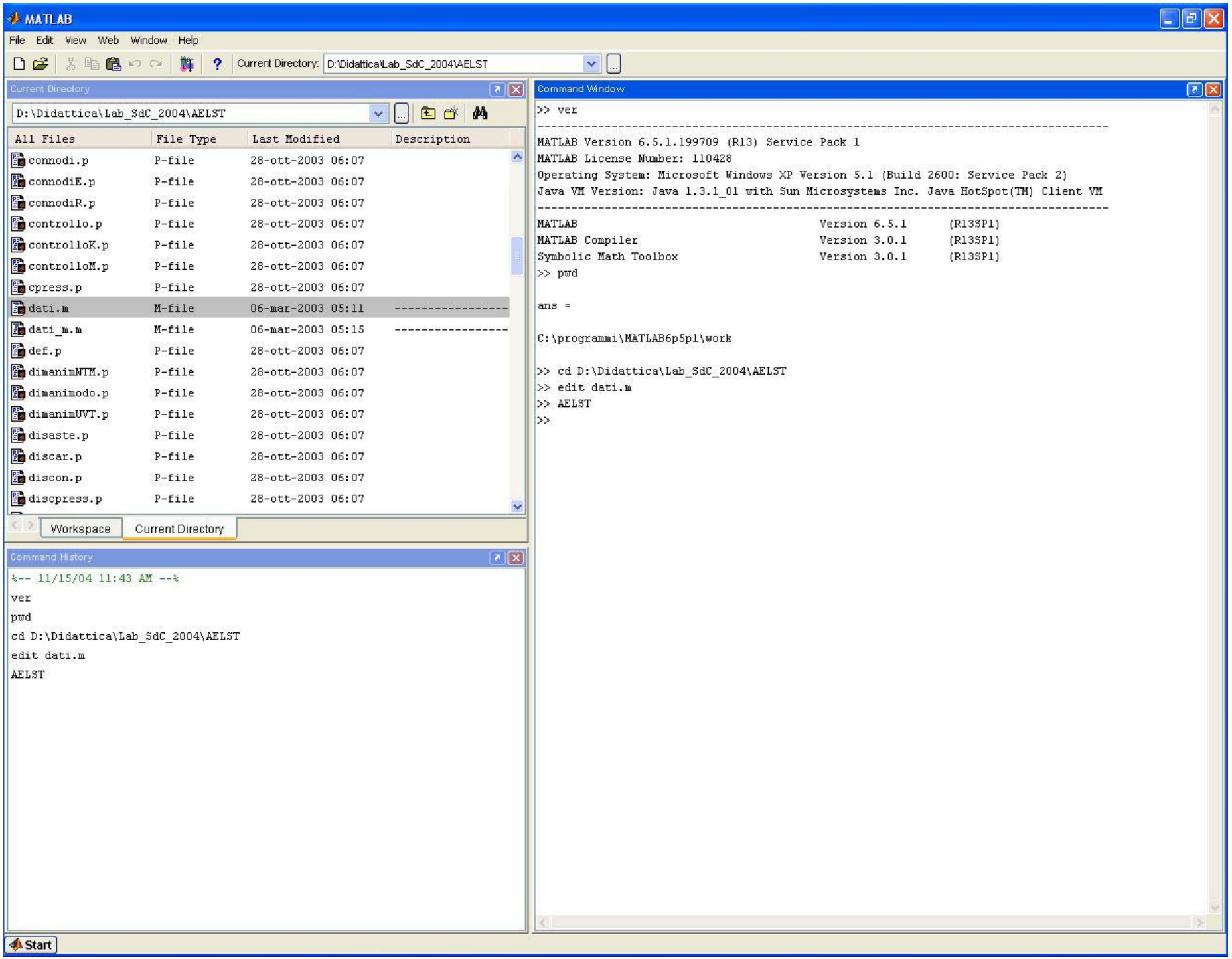
Introduzione all'utilizzo di Matlab

(a cura dell'Ing. G. Castellazzi)

