Scuola universitaria professionale
della Svizzera italiana

Dipartimento
Tecnologie
Innovative

Introduzione a MATLAB

Ing. Roberto Bucher

8 marzo 2011

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Origini

MATLAB è uno dei programmi scientifici di maggior diffusione, grazie alle sue numerose applicazioni in campi quali l'elettronica, la controllistica, l'analisi dei segnali, l'elaborazione di immagini, la chimica, la statistica e numerosi altri.

Viene utilizzato in molti corsi universitari e di ingegneria, e sono ormai numerose le pubblicazioni scientifiche che utilizzano l'ambiente di MATLAB quale sostegno matematico della teoria.

La primissima versione di MATLAB risale alla fine degli anni '70, scritta alla *University of New Mexico* e alla *Stanford University* quale pacchetto software di supporto alle lezioni di *Algebra lineare* e *Analisi numerica*. Oggi MATLAB non si limita più al solo calcolo matriciale e numerico, ma ha sviluppato tutta una serie di funzioni per le applicazioni più diverse nel campo scientifico. La semplicità del linguaggio permette di risolvere problemi molto complessi senza dover sviluppare programmi in C o altri linguaggi di programmazione.

1.2 Piattaforme HW

MATLAB gira su diversi sistemi, che vanno dal PC al Mac, fino alle Workstation Unix (Sun, Alpha). Essendo le funzioni scritte nella maggior parte dei casi come file *testo*, la portabilità da un sistema all'altro di funzioni specifiche scritte dall'utente è garantita praticamente al 100%.

1.3 Toolbox

I toolbox sono una collezione di programmi e funzioni basate su MATLAB, che permettono la soluzione di problemi mirati a campi particolari dell'ingegneria. Ne esistono moltissimi e ogni mese la lista si arricchisce di nuovi programmi.

Nella tabella 1.1 sono riportati alcuni dei toolbox e Add-On a disposizione

SIMULINK	SIMULINK Accelerator	MATLAB Compiler
Real Time Workshop	Symbolic Math	Ext. Symbolic Math
MATLAB C Compiler	Matab C Math Library	Communication
Control System	Robust Control	Autom. Controller Design
Financial	Fuzzy Logic	Optimization
Image Processing	Wavelet	Model Predictive Control
Partial Diff. Equation	Signal Processing	Nonlinear Control Design
DSP Blockset	Chemometrics	Spline
Statistik	Frequency Domain Ident.	Neural Network

Tabella 1.1: Alcuni toolbox di MATLAB

Capitolo 2

Basi

2.1 Introduzione

MATLAB è un ambiente di lavoro basato principalmente su comandi in linea, come potrebbe essere l'ambiente DOS o UNIX. I comandi possono essere digitati direttamente sulla linea di comando o letti da file testo. Esiste la possibilità di scrivere applicazioni con finestre e bottoni, come pure di compilare funzioni scritte in un linguaggio tipo C o Fortran. Come potremo vedere, una grossa quantità di funzioni sono direttamente leggibili, essendo programmate in questi script testo con estensione *.m*.

L'utilizzo di tutte le funzioni risulta semplice e immediato.

Quando facciamo partire MATLAB otteniamo una finestra contenente una semplice linea di comando. Qui l'utente potrà fare tutte le sue richieste e dare i suoi comandi.

Possiamo digitare direttamente sulla linea di comando le operazioni che vogliamo fare, sia che si tratti operazioni, sia che si tratti di funzioni particolari.

2.2 Tipi di dati

MATLAB lavora con alcuni tipi di dati che sono:

- La matrice n-dimensionale di numeri reali, complessi, caratteri o strutture più complesse.
- La cella, un contenitore per svariati tipi di dati.

Per la creazione di variabili occorre tenere presente alcune convenzioni

1. Le variabili sono *case-sensitive*
2. I nomi di variabili possono contenere fino a 19 caratteri
3. I nomi delle variabili devono iniziare con una lettera e possono contenere lettere, numeri e `'_'`.

Per la creazione e l'utilIzzo di matrici multidimensionali, si rimanda alla fine di questa documentazione. Per il momento ci occuperemo unicamente di matrici bidimensionali.

Una matrice può essere creata in differenti modi

- Digitando esplicitamente tutti gli elementi.
- Utilizzando funzioni specifiche di MATLAB.
- Utilizzando matrici create in file `.m` (script).
- Caricando le matrici da files esterni.

In MATLAB non esistono dichiarazioni di tipi o di dimensioni. MATLAB alloca direttamente la memoria necessaria ogni volta che si dichiara o si modifica una variabile.

Utilizzando il primo metodo possiamo dichiarare in modo semplice piccole matrici. Tra un valore e l'altro può essere messa una , o uno spazio. Per introdurre una nuova riga nella matrice, occorre terminare la precedente con un ";", oppure andare a capo.

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

Per matrici di dimensione più grande è spesso utile utilizzare alcuni metodi e alcune funzioni messe a disposizione da MATLAB.

```
>> x=1:10
x =
     1     2     3     4     5     6     7     8     9    10
>> y=linspace(0,1,11)
y =
  Columns 1 through 6
     0    0.1000    0.2000    0.3000    0.4000    0.5000
  Columns 7 through 11
     0.6000    0.7000    0.8000    0.9000    1.0000
>> x=0:0.4:2
x =
   0    0.4000    0.8000    1.2000    1.6000    2.0000
>> y=logspace(-1,1,10)
y =
  Columns 1 through 6
   0.1000    0.1668    0.2783    0.4642    0.7743    1.2915
  Columns 7 through 10
   2.1544   5.9948   10.0000
```


Altri comandi permettono di creare matrici particolari

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
>> ones(2,3)
ans =
     1     1     1
     1     1     1
>> zeros(2,4)
ans =
     0     0     0     0
     0     0     0     0
```

Quando si creano matrici molto grandi si può eliminare l'output terminando il comando con *;*.

I comandi *rand* e *randn* creano matrici di numeri casuali di dimensioni n x n o n x m.

2.3 Indirizzamento di matrici

Gli elementi di una matrice possono essere di qualsiasi tipo

```
>> x=[1 sin(0.7*pi) sqrt(2) 1+2*(4+3)/5]
x =
        1.0000    0.8090    1.4142    3.8000
```

Per accedere ad uno o più elementi di una matrice si utilizzano le parentesi ( ).

```
>> x(4)
ans =
        3.8000
>> x(6)=sin(x(1))
x =
    1.0000    0.8090    1.4142    3.8000         0    0.8415
```

MATLAB ha automaticamente adattato le dimensioni della matrice x alla nuova situazione e ha introdotto uno 0 nel valore $x(5)$ non ancora assegnato.

```
>> A=[1,2,3;4,5,6;7,8,9];
>> r=[10 11 12];
>> A=[A;r]
A =
    1     2     3
    4     5     6
    7     8     9
   10    11    12
```


MATLAB unisce tra di loro matrici e vettori. Si possono estrarre anche più elementi contemporaneamente

```
>> b=A(1:3,:)
b =
    1     2     3
    4     5     6
    7     8     9
```

$1:3$ significa *intervallo da 1 a 3*, mentre *:* significa *tutti*. La riga $A(1:3,:)$ può essere letta come *le righe di A da 1 a 3 e tutte le colonne*.

In quest'ultimo esempio vengono estratte le righe di A da 1 a 3 in ordine inverso, e le colonne 1 e 2.

```
>> c=a(3:-1:1,[1 2])
c =
    7     8
    4     5
    1     2
```

2.4 Manipolazione di matrici

Altre operazioni speciali su una matrice sono descritte nella tabella 2.1

flipud(A)	scambia i valori della matrice A dall'alto in basso
fliplr(A)	scambia i valori della matrice A da sinistra a destra
rot90(A)	ruota la matrice A di 90 gradi in senso antiorario
tril	estrae la parte superiore triangolare
triu	estrae la parte inferiore triangolare
reshape(A,m,n)	ridimensiona la matrice A a m x n. A deve contenere m*n elementi !
end	ultimo indice di una matrice
diag(v)	crea una matrice con sulla diagonale i valori del vettore v
diag(A)	crea un vettore con gli elementi della diagonale della matrice A

Tabella 2.1: Operazioni su matrici

Altre funzioni sono descritte nell'allegato A.

Le operazioni per determinare le dimensioni di una matrice sono

```
>> length(a)  % ritorna il massimo tra righe e colonne
ans =
       3
```

```
>> size(a)
ans =
    3     3
>> [r,c]=size(a)
r =
    3
c =
    3
    1     2
```

È possibile accedere ad un elemento di una matrice con un solo indice. In questo caso gli elementi sono visti come una colonna sotto l'altra

```
a =
       1     2     3
       4     5     6
       7     8     9
>> a(4)
ans =
     2
```

2.5 Espressioni

MATLAB è un linguaggio che si basa su espressioni nella forma

```
variabile = espressione
```

o semplicemente *espressione*

```
>> 100/3
ans =
   33.3333
```

La risposta di un espressione senza assegnamento viene scritta in una variabile di default chiamata *ans* (answer). Da notare inoltre che il simbolo % inizia una parte di commento, e non viene quindi valutato dal sistema.

2.6 Informazioni sul Workspace

Con gli esempi precedenti sono state create delle variabili che si trovano memorizzate nel em workspace. È possibile avere informazioni su queste variabili con i comandi *who* e *whos*.

```
>> who
Your variables are:
A         b         r
ans       c         x
>> whos
```



```
  Name    Size    Elements  Bytes   Density   Complex
   A      4 by 3    12       96       Full      No
   ans    1 by 1    1        8        Full      No
   b      3 by 3    9        72       Full      No
   c      2 by 2    4        32       Full      No
   r      1 by 3    3        24       Full      No
   x      1 by 6    6        48       Full      No
Grand total is 35 elements using 280 bytes
```

Ogni elemento di una matrice reale occupa 8 bytes.
Esistono inoltre alcune variabili predefinite di sistema che sono

1. *ans* nome della variabile di default in uscita
2. *pi* pi greco
3. *eps* il numero più piccolo che addizionato a 1 crea un numero floating- point maggiore di 1 nel computer
4. *inf* infinito
5. *NaN Not-a-Number*, numero indefinito (p.es. 0/0)
6. i e j $i = j = \sqrt{-1}$
7. *realmin* il numero reale positivo più piccolo utilizzabile
8. *realmax* il numero reale positivo più grande utilizzabile

Interessante è la possibilità di avere calcoli che danno risultati infiniti o non definiti senza che il sistema si blocchi o dia particolari errori

```
>> 1/0
Warning: Divide by zero
ans =
   Inf
>> -1/0
Warning: Divide by zero
ans =
  -Inf
>> 0/0
Warning: Divide by zero
ans =
   NaN
```

Mediante il comando *clear* si possono cancellare tutte le variabili definite o solo alcune di esse

```
>> clear A     % elimina la variabile A
>> clear       % elimina tutte le variabili
>> who
Your variables are:
```

È inoltre possibile registrare un'intera sessione di lavoro utilizzando il comando *diary*.

2.7 Accesso al sistema operativo

Dall'interno di MATLAB è possibile accedere direttamente ad informazioni di sistema, mediante alcuni comandi particolari, descritti nella tabella 2.2

what	Lista dei file *.m* nella directory corrente
dir	Lista di tutti i file nella directory corrente
ls	Analogo a *dir*
type test	Visualizza il file test.m
delete test	Elimina il file test.m
cd <path>	Cambia la directory di lavoro
chdir <path>	Come cd <path>
cd	Mostra la directory corrente di lavoro
chdir	Come *cd*
pwd	Come *cd*
which	Mostra tutto il path di un file *.m*
computer	Tipo di computer
web	Apre il Web browser

Tabella 2.2: Accesso al sistema operativo

È inoltre sempre possibile richiamare direttamente un comando del sistema operativo precedendolo con il simbolo *!*

Per esempio

```
>> !edit test.m
```

apre il programma *edit* per modificare il file test.m. In ambiente *Windows* è possibile anche utilizzare il comando

```
>> dos('edit test.m')
```

per effettuare la medesima operazione.

2.8 Salvataggio e richiamo di dati

Tutte le variabili definite o calcolate in una sessione di lavoro possono essere salvate e richiamate a piacimento mediante il comando *save* e *load*.

```
>> save
 Saving to: Matlab.mat
>> save prova
```

e in un futuro

```
>> load
Loading from: Matlab.mat
>> load prova
```


2.9 Formati dei numeri

Nel prossimo esempio possiamo vedere l'uso del comando *format* per la scelta del formato numerico desiderato

```
>> media=40/3
media =
   13.3333
>> format long,media
media =
    13.33333333333334
>> format short e,media
media =
    1.3333e+001
>> format long e,media
media =
    1.333333333333334e+001
>> format hex,media      % formato esadecimale
media =
     402aaaaaaaaaaab
>> format bank,media
media =
     13.33
>> format +,media
media =
    +
>> format rat,media      % formato frazione
media =
    40/3
>> format short,media    % formato di default
media =
    13.3333
```

2.10 Altre caratteristiche di base

È possibile dare più comandi sulla stessa linea, separandoli con , o ;.

```
>> matite=5,gomme=3,penne=2
matite =
       5
gomme =
       3
penne =
       2
```

Se si termina un comando con il carattere ; non viene visualizzato il risultato di questo comando

```
>> matite=5;gomme=3;penne=2;
```

L'uscita da MATLAB è possibile con il comando *quit*.

2.11 Stringhe

Un testo in MATLAB non è nient'altro che un vettore di caratteri. Si possono manipolare facilmente, come pure è possibile costruire frasi concatenando tra di loro delle matrici.

```
>> a='Mi'
a =
  Mi
>> b='chiamo'
b =
  chiamo
>> c='Roberto'
c =
  Roberto
>> d=[a,' ',b,' ',c]
d =
  Mi chiamo Roberto
```

Trattandosi di matrici, non è possibile avere un numero diverso di caratteri tra una riga e un'altra della stessa matrice ! Eventualmente occorre utilizzare il comando *str2mat* che adatta il numero di colonne di una matrice secondo la stringa più lunga.

```
>> str2mat('casa','cassone','nome')
ans =
  casa
  cassone
  nome
>> size(ans)
ans =
  3 7
```

MATLAB utilizza 2 byte per ogni carattere.

```
>> a='abc'
a =
  abc
>> whos
Name      Size            Bytes  Class    Attributes

a         1x3                 6  char
```

Le funzioni per la manipolazione di stringhe sono descritte nella tabella 2.3
Altre funzioni sono descritte nell'allegato A.

abs	Converte una stringa nel corrispondente valore numerico
isstr	vero se la variabile è una stringa
setstr	converte valore numerico in stringa
str2mat	Crea una matrice testo
lower	converte in minuscole
upper	converte in maiuscole
strcmp	confronta 2 stringhe
int2str	converte intero in stringa
num2str	converte numero in stringa
sprintf	converte numero in stringa (come C)
str2num	converte stringa in numero
sscanf	converte stringa in numero (come C)
dec2hex	conversione decimale-esadecimale
hex2dec	conversione esadecimale-decimale
hex2num	conversione esadecimale a floating point
base2dec	conversione da base B a decimale
dec2base	conversione da decimale a base B

Tabella 2.3: Operazioni con stringhe

Capitolo 3

L'Help in linea

3.1 Introduzione

MATLAB contiene un sistema di help in linea che fornisce numerose informazioni sull'uso di tutte le funzioni. I comandi *help* e *lookfor* permettono una ricerca veloce di informazioni.

3.2 Il comando *help*

È sufficiente chiamare il comando *help* seguito dalla funzione desiderata.

```
>> help det
DET         Determinant.
      DET(X) is the determinant of the square matrix X.
```

La chiamata semplicemente di *help* mostra una lista degli argomenti disponibili, che possono poi in seguito essere approfonditi.

