GUIDA ALL'USO DI MATLAB (Matrix Laboratory)

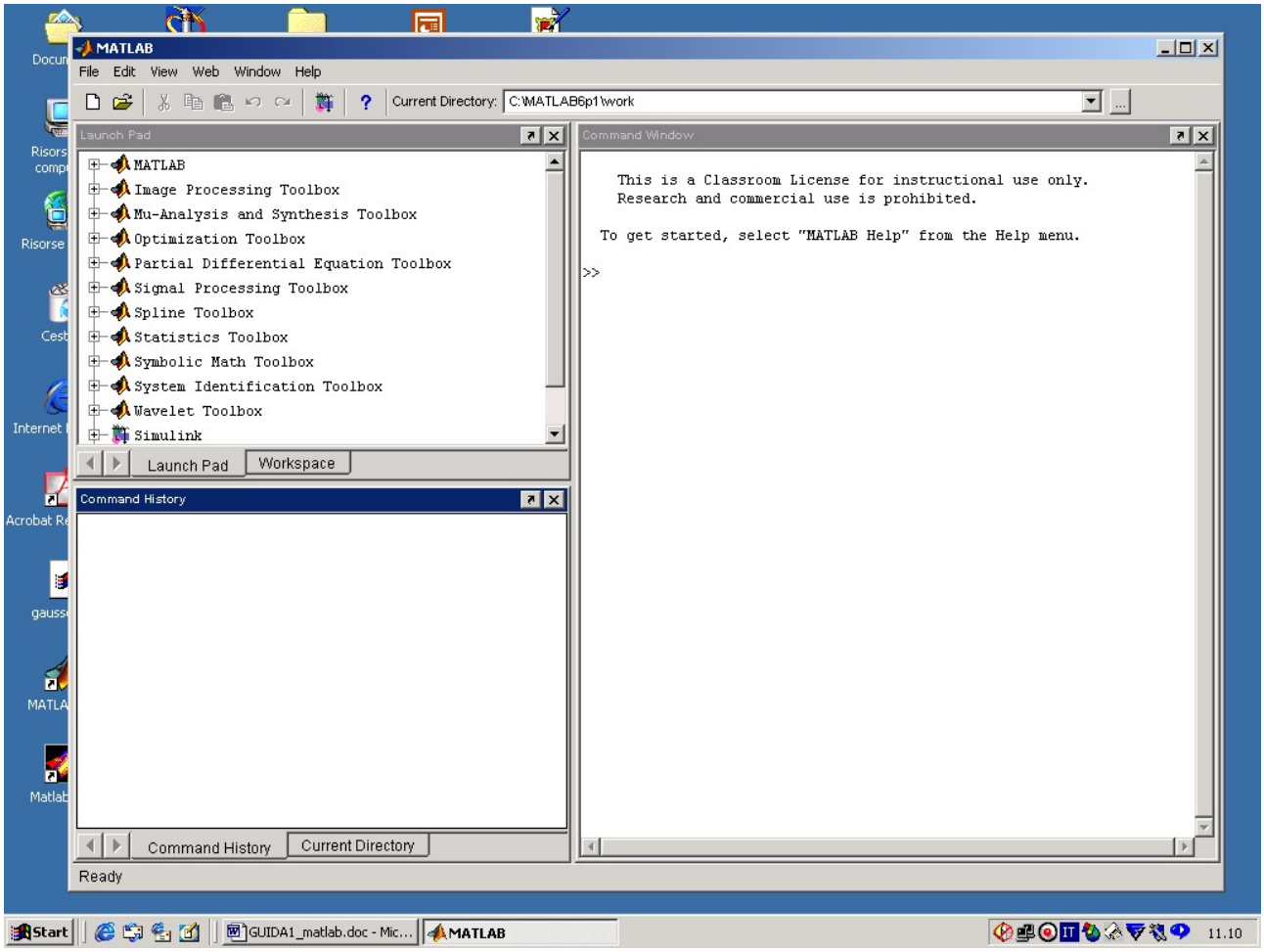
MATLAB è al tempo stesso un linguaggio di **PROGRAMMAZIONE**, al pari di C, FORTRAN e PASCAL ed un ambiente **INTERATTIVO** che permette di gestire variabili, importare ed esportare dati, svolgere calcoli, generare diagrammi, sviluppare e gestire file da utilizzare; ed inoltre è caratterizzato da ottime potenzialità grafiche che consentono un'immediata visualizzazione dei risultati.

