

Introduzione a Matlab

Ing. Anna Maria Vegni

avegni@uniroma3.it

Indice

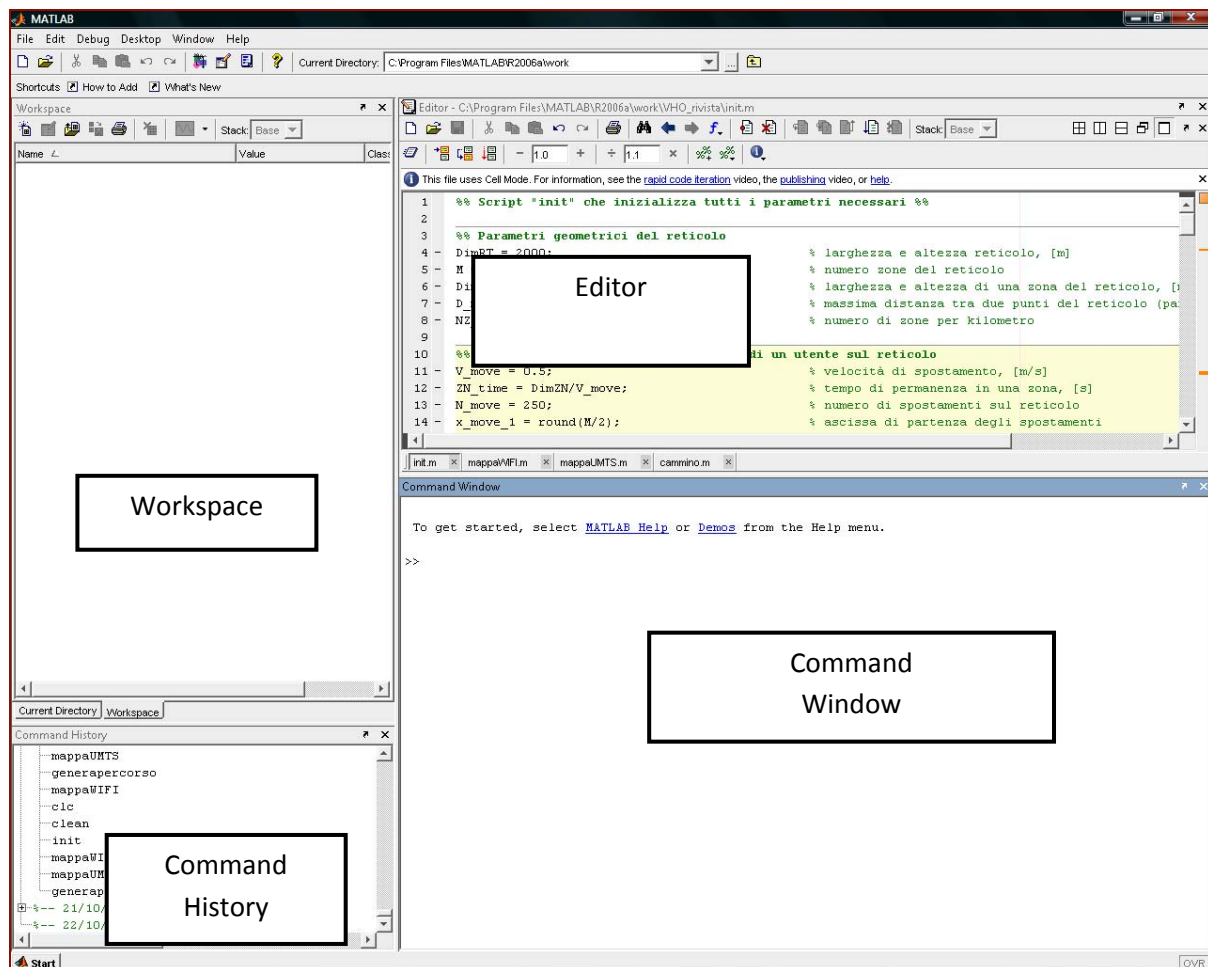
Indice	2
Introduzione	3
Help in Matlab	4
Files di Matlab.....	5
Le variabili in Matlab.....	6
Matrici in Matlab	9
Operazioni matriciali.....	12
Max e Min di una matrice	15
Sort di una matrice	16
Sum e Prod di una matrice.....	17
Strutture di controllo	18
Struttura IF	18
Struttura FOR	19
Struttura WHILE	19
Plot.....	20
Grafici sovrapposti	22
Funzione STEM.....	24
Funzioni speciali.....	26
RAND.....	26
ZEROS.....	26
ONES	27
FIND	28
EYE	29
Operazioni sulle matrici	30
Le stringhe	30
Operatori relazionali	31
Operazioni sui polinomi	34

Introduzione

MATLAB è nato principalmente come programma destinato alla gestione di **matrici**, da qui il nome **MatLab (MATrix LABoratory)**. Successivamente, il programma è stato sviluppato per analisi numeriche molto più complesse.

La linea di comando di MATLAB è indicata da un prompt come in DOS. Accetta dichiarazioni di variabili, espressioni e chiamate a tutte le funzioni disponibili nel programma.

Le funzioni di MATLAB sono file di testo che vengono eseguiti semplicemente digitandone il nome sulla linea di comando.



- Il **Workspace** è un riquadro in cui sono rappresentate tutte le variabili al momento esistenti, e quindi utilizzabili.
- Il **Command History** è un riquadro in cui sono memorizzate tutte le istruzioni digitate nel prompt di MatLab, ordinate per giorno e ora.
- Il **Command Window** è un riquadro dove è possibile digitare le istruzioni di Matlab, (rappresenta il prompt dei comandi).
- L'**Editor** è un riquadro in cui è possibile scrivere funzioni o script di Matlab, e quindi salvare tali file con estensione .m.

Help in Matlab

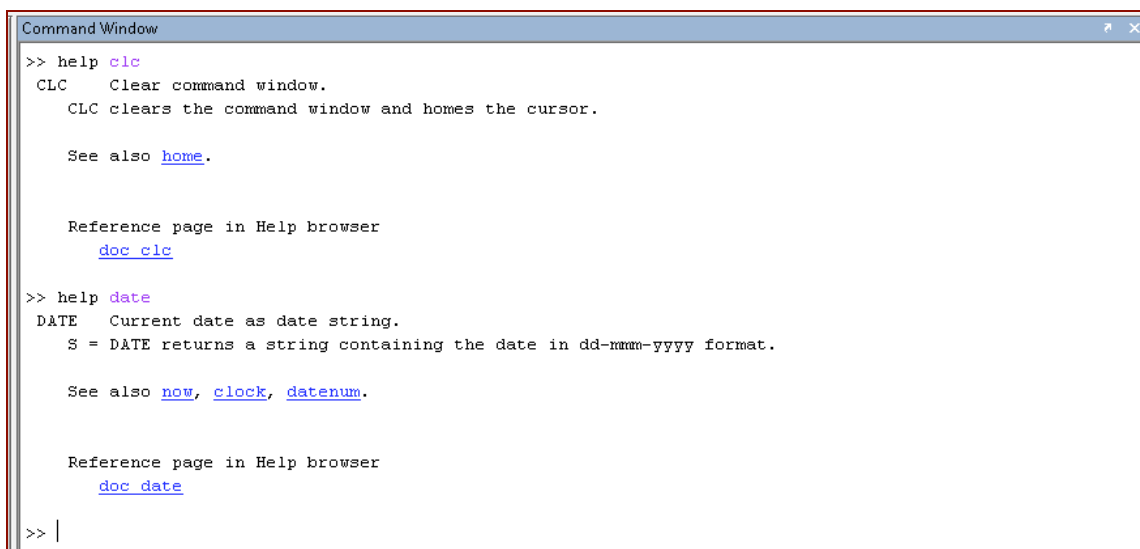
MATLAB presenta un help in linea con informazioni sulla sintassi di tutte le funzioni disponibili. Esistono 3 principali funzioni che possono essere utilizzate per ottenere informazioni relative a una certa funzione:

- **help**, dà informazioni sulle varie categorie di funzioni disponibili (*toolbox*);
- **helpwin**, (abbreviazione per *help window*);
- **doc**, (abbreviazione per *documentation*).

Help e **helpwin** danno la stessa informazione ma in una finestra differente. **Doc** restituisce una pagina HTML con maggiori informazioni.

Per accedere a tali informazioni, basta digitare:

help nome_funzione



```
Command Window
>> help clc
CLC    Clear command window.
       CLC clears the command window and homes the cursor.

       See also home.

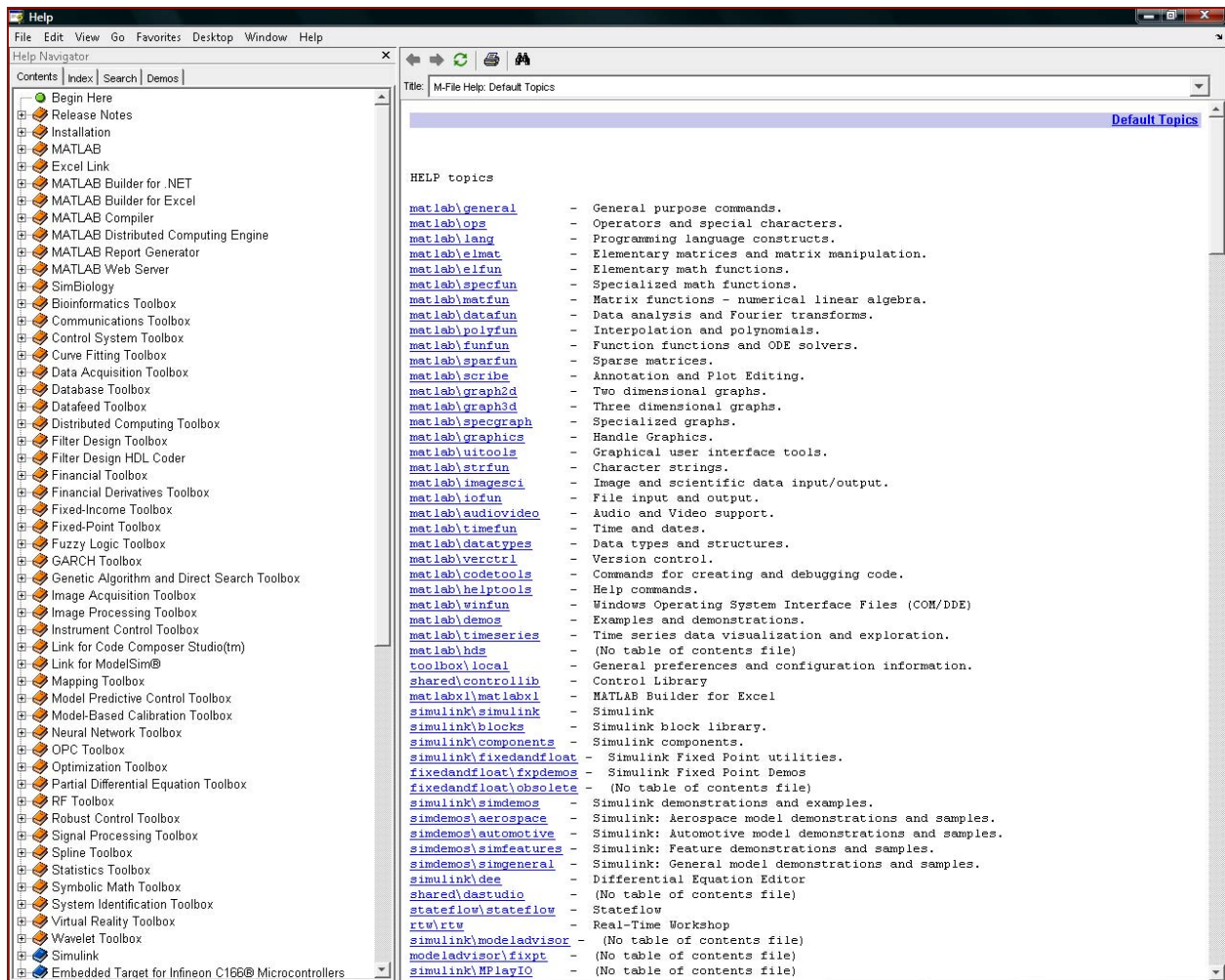
       Reference page in Help browser
       doc clc

>> help date
DATE   Current date as date string.
       S = DATE returns a string containing the date in dd-mm-yy format.

       See also now, clock, datetime.

       Reference page in Help browser
       doc date

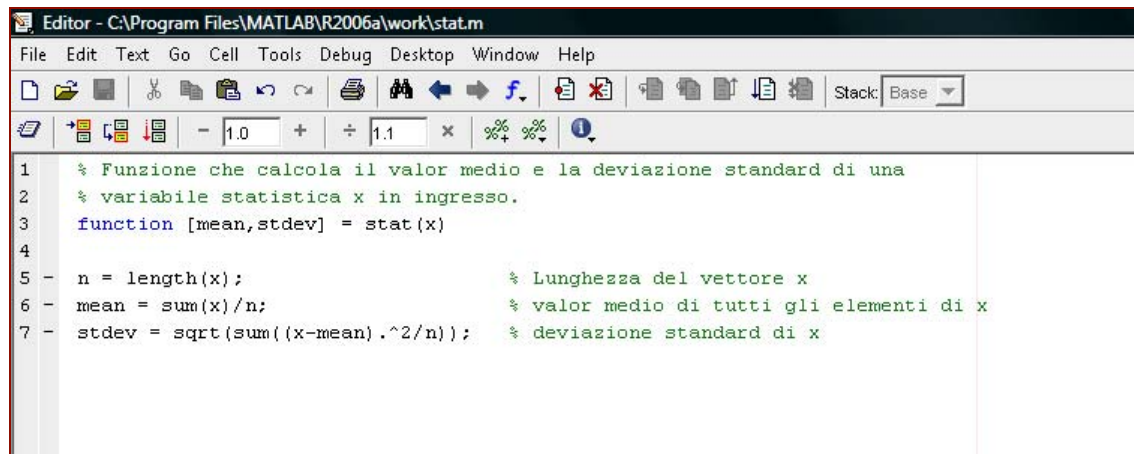
>> |
```



Files di Matlab

I files interpretati da Matlab sono file di testo ASCII con estensione **.m**, generati tramite l'Editor e vengono eseguiti digitando il nome sulla linea di comando (**senza estensione!**).

- **N. B.** Se si scrive una funzione, il nome del file con estensione **.m** deve essere **NECESSARIAMENTE** quello della funzione stessa.
- Le istruzioni possono essere contenute in un file **.m**, oppure digitate direttamente dalla linea di comando.
- **N. B.** Se un'istruzione non viene terminata da un punto e virgola, allora verrà visualizzato il risultato dell'applicazione dell'istruzione.
- I commenti vengono inseriti semplicemente inserendo all'inizio di ogni linea di commento il percento **%**.



```

Editor - C:\Program Files\MATLAB\R2006a\work\stat.m
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons for File, Edit, Text, Go, Cell, Tools, Debug, Desktop, Window, Help]
[Icons for Undo, Redo, Cut, Copy, Paste, Find, Replace, Print, Run, Stop, Step Through, Breakpoints, Command Window, Workspace, Stack]
- 1.0 + 1.1 x %>% %>% %>%
1 % Funzione che calcola il valor medio e la deviazione standard di una
2 % variabile statistica x in ingresso.
3 function [mean,stdev] = stat(x)
4
5 - n = length(x); % Lunghezza del vettore x
6 - mean = sum(x)/n; % valor medio di tutti gli elementi di x
7 - stdev = sqrt(sum((x-mean).^2/n)); % deviazione standard di x
  
```

Le variabili in Matlab

- Le variabili seguono le regole dei linguaggi di programmazione. MATLAB è *case-sensitive* e accetta nomi di variabili lunghi fino ad un massimo di 19 caratteri alfanumerici, con il primo obbligatoriamente alfabetico. Ad esempio, “Pippo”, “PiPPo”, “PIPPo”, e “pippo” vengono considerate come variabili distinte.
- Sono ammessi solo caratteri alfabetici (es., “A-Z”), numeri, e il carattere underscore (es., “_”). Non sono ammessi spazi nei nomi delle variabili. Ad esempio, non si può scrivere come nome di una variabile “la mia variabile”, ma “la_mia_variabile” è accettato.
- L’istruzione **who** dà informazioni sulle variabili presenti nel Workspace. Per cancellare una variabile, basta scrivere l’istruzione **clear nome_variabile**, oppure se si vuole cancellare tutto il contenuto del Workspace, si digita **clear all**. Per cancellare il testo che appare nel CommandWindow, basta scrivere **clc**.

The screenshot shows the MATLAB interface with the Command Window and Workspace. The Command Window contains the following code:

```
>> 20 + 10
ans =
    30
>> a = 20 + 5
a =
    25
>> a = 20 + 5;
>> whos
      Name      Size      Bytes  Class
      a         1x1         8  double array
      ans        1x1         8  double array
Grand total is 2 elements using 16 bytes
>> who
Your variables are:
a      ans
>> clear a
>>
```

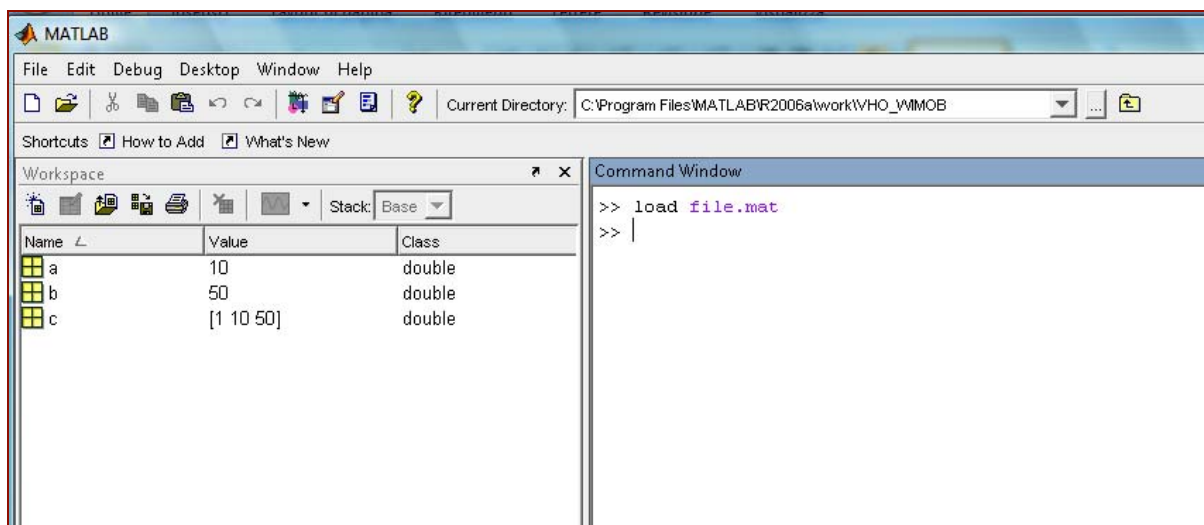
The Workspace window shows the variable 'ans' with a value of 30 and a class of double.

- L'istruzione **save** salva tutte le variabili in memoria sul file specificato, in vari formati; **load** richiama in memoria le variabili salvate sul file specificato; **what** dà l'elenco di tutte le funzioni MATLAB nell'area di lavoro (estensione .m) e dei file di dati che sono stati salvati (estensione .mat)
- L'istruzione **load**, seguita dal nome del file dove sono state salvate le variabili ('file.mat'), permette di richiamare le variabili memorizzate.

The screenshot shows the MATLAB interface with the Command Window and Workspace. The Command Window contains the following code:

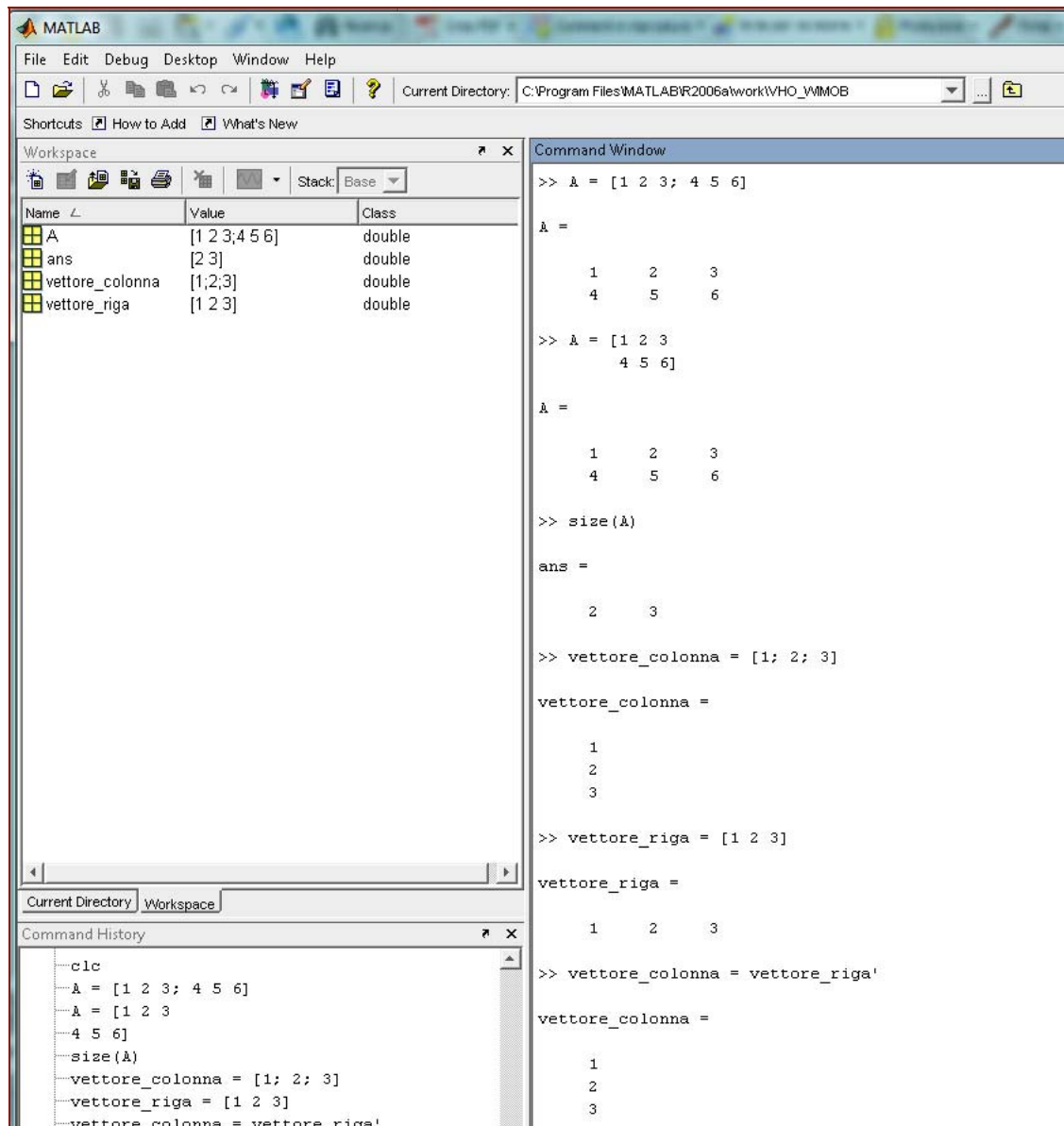
```
>> a = 10;
>> b = 50;
>> c = [1 10 50]
c =
     1    10    50
>> savefile = 'file.mat';
>> save(savefile, 'a', 'b', 'c');
>>
```

The Workspace window shows the variables 'a' (10), 'b' (50), 'c' ([1 10 50]), and 'savefile' ('file.mat').