```
>> help
HELP topics:

toolbox\local    -  Local function library.
matlab\datafun   -  Data analysis and Fourier transform
                    functions.
matlab\elfun     -  Elementary math functions.
matlab\elmat     -  Elementary matrices and matrix
                    manipulation.
matlab\funfun    -  Function functions - nonlinear numerical
                    methods.
matlab\general   -  General purpose commands.
matlab\color     -  Color control and lighting model
                    functions.
matlab\graphics  -  General purpose graphics functions.
matlab\iofun     -  Low-level file I/O functions.
matlab\lang      -  Language constructs and debugging.
matlab\matfun    -  Matrix functions - numerical linear
                    algebra.
```

```
matlab\ops       -  Operators and special characters.
matlab\plotxy    -  Two dimensional graphics.
matlab\plotxyz   -  Three dimensional graphics.
matlab\polyfun   -  Polynomial and interpolation functions.
matlab\sounds    -  Sound processing functions.
matlab \sparfun  -  Sparse matrix functions.
matlab\specfun   -  Specialized math functions.
matlab\specmat   -  Specialized matrices.
matlab\strfun    -  Character string functions.
matlab\dde       -  DDE Toolbox.
matlab\demos     -  The Matlab-Expo and other
                    demonstrations.
simulink\simulink-  Simulink model analysis and construction
                    functions.
simulink\simdemos-  Simulink demonstrations and samples.
simulink\blocks  -  Simulinkblock library.
simulink\sb2sl   -  SystemBuild 3.0 model import into
                    Simulink.
toolbox\control  -  Control System Toolbox.
toolbox\ncd      -  Nonlinear Control Design Toolbox.

For more help on directory/topic, type "help topic".

>> help plotxy

 Two dimensional graphics.

 Elementary X-Y graphs.
   plot              - Linear plot.
   loglog            - Log-log scale plot.
   semilogx          - Semi-log scale plot.
   semilogy          - Semi-log scale plot.
   fill              - Draw filled 2-D polygons.

 Specialized X-Y graphs.
   polar             - Polar coordinate plot.
   bar               - Bar graph.
   stem              - Discrete sequence or "stem" plot.
   stairs            - Stairstep plot.
   errorbar          - Error bar plot.
   hist              - Histogram plot.
   rose              - Angle histogram plot.
   compass           - Compass plot.
   feather           - Feather plot.
   fplot             - Plot function.
   comet             - Comet-like trajectory.
```

```
Graph annotation.
  title             - Graph title.
  xlabel            - X-axis label.
  ylabel            - Y-axis label.
  text              - Text annotation.
  gtext             - Mouse placement of text.
  grid              - Grid lines.

See also PLOTXYZ, GRAPHICS.
```

3.3 Il comando *lookfor*

Questo comando cerca una stringa particolare all'interno della prima riga dell'help di un certo comando, e rimanda una lista di tutti i comandi trovati.

```
>> lookfor clear
 CLC     Clear command window.
 CLEAR   Clear various quantities from the workspace.
 CLA     Clear axis.
 CLF     Clear Figure.
 CLG     Clear Figure (graph window).
 DBCLEAR Remove an M-file breakpoint set by the DBSTOP command.
```


Capitolo 4

Calcolo scientifico

4.1 Introduzione

MATLAB mette a disposizione una quantità di funzioni scientifiche per l'utilizzo in matematica e ingegneria. Inoltre è già implementato tutto il calcolo con i numeri complessi.

4.2 Funzioni matematiche

La chiamata di una funzione in MATLAB è molto semplice.

```
>> x=log(10)
x =
        2.3026
```

Le funzioni matematiche elementari sono elencate nella tabella 4.1
Altre funzioni sono descritte nell'allegato A.
Tutte le funzioni sono applicabile a scalari e a matrici.

```
>> x=(0:0.1:1)*pi;
>> y=sin(x)
y =
 Columns 1 through 7
  0   0.3090   0.5878   0.8090   0.9511   1.0000   0.9511
 Columns 8 through 11
  0.8090   0.5878   0.3090   0.0000
```

La funzione viene applicata ad ogni elemento della matrice x.

Vediamo ora i metodi messi a disposizione da MATLAB per calcolare i risultati di una funzione come ad esempio

$$y = 0.3 + 4 * \cos(x) * \sin(x)$$

per alcuni valori di x.

Il primo metodo consiste nel calcolo diretto della funzione

```
>> x=linspace(-pi/2,pi/2,7)
x =
```



```
 -1.5708  -1.0472  -0.5236   0   0.5236   1.0472   1.5708
>> y=0.3+4*cos(x).*sin(x.*x)
y =
 0.3000   2.0793   1.2379   0.3000   1.2379   2.0793   0.3000
```

Attenzione all'uso del comando ..

Un altro metodo consiste nel definire in forma testuale la funzione e di applicare in seguito il comando *eval*.

```
>> f='0.3+4*cos(x).*sin(x.*x)'
f =
  0.3+4*cos(x).*sin(x.*x)
>> eval(y)
ans =
 0.3000   2.0793   1.2379   0.3000   1.2379   2.0793   0.3000
```

4.3 Numeri complessi

MATLAB lavora senza problemi e senza accorgimenti particolari anche con numeri complessi. Le operazioni vengono svolte esattamente nello stesso modo di quelle con numeri reali.

La parte complessa di un numero viene indicata aggiungendo la lettera i al secondo elemento, esattamente come nella matematica tradizionale.

```
>> c1=4+7i
c1 =
    4.0000 + 7.0000i
>> c2=4+7*i
c2 =
    4.0000 + 7.0000i
>> c3=4+7j
c3 =
    4.0000 + 7.0000i
```

L'utilizzo del segno * è facoltativo, come pure la scelta tra i e j.

Le operazioni si svolgono in modo tradizionale

```
>> c1+c2
ans =
     8.0000 +14.0000i

>> (4+7i)*(5-3i)
ans =
     41.0000 +23.0000i

>> c1+c2
ans =
     8.0000 +14.0000i
```

```
>> (4+7i)/(3+2i)
ans =
   + 1.0000i
```

Le operazioni tradizionali sui numeri complessi sono riportate nella tabella 4.2 Altre funzioni sono descritte nell'allegato A.

abs(x)	valore assoluto o modulo di x
acos(x)	arc cos di x
acosh(x)	arc cos iperbolico di x
angle(x)	angolo di un numero complesso
asin	arc sin di x
asinh	arc sin iperbolico di x
atan(x)	arc tan di x
atan2(x,y)	arc tan su 4 quadranti
atanh	arc tan iperbolica
ceil	arrotondamento verso + ∞
conj(x)	complesso coniugato di x
cos(x)	coseno
cosh(x)	cos iperbolico
exp(x)	esponente e
fix(x)	arrotondamento verso 0
floor(x)	arrotondamento verso - $-\infty$
gcd(x,y)	massimo comun divisore
imag(x)	parte immaginaria di x complesso
lcm	minimo comune multiplo (least common multiple)
log(x)	logaritmo naturale di x
log2(x)	logaritmo base 2
log10(x)	logaritmo base 10 di x
mod	divisione intera
real(x)	parte reale di x complesso
rem(x,y)	resto di x/y
round(x)	arrotondamento verso il numero intero più vicino
sign(x)	segno di x (+, -, 0)
sin(x)	sin di x
sinh(x)	sin iperbolico di x
sqrt(x)	radice quadrata di x
tan(x)	tan di x
tanh(x)	tan iperbolica di x

Tabella 4.1: Funzioni scientifiche

abs	modulo di un numero complesso
angle	fase di un numero complesso
conj	numero complesso coniugato
real	parte reale
imag	parte immaginaria

Tabella 4.2: Operazioni sui numeri complessi

Capitolo 5

Operazioni con matrici

5.1 Matematica tra scalari e matrici

In MATLAB si possono effettuare direttamente operazioni tra matrici e scalari

```
>> a
a =
     1     2     3     4     5
>> 2*a-1
ans =
     1     3     5     7     9
```

In questo caso la matrice a è stata moltiplicata per 2 e ad ogni elemento è stato sottratto 1.

5.2 Matematica tra matrici

È possibile definire operazioni tra gli elementi di due matrici solo nel caso che le due matrici abbiano la stessa lunghezza.

```
>> a
a =
  1     2     3     4     5
>> b
b =
  1     3     5     7     9
>> a+b
ans =
  2     5     8    11    14
>> a-b
ans =
  0    -1    -2    -3    -4
>> a.*b
ans =
  1     6    15    28    45
>> a./b
```



```
ans =
 1.0000    0.6667    0.6000    0.5714    0.5556
>> a.^b
ans =
  1         8       243       16384     1953125
>> 2.^a
ans =
  2     4     8    16    32
>> a.^2
ans =
  1     4     9    16    25
```

Sono particolarmente da notare le ultime operazioni (`.*, ./, .^`) che sono definite come operazioni ., poiché, anche tra matrici, queste operazioni sono normalmente definite in un modo differente (vedi capitolo dedicato all'algebra lineare). Tutte queste operazioni sono svolte elemento per elemento. Il tentativo di eseguire una normale moltiplicazione tra le due matrici a e b porta ad un errore:

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

In MATLAB sono anche definite le due operazioni *cross* e *dot* per il prodotto vettoriale e per il prodotto scalare tra vettori.

5.3 Orientazione delle matrici

Abbiamo visto precedentemente che i valori di una riga di una matrice vengono separati tra di loro con una ,, mentre per passare alla riga successiva si utilizza un ;. Possiamo anche trasporre una matrice riga con l'operatore ', per arrivare ad ottenere un vettore colonna.

```
>> a=1:5
a =
  1     2     3     4     5
>> b=a'
b =
     1
     2
     3
     4
     5
```

L'operazione funziona anche nel modo inverso

```
>> c=b'
c =
   1     2     3     4     5
```

Esiste anche l'operatore .', La differenza esiste unicamente quando viene trasposta una matrice di elementi complessi.

```
>> d=a+i*a
d =
  Columns 1 through 3
  1.0000 + 1.0000i   2.0000 + 2.0000i   3.0000 + 3.0000i
  Column 4 through 5
  4.0000 + 4.0000i   5.0000 + 5.0000i
>> e=d'        % matrice trasposta coniugata
e =
   1.0000 - 1.0000i
   2.0000 - 2.0000i
   3.0000 - 3.0000i
   4.0000 - 4.0000i
   5.0000 - 5.0000i
>> f=d.'        % matrice trasposta
f =
   1.0000 + 1.0000i
   2.0000 + 2.0000i
   3.0000 + 3.0000i
   4.0000 + 4.0000i
   5.0000 + 5.0000i
```

5.4 Operatori relazionali e logiche

5.4.1 Operatori relazionali

Gli operatori relazionali definiti in MATLAB sono descritti nella tabell 5.1

>	maggiore
<	minore
>=	maggiore o uguale
<=	minore o uguale
==	uguale
~=	diverso (not uguale)

Tabella 5.1: Operazioni relazionali

L'utilizzo di questi operatori dà come risultato un valore 0 o 1

```
>> 3+5~=8
ans =
     0
```

Questi operatori possono essere applicati anche a matrici

```
>> a=magic(6)
a =
```



```
35     1     6    26    19    24
 3    32     7    21    23    25
31     9     2    22    27    20
 8    28    33    17    10    15
30     5    34    12    14    16
 4    36    29    13    18    11
```

L'operazione *magic* crea una matrice che contiene tutti i numeri interi da 1 a n^2 con somma delle righe e delle colonne costante. Questa matrice ha la particolarità che ogni terza diagonale contiene solo multipli di 3. Possiamo visualizzare questa caratteristica sfruttando un operatore relazionale

```
>> p=(rem(a,3)==0)
p =
     0     0     1     0     0     1
     1     0     0     1     0     0
     0     1     0     0     1     0
     0     0     1     0     0     1
     1     0     0     1     0     0
     0     1     0     0     1     0
```

Possiamo anche utilizzare il comando *find* per determinare gli elementi di una matrice che soddisfano determinate condizioni

```
>> a=1:36;
>> find(rem(a,3)==0)
ans =
  3   6   9  12  15  18  21  24  27  30  33  36
```

5.4.2 Operatori logici

Gli operatori logici definiti in MATLAB sono riportati nella tabella 5.2

`&`	And
`|`	Or
`~`	Not
`xor`	XOR

Tabella 5.2: Operazioni logiche

L'operazione

```
A & B
```

dà come risultato una matrice di dimensioni uguale a quelle di A e B, con 1 dove gli elementi di entrambe sono diversi da 0, altrimenti 0. È possibile utilizzare le operazioni *any* e *all* per confrontare completamente una matrice o un vettore

```
>> a=0:8
a =
```

```
  0   1   2   3   4   5   6   7   8
>> all(a==0)
ans =
     0
>> any(a==0)
ans =
     1
```

Altri operatori sono riportati nella tabella 5.3

exist	controlla se esiste una variabile o funzione
find	cerca una certa condizione vera in un vettore
finite	controlla se un numero è finito
isempty	controlla se una matrice è vuota
isieee	controlla se il computer usa matematica IEEE
isinf	controlla se un numero è infinito
isnan	controlla se un numero non esiste
issparse	controlla se una matrice è sparsa
isstr	controlla se un vettore rappresenta una stringa

Tabella 5.3: Richieste logiche

5.5 Algebra lineare

MATLAB era stato scritto originariamente per semplificare il lavoro degli studenti in algebra lineare. Esistono quindi una grande quantità di funzioni in questo campo della matematica.

Un sistema lineare di equazioni può essere scritto in forma matriciale nel modo seguente

$$A\mathbf{x} = \mathbf{b} \tag{5.1}$$

dove A è una matrice n x n, $\mathbf{x}$ e $\mathbf{b}$ due vettori colonna di n elementi.

La soluzione tramite MATLAB di questa equazione può avvenire in due modi. Il primo metodo utilizza l'inversa della matrice per risolvere il problema, sapendo che

$$\mathbf{x} = A^{-1}\mathbf{b} \tag{5.2}$$

```
>> a=[1,2,3;4,5,6;7,8,0]
a =
     1     2     3
     4     5     6
     7     8     0

>> b=[12;33;36]
b =
```

```
    12
    33
    36

>> x=inv(a)*b
x =
    4.0000
    1.0000
    2.0000
```

Il secondo metodo utilizza l'operazione definita come divisione sinistra

```
>> a=[1,2,3;4,5,6;7,8,0]
a =
   1     2     3
   4     5     6
   7     8     0

>> b=[12;33;36]
b =
    12
    33
    36

>> x=a\b
x =
    4.0000
    1.0000
    2.0000
```

Attenzione: L'operazione $\mathbf{x}/A$ non è possibile !

In MATLAB troviamo tutta una serie di altre operazioni specifiche di algebra lineare, tra cui quelle riportate nella tabella 5.4

d=eig(A)	Calcola gli autovalori della matrice A
[V,D]=eig(A)	Autovalori e autovettori della matrice A
[L.U]=lu(A)	Fattorizzazione LU della matrice A
[Q,R]=qr(A)	Fattorizzazione QR della matrice A
[U,S,V]=svd(A)	Scomposizione in valori singolari della matrice A
r=rank(A)	rango della matrice A
cond(A)	numero condizionale della matrice A
norm(A)	calcola la norma di A. Supporta 1-norm, 2-norm, F-norm e ∞-norm
poly(A)	Trova il polinomio caratteristico associato alla matrice A
polyvalm(v,A)	Calcola il polinomio v della matrice A

Tabella 5.4: Operazioni di algebra lineare

Capitolo 6

Matrici sparse

Le matrici sparse sono una classe particolare di matrici che contengono un numero rilevante di valori nulli. MATLAB permette di memorizzare queste matrici tenendo conto unicamente degli indici occupati, al fine di occupare meno memoria.

```
>> A=[0 0 0 5;0 2 0 0;1 3 0 0;0 0 4 0]
A =
   0     0     0     5
   0     2     0     0
   1     3     0     0
   0     0     4     0
>> S=sparse(A)
S =
   (3,1)        1
   (2,2)        2
   (3,2)        3
   (4,3)        4
   (1,4)        5
>> whos
  Name      Size         Bytes  Class

   A        4x4            128  double array
   S        4x4             80  sparse array
```

È sempre possibile ricostruire la matrice completa con il comando *full*.

Una matrice sparsa può essere anche creata direttamente con il comando *sparse(i,j,s,m,n)*, con i,j vettori di indici occupati, s contenente i valori, e la coppia m,n a indicare le dimensioni complete della matrice.