- MATLAB è un linguaggio **INTERPRETE** anziché un linguaggio **COMPILATO** ed è disponibile per diversi sistemi operativi (Windows, Ms-DOS, Unix, Macintosh).

Nel seguito assumeremo di lavorare in ambiente Windows.

- **AVVIARE MATLAB:** bisogna selezionare l'**ICONA** corrispondente con il mouse.

Comparirà il PROMPT di MATLAB, cioè il simbolo >>, dal quale potranno essere richiamati tutti i comandi da eseguire in modo interattivo. Ci troviamo di fronte alla seguente finestra Windows:



Il Desktop di MatLab controlla la finestra dei comandi, la guida ed altri strumenti. La configurazione standard è costituita da tre finestre:

- **COMMAND WINDOW** (Finestra dei Comandi);
- **COMMAND HISTORY** (Cronologia dei Comandi);
- **LAUNCH PAD** (Strumenti di MAtLab);
- **WORKSPACE** (Contenuto dell'area di lavoro).

É possibile modificare l'aspetto del Desktop di MatLab: per ripristinare la configurazione standard del Desktop è sufficiente selezionare dal menu **View** l'opzione *Desktop Layout :Default.*

- In MatLab è possibile gestire una sessione di lavoro tramite l'utilizzo di vari COMANDI tra i quali:
  - clc — cancella il contenuto della Command Window
  - clear — elimina tutte le variabili dalla memoria
  - clear var1 var2 — elimina var1 e var2 dalla memoria
  - exist ('nomevar') — determina se un file o una variabile hanno il nome specificato
  - quit — chiude MatLab
  - whos — elenca le variabili presenti in memoria
  - : — genera un vettore di elementi regolarmente intervallati
  - , — separa istruzioni e elementi di una riga di un array
  - ; — esclude la visualizzazione di un risultato di un istruzione e separa le righe di un array
  - ... — continua l'istruzione in una riga successiva

La sostanziale differenza tra COMANDI, ISTRUZIONI E FUNZIONI consiste nel fatto che le funzioni richiedono argomenti racchiusi tra parentesi tonde, i comandi, quando ne richiedono, non devono essere specificati tra parentesi tonde, mentre le istruzioni non hanno argomenti.

- Digitando **help** viene richiamato un comodo help in linea. Per avere un'informazione dettagliata, è sufficiente far seguire al comando **help** il nome del comando. Ad esempio:

```
>> help sin
  SIN     Sine.
          SIN(X) is the sine of the elements of X.

>> help abs
 ABS    Absolute value.
    ABS(X) is the absolute value of the elements of X. When
    X is complex, ABS(X) is the complex modulus (magnitude) of
    the elements of X.
    See also SIGN, ANGLE, UNWRAP.
```

LE VARIABILI IN MATLAB

- IL PUNTO DI FORZA DI MATLAB è rappresentato dalla capacità di gestire grandi insiemi di numeri, array, come se fossero una singola variabile. Ad esempio per sommare due array in MatLab A e B è sufficiente digitare il comando C = A + B e MatLab somma i numeri corrispondenti di A e B per generare C; in altri linguaggi di programmazione questa operazione richiede più di un comando.

- MATLAB lavora tramite espressioni che vengono convertite in variabili. Le espressioni sono date dalla combinazione dei seguenti oggetti:

```
- Operatori (+,-,/,*,^,...)
- Caratteri speciali (%,!,:,;,...)
- Funzioni (sin, exp, roots, spline,...)
- Nomi di variabile
```

Tutte le variabili sono MATRICI:

- matrici 1 x 1 cioè SCALARI
- matrici 1 x n cioè VETTORI RIGA
- matrici n x 1 cioè VETTORI COLONNA
- matrici n x n nel senso classico della parola

N.B. A differenza dei comuni linguaggi di programmazione, MATLAB non richiede all'utente di DICHIARARE esplicitamente il TIPO DI VARIABILE (intero, reale, complesso, stringa) e la DIMENSIONE delle variabili utilizzate. Ciò comporta una sostanziale semplificazione nella stesura dei programmi, ma richiede una certa cautela in fase di programmazione.

- I NOMI DELLE VARIABILI possono essere costituiti da lettere e cifre (e non dai caratteri speciali), con il vincolo che il **primo carattere** debba essere una lettera e che la lunghezza del nome della variabile non superi i 19 caratteri.