Matrici in Matlab

In MATLAB le matrici vengono definite all'interno di una coppia di parentesi quadre ([...]). Per distinguere un elemento dal successivo, e quindi identificare una riga o una colonna di una matrice, si usano rispettivamente gli spazi oppure il punto e virgola.



- È possibile creare matrici anche inserendo due o più vettori/matrici preesistenti.

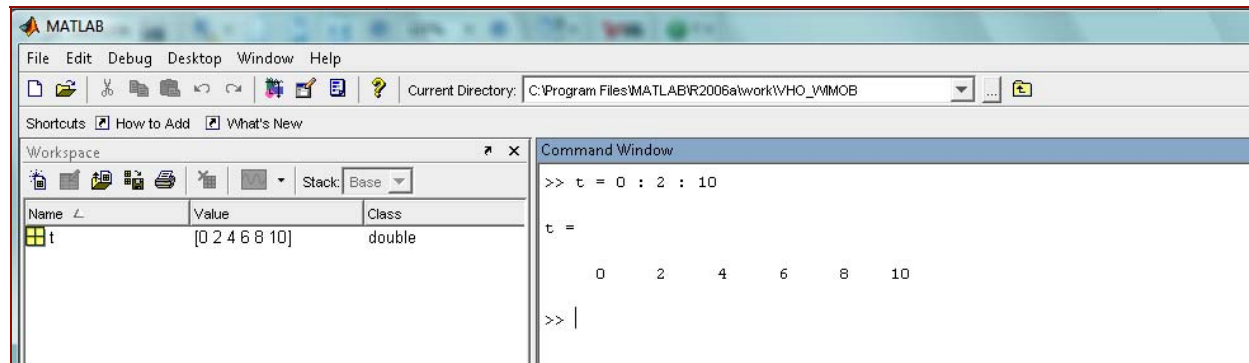
```
>> B = [vettore_riga; vettore_colonna']

B =
     1     2     3
     1     2     3

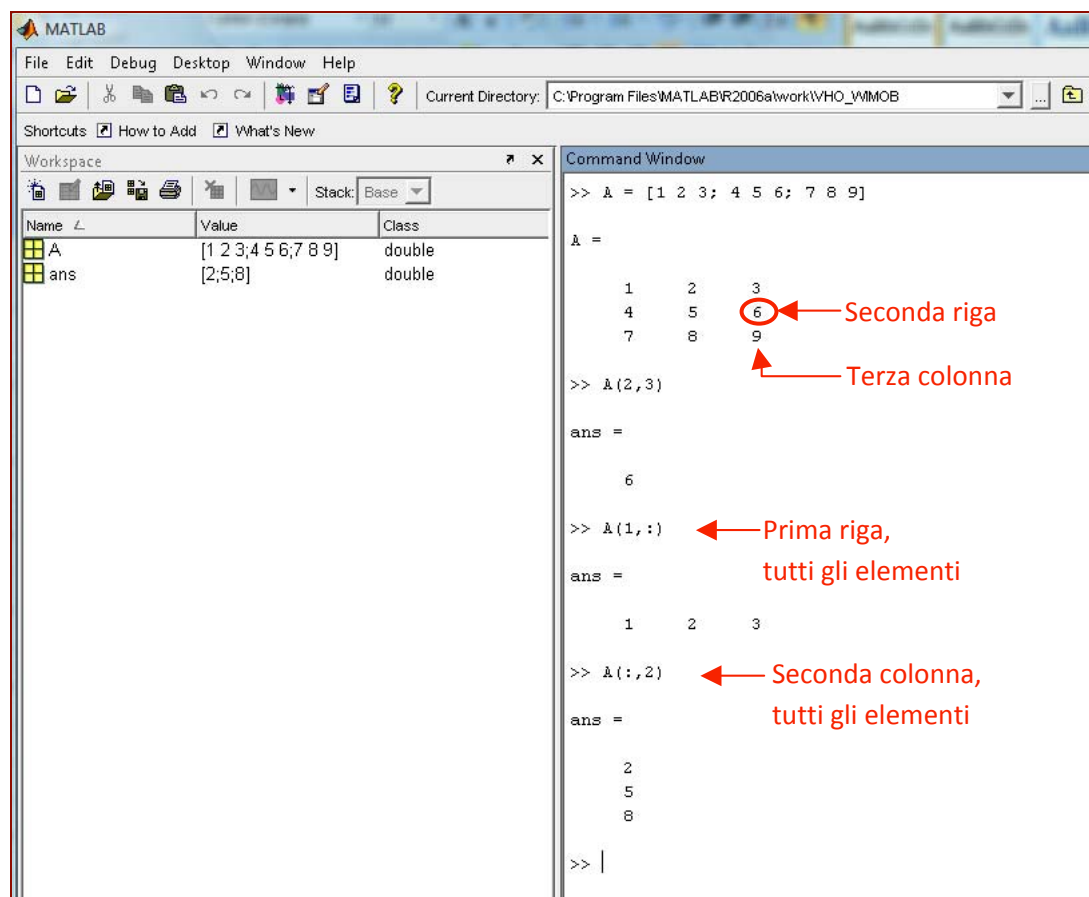
>> |
```

- Per creare vettori manualmente, è possibile usare l'istruzione

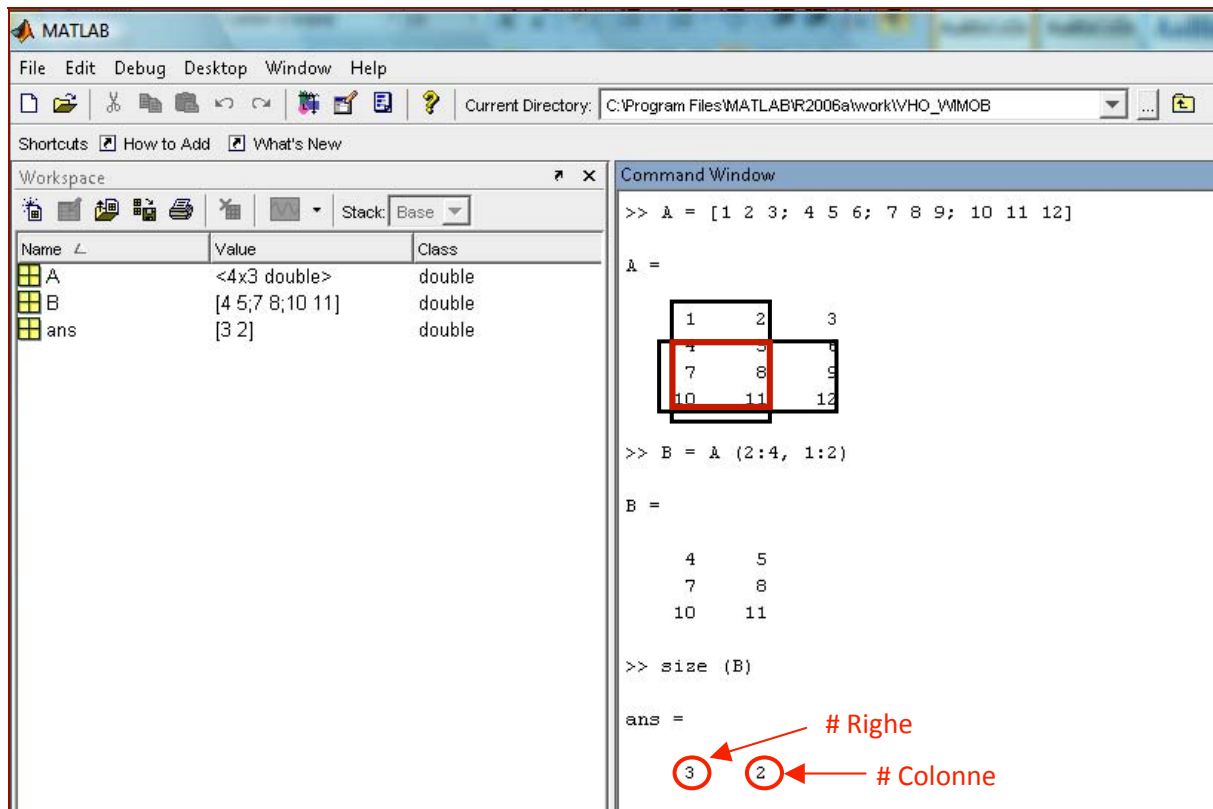
VALORE INIZIALE : INCREMENTO : VALORE FINALE



- Per selezionare un elemento, una riga o una colonna di una matrice, si può utilizzare l'indice dell'elemento, della riga o della colonna. In generale, gli indici iniziano dal valore 1.



- È possibile estrarre un sottoinsieme contiguo di una matrice, facendo riferimento ad un intervallo delle righe ed uno delle colonne.
- Se ad esempio una matrice ha dimensione [4x3], allora si potrà selezionare la sottomatrice che prevede le righe [2:4] e le colonne [1:2].



```

MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\Program Files\MATLAB\R2006a\work\WHO_VMMOB

Workspace
Name | Value | Class
---|---|---
A | <4x3 double> | double
B | [4 5; 7 8; 10 11] | double
ans | [3 2] | double

Command Window
>> A = [1 2 3; 4 5 6; 7 8 9; 10 11 12]
A =
     1     2     3
     4     5     6
     7     8     9
    10    11    12

>> B = A (2:4, 1:2)
B =
     4     5
     7     8
    10    11

>> size (B)
ans =
     3     2
# Righe
# Colonne
  
```

- Per modificare il valore di un elemento all'interno di una matrice, (o una riga o una colonna), basta indicare l'elemento (o la riga o la colonna) e assegnargli un nuovo valore.

```

>> B(1,2) = 0;
>> B(3,:) = [0 0];
>> B
B =
     4     0
     7     8
     0     0

>>
  
```

Operazioni matriciali

L'operazione tra elementi di una matrice viene fatta tramite i seguenti comandi:

- Moltiplicazione *elemento per elemento*: `".*";`
- Divisione *elemento per elemento*: `"./";`
- Addizione *elemento per elemento*: `."+";`
- Sottrazione *elemento per elemento*: `".-";`
- Elevazione a potenza *elemento per elemento*: `".^";`

Attenzione: Le matrici devono essere della stessa dimensione, (es. $A_{n \times m}$ e $B_{n \times m}$).