```
>> S=sparse([3 2 3 4 1],[1 2 2 3 4],[1 2 3 4 5],4,4)
S =
   (3,1)        1
   (2,2)        2
   (3,2)        3
   (4,3)        4
   (1,4)        5
```


Le operazioni sulle matrici sparse possono essere degli stessi tipi di quelle sulle matrici normali, sfruttando però in molti casi algoritmi specifici.

È possibile vedere graficamente l'occupazione di una matrice sparsa mediante il comando spy (figura 6.1).

```
>> spy(S)
```

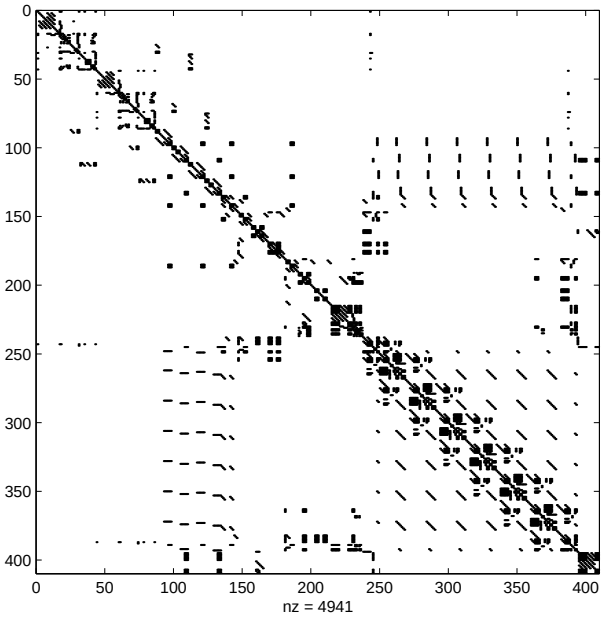

Figura 6.1: Risultato del comando *spy*

Le applicazioni interessanti delle matrici sparse sono quelle legate agli alberi e ai grafi. Un esempio è riportato nella figura 6.2

```
>> [b,v]=bucky; \\
>> gplot(b,v)
```

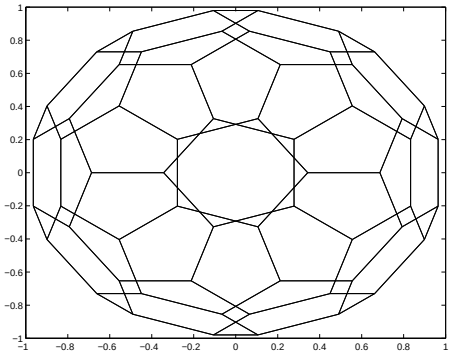

Figura 6.2: Rappresentazione di grafi

Capitolo 7

Semplici script

Tutte le operazioni che vengono date al prompt dei comandi nell'ambiente di MATLAB possono essere integrati in un file testo che può essere richiamato ed eseguito dall'interno dell'ambiente. Questi programmi, chiamati script, devono avere l'estensione *.m*. MATLAB riconosce questi file esattamente come un sistema operativo riconosce i suoi files batch. Con un qualsiasi editor (p. esempio notepad.exe o textedit) posso creare questo file testo, salvarlo con l'estensione *.m* e poi richiamarlo a piacere.

Posso salvare questi comandi come file *esempio.m*

```
a=[1,2 ;3,4]
b=[5 ;7]
c=a*b
```

Posso ora eseguire questo semplice script all'interno di MATLAB

```
>> esempio
a =
     1     2
     3     4
b =
     5
     7
c =
    19
    43
```

Tutte le variabili definite all'interno di uno script restano visibili anche dalla linea di comando. All'interno di uno script posso utilizzare il comando *pause* per far attendere il programma fino a quando non viene premuto un tasto.

Capitolo 8

Programmi e controllo di flusso

8.1 Programmi e funzioni

Abbiamo visto precedentemente come sia possibile costruire degli script contenenti le varie righe di comando da eseguire in MATLAB. Nello stesso modo è possibile costruire delle nuove funzioni che possono poi essere richiamate a piacere. Tutte queste funzioni vanno salvate in file teminanti con l'estensione *.m*, sono files di testo e quindi sono pure portabili da un sistema all'altro. È possibile anche creare funzioni compilate in C, ma per chi è interessato a queste possibilità rimando ai manuali specifici di MATLAB.

Le funzioni vengono caricate e precompilate, e questo codice resta in memoria fino alla fine della sessione o fino ad un comando di *clear*. Questo è il motivo per cui la prima esecuzione di un comando risulta normalmente più lenta, lentezza dovuta alla compilazione del codice.

Dalla versione 5.0 le funzioni possono essere precompilate e salvate in questo formato tramite il comando *pcode*; questo evita il tempo di attesa alla prima esecuzione del comando. Purtroppo, con le funzioni precompilate, si perde la possibilità di accedere all'help interno alla funzione.

Il nome del file dove è memorizzata la funzione è anche il nome della funzione. La dichiarazione interna dovrebbe accordarsi al nome dato, ma non è rilevante. Una funzione viene descritta inizialmente dalla sua definizione nel modo seguente

```
[output1,output2,...]=funzione(input1,input2,...)
```

Nel caso di una sola variabile di output si può evitare di mettere le `[ ]`.

È anche possibile non avere alcun output. Dalla versione 5 è possibile scrivere più funzioni nello stesso M-file. In questo caso però solo la prima funzione può essere chiamata dall'esterno, mentre tutte le altre sono solo accessibili alla funzione principale (funzioni locali).

Vediamo subito un esempio di funzione

```
function y = linspace(d1, d2, n)
%LINSPACE Linearly spaced vector.
%   LINSPACE(x1, x2) generates a row vector of 100 linearly
%   equally spaced points between x1 and x2.
%   LINSPACE(x1, x2, N) generates N points between x1 and x2.
%
```

```
%   See also LOGSPACE, :.

%       Copyright (c) 1984-94 by The MathWorks, Inc.

if nargin == 2
    n = 100;
end
y = [d1+(0:n-2)*(d2-d1)/(n-1) d2];
```

È interessante come la stessa funzione possa essere richiamata passando in entrata 2 o 3 parametri. È possibile distinguere i due casi mediante ilo parametro *nargin* che fornisce il numero di parametri passati in entrata e permette così di distinguere i vari casi. I parametri definiti all'interno di una funzione sono locali, e quindi non vengono passati al em workspace. Se si vogliono utilizzare variabili definite nel em workspace si deve passare tramite la definizione *global*.

I commenti che iniziano alla seconda linea della funzione vengono interpretati da MATLAB come linea di *help* e mostrati quando si richiede sulla linea di comando

```
>> help linspace
```

Molto importante è anche il capire come MATLAB interpreta una linea di comando. Quando su una linea viene dato un comando *nome* l'ordine di esecuzione è il seguente:

- Variabile *nome*
- Funzione interna di MATLAB
- Funzione o script nel path attuale
- Funzione o script nel path definito in matlabrc.m

8.2 Controllo di flusso

Esistono 3 metodi per controllare il flusso di un programma e prendere decisioni all'interno dello stesso:

- for loop
- while loop
- if-elseif-else
- switch-case-otherwise

Tutti questi comandi devono avere alla fine del blocco da eseguire un comando *end*.

Questi comandi possono anche essere utilizzati all'interno del shell di MATLAB e attendono il comando *end* per far partire l'esecuzione. Esiste la possibilità di utilizzare un comando *break* all'interno di questi cicli.

8.2.1 Ciclo *for*

```
for n=1:0.1:5
     y(n)=3*n^2
end
```

8.2.2 Ciclo *while*

```
num=0 ;EPS=1 ;
while (1+EPS)>1)
       EPS=EPS/2 ;
       num=num+1 ;
end
```

8.2.3 Ciclo *if-elseif-else*

```
if a>5
      b=7
elseif a>4
      b=2.7
elseif a>2
      b=1
else
      b=0
end
```

8.2.4 Ciclo *switch-case*

```
switch a
    case {1,2,3}
         y=2*a;
    case {4,5,6}
         y=3*a
    case {7,8,9}
         y=4*a;
    otherwise
         y=a;
end
```


Capitolo 9

Analisi di dati

9.1 Introduzione

MATLAB permette di analizzare statisticamente matrici di dati colonna per colonna
Le operazioni possibili sono riportate nella tabella 9.1

max	valore massimo
min	valore minimo
mean	valore medio
median	valore mediano
std	deviazione standard
sort	sorting
sum	somma elementi
prod	prodotto elementi
cumsum	somma cumulativa
cumprod	prodotto cumulativo
diff	derivata approssimativa
hist	istogramma
corrcoef	coefficienti di correlazione
cov	matrice di covarianza

Tabella 9.1: Operazioni per l'analisi dei dati

9.1.0.1 Esempi

```
>> a =
        21    10
         2     3
        15     4
         3    60
         7    13
>> max(a)
ans =
        21    60
>> mean(a)
```

```
ans =
    9.6000   18.0000
>> median(a)
ans =
     7    10
>> prod(a)
ans =
       13230       93600
>> cumprod(a)
ans =
          21          10
          42          30
         630         120
        1890        7200
       13230       93600
```

Capitolo 10

Polinomi

Un polinomio in MATLAB viene rappresentato mediante un vettore contenente i suoi coefficienti. Per esempio, il polinomio

$$x^3 + 2x^2 + 7x + 5 \tag{10.1}$$

viene rappresentato mediante il vettore p1

```
>> p1=[1 2 7 5]
p1 =
    1     2     7     5
```

Sui polinomi possono essere fatte una grande quantità di operazioni

```
>> p2=poly([0 1])  % polinomio creato mediante i poli
p2 =
     1    -1     0
>> p3=[1 3 2]
p3 =
     1     3     2
>> roots(p3)       % radici di un polinomio
ans =
     -2
     -1
```

Un polinomio può essere derivato

```
>> p1
p1 =
     1     2     7     5
>> polyder(p1)
ans =
     3     4     7
```

La moltiplicazione tra 2 polinomi è un'operazione di convoluzione

```
>> conv(p1,p2)
ans =
  1     1     5    -2    -5     0
```

Esiste anche la possibilità di dividere tra di loro due polinomi utilizzando la funzione *deconv*.

Molto interessanti sono le operazioni messe a disposizione per determinare il polinomio che più approssima una serie di punti

```
>> x=0:0.1:1;
>> y=[-.447,1.978,3.28,6.16,7.08,7.34,7.66,9.56,9.48,9.3,11.2];
>> polyfit(x,y,2)
ans =
   -9.8108   20.1293   -0.0317
```

In quest'ultimo esempio è stato determinato il polinomio di 2. ordine che meglio approssima i punti y. Vediamo un altro esempio di operazioni di questo tipo.

```
>> p=polyfit(x,y,2)
p =
   -9.8108   20.1293   -0.0317

>> x1=0:0.01:1;
>> y1=polyval(p,x1);
>> plot(x,y,'o',x1,y1)
```

Nel grafico 10.1 si può giudicare visivamente il risultato

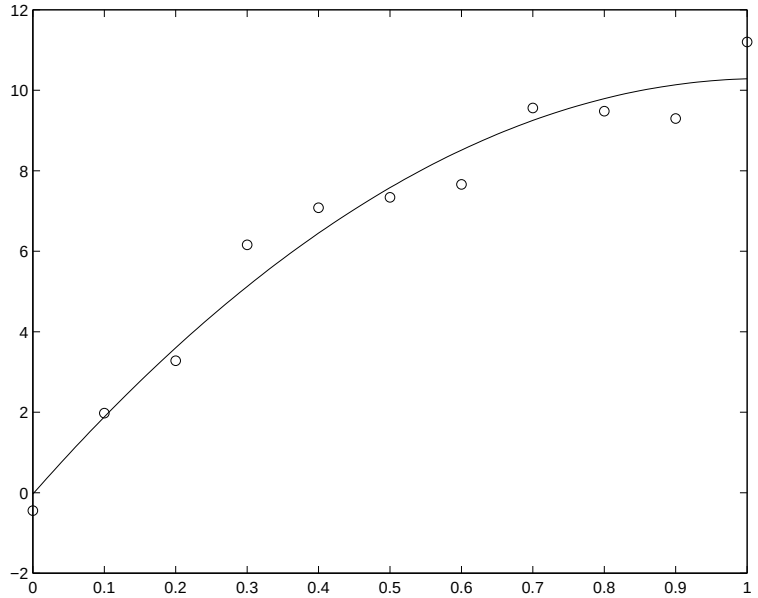

Figura 10.1: Approssimazione di punti con un polinomio

Il comando *polyval* determina il risultato dell'applicazione del polinomio a un vettore di punti *x1*.

Un altro comando interessante è il comando *residue* che effettua una scomposizione in termini di frazioni parziali della divisione di due polinomi

```
>> num=[1 1]
num =
  1     1

>> den=[1 10 31 30 0]
den =
  1    10    31    30     0

>> printsys(num,den)

num/den =

            s + 1
  ---------------------------
  s^4 + 10 s^3 + 31 s^2 + 30 s

>> [r,p,k]=residue(num,den)
r =
    0.1333
   -0.3333
    0.1667
    0.0333

p =
   -5.0000
   -3.0000
   -2.0000
         0

k =
     []
```

Il risultato trovato deve essere visto nella forma

$$\frac{0.1333}{s+5} - \frac{0.3333}{s+3} + \frac{0.1667}{s+2} + \frac{0.0333}{s} \tag{10.2}$$

Capitolo 11

Interpolazione

MATLAB mette a disposizione una serie di comandi per effettuare delle interpolazioni su dati tabellati, visibili nella tabella 11.1.

interp	interpolazione su funzioni a 1 variabile
interp2	interpolazione su funzioni a 2 variabili
interp3	interpolazione su funzioni a 3 variabili
interpn	interpolazione su funzioni a n variabili
interpft	interpolazione con funzioni a 1 variabile con metodo FFT
spline	interpolazione con metodo spline
griddata	interpolazione su funzioni a 2 variabili

Tabella 11.1: Funzioni di interpolazione

Esistono all'interno dei comandi possibilità di scelta del metodo di interpolazione da utilizzare.

