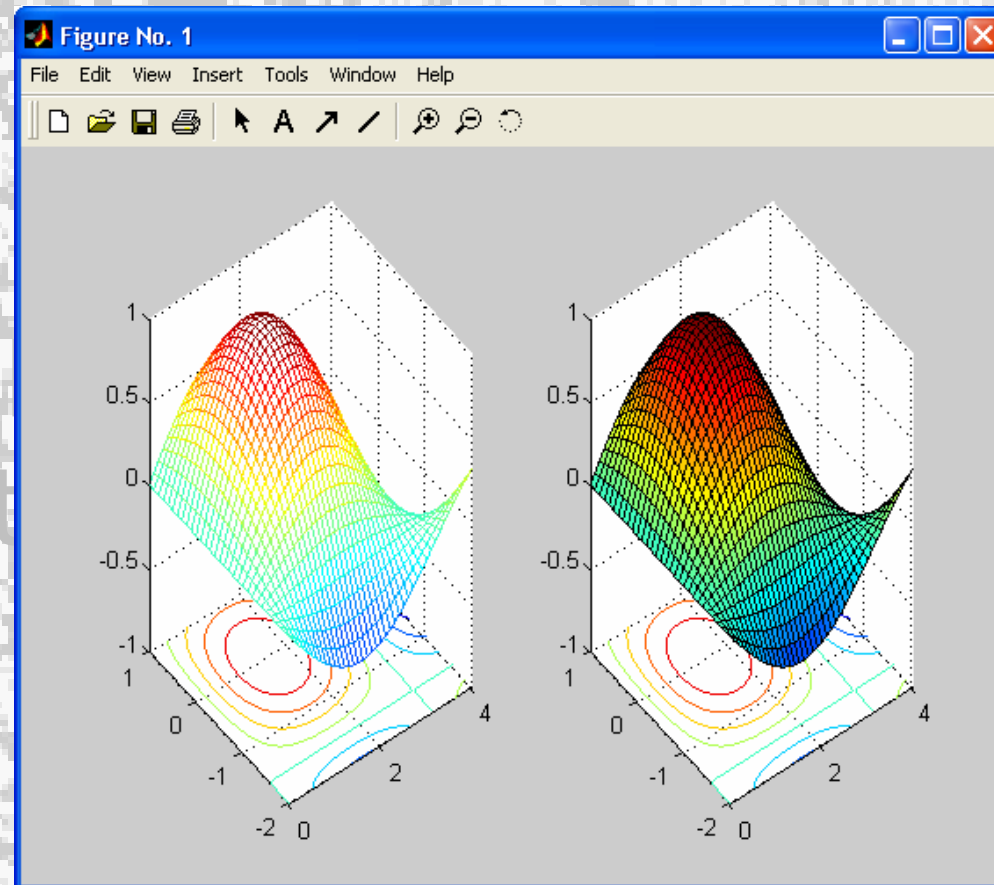


```
>> x=0:1:4;  
>> y=-2:1:1;  
>> [X,Y]=meshgrid(x,y);  
>> Z=sin(X).*cos(Y);  
>> subplot(1,2,1)  
>> mesh(X,Y,Z)  
>> subplot(1,2,2)  
>> surf(X,Y,Z)  
>>
```

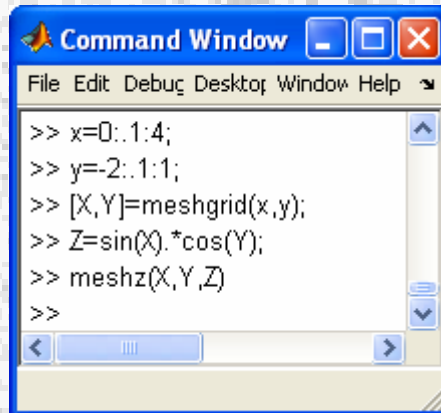


Grafici 3D Funzioni di due variabili

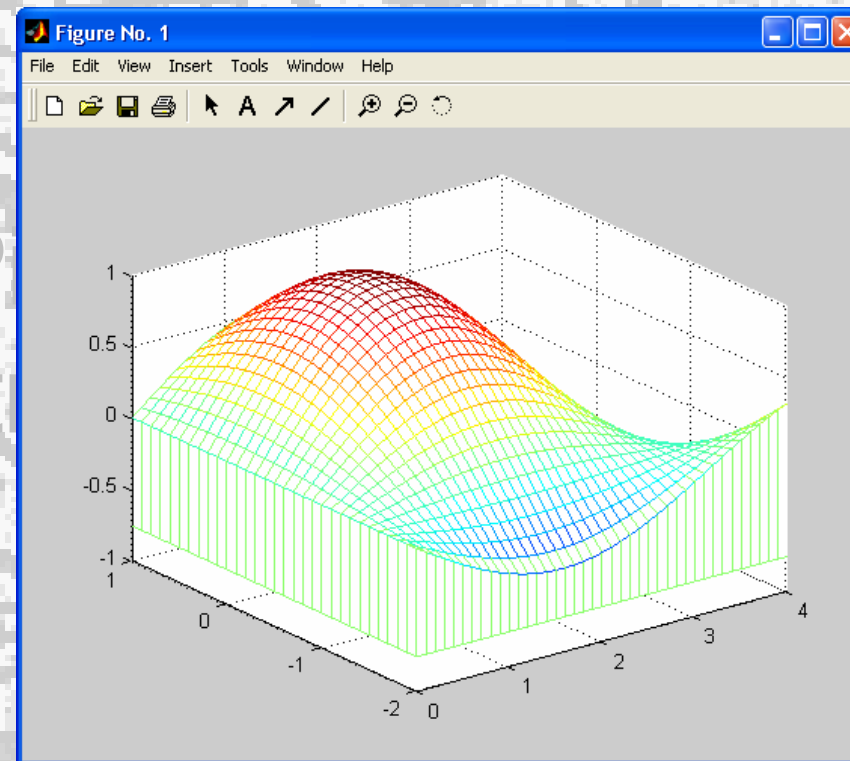
```
Command Window
File Edit Debug Desktop Window Help
>> x=0:1:4;
>> y=-2:1:1;
>> [X,Y]=meshgrid(x,y);
>> Z=sin(X).*cos(Y);
>> subplot(1,2,1)
>> meshc(X,Y,Z)
>> subplot(1,2,2)
>> surf(X,Y,Z)
>>
```



Grafici 3D Funzioni di due variabili



```
>> x=0:1:4;  
>> y=-2:1:1;  
>> [X,Y]=meshgrid(x,y);  
>> Z=sin(X).*cos(Y);  
>> meshz(X,Y,Z)  
>>
```



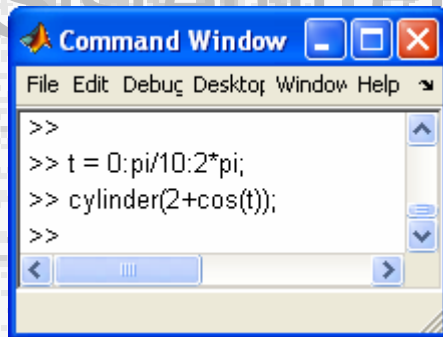
Grafici 3D Solidi di rotazione

Si utilizza la funzione **cylinder**

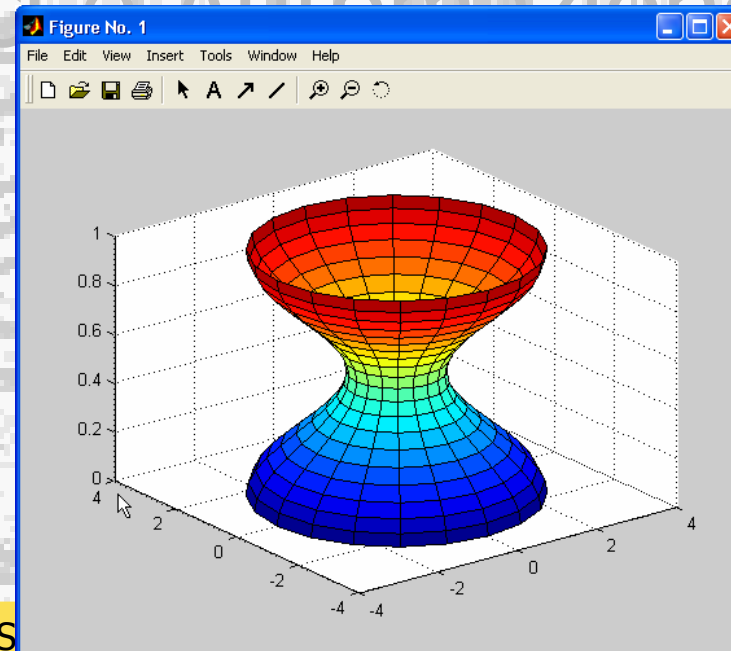
Sintassi

cylinder(r,n) n= numero di punti di rotazione (per default n = 20)

disegna la superficie di rotazione con asse diretto lungo l'asse z e con generatrice la curva descritta dal vettore r.



```
>>  
>> t = 0:pi/10:2*pi;  
>> cylinder(2+cos(t));  
>>
```