1. Introduzione
   - Generalità e definizione di vettori e matrici
   - Variabili, numeri, funzioni base, funzioni ed espressioni.
   - Operatore di assegnazione =.
   - Funzioni di utilizzo generale: **`help`**, **`lookfor`**, **`whos`**, **`clear`**, **`dir`**, **`cd`**, **`clc`**
2. Calcolo Matriciale
   - Generazione di matrici
   - Utilizzo degli indici
   - Operatore **`:`** (colon)
   - Funzioni più comuni : **`'`** (transpose), **`sum`**, **`det`**, **`diag`**, **`eig`**, **`eye`**, **`rank`**, **`trace`**, **`inv`**, **`poly`**
   - Concatenazione di matrici e vettori
   - Eliminazione di righe e colonne
   - Operazioni tra matrici e vettori, funzioni fondamentali: **`+`**, **`-`**, **`.*`**, **`./`**, **`.^`**
   - Funzioni particolari: espansione scalare e indicizzazione logica
   - Creazione di matrici multidimensionali: funzioni **`squeeze`**, **`reshape`**, **`shiftdim`**
3. Tracciamento di Grafici
   - Tracciare il grafico di una funzione, operatore **`plot`**
   - Utilizzo delle finestre e dei comandi integrati
   - Funzioni correlate: **`hold`**, **`zoom`**, **`grid`**, **`clf`**, **`figure`**, **`axis`**, **`xlabel`**, **`ylabel`**, **`title`**
   - Visualizzazione multipla, operatore **`subplot`**
   - Cenni sul tracciamento di grafici tridimensionali: funzioni **`plot3`**, **`mesh`**, **`surf`**, **`shading`**, **`colormap`**


MATLAB
File Edit View Web Window Help
Current Directory: D:\Didattica\Lab_SdC_2004\AELST
Current Directory
D:\Didattica\Lab_SdC_2004\AELST
All Files File Type Last Modified Description
connodi.p P-file 28-ott-2003 06:07
connodiE.p P-file 28-ott-2003 06:07
connodiR.p P-file 28-ott-2003 06:07
controllo.p P-file 28-ott-2003 06:07
controlloK.p P-file 28-ott-2003 06:07
controlloM.p P-file 28-ott-2003 06:07
cpress.p P-file 28-ott-2003 06:07
dati.m M-file 06-mar-2003 05:11 ----------
dati_m.m M-file 06-mar-2003 05:15 ----------
def.p P-file 28-ott-2003 06:07
dimanimNTM.p P-file 28-ott-2003 06:07
dimanimodo.p P-file 28-ott-2003 06:07
dimanimUVT.p P-file 28-ott-2003 06:07
disaste.p P-file 28-ott-2003 06:07
discar.p P-file 28-ott-2003 06:07
discon.p P-file 28-ott-2003 06:07
discpress.p P-file 28-ott-2003 06:07
Workspace
Current Directory
Command History
%-- 11/15/04 11:43 AM --%
ver
pwd
cd D:\Didattica\Lab_SdC_2004\AELST
edit dati.m
AELST
Command Window
>> ver
MATLAB Version 6.5.1.199709 (R13) Service Pack 1
MATLAB License Number: 110428
Operating System: Microsoft Windows XP Version 5.1 (Build 2600: Service Pack 2)
Java VM Version: Java 1.3.1_01 with Sun Microsystems Inc. Java HotSpot(TM) Client VM
MATLAB Version 6.5.1 (R13SP1)
MATLAB Compiler Version 3.0.1 (R13SP1)
Symbolic Math Toolbox Version 3.0.1 (R13SP1)
>> pwd
ans =
C:\programmi\MATLAB6p5p1\work
>> cd D:\Didattica\Lab_SdC_2004\AELST
>> edit dati.m
>> AELST
>>
Start

1 - Introduzione

Definizione di un matrice

```
» A=[1 1;-1 1]
A =
     1     1
    -1     1
» A(2,1)
ans =
    -1
```

Definizione di vettori riga e colonna

```
» a=[2 0]
a =
     2     0
» A
A =
     1     1
    -1     1
» b=[-1;3]
b =
    -1
     3
```

Matlab è case-sensitive, ‘a’ e ‘A’ sono due variabili diverse !
Il Matlab definisce con ‘i’ o ‘j’ l’unità immaginaria, con ‘Inf’ l’estremo superiore dell’insieme dei reali, con NaN il risultato (numerico) di una forma indeterminata.