```
>> orario=1:12;
>> temperature=[5 8 9 15 25 29 31 30 22 25 27 24]
temperature =
  5   8   9  15  25  29  31  30  22  25  27  24

>> interp1(orario,temperature,10.25)
ans =
   25.5000

>> interp1(orario,temperature,10.25,'cubic')
ans =
   25.6875

>> interp1(orario,temperature,10.25,'spline')
ans =
   26.0083
```

La figura 11.1 mostra il risultato di un'interpolazione mediante *spline*

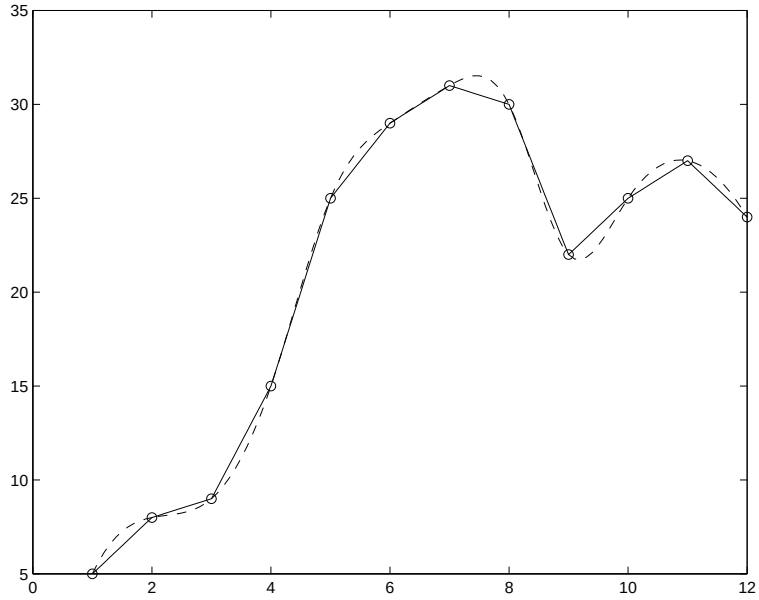

Figura 11.1: Interpolazione spline

I comandi che hanno portato a questo risultato sono

```
>> h=1:0.1:12;
>> t1=interp1(orario,temperature,h);
>> t2=interp1(orario,temperature,h,'spline');
>> plot(orario,temperature,'o',h,t1,'-',h,t2,'--')
```

Capitolo 12

Analisi numerica

12.1 Premessa

Definiamo la funzione seguente sinxsux.m

```
function y=sinxsux(x)
if x==0
        y=1;
else
        y=sin(x)./x;
end
```

Vediamo ora cosa abbiamo a disposizione in MATLAB per analizzare questa funzione.

12.2 Plotting

Possiamo ottenere velocemente il grafico di questa funzione con

```
>> fplot('sinxsux',[-10,10])
>> grid
```

Il risultato è visibile nella figura 12.1

12.3 Ricerca di minimi

Cerchiamo ad esempio il minimo della funzione

$$2e^{-x}\sin(x) \tag{12.1}$$

Possiamo farlo anche scrivendo la funzione in forma testo

```
>> fn='2*exp(-x)*sin(x)'
fn =
 2*exp(-x)*sin(x)
>> fmin(fn,2,5)
ans =
    3.9270
```

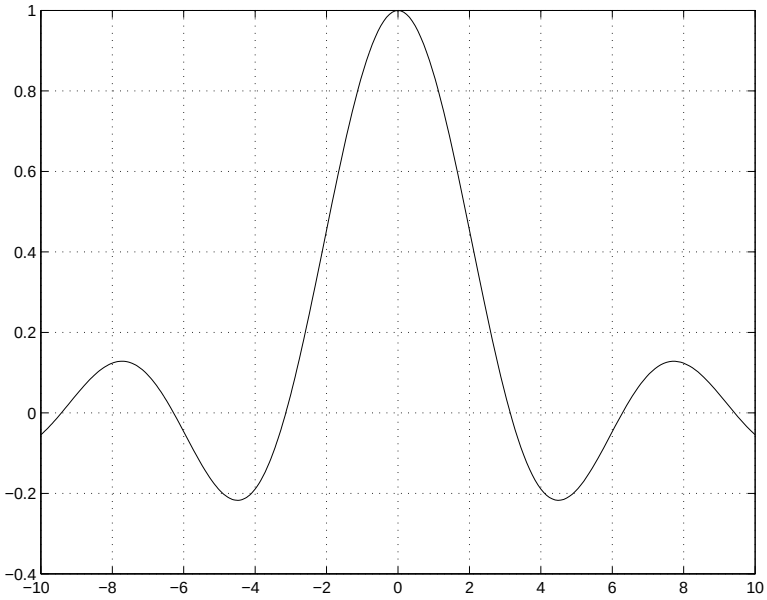

Figura 12.1: Risultato di fplot

12.4 Ricerca di zeri

Per la funzione *sinxsux* vista precedentemente vogliamo cercare i punti in cui il grafico taglia l'asse $x = 0$. Occorre innanzitutto determinare degli intervalli, leggibili dal grafico, e quindi effettuare la ricerca in questi intervalli. Come si può ad esempio vedere nel grafico, tra il valore 2 e il valore 4 c'è uno zero della funzione. La ricerca viene fatta con la funzione *fzero*.

```
>> fzero('sinxsux',2,4)
ans =
    3.6000
```

12.5 Integrazione e derivazione

Esistono diverse funzioni per effettuare l'integrazione di una funzione. Alcune di esse sono riportate nella tabella 12.1.

trapz	Integrazione con approssimazione trapezoidale
quad, quad8	Integrazione con quadratura

Tabella 12.1: Funzioni di integrazione e derivazione

Per derivare una funzione occorre utilizzare metodi più complessi, utilizzando la funzione *diff*, oppure cercando un'approssimazione polinomiale e utilizzando il comando *polyder*.

Capitolo 13

Grafica

13.1 Introduzione

In MATLAB è possibile utilizzare comandi per grafica 2D e 3D. Esistono inoltre comandi per creare delle animazioni.

13.2 Grafica 2D

Le funzioni principali per gestire grafici 2D sono riportate nella tabella 13.1

plot	Crea un grafico da vettori di valori
loglog	Grafico con assi logaritmici
semilogx	Grafico con asse x logaritmico e y lineare
semilogy	Grafico con asse x lineare e y logaritmico
title	Aggiunge un titolo al grafico
xlabel	Label asse x
ylabel	Label asse y
text	Mostra un testo ad una posizione generica
gtext	Mostra un testo ad una posizione acquisita con il mouse
grid	Mostra (toglie) la griglia

Tabella 13.1: Funzioni per grafica 2D

La sequenza di comandi seguente crea un grafico con due funzioni sin(x) e $2\sin(x) + \cos(x)$

```
>> x=linspace(0,2*pi);
>> y1=sin(x);
>> y2=2*sin(x)+cos(x);
>> plot(x,y1,'-',x,y2,':')
>> grid
>> xlabel('Asse x')
>> ylabel('Asse y')
>> title('Grafico delle 2 funzioni')
>> text(x(5),y1(5),'sin(x)')
```

```
>> text(x(30),y2(30),'2sin(x)+cos(x)')
```

Il grafico ottenuto con i comandi precedenti è mostrato nella figura 13.1

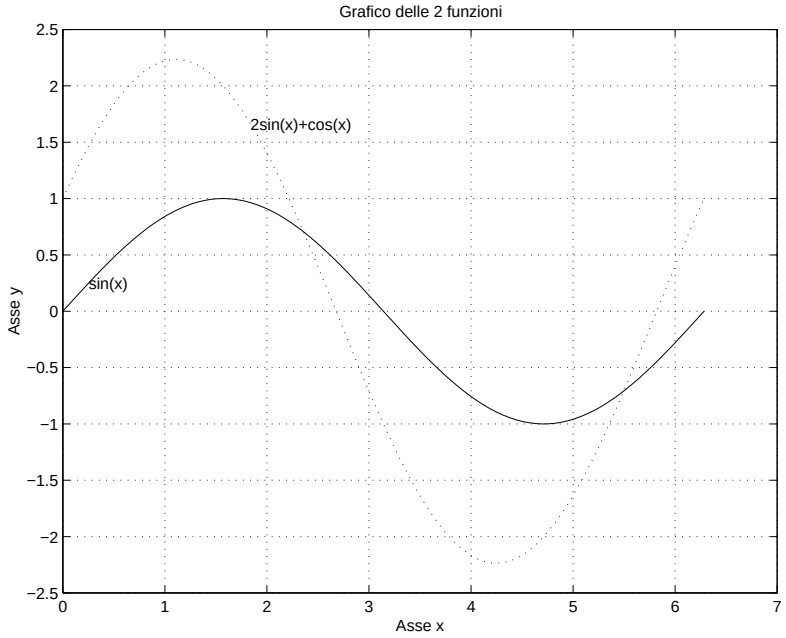

Figura 13.1: Grafico di $2\sin(x) + \cos(x)$

Si può aggiungere una *legenda* mediante il comando *legend*. Il comando *plot* permette una grande variazione di possibilità. Il formato resta però

```
plot(X1,Y1,S1,X2,Y2,S2,...)
```

dove X è il vettore delle ordinate, Y il vettore (o anche la matrice !) delle ascisse, S un formato simbolico con il significato descritto nella tabella 13.2.

Simbolo	Colore	Simbolo	Stile linea
y	giallo	.	punto
m	magenta	o	cerchio
c	ciano	x	croce
r	rosso	+	segno +
g	verde	*	stella
b	blu	-	linea continua
w	bianco	:	linea punteggiata
k	nero	-.	Linea-punto
		–	linea tratteggiata

Tabella 13.2: Formattazione di grafici

Nel prossimo script tutti i comandi di *plot* creano lo stesso grafico.

```
>> plot(x,y1,x,y2);
>> plot(x,[y1;y2])
```

MATLAB riscrive sempre lo stesso grafico, sostituendo quello vecchio con quello nuovo.

È possibile mettere due grafici sulla stessa finestra con due comandi distinti utilizzando il comando *hold on*. I prossimi comandi costruiscono ancora lo stesso grafico dell'esempio precedente

```
>> plot(x,y1)
>> hold on
>> plot(x,y2)
>> hold off
```

Per creare un nuovo grafico senza perdere quello precedente è possibile farlo disegnare in una nuova finestra creata mediante il comando *figure*.

Altri comandi utilizzabili per ottenere grafici 2D sono riportati nella tabella 13.3. Il

axis	possibilità di settare parametri degli assi
bar	grafico a barre
colorbar	grafico a barre
compass	grafico vettori di numeri complessi rappresentati da vettori partenti dall'origine
errorbar	grafico *plot* con barre d'errore
feather	grafico di vettori di numeri complessi partenti da punti dell'asse x a distanza regolare
hist	istogramma
polar	grafico di coordinate polari angolo su raggio
quiver	grafico di gradienti o campi vettoriali
rose	istogramma ad angoli
stairs	grafico a barre non piene
fill	riempie un plot con un colore

Tabella 13.3: Comandi per grafica 2D

comando *zoom on* permette di *cliccare* con il mouse (sinistro) all'interno di un grafico oppure di selezionare una zona, per ingrandire determinati particolari, mentre l'utilizzo del bottone destro riporta gradatamente l'immagine alle dimensioni normali.

13.3 Il comando *subplot*

È possibile portare più grafici sulla stessa finestra, suddividendola in zone con il comando *subplot*.

```
x1=-2.9:0.2:2.9;
y1=randn(5000,1);
subplot(2,2,1);
hist(y1,x1);
title('Histogram')
%
```



```
x2=0:0.1:2;
y2=erf(x2);
e2=rand(size(x2))/10;
subplot(2,2,2);
errorbar(x2,y2,e2);
title('Errorbar')
%
x3=logspace(-2,2);
y3=1+j*x3;
y3=-20*log10(abs(y3));
subplot(2,1,2);
semilogx(x3,y3),grid
title('Semilogx')
```

Il risultato è mostrato nella figura 13.2

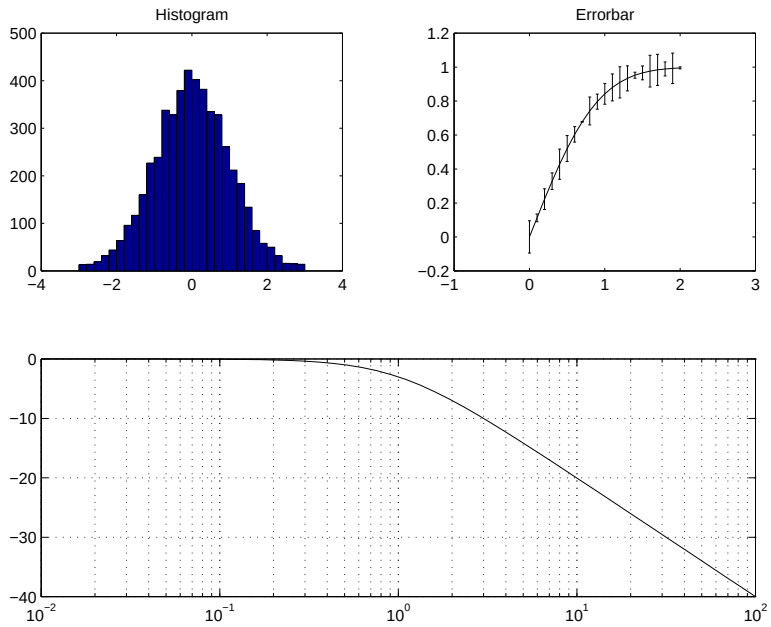

Figura 13.2: Risultato del comando *subplot*

13.4 Trasformazione di coordinate

MATLAB contiene tutta una serie di funzioni che sono in grado di trasformare coordinate da un sistema ad un altro, sia in 2D che in 3D. Le operazioni a disposizione sono riportate nella tabella 13.4

13.5 Grafica 3D

I comandi messi a disposizione da MATLAB per l'elaborazione di immagini e grafici 3D sono riportati nella tabella 13.5

cart2pol	coordinate cartesiane → coordinate polari
cart2sph	coordinate cartesiane → coordinate sferiche
pol2cart	coordinate polari → coordinate cartesiane
sph2cart	coordinate sferiche → coordinate cartesiane

Tabella 13.4: Trasformazione di coordinate

plot3	grafico a linea 3D
contour,contour3	curve di livello
pcolor	grafico 2D in cui il colore rappresenta la 3. Dimensione
image	matrice come immagine
mesh, meshc, meshz	3D prospettici
surf, surfc, surfl	Grafici di superficie, eventualmente con illuminazione
slice	grafico vlolumetrico *a fette*
fill3	Poligono 3D con riempimento
zlabel	Label asse z
clabel	Label curve di livello

Tabella 13.5: Comandi per grafica 3D

Inoltre ci sono altri comandi per modificare le viste di un grafico 3D (tabella 13.6.

Alcuni esempi di grafica 3D sono riportati nelle figure 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9 e 13.10. Da notare anche l'uso della funzione *peaks* che genera un particolare grafico 3D, utilizzato negli esempi.

Il comando *meshgrid*, genera, partendo da due vettori x e y, 2 matrici X e Y in cui ci sono ripetuti tutti i valori di x (riga dopo riga) e y (colonna dopo colonna), in modo da creare una griglia nei cui nodi si ritrovano tutte le possibili combinazioni (x,y).

Il comando *surfl* inserisce una fonte di luce per illuminare la superficie.

Il comando *view* permette di modificare gli angoli di vista di una figura 3D. Questi angoli possono essere modificati anche interattivamente tramite il comando *rotate3d*.

Un'ulteriore manipolazione dei colori dei grafici è possibile con i comandi *colormap* e *shading*.

view	modifica il punto di vista
viewmtx	calcola la matrice di trasformazione per il nuovo punto di vista
rotate3d	rotazione interattiva di una figura con il mouse

Tabella 13.6: Comandi per la vista 3D


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

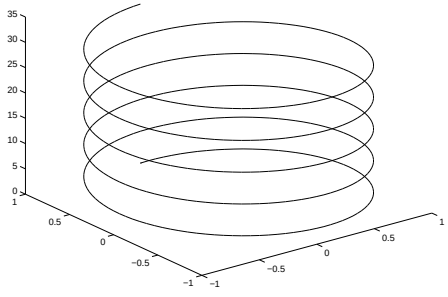

Figura 13.3: Comando *plot3d*

```
>> contour(peaks,20)
```

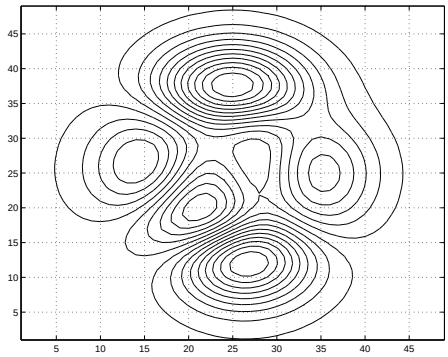

Figura 13.4: Comando *contour*

```
>>contour3(peaks,20)
```

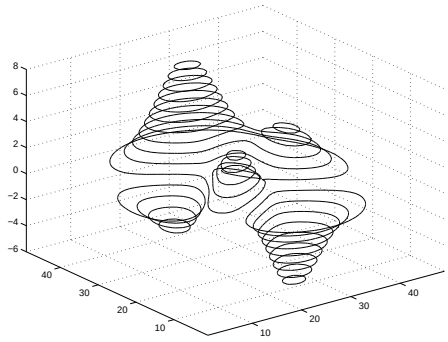

Figura 13.5: Comando *contour3*