Grafici 3D Funzioni di due variabili

Per tracciare le curve di livello di una superficie si usa la funzione **contour** e **clabel**

Sintassi

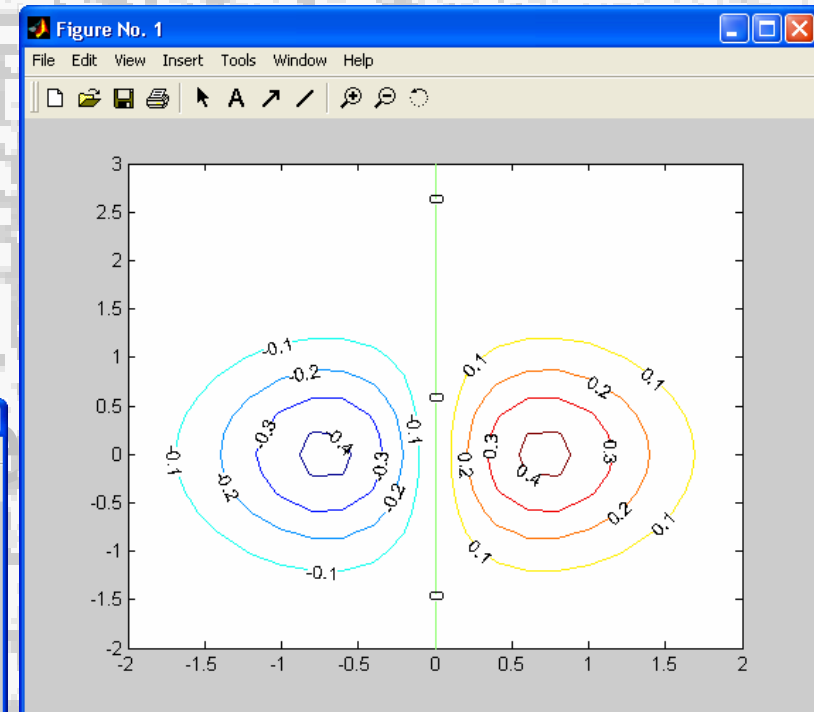
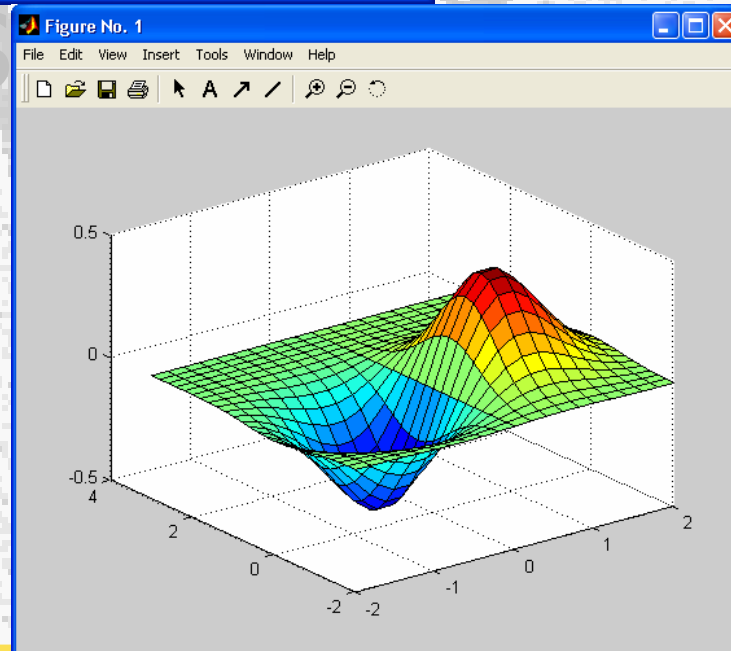
$[C,h]=\text{contour}(X,Y,Z,n)$ disegna n curve di livello della superficie individuata dalle matrici X,Y,Z . Se omesso, n viene scelto in modo automatico.

C è il vettore che contiene i valori di z corrispondenti alle curve di livello.

$\text{clabel}(C,h)$ mette i valori di z alle curve di livello.

Grafici 3D Funzioni di due variabili

```
Command Window
File Edit Debug Desktop Window Help
>> [X,Y]=meshgrid(-2:2:2,-2:2:3);
>> Z = X.*exp(-X.^2-Y.^2);
>> [C,h] = contour(X,Y,Z);
>> clabel(C,h)
>> surf(X,Y,Z)
>> |
```



Grafici 3D Angolo di vista

Per cambiare l'angolo di vista della superficie si usa la funzione **view**

Sintassi

view(azimut, elevazione)

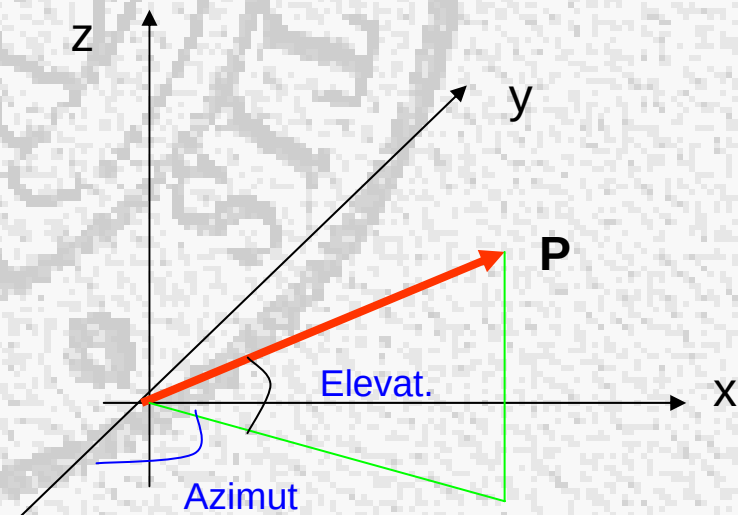
con azimut e elevazione in gradi

i valori di default sono az. = -37.5° e el. = 30°

Oppure

view(x,y,z)

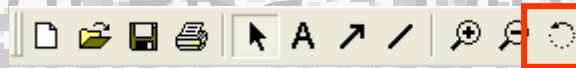
con (x,y,z) coordinate del punto di vista