Funzioni matematiche base

```
» 2*5.6
ans =
   11.2000
» sqrt(3)
ans =
    1.7321
» sin(pi/2)
ans =
     1
```

Funzioni di approssimazione

```
» round(1.8)
ans =
     2
» round(1.2)
ans =
     1
» fix(1.8)
ans =
     1
» ceil(1.1)
ans =
     2
```

Una operazione priva dell’operatore di assegnazione assegna per default il risultato alla variabile di sistema ‘ans’

```
» sqrt(3)
ans =
    1.7321
» r3=sqrt(3)
r3 =
```

```
    1.7321
»
```

La funzione **help** consente di ottenere aiuto su un comando o funzione matlab:

```
» help plot

 PLOT   Linear plot.
    PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.  If X is a scalar and Y is a vector, length(Y)
    disconnected points are plotted.

    PLOT(Y) plots the columns of Y versus their index.
    If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
    In all other uses of PLOT, the imaginary part is ignored.

    Various line types, plot symbols and colors may be obtained with
    PLOT(X,Y,S) where S is a character string made from one element
    from any or all the following 3 colunms:

           y     yellow       .     point              -     solid
           m     magenta      o     circle             :     dotted
           c     cyan         x     x-mark             -.    dashdot
           r     red          +     plus               --    dashed
           g     green        *     star
           b     blue         s     square
           w     white        d     diamond
           k     black        v     triangle (down)
                              ^     triangle (up)
                              <     triangle (left)

 (...)
```

Il comando **lookfor** permette la ricerca a tutto campo di un particolare termine, ad esempio volendo cercare la funzione che calcola gli autovalori di una matrice (eig) è possibile digitare:

```
» lookfor eigenvalues
CONDEIG Condition number with respect to eigenvalues.
EIG    Eigenvalues and eigenvectors.
EXPM3  Matrix exponential via eigenvalues and eigenvectors.
QZ     QZ factorization for generalized eigenvalues.
EIGS   Find a few eigenvalues and eigenvectors.
A2ODE  Stiff problem, linear with real eigenvalues (A2 of EHL).
A3ODE  Stiff problem, linear with real eigenvalues (A3 of EHL).
B5ODE  Stiff problem, linear with complex eigenvalues (B5 of EHL).
D1ODE  Stiff problem, nonlinear with real eigenvalues (D1 of EHL).
EIGSHOW Graphical demonstration of eigenvalues and singular values.
PDEEIGX Exact calculation of eigenvalues for a 2-by-2 matrix.
EIGFUN Function to return sorted eigenvalues (used in GOALDEMO).
DSORT  Sort complex discrete eigenvalues in descending order.
ESORT  Sort complex continuous eigenvalues in descending order.
QZEIG   Computes the generalized eigenvalues of the pencil (A,E)
HANOWA Matrix whose eigenvalues lie on a vertical line.
LESP   Tridiagonal matrix with real, sensitive eigenvalues.
TRIDIEIG  Find a few eigenvalues of a tridiagonal matrix.
»
```

L’istruzione **whos** visualizza le variabili in memoria e le loro proprietà:

```
»whos
  Name      Size         Bytes  Class
```

```
  A         2x2             32  double array
  a         1x2             16  double array
  ans       1x1              8  double array
  b         2x1             16  double array
  r3        1x1              8  double array

Grand total is 10 elements using 80 bytes

»
```

Le variabili vengono aggiunte in memoria man mano che sono definite, possono esserne rimosse tramite la funzione **clear**:

```
» whos
  Name      Size         Bytes  Class

  A         2x2             32  double array
  a         1x2             16  double array
  ans       1x1              8  double array
  b         2x1             16  double array
  r3        1x1              8  double array

Grand total is 10 elements using 80 bytes

» clear a
» whos
  Name      Size         Bytes  Class

  A         2x2             32  double array
  ans       1x1              8  double array
  b         2x1             16  double array
  r3        1x1              8  double array

Grand total is 8 elements using 64 bytes

» clear all
» whos
»
```

Le funzioni **dir** e **cd** sono analoghe a quelle del DOS, in più, per parametri che contengono spazi, possono essere usate mediante una sintassi funzionale:

```
» cd

c:\windows

» cd c:\prova
» dir

.              ..             Nuova cartella dati.txt

» cd('Nuova cartella')
» dir

.  ..

»
```

Il comando **clc** pulisce la finestra di dialogo.