The screenshot shows the MATLAB R2006a environment. The **Workspace** window on the left lists four variables: A, B, C, and D, all of size 4x3 double. The **Command Window** on the right shows the following commands and outputs:

```
>> A
A =
     1     2     3
     4     5     6
     7     8     9
    10    11    12

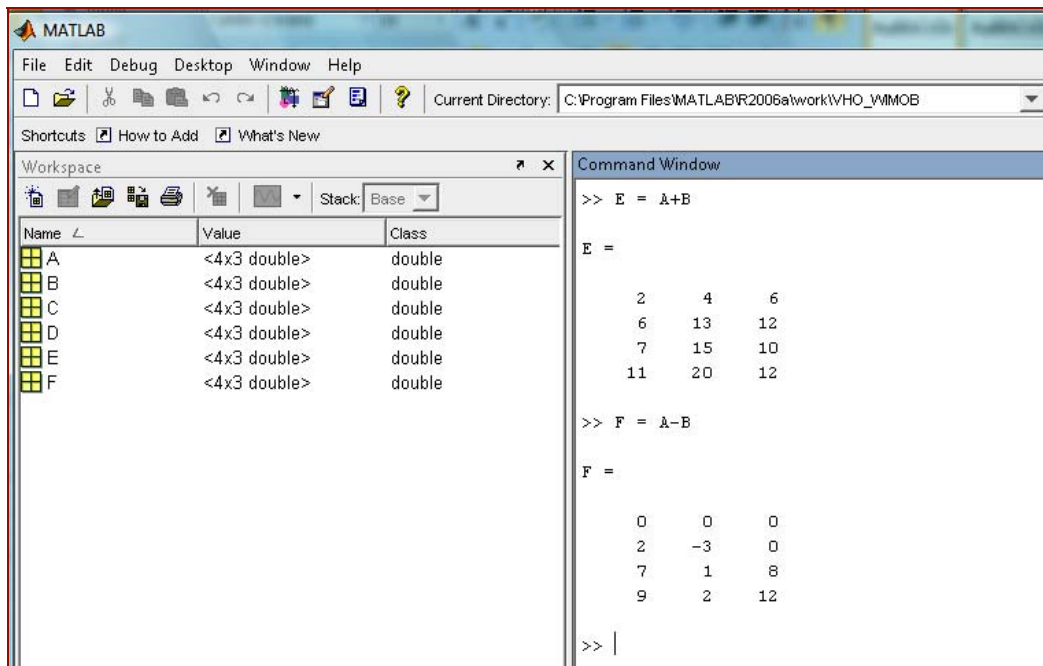
>> B
B =
     1     2     3
     2     8     6
     0     7     1
     1     9     0

>> C = A.*B
C =
     1     4     9
     8    40    36
     0    56     9
    10   99     0

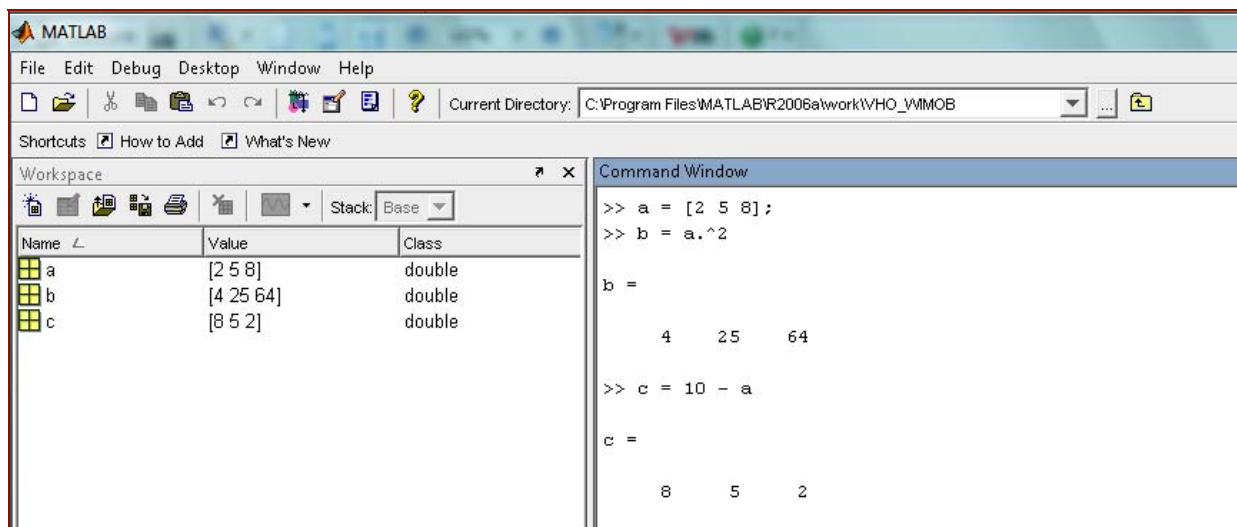
>> D = A./B
Warning: Divide by zero.
D =
    1.0000    1.0000    1.0000
    2.0000    0.6250    1.0000
         Inf    1.1429    9.0000
   10.0000    1.2222         Inf

>> |
```

The **Command History** window at the bottom shows the commands: A, B, clear C, and clc.



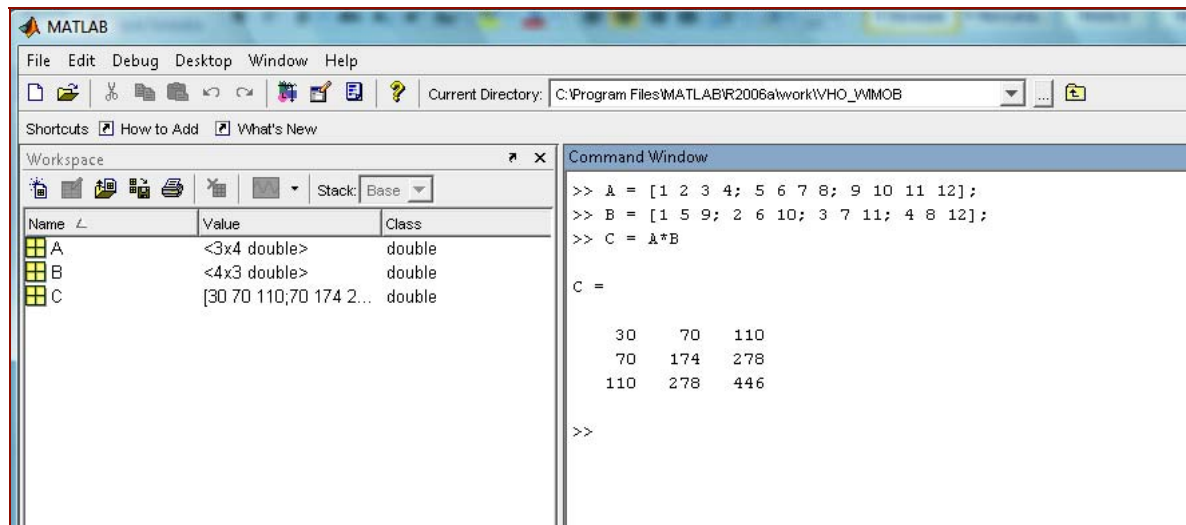
- L'operazione elemento per elemento può essere effettuata anche tra vettori e scalari, ovvero:



- La moltiplicazione tra due matrici è rappresentata dal simbolo *. In questo caso, è necessario che il numero di righe/colonne di una matrice corrispondano al numero di colonne/righe dell'altra matrice, (es. $A_{n \times m}$ e $B_{m \times n}$), ottenendo

$$A_{3 \times 4} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}, B_{4 \times 3} = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix},$$

$$\Rightarrow C_{3 \times 3} = A_{3 \times 4} \times B_{4 \times 3}$$



The screenshot shows the MATLAB interface. The Command Window contains the following code and output:

```
>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12];
>> B = [1 5 9; 2 6 10; 3 7 11; 4 8 12];
>> C = A*B

C =

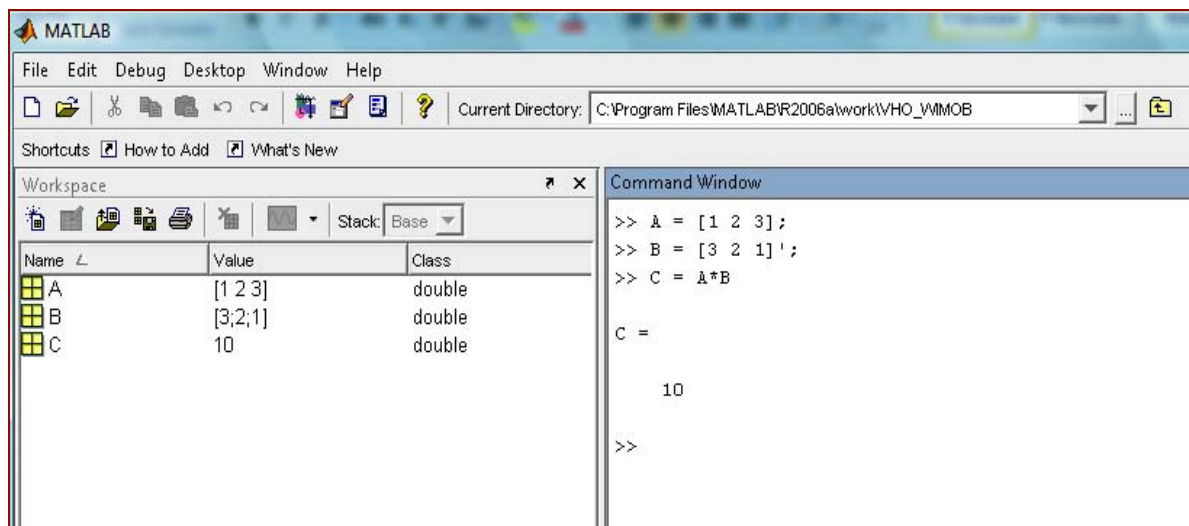
    30    70   110
    70   174   278
   110   278   446
```

The Workspace window shows three variables: A (3x4 double), B (4x3 double), and C (3x3 double).

- Analogamente, vale per i vettori

$$A_{1 \times 3} = [1 \quad 2 \quad 3], B_{3 \times 1} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix},$$

$$\Rightarrow C_{1 \times 1} = A_{1 \times 3} \times B_{3 \times 1}$$



The screenshot shows the MATLAB interface. The Command Window contains the following code and output:

```
>> A = [1 2 3];
>> B = [3 2 1]';
>> C = A*B

C =

    10
```

The Workspace window shows three variables: A (1x3 double), B (3x1 double), and C (1x1 double).