Grafici 3D Angolo di vista

E' possibile attivare la modifica dell'angolo di vista con l'opzione Rotate 3D

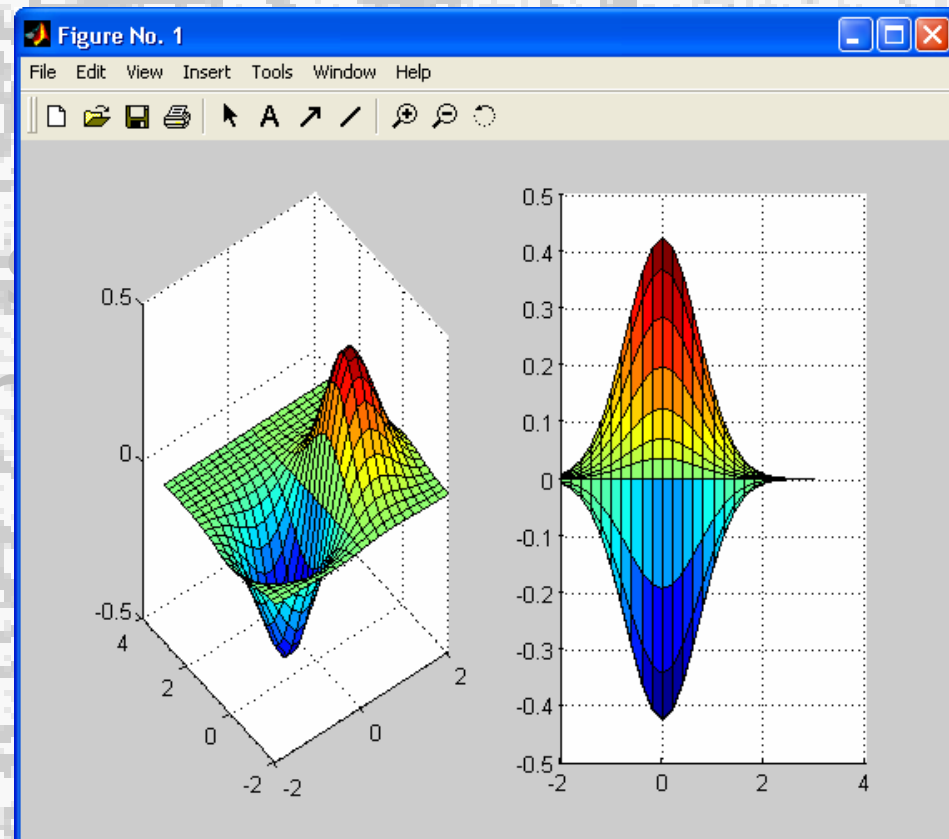
- Dal menu della figura **Tool -> Rotate 3D** oppure dal relativo tasto.



L'angolo di vista viene modificato mediante il mouse.

Grafici 3D Angolo di vista

```
Command Window
File Edit Debug Desktop Window Help
>>
>> [X,Y] = meshgrid(-2:2:2,-2:2:3);
>> Z = X.*exp(-X.^2-Y.^2);
>> subplot(1,2,1)
>> surf(X,Y,Z)
>> subplot(1,2,2)
>> surf(X,Y,Z)
>> view(90,0)
>>
```



Grafici a barre 3D

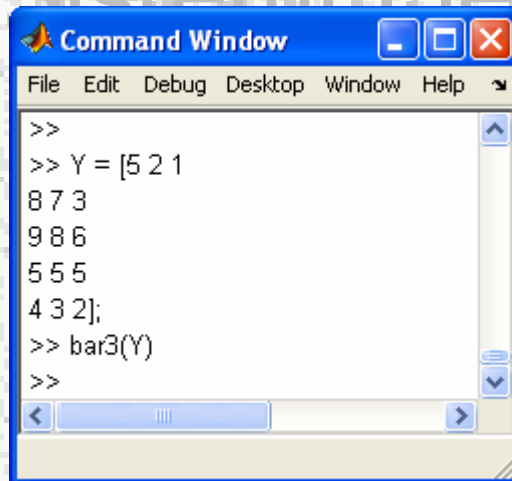
bar3 traccia i grafici 3D a barre verticali

bar3h traccia i grafici 3D a barre orizzontali

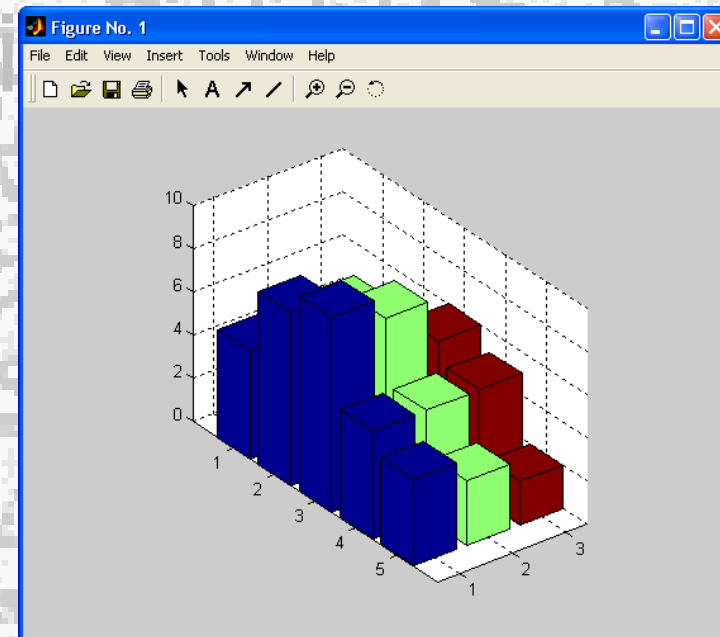
Sintassi (uguale per entrambi)

bar3(x, Y, stile)

valgono le stesse considerazioni viste per il comando bar



```
>>  
>> Y = [5 2 1  
8 7 3  
9 8 6  
5 5 5  
4 3 2];  
>> bar3(Y)  
>>
```

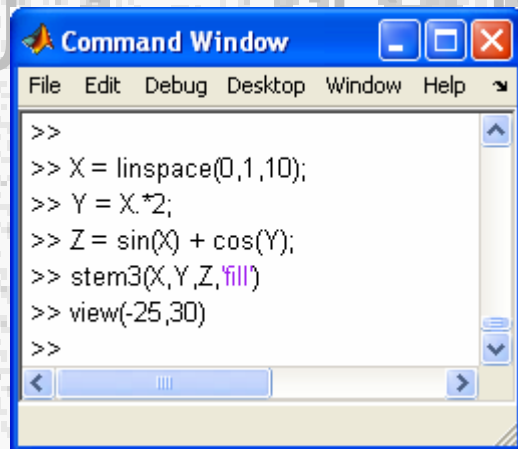


Plot 3D per dati discreti

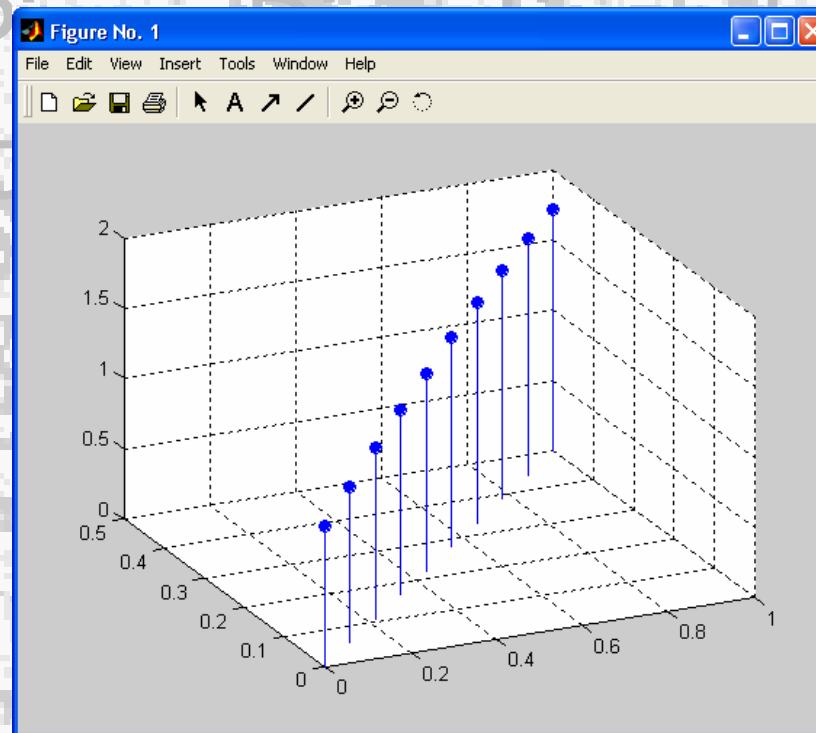
stem3 traccia i dati di una curva come segmenti

Sintassi

stem(X,Y,Z)



```
>>  
>> X = linspace(0,1,10);  
>> Y = X.^2;  
>> Z = sin(X) + cos(Y);  
>> stem3(X,Y,Z,'fill')  
>> view(-25,30)  
>>
```



La programmazione strutturata

Matlab è programmabile sia in maniera strutturata che ad oggetti.