2 – Calcolo Matriciale

Generazione di matrici

```
» M=[1 0 -1;0 2 1;0 0 3]
M =
     1     0    -1
     0     2     1
     0     0     3
```

Estrazione di una sottomatrice

```
» M(1:2,2:3)
ans =
     0    -1
     2     1
```

L'operatore colon (:) serve a definire intervalli. Può essere utile per creare un vettore scegliendo gli estremi ed il passo delle componenti.
Ad esempio, il vettore le cui componenti sono i valori compresi tra 0 e 2 con passo 0.1 è definito come:

```
» a=[0:0.1:2]
a =
  Columns 1 through 7
         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000
  Columns 8 through 14
    0.7000    0.8000    0.9000    1.0000    1.1000    1.2000    1.3000
  Columns 15 through 21
    1.4000    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000
» a(2:6)
ans =
    0.1000    0.2000    0.3000    0.4000    0.5000
»
```

Nel caso di passo negativo:

```
» b=[2:-0.1:0]
b =
  Columns 1 through 7
    2.0000    1.9000    1.8000    1.7000    1.6000    1.5000    1.4000
  Columns 8 through 14
    1.3000    1.2000    1.1000    1.0000    0.9000    0.8000    0.7000
  Columns 15 through 21
    0.6000    0.5000    0.4000    0.3000    0.2000    0.1000         0
»
```

Sono diversi e molto utili gli funzioni matriciali che il Matlab mette a disposizione:

Trasposizione

```
» B=[1 0 -1;0 1 1;0 0 1]
B =
     1     0    -1
     0     1     1
     0     0     1
» B'
ans =
```

```
     1     0     0
     0     1     0
    -1     1     1
»
```

Somma degli elementi di un vettore (o di una matrice)

```
» vettore=[1 0 -1 2 1 4 2]
vettore =
     1     0    -1     2     1     4     2
» sum(vettore)
ans =
     9
```

Elementi della diagonale

```
» A=[1 0 -1;1 2 -1;2 0 -2]
A =
     1     0    -1
     1     2    -1
     2     0    -2
» diag(A)
ans =
     1
     2
    -2
```

Calcolo degli autovalori

```
» A
A =
     1     0    -1
     1     2    -1
     2     0    -2
» eig(A)
ans =
     2
     0
    -1
```

Generazione di matrici particolari (identità, nulle, …)

```
» 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
» zeros(3,2)
ans =
     0     0
     0     0
     0     0
```

Calcolo del rango.

```
» A
A =
     1     0    -1
```

```
     1     2    -1
     2     0    -2
» rank(A)
ans =
     2
```

Traccia.

```
» A
A =
     1     0    -1
     1     2    -1
     2     0    -2
» trace(A)
ans =
     1
```

Inversa.

```
» B=[1 0 -1;0 1 1;0 0 2]
B =
     1     0    -1
     0     1     1
     0     0     2
» inv(B)
ans =
    1.0000         0    0.5000
         0    1.0000   -0.5000
         0         0    0.5000
```

Polinomio caratteristico

```
» A
A =
     1     0    -1
     1     2    -1
     2     0    -2
» poly(A)
ans =
     1    -1    -2     0
»
```

Matrici e vettori possono essere concatenati usando opportunamente gli funzioni di concatenazione , e ; come illustrato nell’esempio:

```
» a=[1 0 -1]
a =
     1     0    -1
» b=[2 2 4]
b =
     2     2     4
» c=[a b]
c =
     1     0    -1     2     2     4
» d=[a;b]
d =
     1     0    -1
     2     2     4
»
» A=[1 2;3 4]
```

```
A =
     1     2
     3     4
» B=[1;1]
B =
     1
     1
» W=[A,B;a]
W =
     1     2     1
     3     4     1
     1     0    -1
»
```

Analogamente è possibile eliminare righe o colonne. La maniera più semplice in tale senso è quella di assegnare alla riga o colonna da eliminare il vettore vuoto [].

Inoltre, tra matrici consistenti è possibile adoperare i classici funzioni matematici:

```
» A=[1 0 -1;1 2 -1;2 0 -2]
A =
     1     0    -1
     1     2    -1
     2     0    -2
» B=[1 0 -1;0 1 1;0 0 1]
B =
     1     0    -1
     0     1     1
     0     0     1
» A+B*A
ans =
     0     0     0
     4     4    -4
     4     0    -4
» A=[2 0 1;1 1 1]
A =
     2     0     1
     1     1     1
» B=[4 0 1]'
B =
     4
     0
     1
» A*B
ans =
     9
     5
»
```