```
>> x=-8:.5:8;
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> R=sqrt(X.^2+Y.^2)+eps;
>> Z=sin(R)./R;
>> mesh(X,Y,Z)
```

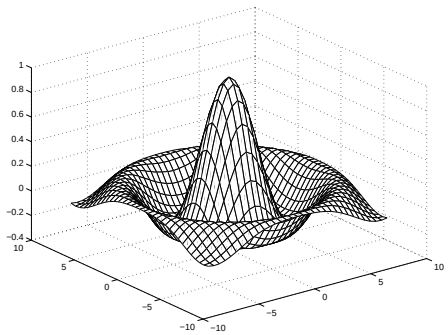

Figura 13.6: Comando *mesh*

```
>> x=-8:.5:8;
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> R=sqrt(X.^2+Y.^2)+eps;
>> Z=sin(R)./R;
>> meshz(X,Y,Z)
```

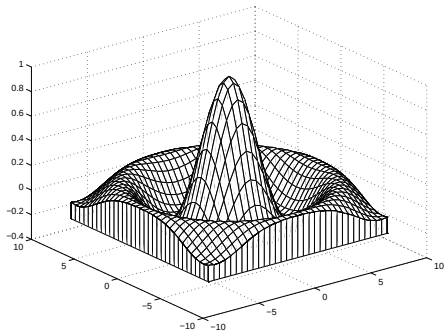

Figura 13.7: Comando *meshz*

```
>> surfl(peaks)
```

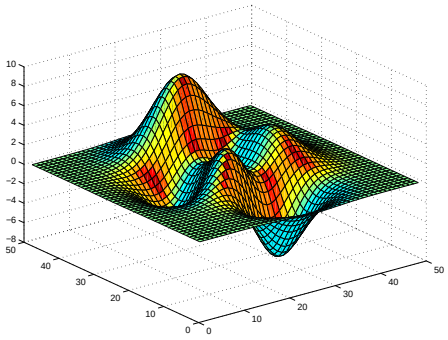

Figura 13.8: Comando *surfl*

```
>> surfl(peaks)
>> view(40,-10)
```

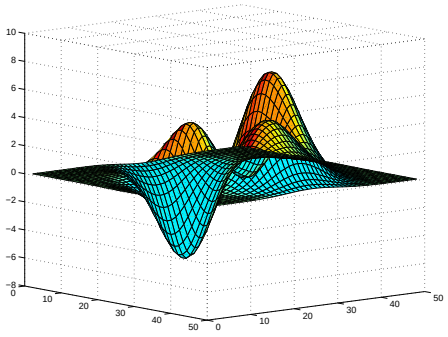

Figura 13.9: Comando *view*


```
>> pcolor(peaks)
```

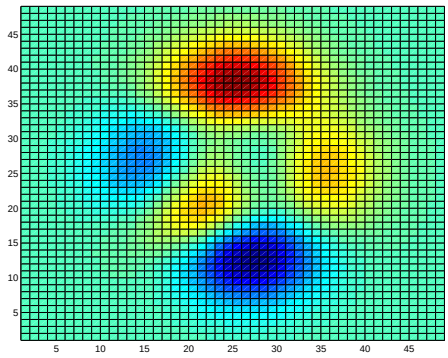

Figura 13.10: Comando *pcolor*

13.6 Stampa di immagini

MATLAB può stampare in 3 modi differenti

- PostScript Driver di MATLAB: ps, psc, ps2, psc2,eps, epsc, eps2, epsc2
- Printer drivers di Windows ; è possibile anche il salvataggio nel clipboard come Bitmap o Metafile.
- GhostScript printer drivers, forniti con MATLAB:
  - HP LaserJet, HP Laserjet+, HP DeskJet 500C, HP PaintJet, GIF ecc.

Per i dettagli si consiglia di chiamare il comando

```
>> help print
```

La modifica del formato di stampa non è semplicissima, poiché deve essere fatta mediante i comandi *get* (vedi capitolo dedicato alle grafica avanzata).

13.7 Animazioni

In MATLAB esiste la possibilità di animare grafici in diversi modi. Quello più semplice, permette di creare un puntino luminoso mobile che disegna gradatamente un grafico di tipo parametrico. Il comando *comet* lavora in 2D, mentre il corrispondente comando per l'ambiente 3D è *comet3*. Un esempio che dimostra ottimamente questi comandi è quello messo a disposizione da MATLAB nella demo sotto MATLAB - *Visualization* - *Animation*, con l'animazione di una carica in un campo elettrico.

Si possono creare anche animazioni registrando le immagini come singoli fotogrammi di un film, mediante il comando *moviein*, *getframe* e *movie*. Un programma che si può scrivere ed eseguire è il seguente

```
z=peaks ;
surf(z) ;
colormap(gray)
shading interp
```

```
lim=axis ;
M=moviein(20) ;    % prepara 20 fotogrammi
for n=1:20        % registra le immagini
    surf(sin(2*pi*n/20)*z,z)
    shading interp
    axis(lim)
    M(:,n)=getframe ;
end
movie(M,20)        % esegui l'animazione
```


Capitolo 14

Equazioni differenziali

La versione di base di MATLAB contiene diverse funzioni per la risoluzione di equazioni differenziali (tabella 14.1

ode23	equazioni differenziali di ordine basso
ode45	equazioni differenziali di ordine medio
ode113	equazioni differenziali di ordine variabile
ode15s	equazioni differenziali *rigide* di ordine variabale
ode23s	equazioni differenziali *rigide* di ordine basso
odeset	crea/modifica i parametri di integrazione
odeget	richiama i parametri di integrazione
odeplot	plot nel tempo
odephas2	diagramma degli stati (fasi) bidimensionale
odephas3	diagramma delle fasi (stati) tridimensionale

Tabella 14.1: Comandi per integrazione numerica

Per risolvere un'equazione differenziale o un sistema di equazioni differenziali occorre innanzitutto ridurre il problema ad un sistema di equazioni differenziali di 1. Ordine, mediante una sostituzione di variabile. Per esempio, l'equazione differenziale di Van der Pol

$$\ddot{x} + \left(x^2 - 1\right)\dot{x} + x = 0 \tag{14.1}$$

viene ridotta con la sostituzione

$$x_1 = x \tag{14.2}$$
$$x_2 = \dot{x}_1 = \dot{x} \tag{14.3}$$

nel sistema

$$\dot{x}_1 = x_2 \tag{14.4}$$
$$\dot{x}_2 = x_2\left(1 - x_1^2\right) - x_1 \tag{14.5}$$

e quindi risolto.

La soluzione di un'equazione o di un sistema di equazioni differenziali richiede la programmazione di una funzione dichiarata come

```
function xdot=<nome funzione>(t,x)
```

dove *xdot* e x sono vettori contenenti le variabili di x_1, x_2 e le loro derivate.

Nel caso della funzione di Van der Pol il programma è

```
function xdot=vdpol(t,x)
xdot=zeros(2,1);
xdot(1)=x(2) ;
xdot(2)=x(2)*(1-x(1).*x(1))-x(1) ;
```

La soluzione di quest'equazione si trova con il comando

```
[t,x]=ode23('vdpol',[time],x0)
```

dove *time* contiene le informazioni relative ai tempi di integrazione e $x0$ è un vettore colonna con le condizioni iniziali x_{10} e x_{20}.

```
>> [t,x]=ode23('vdpol',[0,20],[0;4]);
>> plot(t,x(:,1),t,x(:,2))
```

Il risultato della simulazione è riportato nella figura 14.1

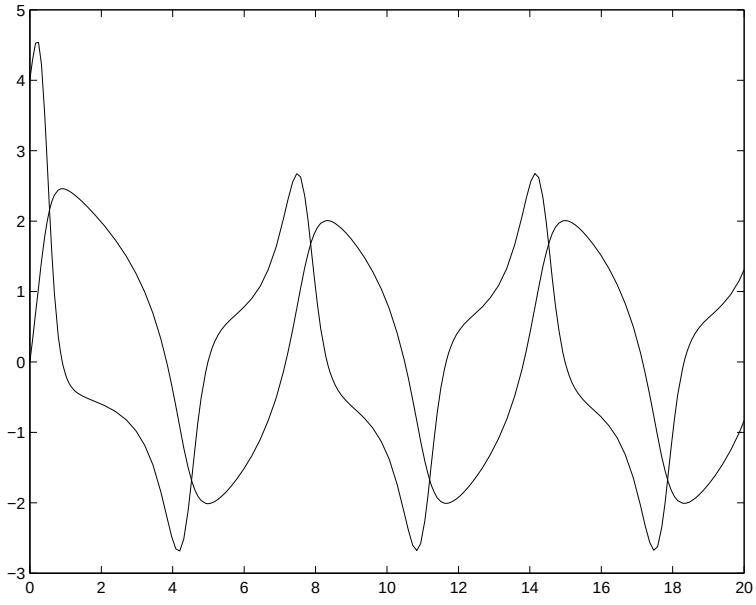

Figura 14.1: Simulazione dell'equazione di Van der Pol

Interessante è la possibilità di vedere il grafico delle due variabili di stato, una in funzione dell'altra, per vedere meglio il ciclo che si forma con diverse condizioni iniziali, con i comandi

```
>> [t,x]=ode23('vdpol',[0,20],[0;4]);
>> plot(x(:,1),x(:,2))
>> hold on
>> [t,x]=ode23('vdpol',[0,20],[0;1]);
>> plot(x(:,1),x(:,2))
>> [t,x]=ode23('vdpol',[0,20],[-1;-2]);
>> plot(x(:,1),x(:,2))
>> hold off
```

Il risultato è mostrato nella figura 14.2.

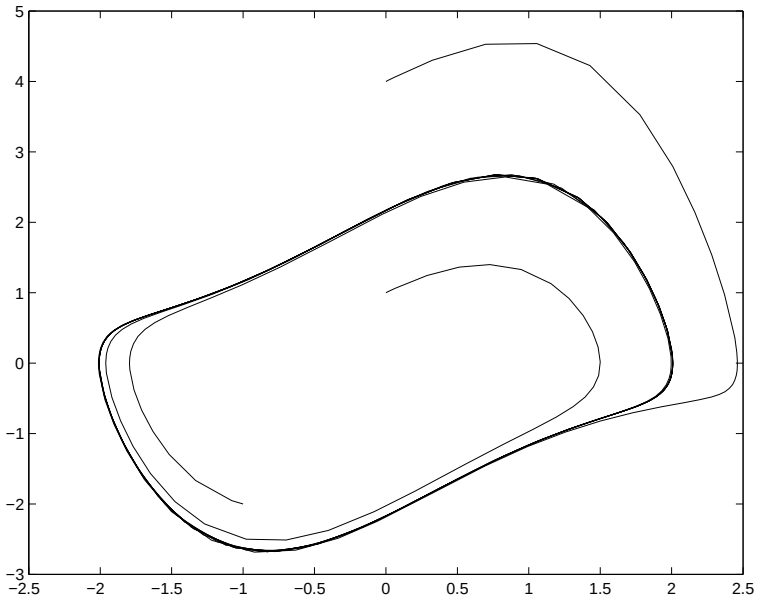

Figura 14.2: Diagramma delle fasi dell'equazione di Van der Pol

Per chi lavora molto a livello di equazioni differenziali è consigliabile l'uso di uno strumento quale SIMULINK che rispetto a MATLAB presenta soluzioni più complete.

Capitolo 15

Input/Output

Spesso si è confrontati con dati raccolti con altri programmi che si vorrebbero analizzare mediante MATLAB. Il metodo più semplice consiste nell'avere questi dati in un file in formato ASCII, come nel prossimo esempio

```
1        2            -5
2        0.2500       -9
3        0.0740       -23
4        0.0310       -53
5        0.0160       -105
6        0.0090       -185
7        0.0050       -299
8        0.0030       -453
9        0.0020       -653
10        0.0020       -905
```

Questi dati possono essere salvati su un file (per esempio *operaz.dat*) e caricati in MATLAB con il comando *load*. In questo esempio, in MATLAB viene creata una matrice *operaz* di grandezza $10x3$.

```
>> load operaz.dat
>> whos
  Name    Size    Elements  Bytes   Density   Complex

 operaz  10 by 3     30      240      Full      No

Grand total is 30 elements using 240 bytes
```

Il salvataggio di dati è possibile specificando un formato con il comando *save*. La sintassi esatta è

```
>> save operaz.dat        % salva in forma binaria
>> save operaz.dat x,y    % salva solo le variabili x e y
>> save operaz.dat -ascii  % salva in forma ascii 8 bit
>> save operaz.dat -double % salva in forma ascii 16 bit
>> save operaz.dat -ascii -tabs  % salva in forma ascii
                                 %con tabulatori
```


Se il nome del file è *stdio* l'output è fatto sullo standard output device, generalmente lo schermo.

I comandi *dlmread* e *dlmwrite* lavorano su files ASCII con elementi delimitati da un separatore.

```
>> A=dlmread('dati.dat',';')   % legge un file
                               %delimitato da ';'
```

Il secondo metodo utilizza tutta una serie di comandi derivati e simili a quelli del linguaggio *C* per effettuare le operazioni di I/O. Questi comandi sono riportati nella tabella 15.1.

fopen,fclose	apertura e chiusura di un file
fread,fwrite	I/O per dati non formattati
fscanf,fprintf	I/O per dati formattati (formattazione con i comandi simili in C
fgetl,fgets	I/O dati formattati
ferror,fseek,ftell, frewind	Come i corrispondenti comandi in C
ssprintf,sscanf	Conversione di stringhe

Tabella 15.1: Comandi di IO

Altri metodi di I/O sono descritti nell'allegato A.

Il primo esempio è un programma che legge un file di testo

```
function str=leggi(nomefile)
       fid=fopen(nomefile,'r') ;
       str=setstr((fread(fid))') ;
       fclose(fid) ;
```

Il prossimo script genera un file binario con 1000 elementi, legge poi un elemento ad una certa posizione.

```
a=rand(1,1000) ;
fid=fopen('dati.bin','w') ;
count=fwrite(fid,a,'float64') ;
fclose(fid) ;
%
posiz=121 ;
fid=fopen('dati.bin','r') ;
status=fseek(fid,(n-1)*8,'bof') ;
ele_1=fread(fid,1,'float64') ;
fclose(fid)
```

Il prossimo esempio scrive su un file testo i valori di una funzione

```
t=0:pi/20:2*pi ;
y=2*sin(t).*cos(t) ;
```

```
funz=[t ;y] ;
fid=fopen('funzione.txt','w') ;
fprintf(fid,'Valori della funzione sin(t)*cos(t) tra 0 e 2*pi\n\n') ;
fprintf(fid,'%4.2f    %10.6f\n',funz) ;
fclose(fid)
```


Capitolo 16

Debugger

Ad eccezione delle versioni per PC e MAC, MATLAB contiene un sistema di debugging utilizzabile solo per le funzioni, e non con script di comandi. Nella versione PC è possibile effettuare il debugger direttamente nell'editor integrato, utilizzando breakpoints e visualizzando variabili. Con Unix, questo non è possibile, e occorre in ogni caso passare dalle procedure descritte in questo capitolo.

Per utilizzare il debugger conviene inizialmente listare i comandi del file con l'indicazione del numero di riga, per poter in seguito inserire dei breakpoints (comando *dbtype*). Quando si raggiunge un punto di break, il sistema si ferma e passa automaticamente in modo *debug*.

Attenzione. Se si edita un programma o si utilizza il comando *clear* vengono persi i breakpoint precedentemente inseriti.

La lista dei comandi di debugger è riportata nella tabella 16.1.

dbtype	mostra un file M con il numero di riga
dbstop	Inserisce un breakpoint all'inizio di una funzione ad una certa riga se c'è un errore se si ottiene inf o NaN
dbclear	elimina i breakpoint
dbstatus	lista dei breakpoint
dbstep	esecuzione passo a passo
dbcont	continuare l'esecuzione
dbup	risalire di un livello di scope per visualizzare variabili locali
dbdown	ritornare al livello più basso
dbstack	visualizzare i comandi collegati al breakpoint attuale
dbquit	abbandonare il modo debug

Tabella 16.1: Comandi del debugger

Esempi di comandi di debugger possono essere

```
>> dbtype linspace

1    function y = linspace(d1, d2, n)
```

```
2    %LINSPACE Linearly spaced vector.
3    %   LINSPACE(x1, x2) generates a row vector of 100 linearly
4    %   equally spaced points between x1 and x2.
5    %   LINSPACE(x1, x2, N) generates N points between x1 and x2.
6    %
7    %   See also LOGSPACE, :.
8
9    %   Copyright (c) 1984-94 by The MathWorks, Inc.
10
11   if nargin == 2
12       n = 100;
13   end
14   y = [d1+(0:n-2)*(d2-d1)/(n-1) d2];

>> dbstop in linspace
>> dbstop at 14 in linspace
>> dbstop if error
```

Vediamo ora una sequenza di debugging completa, dopo aver definito due funzioni *test.m* e *test1.m*.