Fino a questo punto abbiamo usato Matlab lavorando in modalità interattiva, ovvero digitando ed eseguendo le istruzioni da tastiera una per volta.

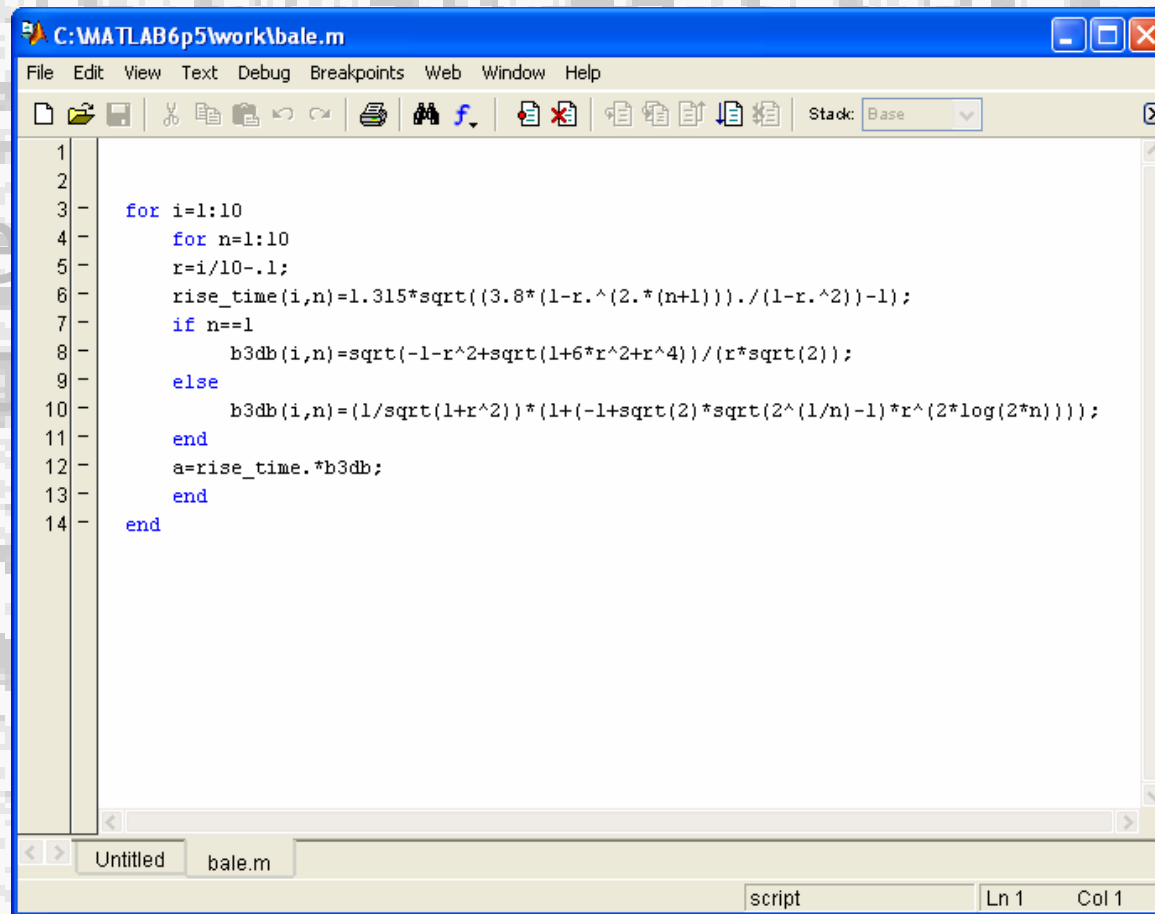
Nel caso si debba ripetere più volte la stessa sequenza di istruzioni o si debbano ripetere le stesse operazioni ma con parametri che cambiano in funzione dei risultati al passo precedente Matlab mette a disposizione due tipi di programmi:

- Script file
- Function file

Matlab mette a disposizione un ambiente di sviluppo detto M-Editor per la creazione dei programmi e il debug.

L'M-Editor

Si lancia da menu **File->new->M-file** oppure dall'icona pagina bianca.



The screenshot shows the MATLAB M-Editor window with the title bar 'C:\MATLAB6p5\work\bale.m'. The menu bar includes File, Edit, View, Text, Debug, Breakpoints, Web, Window, and Help. The toolbar contains icons for file operations and editing. The script content is as follows:

```
1  
2  
3 for i=1:10  
4     for n=1:10  
5         r=i/10-.1;  
6         rise_time(i,n)=1.315*sqrt((3.8*(1-r.^(2.*(n+1)))./(1-r.^2))-1);  
7         if n==1  
8             b3db(i,n)=sqrt(-1-r^2+sqrt(1+6*r^2+r^4))/(r*sqrt(2));  
9         else  
10            b3db(i,n)=(1/sqrt(1+r^2))*(1+(-1+sqrt(2)*sqrt(2^(1/n)-1)*r^(2*log(2*n))));  
11        end  
12        a=rise_time.*b3db;  
13    end  
14 end
```

The status bar at the bottom indicates 'script', 'Ln 1', and 'Col 1'.

Script File

Lo script file è una procedura Matlab scritta con un text editor in ASCII e contiene un insieme di istruzioni da eseguire quando il file è chiamato da Matlab.

In pratica è come scrivere un insieme di istruzioni Matlab al di fuori di Matlab e poi farle eseguire in blocco.

E' un file con estensione .m e per questo è detto anche M-file.

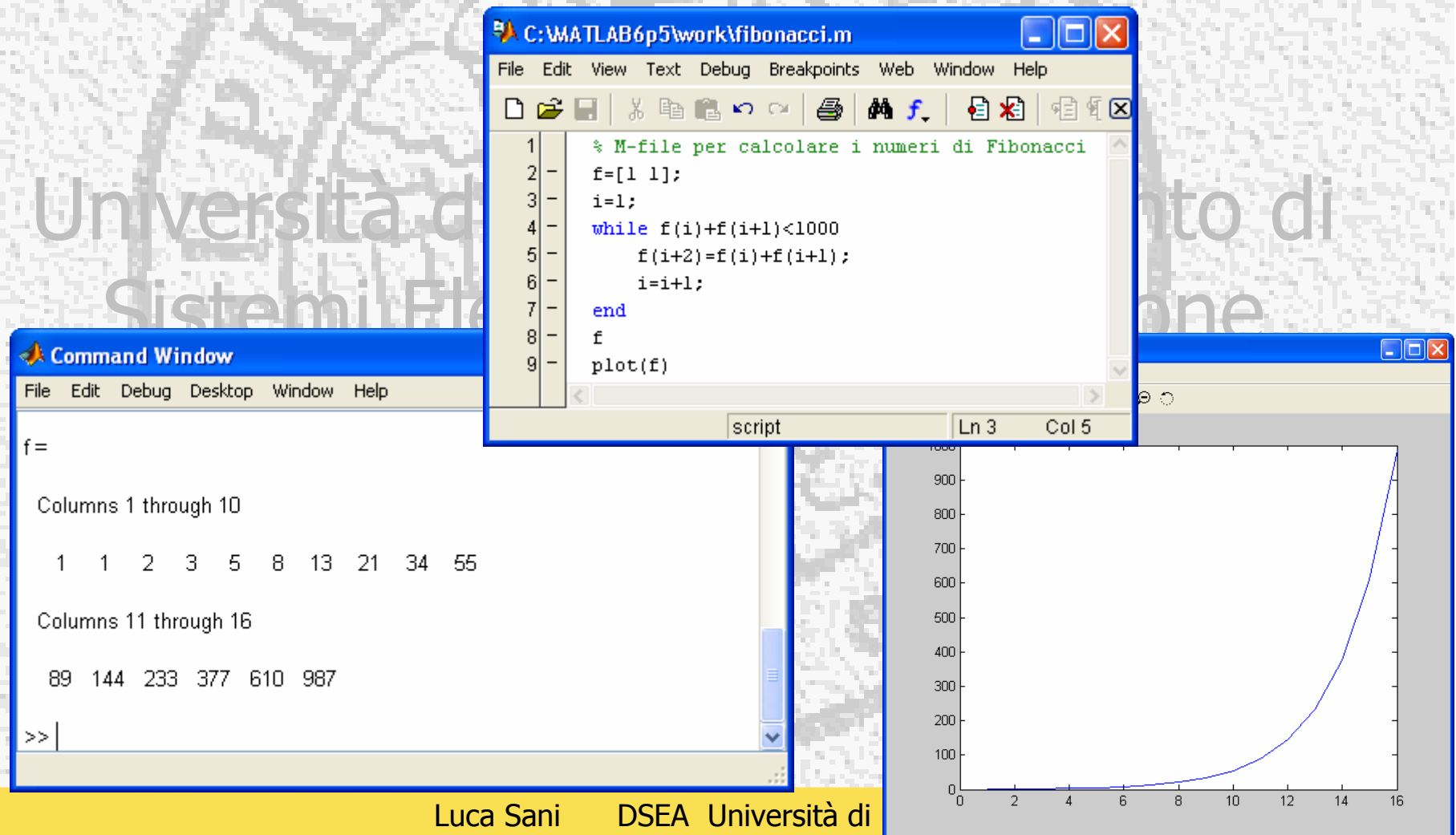
Le righe precedute dal simbolo % sono i commenti

Gli script operano direttamente sulle variabili del workspace e non consentono variabili locali.