Espansione scalare. Tale funzione permette di assegnare in maniera estremamente semplice uno stesso valore scalare ad una matrice senza ricorrere ad una assegnazione consistente:

```
» A=[1 4 5;2 1 0;1 1 0]
A =
     1     4     5
     2     1     0
     1     1     0
» A(2:3,2:3)=100
A =
     1     4     5
     2   100   100
```

```
     1   100   100
»
```

Indicizzazione logica. Tale funzione consente di restringere l'operazione di assegnazione agli elementi della matrice che soddisfano una specificata condizione:

```
» x=randn(1,20)
x =
  Columns 1 through 7
   -0.4326   -1.6656    0.1253    0.2877   -1.1465    1.1909    1.1892
  Columns 8 through 14
   -0.0376    0.3273    0.1746   -0.1867    0.7258   -0.5883    2.1832
  Columns 15 through 20
   -0.1364    0.1139    1.0668    0.0593   -0.0956   -0.8323
» x(x>0)
ans =
  Columns 1 through 7
    0.1253    0.2877    1.1909    1.1892    0.3273    0.1746    0.7258
  Columns 8 through 11
    2.1832    0.1139    1.0668    0.0593
```

Matrici multidimensionali. Tali strutture dati vengono essenzialmente trattate come le classiche strutture bidimensionali.

```
» N=rand(3,4,5)
N(:,:,1) =
    0.9501    0.4860    0.4565    0.4447
    0.2311    0.8913    0.0185    0.6154
    0.6068    0.7621    0.8214    0.7919
N(:,:,2) =
    0.9218    0.4057    0.4103    0.3529
    0.7382    0.9355    0.8936    0.8132
    0.1763    0.9169    0.0579    0.0099
N(:,:,3) =
    0.1389    0.6038    0.0153    0.9318
    0.2028    0.2722    0.7468    0.4660
    0.1987    0.1988    0.4451    0.4186
N(:,:,4) =
    0.8462    0.6721    0.6813    0.5028
    0.5252    0.8381    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289
N(:,:,5) =
    0.3046    0.6822    0.1509    0.8600
    0.1897    0.3028    0.6979    0.8537
    0.1934    0.5417    0.3784    0.5936
```

L'operatore **size** specifica l'estensione della matrice lungo le dimensioni che la caratterizzano.

```
» size(N)
ans =
     3     4     5
» N(:,:,2)
ans =
    0.9218    0.4057    0.4103    0.3529
    0.7382    0.9355    0.8936    0.8132
    0.1763    0.9169    0.0579    0.0099
```

Analogamente alle matrici a due dimensioni, è possibile adoperare la funzionalità di espansione scalare:

```
» N(:,:,3)=0
N(:,:,1) =
    0.9501    0.4860    0.4565    0.4447
    0.2311    0.8913    0.0185    0.6154
    0.6068    0.7621    0.8214    0.7919
N(:,:,2) =
    0.9218    0.4057    0.4103    0.3529
    0.7382    0.9355    0.8936    0.8132
    0.1763    0.9169    0.0579    0.0099
N(:,:,3) =
     0     0     0     0
     0     0     0     0
     0     0     0     0
N(:,:,4) =
    0.8462    0.6721    0.6813    0.5028
    0.5252    0.8381    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289
N(:,:,5) =
    0.3046    0.6822    0.1509    0.8600
    0.1897    0.3028    0.6979    0.8537
    0.1934    0.5417    0.3784    0.5936
»
```

Sono essenziali le funzioni **squeeze**, **reshape** e **shiftdim**.

La funzione **squeeze** rimuove le dimensioni superflue (pari ad 1):

```
» R=rand(2,1,2)
R(:,:,1) =
    0.8385
    0.5681
R(:,:,2) =
    0.3704
    0.7027
» whos
  Name      Size         Bytes  Class

  R         2x1x2           32  double array

Grand total is 4 elements using 32 bytes

» S=squeeze(R)
S =
    0.8385    0.3704
    0.5681    0.7027
» whos
  Name      Size         Bytes  Class

  R         2x1x2           32  double array
  S         2x2             32  double array

Grand total is 8 elements using 64 bytes

»
```

La funzione **reshape** dispone gli elementi di una matrice in una iso-dimensionale (coerente) con diverse estensioni:

```
» A=[1 3 5 1;0 1 2 2;4 1 1 3;0 1 1 2]
```

```
A =
     1     3     5     1
     0     1     2     2
     4     1     1     3
     0     1     1     2
» B=reshape(A,2,8)
B =
     1     4     3     1     5     1     1     3
     0     0     1     1     2     1     2     2
»
```