Altre funzioni che operano su vettori sono:

- **max**, **min**: calcola il massimo o il minimo di un vettore;
- **sort**: ordina gli elementi di un vettore in ordine decrescente o crescente;
- **sum**, **prod**: somma/moltiplica gli elementi di una matrice;

Max e Min di una matrice

1. **Max** e **Min** calcolano per ogni colonna rispettivamente il valore massimo e minimo della matrice A.
2. **Max(A, B)** e **Min(A, B)** calcolano per ogni colonna di A e di B rispettivamente il valore massimo e minimo, (*istruzione di default*).
3. **Max/Min(A, [], 1)** calcolano per ogni riga di A rispettivamente il valore massimo e minimo
4. **Max/Min(A, [], 2)** calcolano per ogni colonna di A e di B rispettivamente il valore massimo e minimo.

```

Command Window

>> A

A =

     3     4     1
     2     8     0
     6     0     5

>> max(A)
ans =

     6     8     5

>> min(A)
ans =

     2     0     0

>> min(A, 1)
ans =

     1     1     1
     1     1     0
     1     0     1

>> min(A, [], 1)
ans =

     2     0     0

>> min(A, [], 2)
ans =

     1
     0
     0

>>

```

Calcola il valore massimo di ogni colonna di A

Calcola il valore minimo di ogni colonna di A

Calcola il valore minimo tra tutti gli elementi di A e 1

Calcola il valore minimo di ogni colonna di A

Calcola il valore minimo di ogni riga di A

```

Command Window

>> max(A, [], 1)
ans =

     6     8     5

>> max(A, [], 2)
ans =

     4
     8
     6

>>

```

Calcola il valore massimo di ogni colonna di A

Calcola il valore massimo di ogni riga di A

Sort di una matrice

- “**Sort (A)**” ordina gli elementi di una matrice A in ordine *crescente*, lungo le colonne della matrice A.
- “**Sort (A, 'descend')**” ordina gli elementi di una matrice A in ordine *decrescente*, lungo le colonne della matrice A.

N.B.: Per default, **sort** ordina gli elementi in ordine **crescente**.

É possibile ordinare gli elementi di una matrice lungo le righe o le colonne, scrivendo:

- “**Sort (A, 1)**” ordina gli elementi di una matrice A in ordine crescente, lungo le **colonne** della matrice A.
- “**Sort (A, 2)**” ordina gli elementi di una matrice A in ordine crescente, lungo le **righe** della matrice A.

```
Command Window
>> A = [3 4 1; 2 8 0; 6 0 5]

A =

     3     4     1
     2     8     0
     6     0     5

>> sort(A)
ans =

     2     0     0
     3     4     1
     6     8     5

>> sort(A, 'ascend')
ans =

     2     0     0
     3     4     1
     6     8     5

>> sort(A, 'descend')
ans =

     6     8     5
     3     4     1
     2     0     0

>> |
```

Ordine **crescente** degli
elementi delle colonne di A

Ordine **crescente** degli
elementi delle colonne di A

Ordine **decrescente** degli
elementi delle colonne di A

```
Command Window
>> sort(A, 1)
ans =

     2     0     0
     3     4     1
     6     8     5

>> sort(A, 1, 'descend')
ans =

     6     8     5
     3     4     1
     2     0     0

>> sort(A, 2)
ans =

     1     3     4
     0     2     8
     0     5     6

>> sort(A, 2, 'descend')
ans =

     4     3     1
     8     2     0
     6     5     0

>> |
```

Ordine **crescente** degli
elementi delle colonne di A

Ordine **decrescente** degli
elementi delle colonne di A

Ordine **crescente** degli
elementi delle righe di A

Ordine **decrescente** degli
elementi delle righe di A

Sum e Prod di una matrice

```
Command Window
>> A
A =
     3     4     1
     2     8     0
     6     0     5

>> sum(A)
ans =
    11    12     6

>> A'
ans =
     3     2     6
     4     8     0
     1     0     5

>> sum(A')
ans =
     8    10    11

>>
```

Somma degli elementi
delle **colonne** di A

Trasposta di A

Somma degli elementi
delle **colonne** di A'

```
Command Window
>> A
A =
     3     4     1
     2     8     0
     6     0     5

>> prod(A)
ans =
    36     0     0

>> prod(A,2)
ans =
    12
     0
     0

>>
```

Moltiplicazione degli
elementi delle **colonne** di A

Moltiplicazione degli
elementi delle **righe** di A

Altri operatori che MatLab utilizza sono:

- **sin, cos, tan,**
- **asin, acos, atan, atand**
- **exp, log (naturale), log10 (in base 10),**
- **abs, sqrt, sign**

Per i numeri complessi, l'unità complessa è i o j ed è predefinita. Un numero complesso si scrive nella forma

$$z = a + jb, \quad (\text{es. } a = 3, \quad b = 5) \Rightarrow z = 3 + 5j,$$

o anche

$$z = r(\cos \varphi + j \sin \varphi),$$

dove r è il modulo di z , e φ rappresenta la sua fase,

$$r = \sqrt{a^2 + b^2}; \quad \varphi = \arctan\left(\frac{b}{a}\right).$$

Gli operatori che si possono utilizzare sono, pertanto:

- **abs** : calcola il modulo di z , (es. $\text{abs}(z)$);
- **angle** : calcola la fase di z , (es. $\text{angle}(z)$);
- **real** : calcola la parte reale di z , (es. $\text{real}(z)$);
- **imag** : calcola la parte immaginaria di z , (es. $\text{imag}(z)$);

```
Command Window
>> z = 3+5j;
>> abs(z)

ans =

    5.8310

>> angle(z)

ans =

    1.0304

>> real(z)

ans =

     3

>> imag(z)

ans =

     5
```

Strutture di controllo

In MATLAB è possibile utilizzare strutture di controllo, quali **if**, **for**, e **while**. Esse permettono la concatenazione di diverse istruzioni, le quali vanno separate con delle virgole.

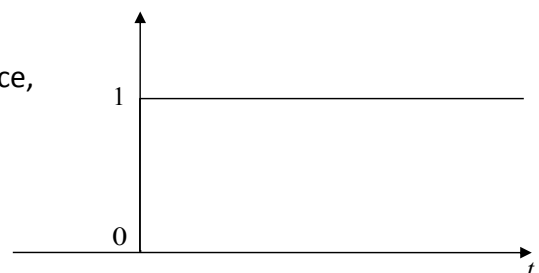
Struttura IF

Es: **if flag==0,**

<istruzioni separate da virgole>;

end;

Esempio: La funzione Heaviside (*funzione gradino*) fornisce, nella variabile x , il valore della funzione gradino a tempo continuo, calcolata in t .



```
if t >= 0  
    x=1;  
    else x=0;  
end
```

Struttura FOR

Si considerino le seguenti istruzioni:

```
[m n] = size (a);  
for i = 1 : m  
    for j = 1 : n  
        c (i, j) = a (i, j)^2;  
    end;  
end;
```

esse creano e visualizzano la matrice **c** ottenuta elevando al quadrato gli elementi della matrice **a**, ovvero $c = a^2$.

Struttura WHILE

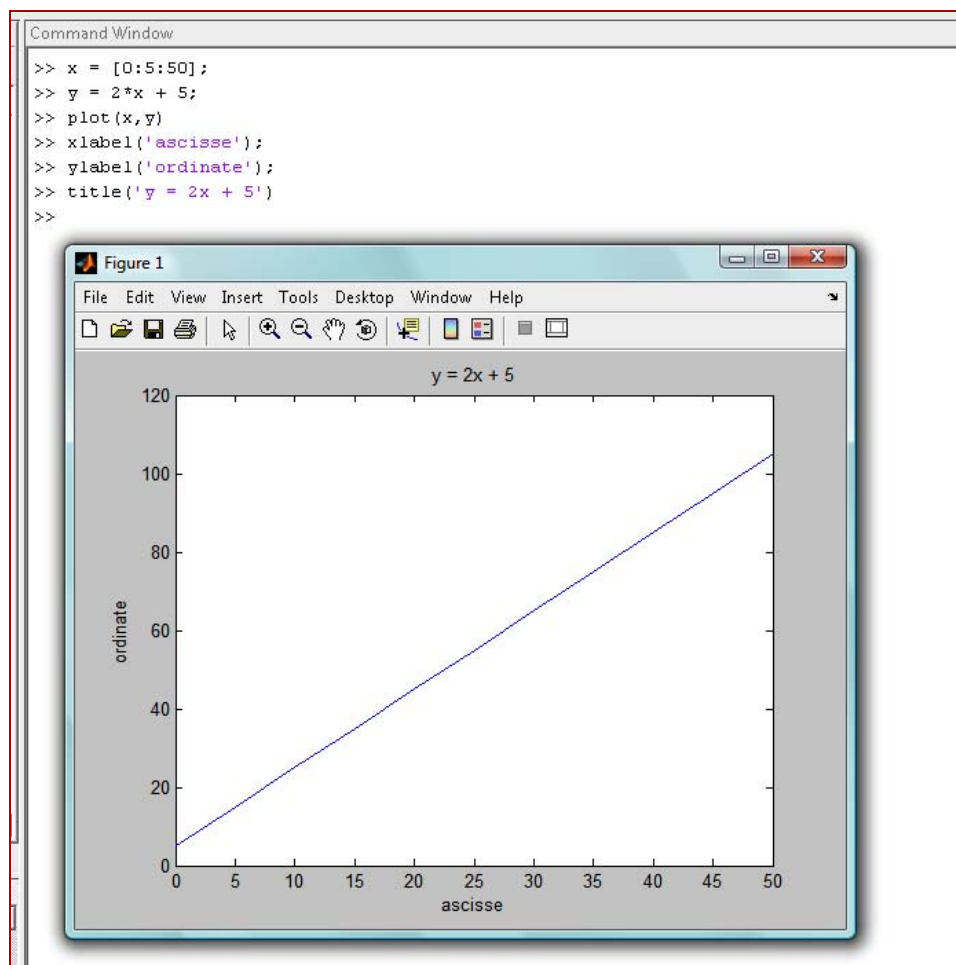
Si considerino le seguenti istruzioni:

```
a=1; b=5; c = b - a;  
while c >= 0  
    c = c - a;  
    a = a + 1;  
end;
```

Plot

L'istruzione per visualizzare il grafico di una funzione $y = f(x)$ è **PLOT**, in riferimento a due variabili della stessa dimensione (es. vettore x e y), quindi ottenendo un grafico su un piano cartesiano.