```
>> dbtype test

1    function a=test(b)
2    c=sqrt(b)*cos(b);
3    a=test1(b,c);
>> dbtype test1

1    function a=test1(b,c)
2    q=cond(b);
3    [w,e]=eig(c);
4    a=w*q;
>> dbtype cond

1    function y = cond(x)
2    %COND        Matrix condition number.
3    %   COND(X) is the ratio of the largest singular value of X
4    %   to the smallest, which is the condition number of X in 2-norm.
5    %
6    %   See also RCOND, NORM, CONDEST, NORMEST.
7
8    %   J.N. Little 11-15-85
9    %   Revised 3-9-87 JNL, 2-11-92 LS.
10   %   Copyright (c) 1984-94 by The MathWorks, Inc.
11
12   if length(x) == 0  % Handle null matrix
13       y = NaN;
14       return
```

```
15   end
16   if issparse(x)
17           error('Matrix must be non-sparse.')
18   end
19   s = svd(x);
20   if any(s == 0)   % Handle singular matrix
21       disp('Condition is infinite')
22       y = Inf;
23       return
24   end
25   y = max(s)./min(s);

>> dbstop in test         % breakpoint 1
>> dbstop at 19 in cond   % breakpoint 2

>> test(magic(3))
2   c=sqrt(b)*cos(b);
K>> dbstack               % K>> indica modo debugger
In d:\matlab\toolbox\bucher\corso\test.m at line 2
K>> dbcont
19  s = svd(x);
K>> dbstack
In d:\matlab\toolbox\matlab\matfun\cond.m at line 19
In d:\matlab\toolbox\bucher\corso\test1.m at line 2
In d:\matlab\toolbox\bucher\corso\test.m at line 3
K>> who

Your variables are:

x         y

K>> x            % posso visualizzare ed ev.
                 % modificare il cont di una variabile
x =
     8     1     6
     3     5     7
     4     9     2
K>> dbstep
20  if any(s == 0)   % Handle singular matrix
K>> s
s =
   15.0000
    6.9282
    3.4641
K>> dbup
In workspace belonging to d:\matlab\toolbox\bucher\corso\test1.m.
K>> who
```



```
Your variables are:

a         b         c

K>> dbstep
25  y = max(s)./min(s);
K>> who

Your variables are:

s         x         y

K>> y
y =
     []
K>> dbcont
ans =
   2.0428     -1.9138 - 0.9902i  -1.9138 + 0.9902i
   3.6832      0.5722 - 0.5802i   0.5722 + 0.5802i
   1.0056     -2.9190 - 2.2187i  -2.9190 + 2.2187i
```

Capitolo 17

Grafica avanzata (GUI)

È possibile agire sui grafici creati da MATLAB modificando direttamente le proprietà degli oggetti disegnati, come i colori delle curve, i fonts, lo sfondo ecc. Nelle nuove release di MATLAB queste modifiche possono essere fatte direttamente tramite menu e bottoni sulla finestra del grafico, ma spesso può essere necessario farle in modo interattivo con comandi in linea nella shell o in un programma.

È importante sapere come sono ordinati gerarchicamente all'interno del sistema i diversi oggetti. In alto troviamo la *root* che ha identificatore o handle 0. Qindi si trovano i diversi oggetti, secondo la gerarchia visibile nella figura 17.1.

Gli oggetti di MATLAB sono principalmente 11, e sono riportati nella tabella 17.1.

root	finestra di comando di MATLAB
figure	figura grafica
axes	assi di oggetti complessi
uicontrol	bottoni, box di editor, popup, slide
uimenu	menu a tendina
line	linea di base di un plot
patch	poligono chiuso colorato
surface	superficie rttangolare (nxm poligoni). Può essere un cono, una sfera, un cilindro o qualsasi altra figura nello spazio
image	oggetto bidimensionale (foto, ecc.)
text	testo
light	luci

Tabella 17.1: Oggetti della GUI

È possibile modificare i settaggi di default di ogni oggetto utilizzando il comando *set* e specificando l'oggetto e la proprietà da modificare nel modo seguente

```
set(0,DefaultObjectProperty,Valore)
```

Ad esempio, se volessimo cambiare il font di default per i testi sugli assi di una figura possiamo dare

```
>> set(0,'DefaultAxesFontName','Times Roman') ;
```


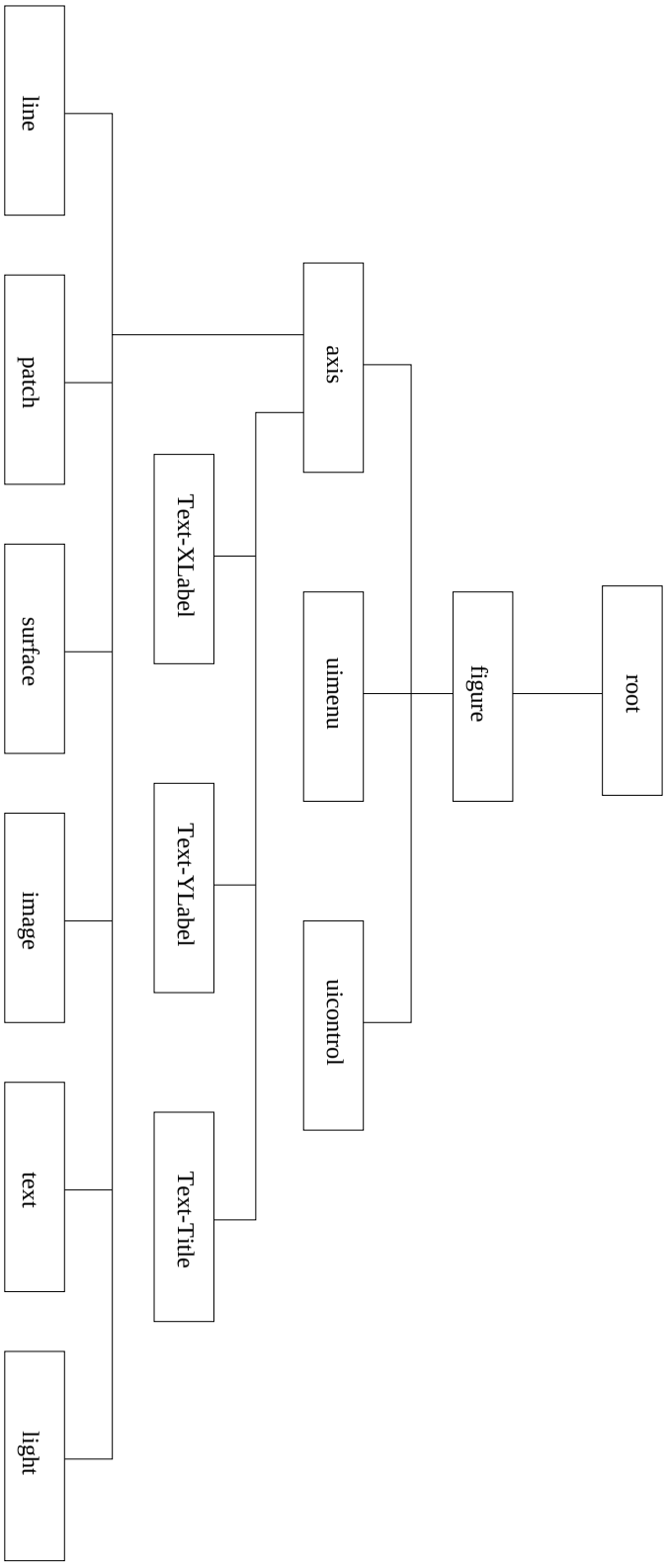

Figura 17.1: Gerarchia degli oggetti della GUI

Questi comandi possono essere inseriti nel file di partenza di MATLAB (matlabrc.m) o nel file *startup.m*, in modo da renderli attivi ad ogni nuova sessione di lavoro.

Vediamo ora come è possibile modificare alcune proprietà di oggetti grafici, partendo da un esempio semplice. Creiamo per prima cosa una figura con una scritta:

```
>> figure
>> text(0.5,0.5,'Ciao a tutti')
```

La figura ottenuta è visibile nella figura 17.2

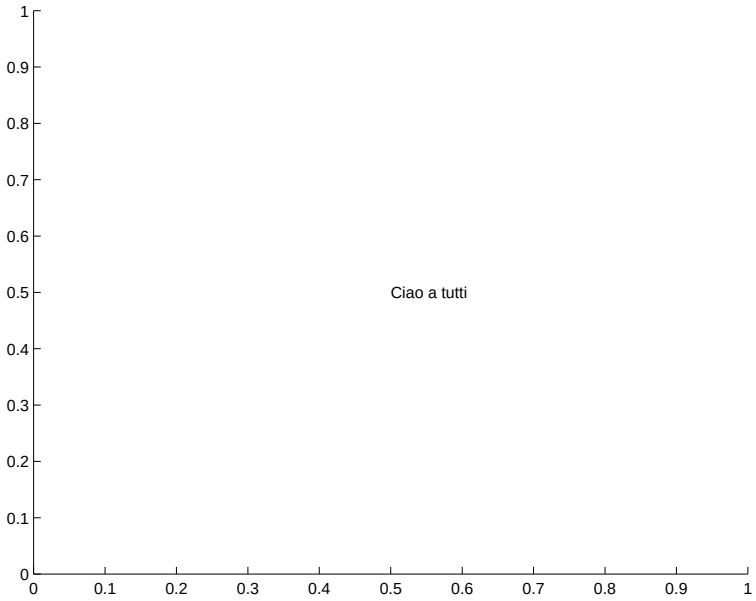

Figura 17.2: Figura di partenza

Ora leggiamo l'handle di questa figura

```
>> h=gcf
h =
     1
```

Possiamo ora vedere le caratteristiche messa di default di questa figura

```
>> get(h)
       BackingStore = on
       Color = [0 0 0]
       Colormap = [ (64 by 3) ]
       CurrentAxes = [58.0002]
       CurrentCharacter =
       CurrentMenu = [1]
       CurrentObject = []
       CurrentPoint = [0 0]
       FixedColors = [ (8 by 3) ]
```



```
        InvertHardcopy = on
        KeyPressFcn =
        MenuBar = figure
        MinColormap = [64]
        Name =
        NextPlot = add
        NumberTitle = on
        PaperUnits = inches
        PaperOrientation = portrait
        PaperPosition = [0.25 2.5 8 6]
        PaperSize = [8.5 11]
        PaperType = usletter
        Pointer = arrow
        Position = [23 299 560 420]
        Resize = on
        SelectionType = normal
        ShareColors = yes
        Units = pixels
        WindowButtonDownFcn =
        WindowButtonMotionFcn =
        WindowButtonUpFcn =

        ButtonDownFcn =
        Children = [58.0002]
        Clipping = on
        Interruptible = no
        Parent = [0]
        Type = figure
        UserData = []
        Visible = on

ans =
      []
```

Settiamo ora il colore dello sfondo a grigio

```
>> set(h,'Color',[0.5,0.5,0.5])
```

Vediamo ora l'handle agli assi, per poter accedere agli oggetti della figura

```
>> a=gca
a =
   58.0002
```

Le caratteristiche di default degli assi sono

```
>> get(a)
        AspectRatio = [NaN NaN]
```

```
Box = off
CLim = [0 1]
CLimMode = auto
Color = none
CurrentPoint = [ (2 by 3) ]
ColorOrder = [ (6 by 3) ]
DrawMode = normal
FontAngle = normal
FontName = Helvetica
FontSize = [12]
FontStrikeThrough = off
FontUnderline = off
FontWeight = normal
GridLineStyle = :
LineStyleOrder = -
LineWidth = [0.5]
NextPlot = replace
Position = [0.13 0.11 0.775 0.815]
TickLength = [0.01 0.025]
TickDir = in
Title = [63.0001]
Units = normalized
View = [0 90]
XColor = [1 1 1]
XDir = normal
Xform = [ (4 by 4) ]
XGrid = off
XLabel = [60.0002]
XLim = [0 1]
XLimMode = auto
XScale = linear
XTick = [0 0.2 0.4 0.6 0.8 1]
XTickLabels =
        0
        0.2
        0.4
        0.6
        0.8
        1
XTickLabelMode = auto
XTickMode = auto
YColor = [1 1 1]
YDir = normal
YGrid = off
YLabel = [61.0002]
YLim = [0 1]
YLimMode = auto
```



```
        YScale = linear
        YTick = [ (1 by 11) ]
        YTickLabels = [ (11 by 3) ]
        YTickLabelMode = auto
        YTickMode = auto
        ZColor = [1 1 1]
        ZDir = normal
        ZGrid = off
        ZLabel = [62.0002]
        ZLim = [0 1]
        ZLimMode = auto
        ZScale = linear
        ZTick = [0 0.5 1]
        ZTickLabels =
        ZTickLabelMode = auto
        ZTickMode = auto

        ButtonDownFcn =
        Children = [59.0002]
        Clipping = on
        Interruptible = no
        Parent = [1]
        Type = axes
        UserData = []
        Visible = on

    ans =
        []
```

Vediamo che è definito un unico figlio (Children) di questi assi. Possiamo vederne l'handle e le caratteristiche

```
>> ch=get(a,'Children')
ch =
   59.0002
>> get(ch)
        Color = [1 1 1]
        EraseMode = normal
        Extent = [0.498845 0.469208 0.17321 0.0527859]
        FontAngle = normal
        FontName = Helvetica
        FontSize = [12]
        FontStrikeThrough = off
        FontUnderline = off
        FontWeight = normal
        HorizontalAlignment = left
        Position = [0.5 0.5 0]
        Rotation = [0]
```

```
        String = Ciao a tutti
        Units = data
        VerticalAlignment = middle

        ButtonDownFcn =
        Children = []
        Clipping = off
        Interruptible = no
        Parent = [58.0002]
        Type = text
        UserData = []
        Visible = on

ans =
    []
```

Cambiamo il colore in verde, il testo della stringa, ruotiamo il testo di 90 gradi e aumentiamo la grandezza del font

```
>> set(ch,'Color',[0,1,0])
>> set(ch,'String','Ciao')
>> set(ch,'Rotation',90)
>> set(ch,'FontSize',30)
>> set(ch,'HorizontalAlignment','center')
>> set(ch,'FontSize',30)
```

Il risultato finale ottenuto è visibile nella figura 17.3.

La manipolazione di immagini non è difficile. Utilizzando unicamente i comandi *get* e *set*, si possono modificare a piacimento le caratteristiche di un grafico. Chiaramente si tratta di un lavoro relativamente lungo, che può essere parzialmente automatizzato con degli script e delle funzioni.

Le proprietà di ogni oggetto e i possibili valori a disposizione, sono riportati nell'allegato B.

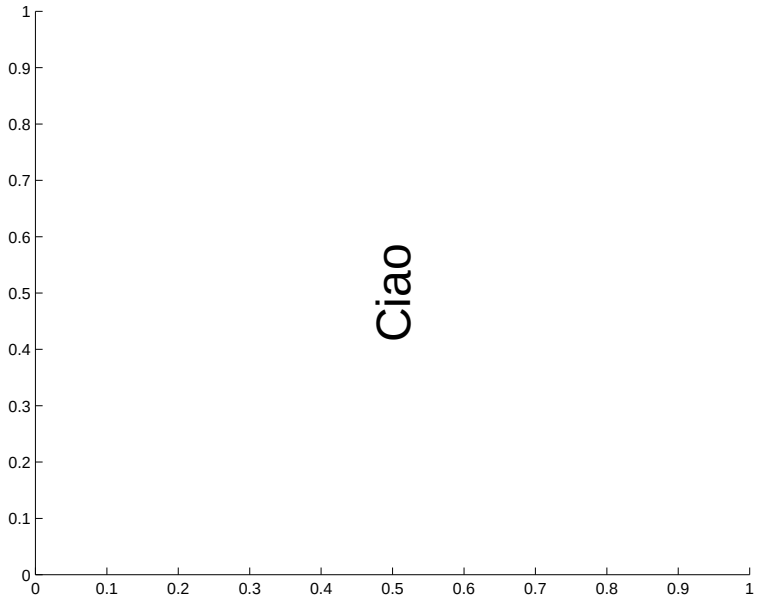

Figura 17.3: Risultato finale della manipolazione

Capitolo 18

Strutture particolari

18.1 Matrici multidimensionali

Dalla versione 5.0 di MATLAB le dimensioni delle matrici non sono più limitate a 2, ma possono essere più grandi. Le funzioni che permettono di creare matrici speciali nelle due dimensioni, possono essere utilizzate anche su n dimensioni. Per esempio, il comando $ones(2, 3, 4, 1)$ permette di creare una matrice di uni su 4 dimensioni.

Quando vogliamo utilizzare una matrice a 3 dimensioni, ad esempio, la terza dimensione è definita *pagina* (*page*). Per crearla occorre utilizzare il comando *cat*, per concatenare delle matrici bidimensionali sulla 3. dimensione.