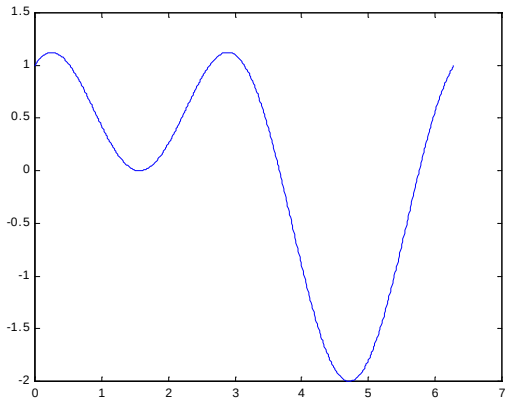
la funzione **shiftdim** trasla verso destra di una quantità specificata le dimensioni della variabile:

```
» a=rand(2,1,3);
» whos
  Name      Size         Bytes  Class

  a         2x1x3           48  double array

Grand total is 6 elements using 48 bytes

» b=shiftdim(a,1);
» whos
  Name      Size         Bytes  Class

  a         2x1x3           48  double array
  b         1x3x2           48  double array

Grand total is 12 elements using 96 bytes
```

3 – Tracciamento di Grafici

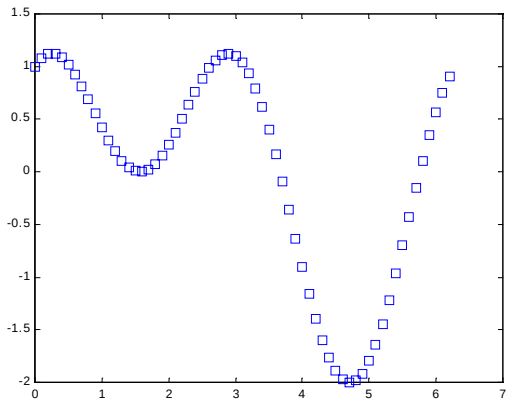
La funzione più significativa è indubbiamente la **plot**:

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

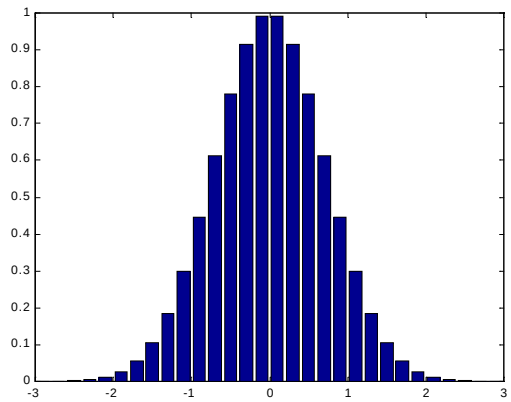


La funzione plot è in grado di tracciare le curve impiegando svariati simboli:

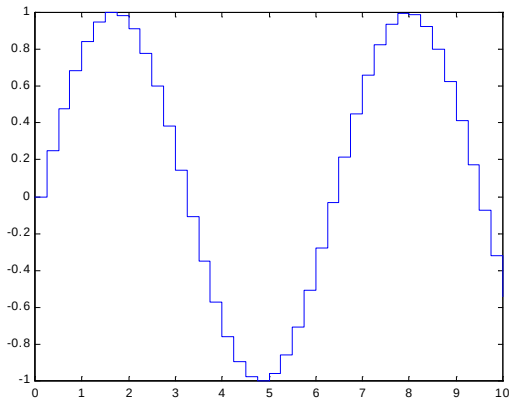
```
» x=[0:0.1:2*pi];
» y=sin(x)+cos(2*x);
» plot(x,y,'s');
```



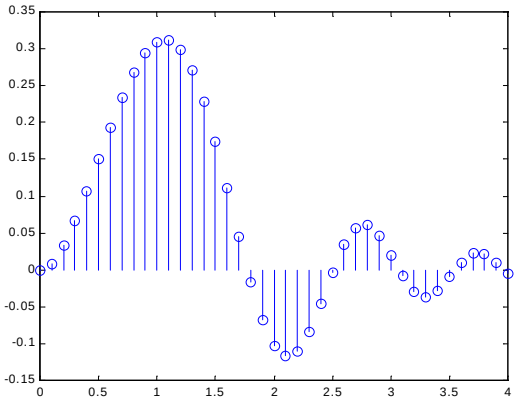
```
» x=-2.9:0.2:2.9;
» bar(x,exp(-x.*x));
```



```
» x=0:0.25:10;
» stairs(x,sin(x));
```