$$x = [0, 5, 10, \dots, 50], \quad y = 2x + 5,$$

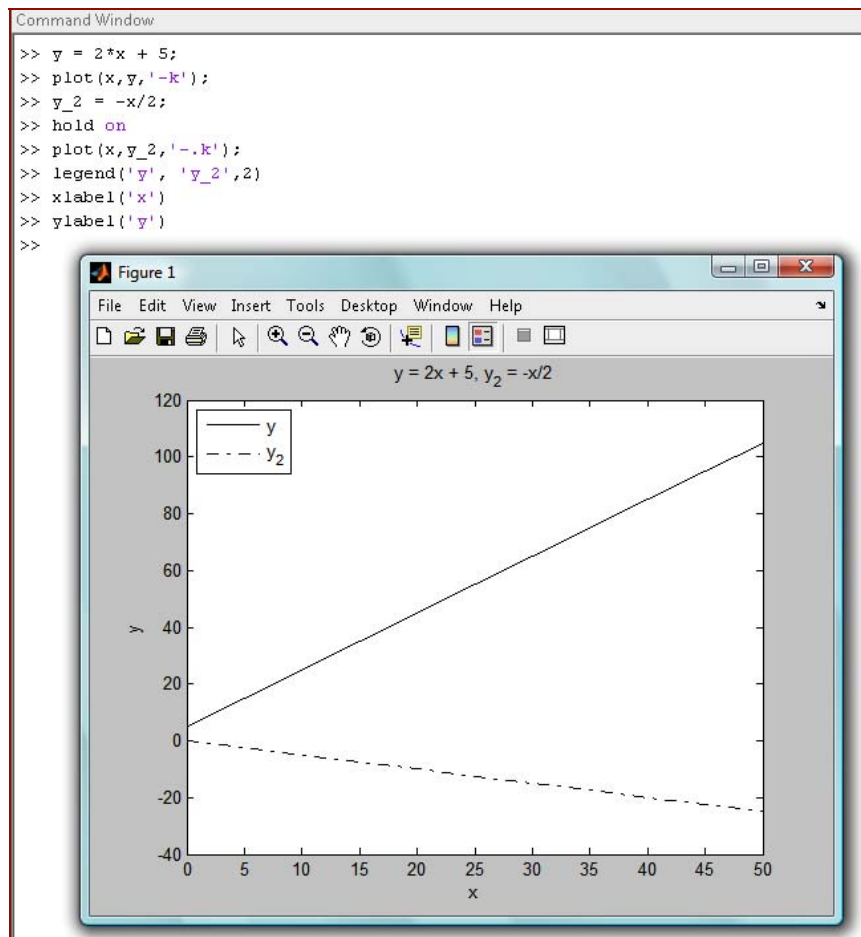


- Se si vuole visualizzare l'andamento di una seconda funzione (ad es. $y_2 = -x/2$) nella stessa finestra *figure* preesistente, si scrive l'istruzione **hold on**, seguita da **plot(x, y_2)**.

$$x = [0, 5, 10, \dots, 50], \quad y_2 = -\frac{x}{2},$$

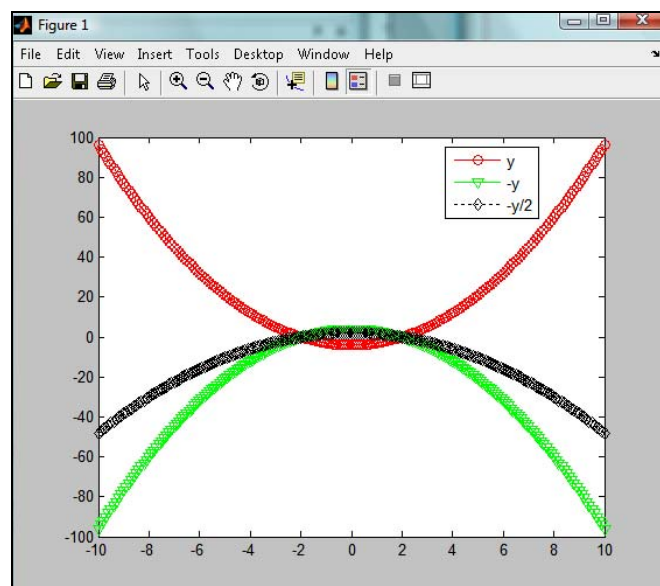
- **xlabel ('etichetta')** e **ylabel ('etichetta')** generano le etichette per gli assi x e y , rispettivamente.
- **Legend('y', 'y_2')** scrive la legenda delle variabili rappresentate.

- Il comando **close all** chiude tutte le figure aperte in finestre pop-up. Per chiudere una sola figura basta scrivere **close <numero della figura>**.
- Per dichiarare il numero di una nuova figura si scrivere **figure(<numero della figura>)**.



- Per creare grafici di colori diversi e usare marcatori diversi, si può specificare una stringa che rappresenta il colore del grafico e il simbolo (*marker*) usato per il plot.

```
Command Window
>> x = [-10:0.1:10];
>> y = x.^2-4;
>> plot(x,y,'-ro')
>> hold on
>> plot(x,-y,'-gv')
>> hold on
>> plot(x,-y/2,':kd')
>> legend('y',' -y',' -y/2')
>>
```



- Tutte le possibili scelte sono elencate nella tabella di seguito:

SIMBOLO	COLORE	SIMBOLO	MARKER
r	red	.	point
g	green	o	circle
b	blue	x	x-mark
w	white	+	plus
m	magenta	*	star
c	cyan	-	solid
y	yellow	:	dotted
k	black	--	dashed

Grafici sovrapposti

- Per visualizzare più grafici in un'unica finestra, occorre usare l'istruzione **subplot** (r, c, p), dove r e c rappresentano rispettivamente le righe e le colonne che dividono la figura. Ogni riga e colonna individuerà una posizione all'interno della figura.
- La variabile p rappresenta la posizione individuata dalla riga r , e dalla colonna c .

Esempio. **subplot** (3, 2, 1) divide la figura in 3×2 rettangoli e seleziona il rettangolo numero 1. Analogamente, **subplot** (3, 2, 5) divide la figura in 3×2 rettangoli e seleziona il rettangolo numero 5. Per ogni rettangolo selezionato, sarà possibile graficare una certa funzione.

Esempio. **subplot** (3, 2, 1);
 plot (x, y) # Grafica la funzione $y=f(x)$ nel primo rettangolo.

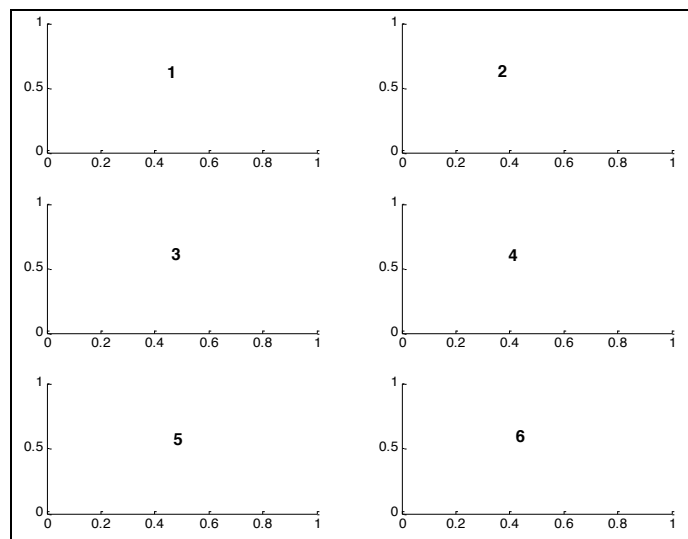
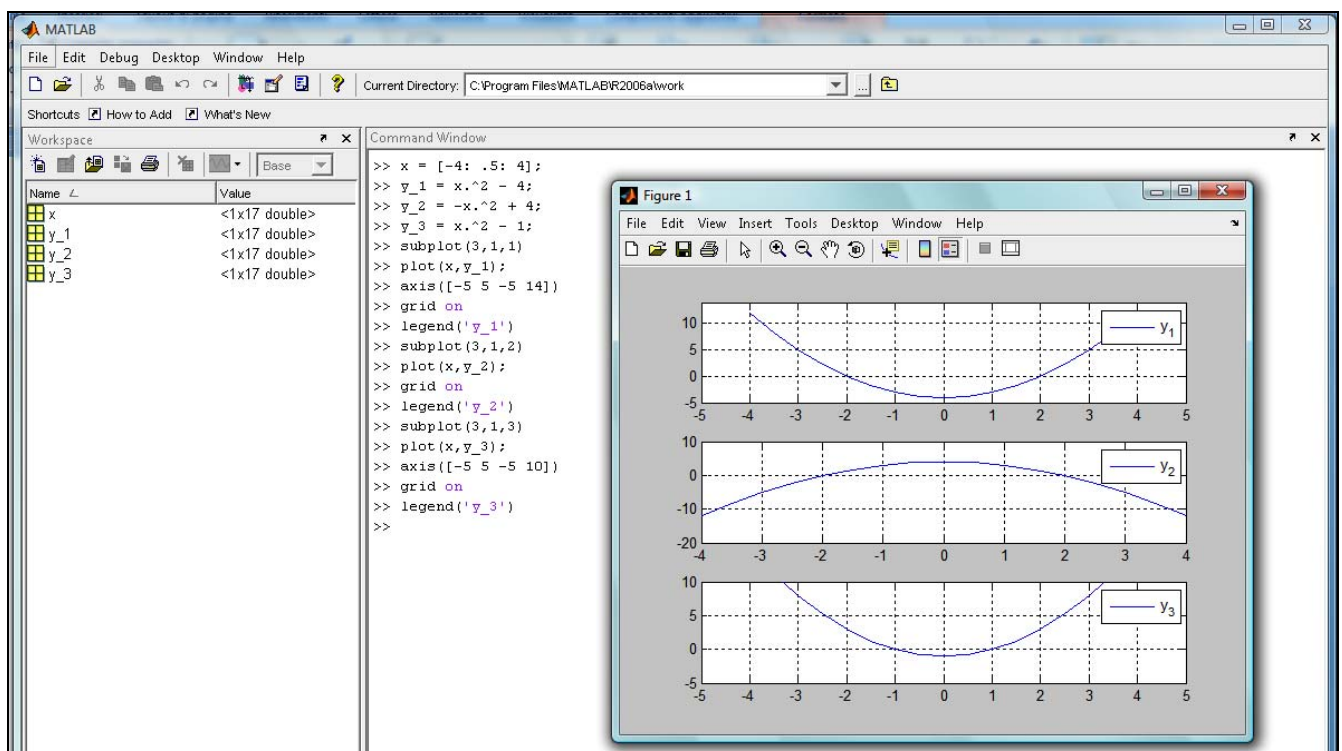
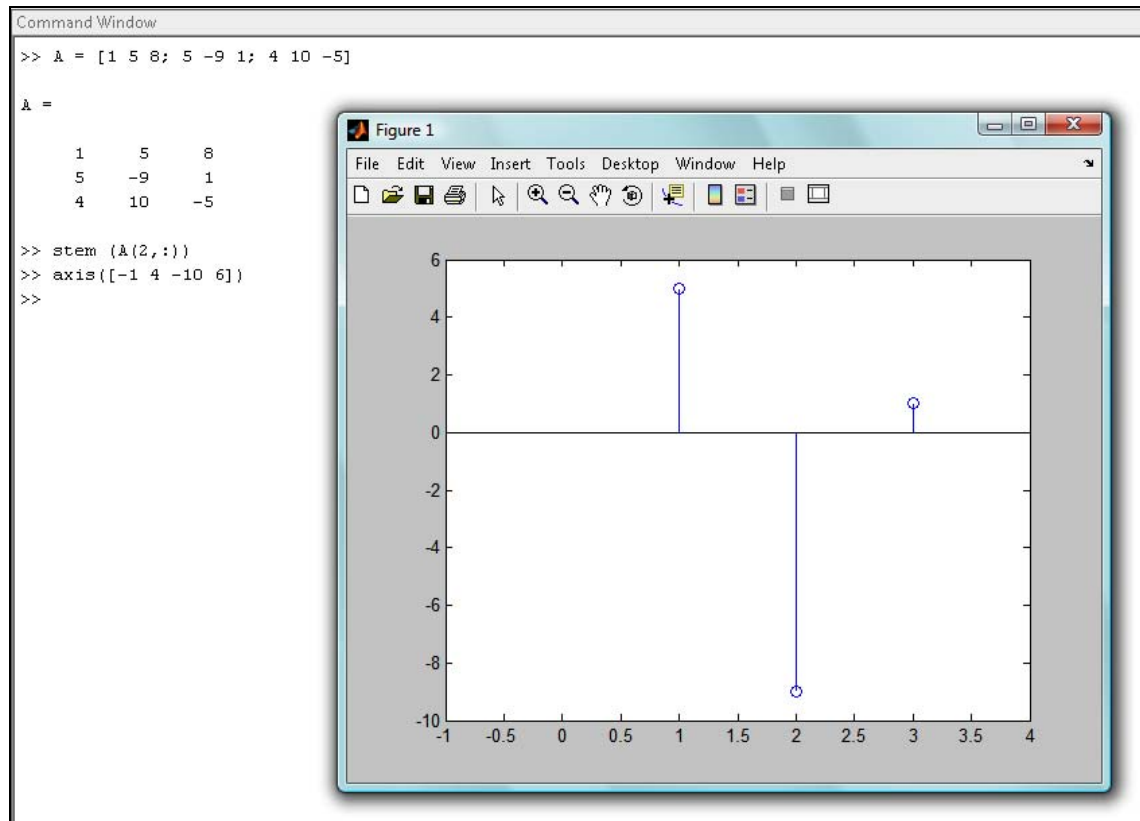