- MATLAB distingue tra lettere maiuscole e lettere minuscole (*case sensitive*).
  (per eliminare questa caratteristica basta digitare il comando **casesen off**, mentre per abilitarla nuovamente si darà **casesen on**).

- Il comando **who** elenca tutte le variabili definite durante la sessione di lavoro.
  Ad esempio

```
>> x=[pi pi/2];
>> y=sin(x)

y =

   0.0000    1.0000
```

```
>> who

Your variables are:
x  y
```

- MATLAB lavora con 16 cifre significative, tuttavia queste non vengono sempre scritte tutte nell'output.

  Se la variabile considerata è un intero, essa verrà presentata in un formato privo di punto decimale:

```
>>  i2 = 5

i2 =
    5
```

  Se si tratta di un numero decimale, essa verrà presentata di default con solo 4 cifre significative (ossia nel cosiddetto **format short**):

```
>> numero = 1.234567

numero=
   1.2346
```

  Per visualizzare più cifre decimali, è necessario ricorrere al comando **format long**:

```
>>  numero = 1.234567
numero=
   1.2346
>>  format long
>>  numero
numero =
  1.2345670000000
```

  Per tornare al formato short (di default) è sufficiente dare il comando **format short**. Usando **format short e** e **format long** i numeri vengono rappresentati in notazione esponenziale:

```
>> format short e
>> numero
numero =
  1. 2346e+00
>> format long e
>> numero
numero =
  1.23456700000000e+00
```

- Alcune variabili sono predefinite, come:

eps zero macchina
pi π
i, j $\sqrt{-1}$
NaN "Not-a-Number" (l'espressione calcolata non è un numero macchina)
flops numero di operazioni macchina effettuate

Il comando **flops** non tiene conto di tutte le singole operazioni effettuate, ma solo delle più importanti.

- Le variabili predefinite possono essere modificate dall'utente durante la sessione di lavoro, ma è meglio non modificarle.

- Il comando **whos** dà un maggior numero d'informazioni sulle variabili in memoria rispetto al comando **who** sopra descritto, fornisce anche un'indicazione sulla quantità di memoria allocata:

```
>> whos
 Name      Size            Bytes  Class

 x         1x2                16  double array
 y         1x2                16  double array

Grand total is 4 elements using 32 bytes
```

- MATLAB memorizza tutte le variabili definite nel corso della sessione di lavoro e quindi si rischia facilmente di saturare rapidamente la memoria disponibile.
  In tal caso può comparire un messaggio d'errore del tipo **out of memory**, accompagnato talvolta da una brusca interruzione del programma (con conseguente perdita del lavoro svolto).

- Per ovviare a questo tipo di problemi, soprattutto se si lavora su macchine con poca memoria, conviene controllare spesso l'occupazione di memoria tramite il comando **whos** e CANCELLARE LE VARIABILI INUTILI.

clear cancella tutte le variabili definite dall'utente

clear seguito dai nomi di alcune variabili cancella soltanto quelle variabili

Esempio: **clear a** cancella la variabile a, **clear a b** cancella a e b

ASSEGNAZIONE VARIABILI

- **L'ASSEGNAZIONE DI UNO SCALARE** viene effettuata semplicemente ponendo il nome della variabile uguale al valore prescelto:

```
>> b=2.34
b=
  2.3400
```

L'espressione a sinistra dell'operatore "=" deve avere un valore calcolabile. Ad esempio se alla variabile y non è stato assegnato alcun valore l'espressione che segue genera un messaggio d'errore:

```
>>  x=y+5;
```

Se il nome della variabile viene omesso, MATLAB crea automaticamente la variabile **ans** (che significa answer, risposta) contenente il valore assegnato.

```
>>  2.34
ans =
   2.3400
```

Ciascuna frase deve terminare con il comando di invio (return), indicato con ↵ , dopo il quale MATLAB visualizza sullo schermo la variabile ed il suo contenuto.
L'output viene soppresso se si conclude la frase con il carattere **;**

Quest'ultima opzione è consigliabile durante l'elaborazione di un complesso flusso di istruzioni; per contro, qualora si desideri verificare la correttezza di un algoritmo, può essere vantaggioso lasciare attiva la fase di display su schermo allo scopo di evidenziare possibili errori o risultati parziali.