```
>> a=[1,2,3;4,5,6;7,8,9]
a =
     1     2     3
     4     5     6
     7     8     9
>> b=10*a
b =
    10    20    30
    40    50    60
    70    80    90
>> c=cat(3,a,b)
c(:,:,1) =
     1     2     3
     4     5     6
     7     8     9
c(:,:,2) =
    10    20    30
    40    50    60
    70    80    90
```

La rappresentazione di una matrice multidimensionale viene fatta pagina per pagina.

Con i comando *repmat* è possibile creare una matrice multidimensionale con contenuto costante

```
>> B=repmat(pi,[2 2 3 3])
```

```
B(:,:,1,1) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,2,1) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,3,1) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,1,2) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,2,2) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,3,2) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,1,3) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,2,3) =
    3.1416    3.1416
    3.1416    3.1416
B(:,:,3,3) =
    3.1416    3.1416
    3.1416    3.1416
```

Ta tabella 18.1mostra altri possibili comandi su matrici multidimensionali.

flipdim	gira una matrice su una determinata dimensione
ndgrid	genera una matrice multidimensionale per funzioni o interpolazione
ndims	numero delle dimensioni di una matrice
permute,ipermute	permuta le dimensioni di una matrice
reshape	cambia le dimensioni
shiftdim	sposta le dimensioni
squeeze	elimina dimensioni singole
sub2ind,ind2sub	indice lineare corrispondente

Tabella 18.1: Operazioni su matrici multidimensionali

Anche diversi comandi come *sum*, *prod*, *cumsum* ed altri ancora, sono stati completati con un'informazione supplementare che permette di decidere su quale dimensione vanno applicati (vedi help di ogni comando).

18.2 Celle

Le celle sono matrici che possono contenere elementi di tipo diverso fra di loro. Le celle possono essere create direttamente per assegnamento, o utilizzando il comando cell. Da notare che l'indicizzazione ad una cella viene fatto utilizzando le parentesi graffe.

```
>> A(1,1)={[1,2,3;4,5,6;7,8,9]}
A =
    [3x3 double]
```

oppure

```
>> A{1,1}=[1,2,3;4,5,6;7,8,9]
A =
    [3x3 double]
```

Completiamo ora questa matrice di celle

```
>> A{1,2}=2+3i
A =
   [3x3 double]    [2.0000+ 3.0000i]
>> A{2,1}='Ciao a tutti'
A =
   [3x3 double]    [2.0000+ 3.0000i]
   'Ciao a tutti'                  []
>> A{2,2}=10:-1:1;
```

Visualizziamo il contenuto di A

```
>> A
A =
   [3x3 double]    [2.0000+ 3.0000i]
  'Ciao a tutti'        [1x10 double]
```

Possiamo vedere come si visualizzino informazioni sul contenuto di ogni cella, ma non il contenuto stesso. Per accedere ad un contenuto possiamo farlo sfruttando le parentesi graffe

```
>> A{2,2}
ans =
   10   9   8   7   6   5   4   3   2   1
```

oppure

```
>> A{2,2}(2)
ans =
     9
```

La matrice di celle può anche essere visualizzata in forma grafica (vedi figura 18.1.

Le matrici di celle possono essere modificate e manipolate come normali matrici (concatenazione, scambio di elementi, ecc.). L'uso principale previsto è quello della gestione delle stringhe, senza dover a tutti i costi ridimensionare ogni riga di una matrice alla dimensione più grande (vedi capitolo sulle stringhe).

La tabella 18.2 riporta alcuni comandi applicabili su celle.


```
>>cellplot(A)
```

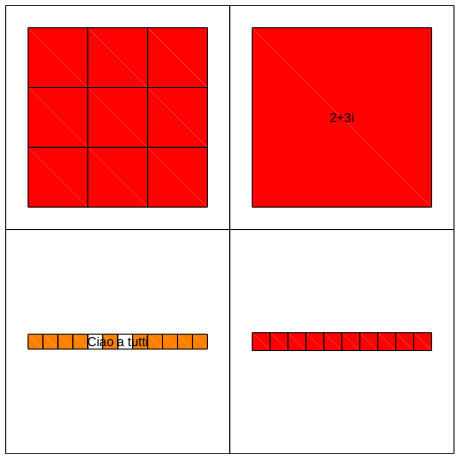

Figura 18.1: Risultato del comando *cellplot*

cell	crea una cella
cell2struct	conversione da cella astruttura
celldisp	mostra la struttura della cella
cellplot	mostra la struttura della cella in forma grafica
num2cell	converte una matrice in una cella

Tabella 18.2: Comandi applicabili a celle

18.3 Strutture

Un'altra novità introdotta dalla versione 5 sono le strutture. Le strutture sono particolari variabili che contengono dei campi con informazioni di tipi differente. Le strutture utilizzano la sintassi con il punto, sintassi comune a molti linguaggi di programmazione. I campi possono essere generati dinamicamente nel momento in cui vengono utilizzati per la prima volta, oppure possono essere stati dichiarati prima dell'uso.

```
>> professore.cognome='Bucher'
professore =
    cognome: 'Bucher'
>> professore.nome='Roberto'
professore =
    cognome: 'Bucher'
       nome: 'Roberto'
>> professore.materia={'Regolazione','Automazione'}
professore =
    cognome: 'Bucher'
       nome: 'Roberto'
    materia: {1x2 cell}
>> professore.materia{1}
ans =
   Regolazione
```


Come si può vedere l'elemento di una struttura può essere una matrice, una cella, o un'altra struttura.

I dati possono essere immessi anche utilizzando il comando *struct*

```
>> Cognomi={'Bucher','Bernasconi','Tizio'};
>> Nomi={'Roberto','Giovanni','Caio'};
>> Materie={{'Regolazione','Automazione'},{'Matematica'},{'Latino'}};
>> professore=struct('cognome',Cognomi,'nome',Nomi,'materia',Materie)
professore =
 1x3 struct array with fields:
    cognome
    nome
    materia
```

L'accesso ai dati può essere fatto come per le normali matrici

```
>> professore(1)
ans =
    cognome: 'Bucher'
       nome: 'Roberto'
    materia: {1x2 cell}
>> professore.nome
ans =
    Roberto
ans =
    Giovanni
ans =
     Caio
>> getfield(professore,{1},'nome')
ans =
     Roberto
```

È possibile accedere al nome dei campi tramite il comando *fieldnames*, in modo da poter utilizzare il risultato ad esempio in un programma

```
>> T=fieldnames(professore)
T =
    'cognome'
    'nome'
    'materia'
```

Altri comandi applicabili alle strutture sono riportati nella tabella 18.3.

Ulteriori informazioni su celle e strutture possono essere trovate con il comando *help datatypes*.

rmfield	cancellazione di un campo
setfield	sostituzione del contenuto di un campo
struct2cell	conversione da struttura a cella
isstruct	controllo se un certo elemento è una struttura

Tabella 18.3: Comandi su strutture

Capitolo 19

Programmazione orientata agli oggetti

19.1 Introduzione

Tramite gli oggetti e le classi è possibile introdurre in MATLAB nuovi tipi di dati e nuove operazioni. La classe indica la struttura di una variabile e le operazioni e funzioni che possono essere applicate a questa variabile.

In MATLAB a partire dalla versione 5, i tipi *double*, *sparse*, *char*, *struct* e *cell* sono già implementati come oggetti. In alcuni toolbox troviamo oggetti particolari, come ad esempio nel toolbox di controllo con gli oggetti LTI.

Oggetti possono essere implementati specificandone la struttura e creando direttori di funzioni che agiscono su questi oggetti (metodi). Esiste anche la possibilità di sovrascrivere alcuni metodi (overloading).

19.2 Costruttori

Oggetti di una classe possono essere instanziati senza doverli dichiarare in precedenza; gli oggetti vengono creati dinamicamente tramite un costruttore della classe.

Quale esempio utilizzeremo un nuovo oggetto da creare chiamato *polinomio*. Come in altri linguaggi ad oggetti il costruttore mantiene il nome della classe. Le informazioni e i metodi di una classe vanno messi in un direttorio con il nome *@nomeclasse*. Nel nostro esempio dobbiamo creare un direttorio chiamato *@polinomio*.

Il nostro oggetto sarà una struttura con un solo campo contenente una matrice di coefficienti. Creiamo ora il costruttore, che verrà chiamato passando come parametro la matrice dei coefficienti, oppure un altro polinomio.

```
function p=polinomio(coeff)
%
%   Costruttore della classe polinomio
%
%   p=polinomio(a) costruisce un polinomio
%   partendo dai coefficienti a o da un altro polinomio
%   in coeff
%
```



```
if nargin==0
   p.c=[];
   p=class(p,'polinomio');
elseif isa(coeff,'polinomio')
   p=coeff;
else
   p.c=coeff(:).';
   p=class(p,'polinomio');
end
```

La funzione *isa* controlla se l'oggetto passato $coeff$ appartiene già alla classe *polinomio*. In quest'ultimo caso l'oggetto viene semplicemente copiato.

L'uso di questo costruttore è molto semplice

```
>> p=polinomio([1 2 3])
p =
        polinomio object: 1-by-1
>> p1=polinomio(p)
p1 =
        polinomio object: 1-by-1
>> whos
  Name     Size      Bytes  Class

  p        1x1         148  polinomio object
  p1       1x1         148  polinomio object

Grand total is 8 elements using 296 bytes
```

19.3 Funzioni di conversione

Creiamo ora una funzione di conversione da *polinomio* a *double*.

La funzione deve avere il nome della classe del risultato

```
function c=double(p)
% c=double(p)
% Estrazione dei coefficienti del polinomio
%
c=p.c;
```

Applicando ora la funzione

```
>> double(p)
ans =
     1     2     3
```

Nello stesso modo possiamo creare un metodo di conversione a *char*, che ci fornisce il polinomio in forma di stringa

```
function s=char(p)
% s=char(p)
% Estrazione del polinomio come stringa
%
c=p.c;
if all(c==0)
   s='0';
else
   d=length(p.c)-1;
   s=[];
   for a=c;
      if a~=0;
         if ~isempty(s)
            if a>0
               s=[s ' + '];
            else
               s=[s ' - '];
               a=-a;
            end
         end
         if a~=1 | d==0
            s=[s num2str(a)];
            if d>0
               s=[s '*'];
            end
         end
         if d>=2
            s=[s 'x^' int2str(d)];
         elseif d==1
            s=[s 'x'];
         end
      end
      d=d-1;
   end
end
```

Il risultato della conversione è

```
>> p=polinomio([1 2 3 4 5]);
>> char(p)
ans =
x^4 + 2*x^3 + 3*x^2 + 4*x + 5
```

Si può definire un metodo chiamato *display*, che sarà il metodo chiamato ogni volta che si deve visualizzare una variabile di tipo polinomio

```
function display(p)
%
```

```
%  Mostra il polinomio
%
disp(' ');
disp([inputname(1),' = ',])
disp(' ');
disp(['   ' char(p)])
disp(' ');
```

Quando ora si crea un elemento di tipo polinomio otteniamo

```
>> x=polinomio([1,2,2])
x =
   x^2 + 2*x + 2
```

19.4 Overloading

All'interno di una classe è possibile sovrascrivere gli operatori classici con nuove operazioni inerenti gli oggetti di una classe. Gli operatori che possono essere sovrascritti sono definiti nella tabella 19.1.

Vediamo ora di sovrascrivere l'operatore `*` , in modo da moltiplicare tra di loro due polinomi

```
function res=mtimes(p1,p2)
%
%   res=p1*p2 moltiplicazione polinomiale
%
p1=polinomio(p1);
p2=polinomio(p2);
res=polinomio(conv(p1.c,p2.c));
>> p1=polinomio([1,2,1])
p1 =
   x^2 + 2*x + 1
>> p2=polinomio([1,0,0])
p2 =
   x^2
>> p3=p1*p2
p3 =
   x^4 + 2*x^3 + x^2
```

Possiamo anche sovrascrivere altre funzioni come ad esempio il comando *roots*, così da utilizzarlo con la nuova classe *polinomio*.

```
function r=roots(p)
%
%   r=roots(p)
%
%   Radici del polinomio
%
```

Operazione	M-file	Descrizione
`a + b`	plus(a,b)	somma
`a - b`	minus(a,b)	sottrazione
`-a`	uminus(a)	decremento
`+a`	uplus(a)	incremento
`a.*b`	times(a,b)	moltiplicazione elemento x elemento
`a*b`	mtimes	moltiplicazione matriciale
`a./b`	rdivide(a,b)	divisione a destra
`a.\b`	ldivide(a,b)	divisione a sinistra
`a/b`	mrdivide(a,b)	divisione matriciale a destra
`a\b`	mldivide(a,b)	divisione matriciale a sinistra
`a.^b`	power(a,b)	potenza elemento x elemento
`a^b`	mpower(a,b)	potenza matriciale
`a < b`	lt(a,b)	minore di
`a > b`	gt(a,b)	maggiore di
`a <= b`	le(a,b)	minore o uguale di
`a >= b`	ge(a,b)	maggiore o uguale di
`a ~=b`	ne(a,b)	diverso da
`a == b`	eq(a,b)	uguale a
`a & b`	and(a,b)	and logico
`a| b`	or(a,b)	or logico
`~a`	not(a)	negazione logica
`a:d:b`	colon(a,d,b)	operatore :
`a:b`	colon(a,b)	operatore :
`a'`	ctranspose(a)	trasposizione complessa coniugata
`a.'`	transpose(a)	trasposizione matriciale
`[a b]`	horzcat(a,b)	concatenazione orizzontale
`[a;b]`	vertcat	concatenazione verticale

Tabella 19.1: Operatori sovrascrivibili

```
r=roots(p.c);
>> p=polinomio([1 3 2 0])
p =
    x^3 + 3*x^2 + 2*x
>> roots(p)
ans =
     0
    -2
    -1
```

19.5 Ereditarietà e aggregazione

Una classe può ereditare le caratteristiche di un'altra classe ed aggiungere in seguito proprie particolarità o modificare metodi della classe superiore. Si può analizzare l'oggetto LTI

del *Control System Toolbox*, che utilizza questi metodi per descrivere i processi in diverse forme (Funzione di trasferimento, Piano degli stati, Zeri-Poli-Guadagno).

È possibile avere un'eredità semplice o multipla, per mezzo del comando *class*, specificando la struttura, il nome della classe e il nome della o delle classi da cui si ereditano le caratteristiche.

Per quanto riguarda l'aggregazione, un oggetto può contenere quali campi altri oggetti. Si può inoltre creare una sottodirectory *@private*, che contiene metodi privati della classe in questione.

Appendice A

Comandi di Matlab

A.1 Preferences.

A.1.1 Saved preferences files.