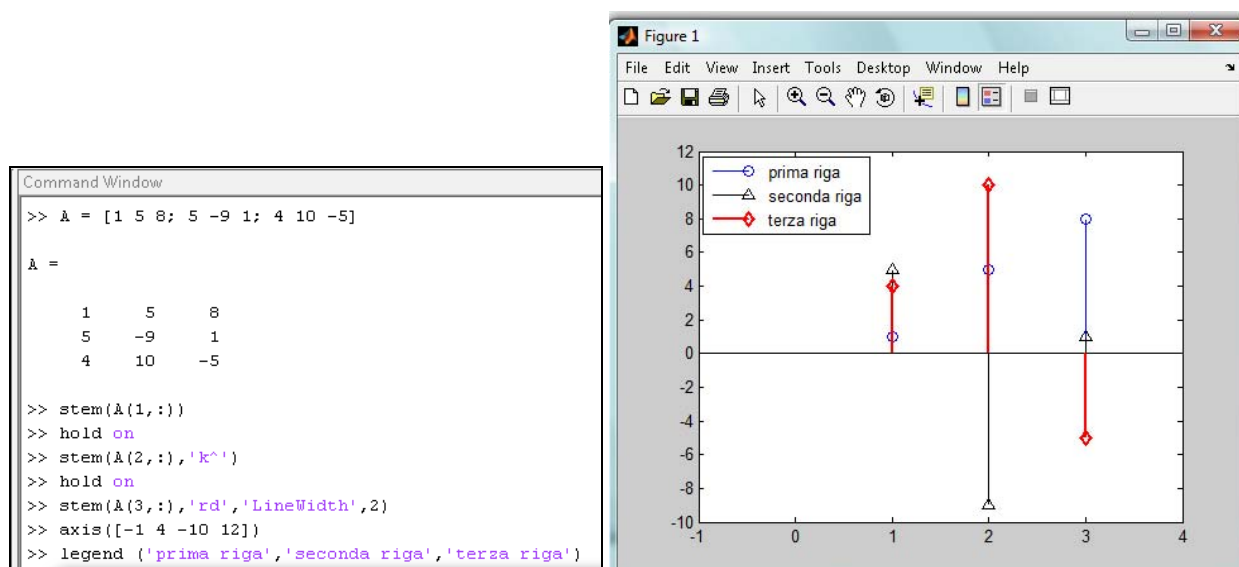
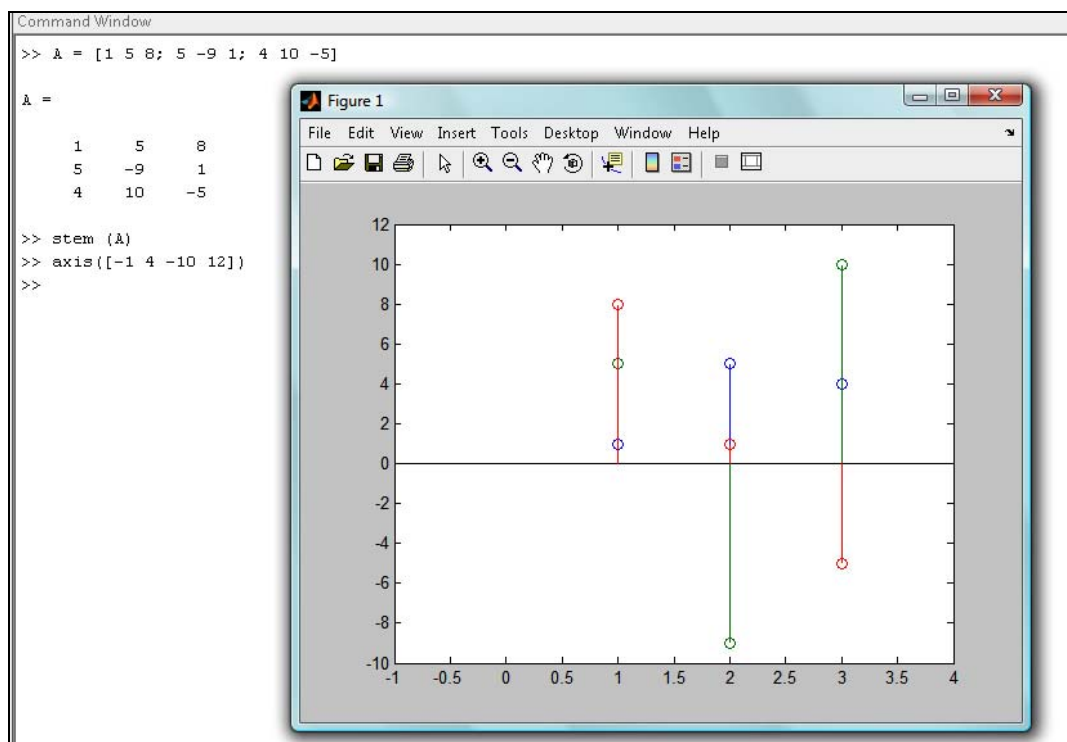
Figura 1. Comando subplot per sottografici all'interno della stessa figura.



Funzione STEM

- **stem(x)** grafica gli elementi del vettore x come “impulsi matematici”. Se x è una matrice, **stem** grafica tutti gli elementi di una riga di x.





Funzioni speciali

RAND

- **rand**, genera numeri random uniformemente distribuiti;
- **rand(N)** genera una matrice N×N composta da elementi random, scelti da una distribuzione uniforme nell'intervallo (0, 1).
- **rand(M, N)** genera una matrice M×N composta da elementi random nell'intervallo (0, 1).

```
Command Window
>> rand

ans =

    0.9501

>> rand(4)

ans =

    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
    0.8913    0.8214    0.9218    0.9355

>> rand(2,5)

ans =

    0.9169    0.8936    0.3529    0.0099    0.2028
    0.4103    0.0579    0.8132    0.1389    0.1987

>>
```

ZEROS

- **zeros(N)** genera una matrice quadrata di dimensione N, composta da soli zeri.
- **zeros(M,N)** genera una matrice M×N, composta da elementi nulli.

```
Command Window
>> zeros(1,3)

ans =

     0     0     0

>> zeros(3)

ans =

     0     0     0
     0     0     0
     0     0     0

>>
```

ONES

- **ones(N)** genera una matrice quadrata di dimensione N, composta da elementi unitari (1).
- **ones(M,N)** genera una matrice M×N, composta da elementi uguali a 1.

```
Command Window

>> ones(3)

ans =

     1     1     1
     1     1     1
     1     1     1

>> ones(3,5)

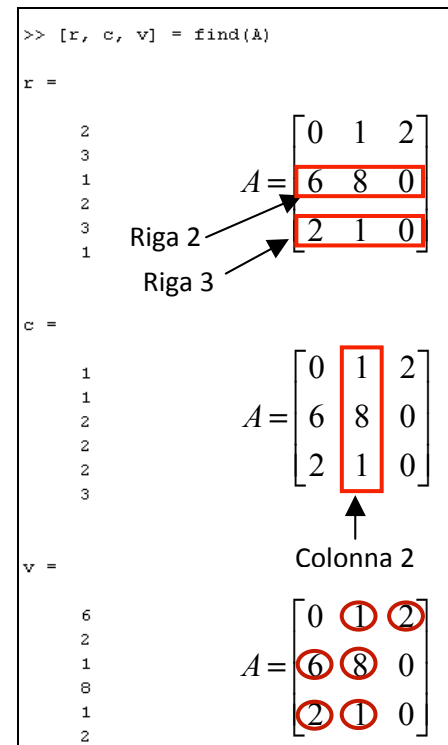
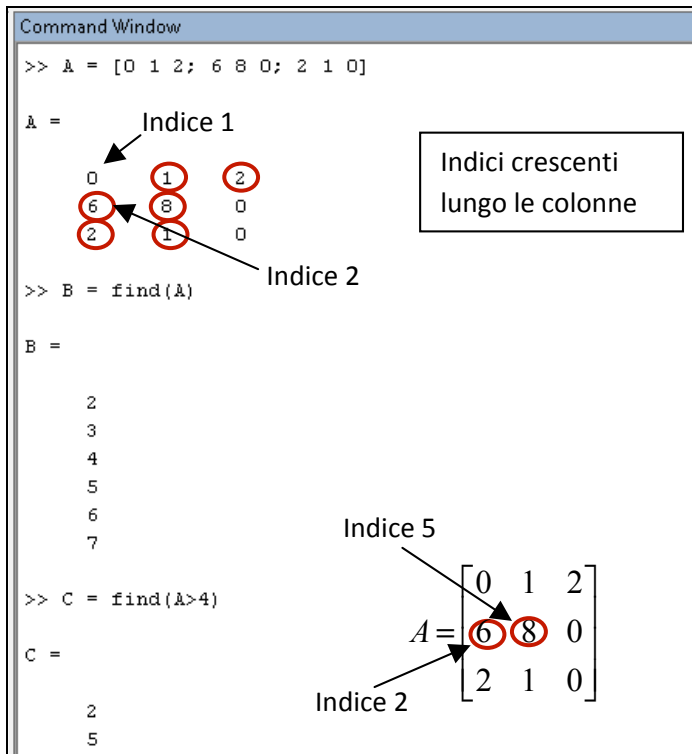
ans =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

>>
```

FIND

- La funzione “find” determina gli indici degli elementi non nulli.
- B = find(A)** restituisce gli indici del vettore A che sono non nulli.
- L’istruzione **I = find(A>x)** restituisce gli indici degli elementi di A, per cui A è maggiore di x.