- Per ASSEGNARE MATRICI O VETTORI bisogna ricorrere ai caratteri speciali **[ ]**

Gli elementi di una stessa riga devono essere separati da spazi vuoti (o da virgole); le diverse righe devono essere separate dal punto e virgola (o dall'andata a capo).

Assegnazione di un VETTORE RIGA:

```
> >  b = [2  1  5  4]
b=
   2    1    5    4
```

oppure:

```
> >  b = [2, 1, 5, 4]
b=
    2     1    5     4
```

Assegnazione di un VETTORE COLONNA:

```
> >  c = [1;2;3;4]
c =
     1
     2
     3
     4
```

oppure:

```
> >  c = [1
2
3
4]
c =
     1
     2
     3
     4
```

Assegnazione di una MATRICE QUADRATA 2 x 2:

```
>> matrice = [1 2 ; 3 4]
matrice =
    1     2
    3     4
```

Assegnazione di una MATRICE RETTANGOLARE 2 x 3:

```
>> matrice_ret = [2 2 0 ; 3 4 1]
matrice_ret =
        2   2  0
        3   4  1
```

- Il comando **size** fornisce la dimensione della matrice:

```
>> vet = [1 2 3 4];
>> size(vet)
ans =
     1      4
```

Possiamo memorizzare il risultato di **size** in un vettore scrivendo:

```
>> [n,m] = size (b)
```

MATLAB lavora con matrici e non può trattare oggetti che non siano tali. Se costruiamo un vettore con un numero di colonne variabile da riga a riga otteniamo un messaggio d'errore:

```
>> A = [1  2  ;  1  2  3]
???  All rows in the bracketed expression must have the same
number of columns.
>>
```

- Quando un vettore sia costituito da elementi con differenza costante, ad esempio

  vet = [1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0],

  si può assegnare nel seguente modo:

```
>> vet = [1 : 0.1 : 2]

 vet =

Columns 1 through 7

   1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000

  Columns 8 through 11

   1.7000    1.8000    1.9000    2.0000
```

Cioè si scrive:

NOME_VETTORE = [VALORE_INIZIALE : INCREMENTO : VALORE_FINALE]

Anche il comando linspace crea un vettore riga di elementi linearmente intervallati, con la differenza che questo comando richiede il numero di valori e non l'incremento, la sintassi è:

linspace (VALORE_INIZIALE , VALORE_FINALE,n)

dove n è il numero di elementi del vettore riga. Ad esempio il seguente comando

```
>> vet = linspace(1,2,11)
```

è equivalente a

```
>> vet = [1 : 0.1 : 2]
```

Il comando logspace genera un array di elementi intervallati logaritmicamente, la sintassi è: **logspace (** a **,** b **,** n **)**, dove n è il numero di punti tra 10^a e 10^b.

- E’ possibile estrarre singoli elementi o blocchi da matrici già definite:

```
>> c =  [ 1 2 3 ; 4 5 6]
c=
     1    2     3
     4    5     6

>> d = c(2,3)          (l'elemento 2,3 della matrice)
d=
     6

>> d = c(1, :)          (estrazione della prima riga)
d =
     1    2     3

>> d = c(:,2)          (estrazione della seconda colonna)
d=
    2
    5

>> d = c(:,1:2)        (estrazione delle prime due colonne)
d=
     1    2
     4    5

>> d = c(:)            (sposta la matrice c in un vettore d)
d=
     1
     4
     2
     5
     3
     6
```

Il carattere speciale **:** significa dal **valore iniziale** al **valore finale** con un **certo incremento**. Quando l’incremento non compare esplicitamente vale 1.

2 : 7 significa da 2 a 7 con incremento 1

2 : 0.5 :7 significa da 2 a 7 con incremento 0.5

: significa tutti gli elementi di riga o di colonne

- Si possono costruire matrici a partire da altre matrici:

```
>>  c = [1 2 3 ; 4 5 6];
>>  c = [ c ; c ]
```

```
c =
     1     2     3
     4     5     6
     1     2     3
     4     5     6
```

E’ possibile creare una matrice vuota (di dimensioni 0 x 0) con il comando

```
>> vuota=[ ]

vuota =

    []
```

Esempio dell’uso della matrice vuota: rimozione di una riga

```
>> mat=[1 1 ; 2 2 ; 3 3]

mat =

    1     1
    2     2
    3     3

>> mat([1],:)=[]

mat =

    2     2
    3     3
```

- **OPERAZIONI ELEMENTARI**

Sono definite le operazioni aritmetiche:

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

- Quando l'espressione da valutare è troppo lunga perché stia su di un'unica riga di comando si utilizza il carattere di continuazione dato da **...** (tre punti):

```
>> a=1+2+3+4+...
5

a =

   15
```

- E' possibile alterare le precedenze classiche delle operazioni aritmetiche mediante l'uso opportuno delle parentesi tonde.
  Nel caso in cui via siano più coppie di parentesi, si usano sempre le tonde, non si usano mai le quadre e le graffe:

```
>> ((5+2)*0.5)^3
ans =
  42.8750

>> b=[1, 2*sqrt(2), sin(pi/4)*(cos(pi/4)+exp(0.5))]

b =

   1.0000    2.8284    1.6658


>> c=[1, 2*sqrt(2), sin(pi/4)*(cos(pi/4)+exp(0.5)) ;
       (2+sqrt(2))^0.5 1/(2+sqrt(3)) 1]

c =

   1.0000    2.8284    1.6658
   1.8478    0.2679    1.0000
```

- Le operazioni elementari **+ - *** si estendono a VETTORI e MATRICI, non si estendono le operazioni di divisione e d'elevamento a potenza:

```
>> a = 1:4;
>> b = 1:3;
>> c = [3 2 6 -1];
>> a + c                    (somma di vettori riga della stessa dimensione)
ans =
    4     4     9     3


>> a – c                    (sottrazione di vettori riga della stessa dimensione)
ans =
   -2     0    -3     5

>> a + b                    (tentativo di somma di due vettori riga di dimensione diversa)
??? Error using ==> +
Matrix dimensions must agree.

>> a * c
??? Error using ==> *
Inner matrix dimensions must agree.
```

N.B. La somma **+** e la sottrazione **-** tra vettori si possono fare soltanto se i vettori hanno la stessa dimensione.

- Il prodotto ***** tra vettori è in realtà un PRODOTTO RIGHE PER COLONNE TRA MATRICI.

Quindi tra vettori si può fare soltanto tra un vettore riga di dimensione 1 x n ed un vettore colonna di dimensione n x 1 ed esso coincide con il prodotto scalare tra vettori:

```
>> r = [1 0 2 1];
>> c = [1 ; 1 ; 2 ; 2];
>> r*c
ans =
    7
```

Il prodotto c * r fornisce una matrice di dimensione n x n:

```
>> r = [1 0 2 1];
>> c = [1 ; 1 ; 2 ; 2];
>> c*r
ans =
    1     0     2     1
    1     0     2     1
    2     0     4     2
    2     0     4     2
```

- Il prodotto ***** tra due MATRICI è un PRODOTTO RIGHE PER COLONNE e si può fare solo se il numero di colonne della prima matrice è uguale al numero di righe della seconda matrice.

```
>> a = [ 1  2  3 ; 2  1  0 ; 1  1  0 ; 2  1  3 ]
a =
    1    2    3
    2    1    0
    1    1    0
    2    1    3
>> b = [ 1  0 ; 1  2 ; 1  0 ]
b =
    1    0
    1    2
    1    0
>> a * b
ans =
    6    4
    3    2
    2    2
    6    2
```

- Altre operazioni caratteristiche delle matrici:

  OPERAZIONE DI TRASPOSIZIONE: si pospone al nome della matrice il carattere APICE **'**
  Se l'array contiene elementi complessi, l'operatore di trasposizione produce la matrice trasposta complessa e coniugata, cioè gli elementi risultanti sono complessi e coniugati degli elementi trasposti dell'array originale. In alternativa è possibile utilizzare l'operatore di trasposizione **.'** per trasporre l'array senza produrre elementi complessi e coniugati.

```
>> r = 1 : 4
r =
    1    2    3    4
>> r'
ans =
    1
    2
    3
    4
>> mat= [1 2 ; 3 4 ]
mat =
    1    2
    3    4
```

```
>> mat'
ans =
    1    3
    2    4
```

OPERAZIONE D'INVERSIONE (per matrici quadrate non singolari): si usa il comando **inv**

```
>> A = [ 1 2 ; 3 4 ];
>> inv(A)
ans =
  -2.0000    1.0000
   1.5000   -0.5000

>> B = [ 0 0 ; 0 1 ];
>> inv(B)
Warning:
Matrix is singular to working precision.
ans =
   Inf   Inf
   Inf   Inf
```

N.B. Il calcolo dell'inversa è effettuato per via numerica e quindi può essere affetto da grandi errori nel caso in cui la matrice da invertire risulti **mal condizionata**.