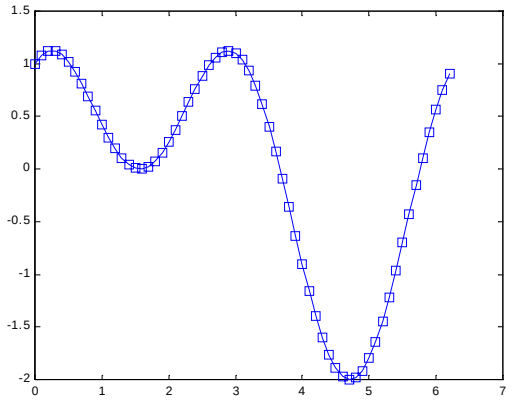


```
» x=0:0.1:4;
» y=sin(x.^2).*exp(-x);
» stem(x,y)
```



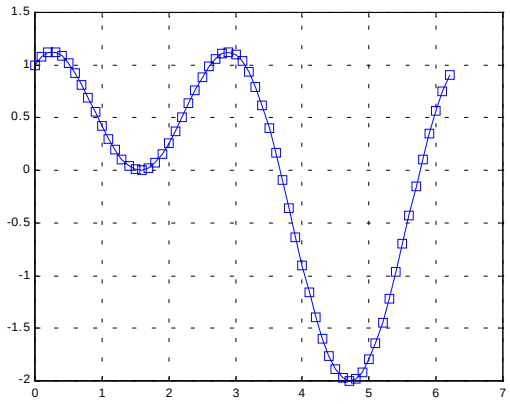
Insieme alla funzione **plot** e alla varietà di parametri impiegabili che la caratterizzano è possibile utilizzare una serie di funzioni di fondamentale ausilio nel tracciamento di grafici. La funzione **hold**, se attivata dal parametro **on**, consente il tracciamento di più grafici nella stessa finestra:

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



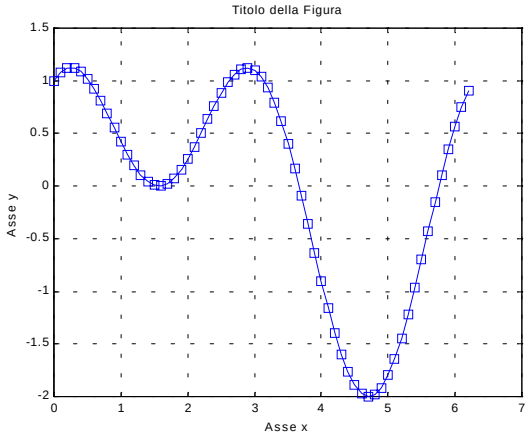
La funzione **zoom** (attivabile anche dall'icona sulla finestra stessa) permette l'ingrandimento di regioni del grafico. L'attivazione della funzione **grid** traccia un reticolato sul grafico.

```
» grid on
```



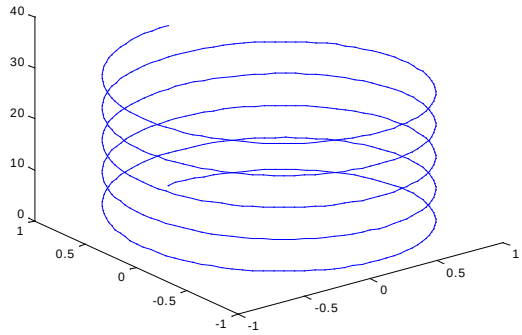
Il comando **clf** pulisce la finestra corrente, mentre **figure** ne apre una nuova. L'istruzione **axis**, oltre a definire la scalatura degli assi, permette di definire l'apparenza della figura. Le istruzioni **xlabel**, **ylabel** e **title** etichettano gli assi e la figura:

```
» xlabel('Asse x')
» ylabel('Asse y')
» title('Titolo della Figura')
```