- L’istruzione **[r, c, v] = find(A)** restituisce tre vettori r, c e v che rappresentano rispettivamente:
 - r = vettore degli indici delle righe della matrice A che individuano gli elementi non nulli;
 - c = vettore degli indici delle colonne della matrice A che individuano gli elementi non nulli;
 - v = vettore degli elementi non nulli della matrice A.

EYE

- La funzione **eye(n)** genera una matrice identità di dimensioni $n \times n$.
- La funzione **eye(n, m)** genera una matrice sulla cui diagonale principale vi sono gli 1, e gli zero altrove.

```
Command Window

>> eye (3)

ans =

    1    0    0
    0    1    0
    0    0    1

>> 5.*eye (3)

ans =

    5    0    0
    0    5    0
    0    0    5

>> 5.*eye (3,4)

ans =

    5    0    0    0
    0    5    0    0
    0    0    5    0

>> 5.*eye (3,6)

ans =

    5    0    0    0    0    0
    0    5    0    0    0    0
    0    0    5    0    0    0

>> |
```


Operazioni sulle matrici

- Per cambiare l'orientamento delle matrici si usano le funzioni **flipud**, **fliplr**, e **rot90**

```
Command Window
>> A = [1 2 3; 0 5 2; 6 8 7]

A =
     1     2     3
     0     5     2
     6     8     7

>> flipud(A)

ans =
     6     8     7
     0     5     2
     1     2     3

>> fliplr(A)

ans =
     3     2     1
     2     5     0
     7     8     6
```

FLIPLR inverte le colonne

FLIPUD inverte le righe

- La funzione **rot90** inverte una matrice in senso antiorario di 90 gradi.

```
>> rot90(A)

ans =
     3     2     7
     2     5     8
     1     0     6
```

$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 2 \\ 6 & 8 & 7 \end{bmatrix}$

Le stringhe

- Si definisce **STRINGA** una sequenza di caratteri, racchiusa tra due apici

MATLAB

File Edit Debug Desktop Window Help

Current Directory: C:\Program Files\MATLAB\R2006a\work

Shortcuts How to Add What's New

Workspace

Name	Value
stringa	'come visualizzare u...'

Command Window

```
>> stringa = 'come visualizzare una stringa';
>> disp(stringa)
come visualizzare una stringa
>>
```

- Per parole accentate, si sostituisce l'accento con due apici successivi, in modo da evitare conflitti con gli apici per definire la stringa. Ad esempio:
 - La stringa *"l'anno dell'invenzione di Internet"*, viene scritta in MATLAB come *'l''anno dell''invenzione di Internet'*.

The screenshot shows the MATLAB environment. The Command Window contains the following code and output:

```
>> stringa = 'l''anno dell''invenzione di Internet';
>> disp (stringa)
l'anno dell'invenzione di Internet
>> |
```

The Workspace window shows a variable named 'stringa' with the value 'l'anno dell'invenzione di Internet'.

- In MATLAB le stringhe sono considerate dei vettori riga. Valgono quindi le stesse regole degli array.

The screenshot shows the MATLAB environment with the following code and output in the Command Window:

```
>> stringa = 'l''anno dell''invenzione di Internet';
>> disp (stringa)
l'anno dell'invenzione di Internet
>> a = stringa(1:7)

a =

l'anno

>> length(stringa)

ans =

    34

>> stringa

stringa =

l'anno dell'invenzione di Internet
>>
```

The Workspace window shows three variables: 'a' with value 'l'anno', 'ans' with value 34, and 'stringa' with value 'l'anno dell'invenzione di Internet'.

Operatori relazionali

- == uguale
- ~= diverso da
- < minore di
- <= minore o uguale

- Tali istruzioni generano un risultato logico, ovvero con valore 1 (*risposta affermativa*) oppure 0 (*risposta negativa*).

The screenshot shows the MATLAB Command Window and Workspace. The Workspace pane on the left lists variables A (4x4 double), B (4x4 double), and ans (4x4 logical). The Command Window on the right shows the following commands and outputs:

```
>> A=eye(4)
A =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1

>> B=zeros(4)
B =
    0    0    0    0
    0    0    0    0
    0    0    0    0
    0    0    0    0

>> A>B
ans =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1

>> A~=B
ans =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1
```

Annotations in the screenshot explain the results of the logical operations:

- For `A>B`, the output is a 4x4 matrix of 1s and 0s. The first row is `1 0 0 0`, the second row is `0 1 0 0`, the third row is `0 0 1 0`, and the fourth row is `0 0 0 1`.
- For `A~=B`, the output is a 4x4 matrix of 1s and 0s. The first row is `1 0 0 0`, the second row is `0 1 0 0`, the third row is `0 0 1 0`, and the fourth row is `0 0 0 1`.

Arrows point to specific elements in the Command Window output:

- An arrow points to the `1` in the first row, first column of the `A>B` output, with the text "1 è diverso da 0, risposta negativa".
- An arrow points to the `0` in the first row, fourth column of the `A>B` output, with the text "0 è uguale a 0, risposta affermativa".

- any (x)** restituisce 1 se c'è un elemento di x diverso da zero;
- all (x)** restituisce 1 se tutti gli elementi di x sono diversi da zero;
- isnan (x)** restituisce 1 per ogni elemento NaN (*Not-a-Number*) in x. NaN è generato per operazioni $0/0$ oppure ∞/∞ .
- isinf (x)** restituisce 1 per ogni elemento infinito in x;
- finite (x)** restituisce 1 per ogni elemento finito in x.

Workspace

Name	Value	Class
A	<4x4 double>	double
ans	true	logical

Command Window

```
>> A = [0 4 Inf 7; 6 -2 0 NaN; 7 -1 Inf 4; 0 0 0 0]
```

A =

```
0     4    Inf     7
6    -2     0    NaN
7    -1    Inf     4
0     0     0     0
```

>> any(A)

ans =

```
1     1     1     1
```

>> any(A(4,:))

ans =

```
0
```

>> any(A(3,:))

ans =

```
1
```

Riga 4, in cui tutti gli elementi sono nulli

Riga 3, in cui nessun elemento è nullo

```
>> all(A)

ans =

0     0     0     0

>> all(A(3,:))

ans =

1
```

```
>> isnan(A)

ans =

0     0     0     0
0     0     0     1
0     0     0     0
0     0     0     0

>> isinf(A)

ans =

0     0     1     0
0     0     0     0
0     0     1     0
0     0     0     0
```

```
>> finite(A)

ans =

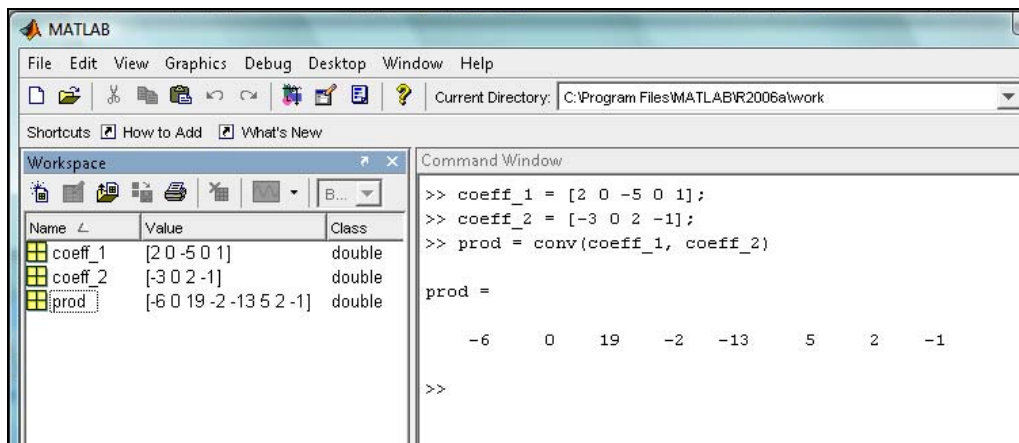
1     1     0     1
1     1     1     0
1     1     0     1
1     1     1     1
```

```
A =

0     4    Inf     7
6    -2     0    NaN
7    -1    Inf     4
0     0     0     0
```

Operazioni sui polinomi

- MATLAB considera un polinomio come vettore riga, i cui elementi sono i coefficienti del polinomio, in ordine di potenza decrescente.
- Ad es. il polinomio $x^5 + 2x^4 - 5x^2 + 1$ può essere rappresentato dal vettore dei coefficienti $coeff = [1 \ 2 \ 0 \ -5 \ 0 \ 1]$
- La funzione **conv** moltiplica due vettori. Può essere usata per il prodotto tra due polinomi.
- Ad es. il prodotto tra polinomi $P_1(x) = 2x^4 - 5x^2 + 1$ e $P_2(x) = -3x^3 + 2x - 1$ sarà la moltiplicazione tra i vettori $coeff_1 = [2 \ 0 \ -5 \ 0 \ 1]$ e $coeff_2 = [-3 \ 0 \ 2 \ -1]$, il cui risultato sarà un vettore con i coefficienti del polinomio prodotto $P_1(x) \cdot P_2(x)$.



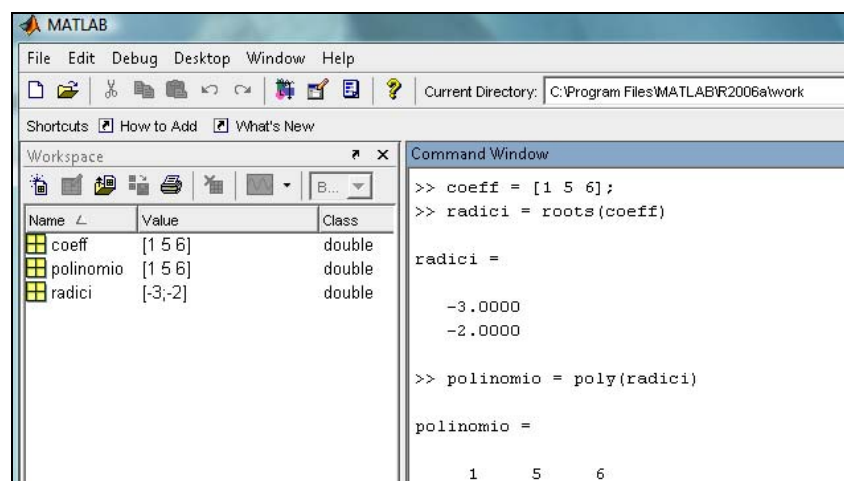
```

>> coeff_1 = [2 0 -5 0 1];
>> coeff_2 = [-3 0 2 -1];
>> prod = conv(coeff_1, coeff_2)

prod =

    -6     0    19     -2    -13     5     2    -1
  
```

- La funzione **roots** calcola le radici del polinomio.
- Ad es. il polinomio $P(x) = x^2 + 5x + 6$ viene rappresentato tramite il vettore riga $coeff = [1 \ 5 \ 6]$.
- L'istruzione **radici = root(coeff)** calcola le radici del polinomio $P(x)$.
- L'istruzione **polinomio = poly(radici)** genera il polinomio $P(x)$ a partire dal vettore **radici**.



```

>> coeff = [1 5 6];
>> radici = roots(coeff)

radici =

   -3.0000
   -2.0000

>> polinomio = poly(radici)

polinomio =

     1     5     6
  
```