OPERAZIONI DI CALCOLO DEL DETERMINANTE (nel caso di una matrice quadrata non singolare) e del RANGO di una matrice (dato dal massimo numero di colonne (o righe) linearmente indipendenti), per le quali valgono le stesse considerazioni svolte a livello del calcolo dell'inversa.
Si usano i comandi **det** e **rank** :

```
>> A = [1 2 ; 3 4];
>> det(A)
ans =
   -2

>> A=[1 1 1 ; 2 2 2 ; -1 2 3]
A =
    1    1    1
    2    2    2
   -1    2    3

>> rank(A)
ans =
    2
```

- Tra le altre numerose funzioni che consentono di manipolare matrice e vettori ricordiamo:

diag tril triu abs sum max min norm cond eig eye zeros ones rand

- **diag** : se applicata ad una matrice estrae la diagonale della matrice, se applicata ad un vettore genera una matrice diagonale

```
>> a=[1:1:3;3:1:5;3:-1:1]

a =

    1    2    3
    3    4    5
    3    2    1

>> b=diag(a)

b =

    1
    4
    1

>> diag(b)

ans =

    1    0    0
    0    4    0
    0    0    1
```

- **tril (triu)**: estrae da una matrice la parte triangolare inferiore (triangolare superiore). Indicando come secondo argomento della funzione un numero intero positivo o negativo è possibile estrarre anche le co-diagonali superiori o inferiori della matrice.

```
>> a=[1:1:3;3:-1:1;0:1:2]
a =

    1    2    3
    3    2    1
    0    1    2

>> b=tril(a)
b =

    1    0    0
    3    2    0
    0    1    2
```

```
>> c=triu(a)
c =

    1    2    3
    0    2    1
    0    0    2

>> b1=tril(a,1)
b1 =

    1    2    0
    3    2    1
    0    1    2

>> c1=triu(a,-1)
c1 =

    1    2    3
    3    2    1
    0    1    2

>> c1=triu(a,1)

c1 =
    0    2    3
    0    0    1
    0    0    0
>> d=b+c-diag(diag(a))

d =
    1    2    3
    3    2    1
    0    1    2
```

- **abs :** applicata ad vettore (matrice) ne calcola il vettore (matrice) dei valori assoluti se gli elementi sono tutti reali. Applicata ad un vettore (matrice) con elementi complessi calcola il vettore (matrice) dei moduli.

```
>> a=[1 -1]
a =

    1   -1

>> abs(a)
ans =

    1    1

>> x=0.5+2*i
x =
```

```
   0.5000 + 2.0000i

>> abs(x)
ans =

    2.0616

>> b=[x 2]
b =

   0.5000 + 2.0000i   2.0000

>> abs(b)
ans =

    2.0616    2.0000

>> e=[i 2+i; 3+i*5 2]
e =

        0 + 1.0000i   2.0000 + 1.0000i
   3.0000 + 5.0000i   2.0000

>> abs(e)
ans =

    1.0000    2.2361
    5.8310    2.0000
```

- **sum :** Applicata ad un vettore restituisce la somma delle componenti del vettore. Applicata ad una matrice restituisce un vettore le cui componenti sono le somme per colonna degli elementi della matrice

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

a =

     1     3     5
     2     4     6
     7     8     9

>> sum(a)

ans =

    10    15    20

>> sum(diag(a))

ans =
    14
```

- **max (min) :** Applicata ad un vettore restituisce il massimo (minimo) delle componenti del vettore. Applicata ad una matrice restituisce un vettore le cui componenti sono il massimo (minimo) degli elementi della matrice per colonna.

```
>> a=[1:0.2:2 ; 0 : 0.2 :1]
a =

   1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
        0    0.2000    0.4000    0.6000    0.8000    1.0000

>> max(a)
ans =

   1.0000    1.2000    1.4000    1.6000    1.8000    2.0000

>> min(a)
ans =

        0    0.2000    0.4000    0.6000    0.8000    1.0000

>> min(a([1],:))
ans =

    1
```

- **norm :** Calcola la norma 2 di una matrice o di un vettore.