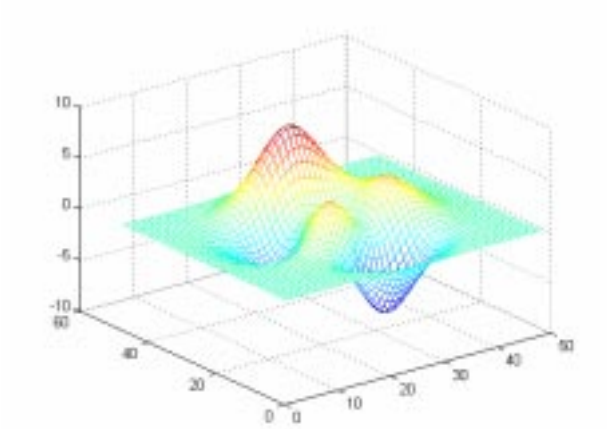
Grafici tridimensionali sono tipicamente tracciati per mezzo delle istruzioni **plot3**, **mesh** e **surf**.
La **plot3** consente di tracciare curve:

```
» t=0:pi/50:10*pi;
» plot3(sin(t),cos(t),t);
```



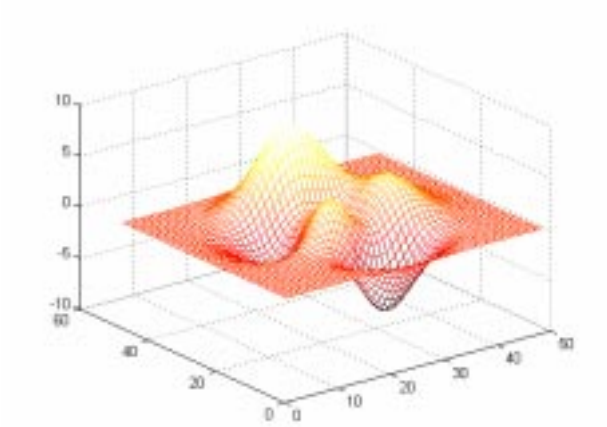
L'istruzione **mesh** traccia il grafico di una superficie. Ha come parametro la matrice che descrive l'andamento di tale superficie.

```
» Z=peaks;
» mesh(Z)
```



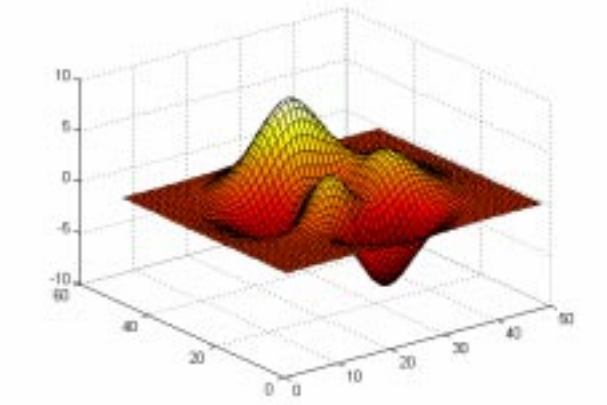
Tramite **colormap** possono essere impiegate differenti mappe di colori.

```
» colormap hot
```



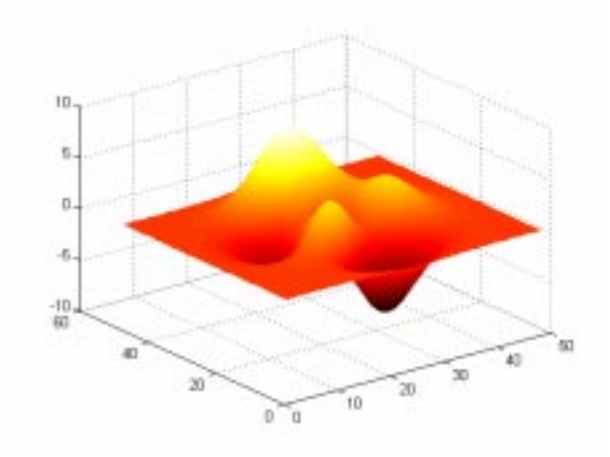
Alternativamente a **mesh** può essere impiegata **surf**, quest'ultima non colora semplicemente il reticolo ma l'intera superficie.

```
» surf(Z)
```



L'istruzione **shading**, mettendo a disposizione diversi parametri, permette di raffinare ulteriormente l'esposizione della superficie tracciata:

```
» shading interp
```


10
5
0
-5
-10
60
40
20
0
0
10
20
30
40
50