Si avvia digitando dal prompt della Command Window il nome del file.

Script File

Ad esempio il codice relativo al calcolo dei numeri di Fibonacci è:



Function File

Una funzione si crea in Matlab in maniera analoga a un file script, tranne che per l'intestazione:

function [var. di uscita]= nomefunzione(var. di ingresso)

In questo caso tutte le variabili sono locali. Il passaggio dei parametri di ingresso avviene per valori.

Per poter lavorare su variabili del workspace direttamente occorre definire tali variabili come **global** dentro e fuori la funzione.

La function deve essere salvata in un file con estensione .m che deve avere lo stesso nome della function.

E' possibile inserire nello stesso file altre functions. Queste saranno visibili solo dalla funzione che dà il nome al file.

Le righe di commento dopo l'intestazione della function sono visualizzate con il comando **help** nomefunzione.

Function File

E' possibile rendere una variabile interna statica con il comando **mlock**

Con il comando **munlock** si annulla il comando precedente.

Una funzione viene eseguita dal prompt dei comandi o all'interno di un file script richiamandola con la seguente sintassi:

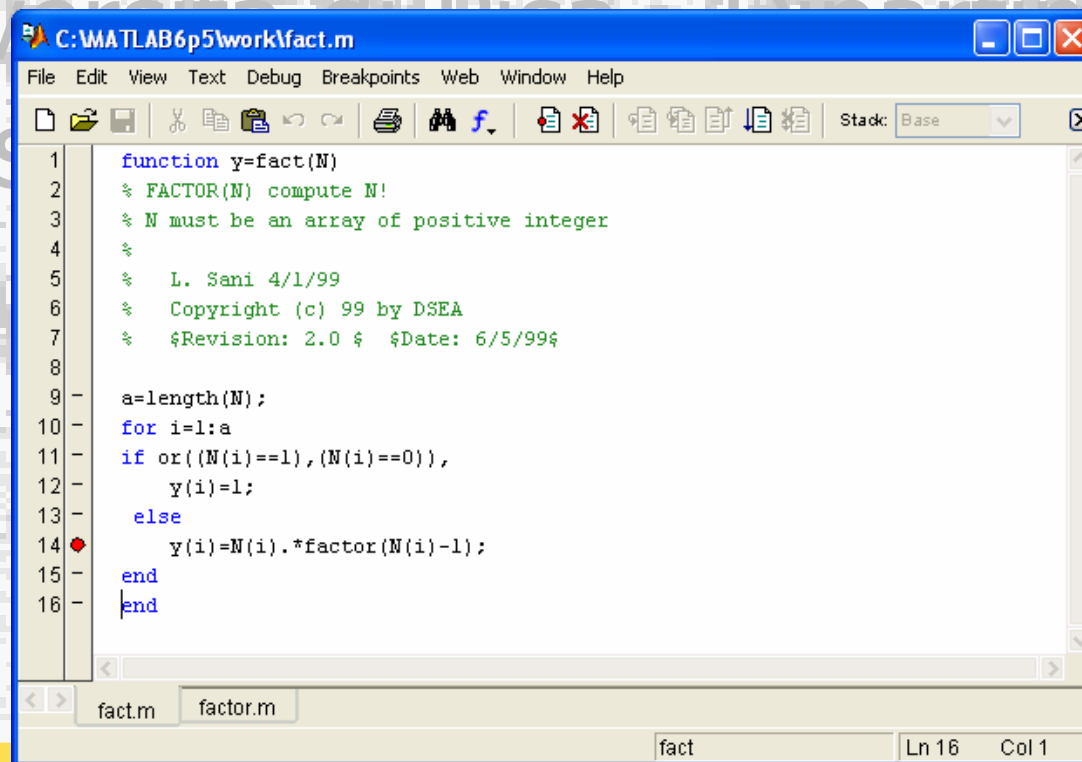
```
[var_di_uscita] = nomefunzione(var ingresso)
```

Se var_di_uscita è omissso il risultato della funzione è memorizzato nella variabile **ans**.

Function File

E' possibile determinare il numero di variabili di ingresso con cui la funzione viene richiamata utilizzando il comando **nargin**.

E' possibile determinare il numero di variabili di uscita con cui la funzione viene richiamata utilizzando il comando **nargout**.



The screenshot shows a MATLAB function file editor window titled 'C:\MATLAB6p5\work\fact.m'. The window contains the following code:

```
1 function y=fact(N)
2 % FACTOR(N) compute N!
3 % N must be an array of positive integer
4 %
5 % L. Sani 4/1/99
6 % Copyright (c) 99 by DSEA
7 % $Revision: 2.0 $ $Date: 6/5/99$
8
9 a=length(N);
10 for i=1:a
11 if or( (N(i)==1), (N(i)==0) ),
12     y(i)=1;
13 else
14     y(i)=N(i).*factor(N(i)-1);
15 end
16 end
```

The code defines a function `fact(N)` that computes the factorial of `N`. It includes comments about the function's purpose, input requirements, and author information. The function uses a loop to calculate the factorial for each element in the input array `N`.

Strutture di controllo

Matlab mette a disposizione otto diversi tipi di strutture di controllo del flusso di istruzioni:

- if, else, elseif
- switch
- while
- for
- continue
- break
- try, catch
- return

if, else, elseif

- if

Sintassi

if **condizione logica**

istruzioni

end

Esempio:

```
if rem(a,2) == 0
```

```
    disp('a pari')
```

```
    b = a/2;
```

```
end
```

if, else, elseif

- if else

Sintassi

if **condizione logica**

istruzioni

else

istruzioni

end

if, else, elseif

Esempio:

```
if rem(a,2) == 0
    disp('a pari')
else
    disp('a dispari')
end
```

if, else, elseif

- if elseif

Sintassi

if **condizione logica**

istruzioni

elseif **condizione logica**

istruzioni

else

istruzioni

end

switch

Sintassi

swich **variabile**

case **valore 1**

istruzioni

case **valore 2**

istruzioni

...

otherwise

istruzioni

end

a differenza del C, se ad esempio, 'case valore 1' è vero viene eseguito il blocco di istruzioni relativo e poi si esce dal case, non occorre inserire un'istruzione di break.

switch

Esempio:

```
switch input_num
```

```
    case -1
```

```
        disp('Negativo e pari a -1');
```

```
    case 0
```

```
        disp('zero');
```

```
    case 1
```

```
        disp('Positivo e pari a 1');
```

```
    otherwise
```

```
        disp('Altro valore');
```

```
end
```

while

Sintassi

while **condizione**

istruzioni

end

Esempio:

while a<10

a=a+1

end

for

Sintassi

for **indice=inizio:incremento:fine**

istruzioni

end

L'incremento può essere omesso e per default è assunto pari a 1.

Es

for i = 2:6

$x(i) = 2 * x(i-1);$

end

for i = 1:m

 for j = 1:n

$A(i,j) = 1/(i + j - 1);$

 end

end

continue

In un ciclo for o while il controllo passa all'istruzione successiva.

break

Termina l'esecuzione di un ciclo for o while passando il controllo all'istruzione successiva.

return

Termina l'esecuzione di una funzione e ritorna al programma chiamante.

try

Sintassi

try

istruzioni

catch

istruzioni

end

Sono eseguite le istruzioni tra try e catch finché non si verificano errori.
In questo caso sono eseguite le istruzioni dopo catch.