```
>> vet=[1 -2 3 -4 5];
>> norm(vet)

ans =

   7.4162

>> mat=diag(vet)

mat =

    1    0    0    0    0
    0   -2    0    0    0
    0    0    3    0    0
    0    0    0   -4    0
    0    0    0    0    5

>> norm(mat)

ans =

    5

>> v=vander([1:0.2:2])
```

```
v =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    2.4883    2.0736    1.7280    1.4400    1.2000    1.0000
    5.3782    3.8416    2.7440    1.9600    1.4000    1.0000
   10.4858    6.5536    4.0960    2.5600    1.6000    1.0000
   18.8957   10.4976    5.8320    3.2400    1.8000    1.0000
   32.0000   16.0000    8.0000    4.0000    2.0000    1.0000

>> norm(v)

ans =

   46.1026
```

- **cond :** Calcola l'indice di condizionamento in norma 2 di una matrice.

```
>> cond(v)

ans =

  8.9017e+005

>> eye(5)

ans =

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

>> cond(ans)

ans =

     1
```

- **eig :** Calcola gli autovalori ed autovettori di una matrice quadrata.

```
>> [Y,D]=eig(v)
```

```
Y =

    0.1268    0.1714   -0.1822   -0.1345   -0.0692    0.0219
    0.1754    0.1287    0.1060    0.3236    0.3259   -0.1632
    0.2478    0.0346    0.3298    0.1371   -0.3926    0.4793
    0.3536   -0.1380    0.3697   -0.3773   -0.2447   -0.6932
    0.5046   -0.4242    0.0556   -0.5673    0.7391    0.4935
    0.7156   -0.8683   -0.8409    0.6279   -0.3587   -0.1383


D =

   16.7493         0         0         0         0         0
         0   -6. 3939         0         0         0         0
         0         0    0.8898         0         0         0
         0         0         0   -0.0707         0         0
         0         0         0         0    0.0030         0
         0         0         0         0         0   -0.0001
```

- **eye :** Genera la matrice identità di ordine assegnato.

- **zeros :** Genera un vettore o una matrice nulla.

```
>> zeros(3)

ans =

    0    0    0
    0    0    0
    0    0    0

>> b=zeros(1:2)

b =

    0    0

>> c=[b ; 1 2]

c =

    0    0
    1    2
```

- **ones :** Genera un vettore o una matrice con elementi tutti uguali ad 1.

- **rand :** Genera un vettore o una matrice di numeri casuali.

```
>> c=rand(3)

c =

   0.6665   0.7701   0.9909
   0.6768   0.7374   0.5039
   0.9425   0.8663   0.6291

>> b=rand(1,4)

b =

   0.7926   0.4486   0.5244   0.1715
```

- **VETTORI E MATRICI: OPERAZIONI ELEMENTO PER ELEMENTO**

- Le operazioni matematiche **+ - * / ^** precedute dal carattere punto **.** agiscono elemento per elemento. Nel caso di vettori:

```
>> u = 1 : 4 ;
>> v = [ 3 2 6 -1];
>> w = u .* v
w =
    3    4   18   -4

>> w = u ./ v
w =
   0.3333   1.0000   0.5000  -4.0000

>> w = u .^ v
w =
   1.0000   4.0000 729.0000   0.2500
>>
```

Nel caso di matrici:

```
>> A = [ 1 2 ;  3 4];
>> B = [ 1 2 ; -1 1];
>> C = A ./ B
C =
    1    1
   -3    4

>> C = A.*B
```

```
C =
     1     4
    -3     4

>> C = A.^ B
C =
    1.0000    4.0000
    0.3333    4.0000
```

- **OPERAZIONI CON I NUMERI COMPESSI**

MatLab gestisce in modo automatico l'algebra dei numeri complessi. Si possono utilizzare i simboli i e j per specificare la parte immaginaria; per esempio il numero $c_1 = 1 - 2i$ viene immesso così:

```
>> c1 = 1 – 2i;
```

Non occorre porre un asterisco tra i o j e un numero, sebbene sia richiesto in altri casi, come :

```
>> c2 = 5 – i*c1;
```

Questa convenzione può causare degli errori se non si sta attenti; per esempio le espressioni y = 7/2*i e x = 7/2i danno due risultati diversi.

È semplice sommare, sottrarre, moltiplicare e dividere i numeri complessi, come mostrano i seguenti esempi:

```
>> s=3+7i;
>> w=5-9i;
>> w+s

ans =

   8.0000 - 2.0000i

>> w*s

ans =

  78.0000 + 8.0000i

>> w/s

ans =

  -0.8276 - 1.0690i
```