Funzioni di input

Il comando **input** visualizza un testo sullo schermo e aspetta che l'utente digiti qualcosa e poi registra l'input nella variabile specifica.

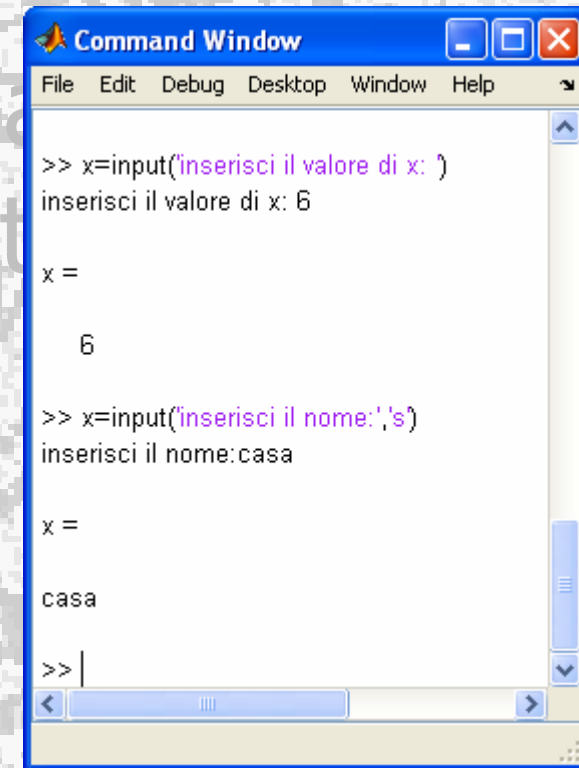
Sintassi

nel caso di input numerico

`Var_input=input('Testo')`

nel caso di input di una stringa

`Var_input=input('Testo'; 's')`



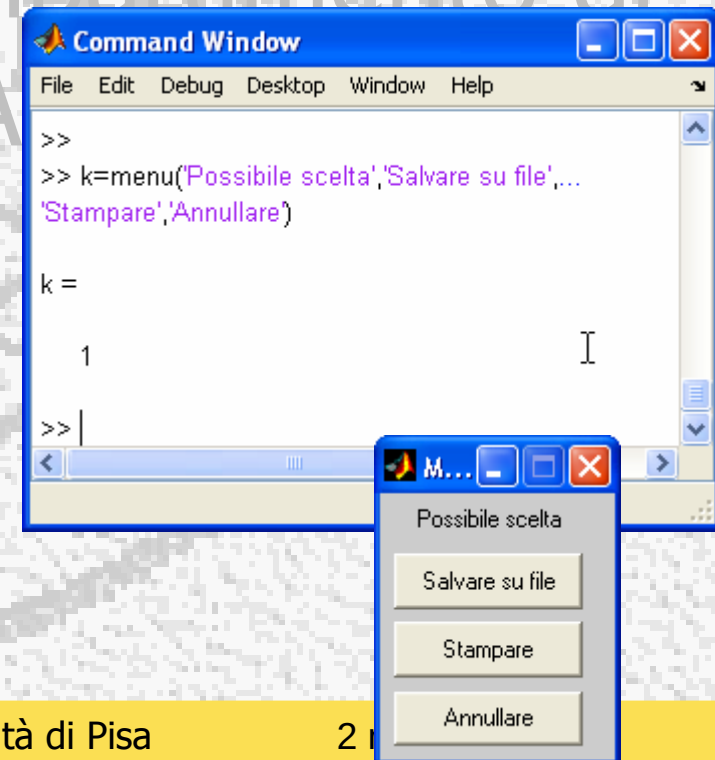
Funzioni di input

Il comando **menu** genera un menù di opzioni all'utente

Sintassi

Var_input=menu('Titolo', 'opzione1', 'opzione2,...')

Questo comando visualizza un menu il cui titolo è quello specificato dalla variabile stringa titolo e le cui opzioni sono opzione1, opzione2... Il valore che assume var_input dipende dall'opzione scelta: 1 per opzione1, 2 per opzione2 e così via.



Funzioni di sospensione

Il comando **pause** sospende l'esecuzione dello script

Sintassi

pause

Sospende l'esecuzione finché l'utente non preme un tasto.

pause(n)

Sospende l'esecuzione per n secondi.

Il comando **keyboard** sospende l'esecuzione dello script

Il prompt diventa K>>

E' possibile esaminare lo stato della memoria o eseguire altri programmi.

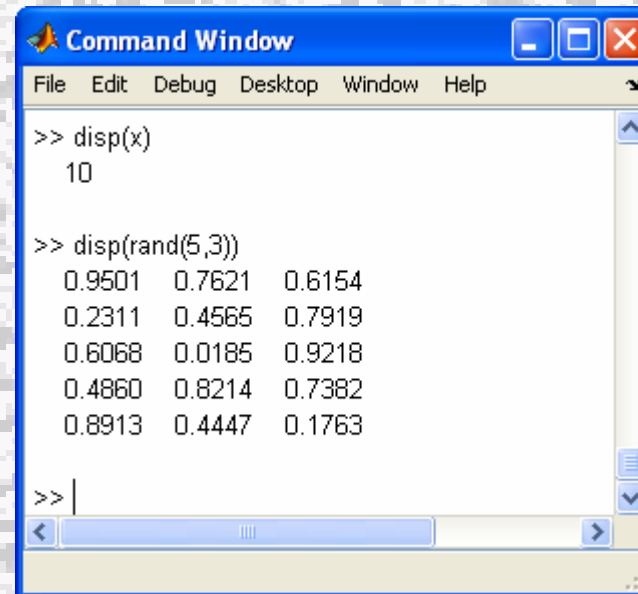
Si riprende l'esecuzione del programma con la funzione return.

Funzioni di output

Il comando **disp** visualizza su schermo il contenuto di una variabile.

Sintassi

disp(var)



The screenshot shows the MATLAB Command Window with the following content:

```
>> disp(x)
10

>> disp(rand(5,3))
0.9501 0.7621 0.6154
0.2311 0.4565 0.7919
0.6068 0.0185 0.9218
0.4860 0.8214 0.7382
0.8913 0.4447 0.1763

>> |
```

The window has a menu bar with File, Edit, Debug, Desktop, Window, and Help. The output is displayed in a scrollable area with a scrollbar on the right.

Lo studio di funzioni matematiche

Matlab mette a disposizione una serie di funzioni per lo studio di funzioni matematiche in una o più variabili.

Definizione di funzione

Esistono due modalità:

- con una function
- con il comando **inline**

Esempio

$$humps(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

il function file é

```
function y=humps(x)
```

```
y=1./((x-.3).^2+.01)+1./((x-.9).^2+.04)-6;
```

Lo studio di funzioni matematiche

$$humps(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

con il comando **inline**

```
f=inline('1./((x-.3).^2+.01)+1./((x-.9).^2+.04)-6')
```

Lo studio di funzioni matematiche

E' possibile definire in un function file più di una funzione.

Ad ogni funzione matematica deve corrispondere una colonna della matrice di uscita.

Esempio.

```
function Y = myfun(x)
```

```
Y(:,1) = 200*sin(x(:))./x(:);
```

```
Y(:,2) = x(:).^2;
```

Lo studio di funzioni matematiche a più variabili

E' possibile definire un function file per una funzione a più variabili. In questo caso le varie variabili devono essere le componenti del vettore di ingresso.

Esempio.

```
function Y = seno2(u)
```

```
x=u(1);
```

```
y=u(2);
```

```
z=sin(x).*cos(y)
```

Lo studio di funzioni matematiche

Per valutare il valore della funzione in un set di valori si utilizza la funzione **feval**

Sintassi

`y=feval(nome_f, [x1 x2 .. xn])` se la funzione è definita con inline di

`y=feval(@nome_f, [x1 x2 .. xn])` se la funzione è definita con un function file

y è il vettore che contiene i valori della funzione valutata nei punti x1, x2, .. xn.

Se il function file definisce più funzioni, y è una matrice in cui la i-esima colonna rappresenta i valori della i-esima funzione valutata nei punti x1, x2, .. xn.

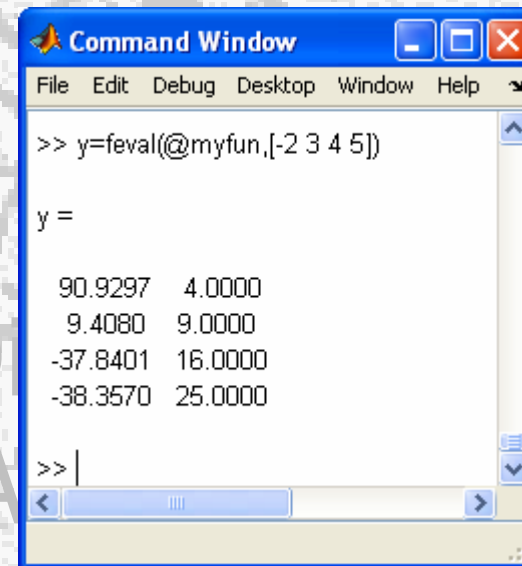
Lo studio di funzioni matematiche

Esempio.

```
function Y = myfun(x)
```

```
Y(:,1) = 200*sin(x(:))./x(:);
```

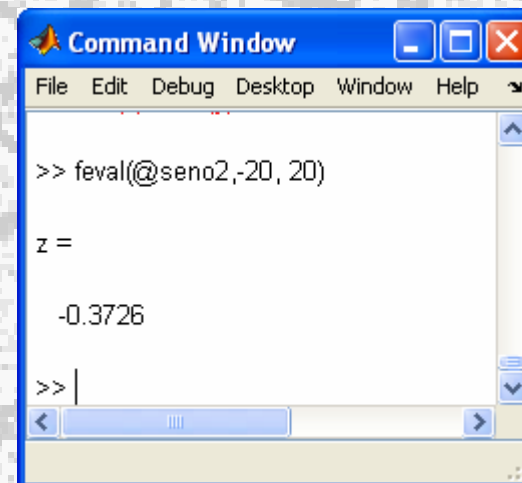
```
Y(:,2) = x(:).^2;
```



```
Command Window
File Edit Debug Desktop Window Help
>> y=feval(@myfun,[-2 3 4 5])
y =
    90.9297    4.0000
     9.4080    9.0000
   -37.8401   16.0000
   -38.3570   25.0000
>> |
```

```
function Y = seno2(x,y)
```

```
z=sin(x).*cos(y)
```



```
Command Window
File Edit Debug Desktop Window Help
>> feval(@seno2,-20,20)
z =
   -0.3726
>> |
```

Lo studio di funzioni matematiche Plot

Per tracciare il grafico si utilizza la funzione **fplot**

Sintassi

fplot(nome_f, [xmin xmax]) se la funzione è definita con inline

fplot(@nome_f, [xmin xmax]) se la funzione è definita con un
function file

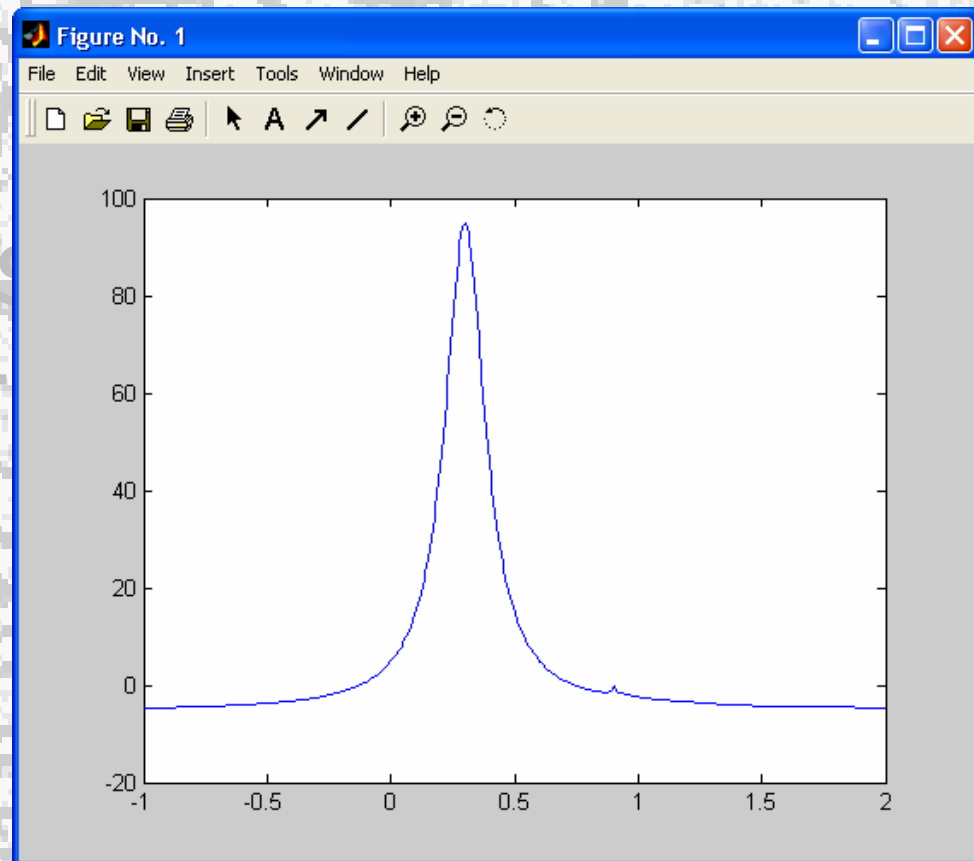
funziona solo per funzioni in una variabile

Lo studio di funzioni matematiche Plot

Esempio

```
fplot(@humps, [-1 2])
```

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



Lo studio di funzioni matematiche Plot

Nel caso in cui nel fuction file si abbia definito più di una funzione con l'istruzione fplot vengono visualizzate tutte le funzioni.

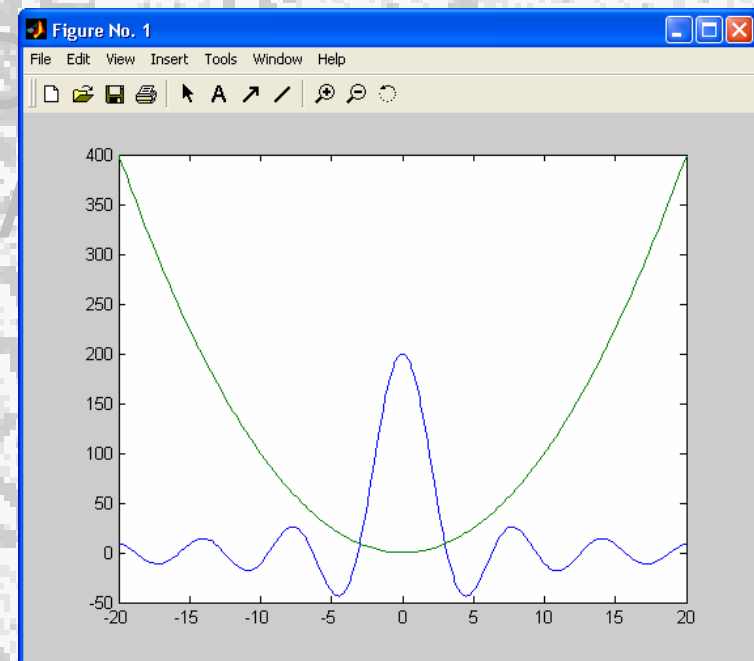
Esempio.

```
function Y = myfun(x)
```

```
Y(:,1) = 200*sin(x(:))./x(:);
```

```
Y(:,2) = x(:).^2;
```

```
fplot(@myfun, [-20 20])
```



Lo studio di funzioni matematiche. Integrale definito per funzioni in una variabile

Si utilizza la funzione **quad**

Sintassi

var=quad(@nome_f, a,b)

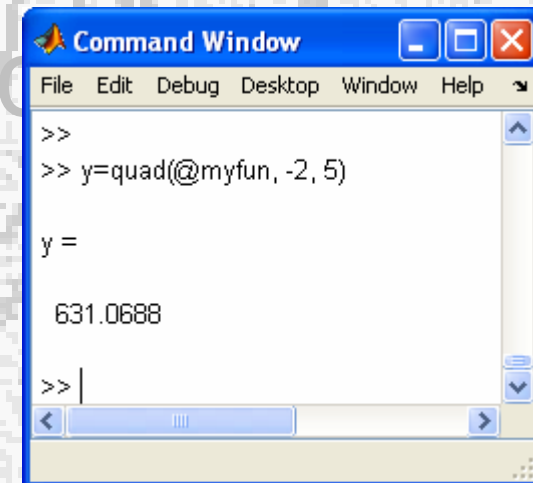
a e **b** sono gli estremi di integrazione

Esempio.

function Y = myfun(x)

Y(:,1) = 200*sin(x(:))./x(:);

Y(:,2) = x(:).^2;



Lo studio di funzioni matematiche. Integrale definito per funzioni in una variabile

Si utilizza la funzione **dblquad**

Sintassi

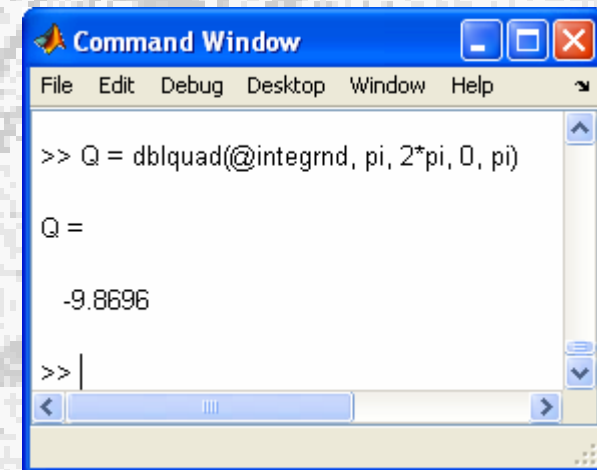
var=dblquad(@nome_f, xmin, xmax, ymin, ymax)

xmin, xmax, ymin, ymax sono gli estremi di integrazione

Esempio.

function z = integrnd(x, y)

z = y*sin(x)+x*cos(y);



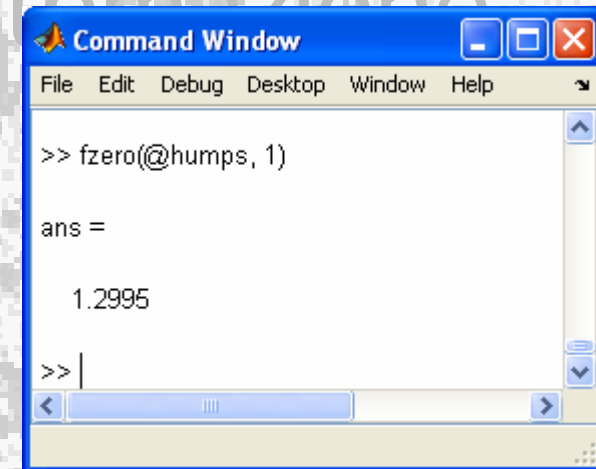
Lo studio di funzioni matematiche. Ricerca di zeri

Per le funzioni in una variabile si usa la funzione **fzero**

Sintassi

var=fzero(@nome_f, x0)

x0 punto prossimo allo zero



Lo studio di funzioni matematiche Ricerca di minimi

Per le funzioni in una variabile si usa la funzione **fminbnd**

Sintassi

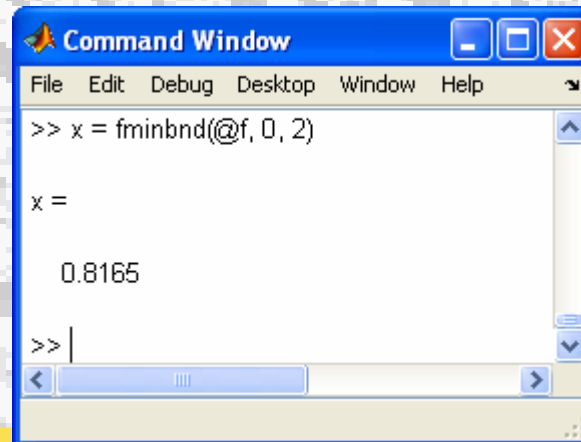
var=fminbnd(@nome_f, x1,x2)

x1, x2 è il punto prossimo al il minimo locale

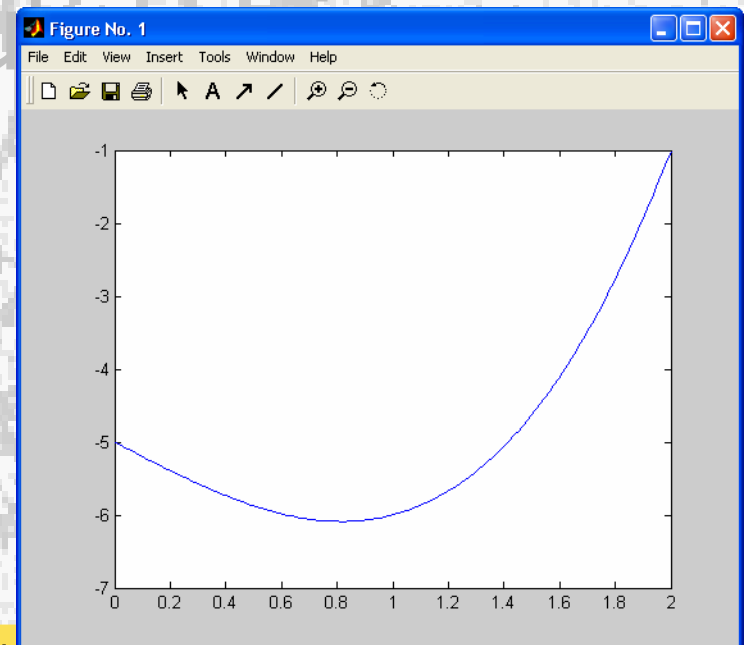
Esempio:

function $y = f(x)$

$y = x.^3 - 2*x - 5;$



```
Command Window
File Edit Debug Desktop Window Help
>> x = fminbnd(@f, 0, 2)
x =
    0.8165
>> |
```



Lo studio di funzioni matematiche Ricerca di minimi

Per le funzioni in più variabili si usa la funzione **fminsearch**

Sintassi

$[x, fval] = fmin(@nome_f, [x1, x2])$

$x1, x2$ è l'intervallo in cui si ricerca il minimo locale

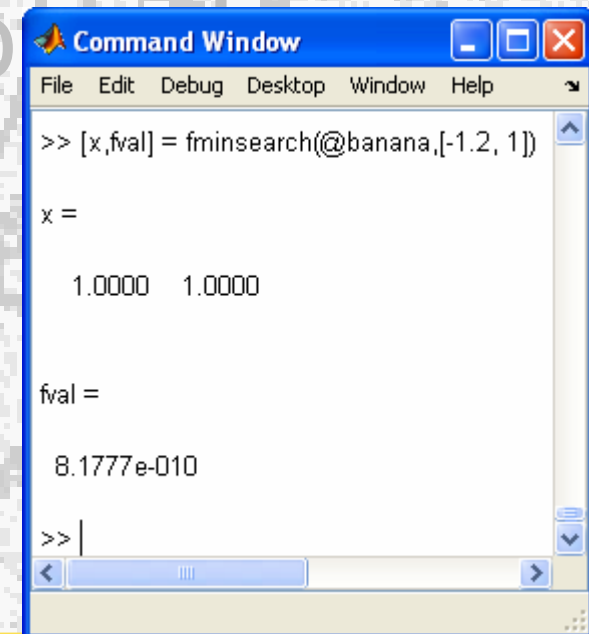
x coordinate del minimo

$fval$ valore del minimo

Esempio:

function f = banana(x)

f = 100*(x(2)-x(1)^2)^2 +
+(1-x(1))^2;



```
Command Window
File Edit Debug Desktop Window Help
>> [x,fval] = fminsearch(@banana,[-1.2, 1])
x =
    1.0000    1.0000
fval =
    8.1777e-010
>>
```