```
startup     - User startup M-file.
finish      - User finish M-file.
matlabrc    - Master startup M-file.
pathdef     - Search path defaults.
docopt      - Web browser defaults.
printopt    - Printer defaults.
```

A.1.2 Preference commands.

```
cedit       - Set command line editor keys.
terminal    - Set graphics terminal type.
colordef    - Set color defaults.
graymon     - Set graphics window defaults for gray-scale monitors.
whitebg     - Change axes background color.
```

A.1.3 Configuration information.

```
hostid      - MATLAB server host identification number.
license     - License number.
version     - MATLAB version number.
```

A.2 General purpose commands.

A.2.1 General information

```
help        - On-line help, display text at command line.
helpwin     - On-line help, separate window for navigation.
helpdesk    - Comprehensive hypertext documentation and troubleshooting.
demo        - Run demonstrations.
ver         - MATLAB, SIMULINK, and toolbox version information.
whatsnew    - Display Readme files.
Readme      - Whats new in MATLAB 5.
```

A.2.2 Managing the workspace.

```
who         - List current variables.
whos        - List current variables, long form.
clear       - Clear variables and functions from memory.
pack        - Consolidate workspace memory.
load        - Load workspace variables from disk.
save        - Save workspace variables to disk.
quit        - Quit MATLAB session.
```

A.2.3 Managing commands and functions.

```
what        - List MATLAB-specific files in directory.
type        - List M-file.
edit        - Edit M-file.
lookfor     - Search all M-files for keyword.
which       - Locate functions and files.
pcode       - Create pre-parsed pseudo-code file (P-file).
inmem       - List functions in memory.
mex         - Compile MEX-function.
```

A.2.4 Managing the search path

```
path        - Get/set search path.
addpath     - Add directory to search path.
rmpath      - Remove directory from search path.
editpath    - Modify search path.
```

A.2.5 Controlling the command window.

```
echo        - Echo commands in M-files.
more        - Control paged output in command window.
diary       - Save text of MATLAB session.
format      - Set output format.
```

A.2.6 Operating system commands

```
cd          - Change current working directory.
pwd         - Show (print) current working directory.
dir         - List directory.
delete      - Delete file.
getenv      - Get environment variable.
!           - Execute operating system command.
dos         - Execute DOS command and return result.
unix        - Execute UNIX command and return result.
vms         - Execute VMS DCL command and return result.
web         - Open Web browser on site or files.
computer    - Computer type.
```

A.2.7 Debugging M-files.

```
debug       - List debugging commands.
dbstop      - Set breakpoint.
dbclear     - Remove breakpoint.
dbcont      - Continue execution.
dbdown      - Change local workspace context.
dbstack     - Display function call stack.
dbstatus    - List all breakpoints.
dbstep      - Execute one or more lines.
dbtype      - List M-file with line numbers.
dbup        - Change local workspace context.
dbquit      - Quit debug mode.
dbmex       - Debug MEX-files (UNIX only).
```

A.2.8 Profiling M-files.

```
profile     - Profile M-file execution time.
```

A.3 Operators and special characters.

A.3.1 Arithmetic operators.

```
plus        - Plus                            +
uplus       - Unary plus                      +
minus       - Minus                           -
uminus      - Unary minus                     -
mtimes      - Matrix multiply                 *
times       - Array multiply                 .*
mpower      - Matrix power                    ^
power       - Array power                    .^
mldivide    - Backslash or left matrix divide  \
mrdivide    - Slash or right matrix divide     /
ldivide     - Left array divide              .\
rdivide     - Right array divide             ./
kron        - Kronecker tensor product      kron
```

A.3.2 Relational operators.

```
eq          - Equal                           ==
ne          - Not equal                       ~=
lt          - Less than                        <
gt          - Greater than                     >
le          - Less than or equal              <=
ge          - Greater than or equal           >=
```

A.3.3 Logical operators.

```
and         - Logical AND                      &
```

```
or          - Logical OR                          |
not         - Logical NOT                         ~
xor         - Logical EXCLUSIVE OR
any         - True if any element of vector is nonzero
all         - True if all elements of vector are nonzero
```

A.3.4 Special characters.

```
colon       - Colon                               :
paren       - Parentheses and subscripting      ( )
paren       - Parentheses and subscripting      ( )
paren       - Brackets                          [ ]
paren       - Braces and subscripting           { }
paren       - Braces and subscripting           { }
punct       - Decimal point                      .
punct       - Structure field access             .
punct       - Parent directory                   ..
punct       - Continuation                       ...
punct       - Separator                          ,
punct       - Semicolon                          ;
punct       - Comment                            %
punct       - Invoke operating system command    !
punct       - Assignment                         =
punct       - Quote
transpose   - Transpose                         .
ctranspose - Complex conjugate transpose
horzcat     - Horizontal concatenation          [,]
vertcat     - Vertical concatenation            [;]
subsasgn    - Subscripted assignment         ( ),{ },.
subsref     - Subscripted reference          ( ),{ },.
subsindex   - Subscript index
```

A.3.5 Bitwise operators.

```
bitand      - Bit-wise AND.
bitcmp      - Complement bits.
bitor       - Bit-wise OR.
bitmax      - Maximum floating point integer.
bitxor      - Bit-wise XOR.
bitset      - Set bit.
bitget      - Get bit.
bitshift    - Bit-wise shift.
```

A.3.6 Set operators.

```
union       - Set union.
unique      - Set unique.
intersect   - Set intersection.
```

```
setdiff    - Set difference.
setxor     - Set exclusive-or.
ismember   - True for set member.
```

A.4 Programming language constructs.

A.4.1 Control flow.

```
if          - Conditionally execute statements.
else        - IF statement condition.
elseif      - IF statement condition.
end         - Terminate scope of FOR, WHILE, SWITCH and IF statements.
for         - Repeat statements a specific number of times.
while       - Repeat statements an indefinite number of times.
break       - Terminate execution of WHILE or FOR loop.
switch      - Switch among several cases based on expression.
case        - SWITCH statement case.
otherwise   - Default SWITCH statement case.
return      - Return to invoking function.
```

A.4.2 Evaluation and execution.

```
eval        - Execute string with MATLAB expression.
feval       - Execute function specified by string.
evalin      - Evaluate expression in workspace.
builtin     - Execute built-in function from overloaded method.
assignin    - Assign variable in workspace.
run         - Run script.
```

A.4.3 Scripts, functions, and variables.

```
script      - About MATLAB scripts and M-files.
function    - Add new function.
global      - Define global variable.
mfilename   - Name of currently executing M-file.
lists       - Comma separated lists.
exist       - Check if variables or functions are defined.
isglobal    - True for global variables.
```

A.4.4 Argument handling.

```
nargchk     - Validate number of input arguments.
nargin      - Number of function input arguments.
nargout     - Number of function output arguments.
varargin    - Variable length input argument list.
varargout   - Variable length output argument list.
inputname   - Input argument name.
```


A.4.5 Message display.

```
error       - Display error message and abort function.
warning     - Display warning message.
lasterr     - Last error message.
errortrap   - Skip error during testing.
disp        - Display an array.
fprintf     - Display formatted message.
sprintf     - Write formatted data to a string.
```

A.4.6 Interactive input.

```
input       - Prompt for user input.
keyboard    - Invoke keyboard from M-file.
pause       - Wait for user response.
uimenu      - Create user interface menu.
uicontrol   - Create user interface control.
```

A.5 Elementary matrices and matrix manipulation.

A.5.1 Elementary matrices.

```
zeros       - Zeros array.
ones        - Ones array.
eye         - Identity matrix.
repmat      - Replicate and tile array.
rand        - Uniformly distributed random numbers.
randn       - Normally distributed random numbers.
linspace    - Linearly spaced vector.
logspace    - Logarithmically spaced vector.
meshgrid    - X and Y arrays for 3-D plots.
:           - Regularly spaced vector and index into matrix.
```

A.5.2 Basic array information.

```
size        - Size of matrix.
length      - Length of vector.
ndims       - Number of dimensions.
disp        - Display matrix or text.
isempty     - True for empty matrix.
isequal     - True if arrays are identical.
isnumeric   - True for numeric arrays.
islogical   - True for logical array.
logical     - Convert numeric values to logical.
```

A.5.3 Matrix manipulation.

```
reshape     - Change size.
diag        - Diagonal matrices and diagonals of matrix.
```

```
tril        - Extract lower triangular part.
triu        - Extract upper triangular part.
fliplr      - Flip matrix in left/right direction.
flipud      - Flip matrix in up/down direction.
flipdim     - Flip matrix along specified dimension.
rot90       - Rotate matrix 90 degrees.
:           - Regularly spaced vector and index into matrix.
find        - Find indices of nonzero elements.
end         - Last index.
sub2ind     - Linear index from multiple subscripts.
ind2sub     - Multiple subscripts from linear index.
```

A.5.4 Special variables and constants.

```
ans         - Most recent answer.
eps         - Floating point relative accuracy.
realmax     - Largest positive floating point number.
realmin     - Smallest positive floating point number.
pi          - 3.1415926535897....
i, j        - Imaginary unit.
inf         - Infinity.
NaN         - Not-a-Number.
isnan       - True for Not-a-Number.
isinf       - True for infinite elements.
isfinite    - True for finite elements.
flops       - Floating point operation count.
why         - Succinct answer.
```

A.5.5 Specialized matrices.

```
compan      - Companion matrix.
gallery     - Higham test matrices.
hadamard    - Hadamard matrix.
hankel      - Hankel matrix.
hilb        - Hilbert matrix.
invhilb     - Inverse Hilbert matrix.
magic       - Magic square.
pascal      - Pascal matrix.
rosser      - Classic symmetric eigenvalue test problem.
toeplitz    - Toeplitz matrix.
vander      - Vandermonde matrix.
wilkinson   - Wilkinsons eigenvalue test matrix.
```

A.6 Elementary math functions.

A.6.1 Trigonometric.

```
sin         - Sine.
```



```
sinh        - Hyperbolic sine.
asin        - Inverse sine.
asinh       - Inverse hyperbolic sine.
cos         - Cosine.
cosh        - Hyperbolic cosine.
acos        - Inverse cosine.
acosh       - Inverse hyperbolic cosine.
tan         - Tangent.
tanh        - Hyperbolic tangent.
atan        - Inverse tangent.
atan2       - Four quadrant inverse tangent.
atanh       - Inverse hyperbolic tangent.
sec         - Secant.
sech        - Hyperbolic secant.
asec        - Inverse secant.
asech       - Inverse hyperbolic secant.
csc         - Cosecant.
csch        - Hyperbolic cosecant.
acsc        - Inverse cosecant.
acsch       - Inverse hyperbolic cosecant.
cot         - Cotangent.
coth        - Hyperbolic cotangent.
acot        - Inverse cotangent.
acoth       - Inverse hyperbolic cotangent.
```

A.6.2 Exponential.

```
exp         - Exponential.
log         - Natural logarithm.
log10       - Common (base 10) logarithm.
log2        - Base 2 logarithm and dissect floating point number.
pow2        - Base 2 power and scale floating point number.
sqrt        - Square root.
nextpow2    - Next higher power of 2.
```

A.6.3 Complex.

```
abs         - Absolute value.
angle       - Phase angle.
conj        - Complex conjugate.
imag        - Complex imaginary part.
real        - Complex real part.
unwrap      - Unwrap phase angle.
isreal      - True for real array.
cplxpair    - Sort numbers into complex conjugate pairs.
```

A.6.4 Rounding and remainder.

```
fix         - Round towards zero.
floor       - Round towards minus infinity.
ceil        - Round towards plus infinity.
round       - Round towards nearest integer.
mod         - Modulus (signed remainder after division).
rem         - Remainder after division.
sign        - Signum.
```

A.7 Specialized math functions.

A.7.1 Specialized math functions.

```
airy        - Airy functions.
besselj     - Bessel function of the first kind.
bessely     - Bessel function of the second kind.
besselh     - Bessel functions of the third kind (Hankel function).
besseli     - Modified Bessel function of the first kind.
besselk     - Modified Bessel function of the second kind.
beta        - Beta function.
betainc     - Incomplete beta function.
betaln      - Logarithm of beta function.
ellipj      - Jacobi elliptic functions.
ellipke     - Complete elliptic integral.
erf         - Error function.
erfc        - Complementary error function.
erfcx       - Scaled complementary error function.
erfinv      - Inverse error function.
expint      - Exponential integral function.
gamma       - Gamma function.
gammainc    - Incomplete gamma function.
gammaln     - Logarithm of gamma function.
legendre    - Associated Legendre function.
cross       - Vector cross product.
```

A.7.2 Number theoretic functions.

```
factor      - Prime factors.
isprime     - True for prime numbers.
primes      - Generate list of prime numbers.
gcd         - Greatest common divisor.
lcm         - Least common multiple.
rat         - Rational approximation.
rats        - Rational output.
perms       - All possible permutations.
nchoosek    - All combinations of N elements taken K at a time.
```


A.7.3 Coordinate transforms.

```
cart2sph    - Transform Cartesian to spherical coordinates.
cart2pol    - Transform Cartesian to polar coordinates.
pol2cart    - Transform polar to Cartesian coordinates.
sph2cart    - Transform spherical to Cartesian coordinates.
hsv2rgb     - Convert hue-saturation-value colors to red-green-blue.
rgb2hsv     - Convert red-green-blue colors to hue-saturation-value.
```

A.8 Matrix functions - numerical linear algebra.

A.8.1 Matrix analysis.

```
norm        - Matrix or vector norm.
normest     - Estimate the matrix 2-norm.
rank        - Matrix rank.
det         - Determinant.
trace       - Sum of diagonal elements.
null        - Null space.
orth        - Orthogonalization.
rref        - Reduced row echelon form.
subspace    - Angle between two subspaces.
```

A.8.2 Linear equations.

```
\ and /     - Linear equation solution; use "help slash".
inv         - Matrix inverse.
cond        - Condition number with respect to inversion.
condest     - 1-norm condition number estimate.
chol        - Cholesky factorization.
cholinc     - Incomplete Cholesky factorization.
lu          - LU factorization.
luinc       - Incomplete LU factorization.
qr          - Orthogonal-triangular decomposition.
nnls        - Non-negative least-squares.
pinv        - Pseudoinverse.
lscov       - Least squares with known covariance.
```

A.8.3 Eigenvalues and singular values.

```
eig         - Eigenvalues and eigenvectors.
svd         - Singular value decomposition.
eigs        - A few eigenvalues.
svds        - A few singular values.
poly        - Characteristic polynomial.
polyeig     - Polynomial eigenvalue problem.
condeig     - Condition number with respect to eigenvalues.
hess        - Hessenberg form.
```

```
qz          - QZ factorization for generalized eigenvalues.
schur       - Schur decomposition.
```

A.8.4 Matrix functions.

```
expm        - Matrix exponential.
logm        - Matrix logarithm.
sqrtm       - Matrix square root.
funm        - Evaluate general matrix function.
```

A.8.5 Factorization utilities

```
qrdelete    - Delete column from QR factorization.
qrinsert    - Insert column in QR factorization.
rsf2csf     - Real block diagonal form to complex diagonal form.
cdf2rdf     - Complex diagonal form to real block diagonal form.
balance     - Diagonal scaling to improve eigenvalue accuracy.
planerot    - Givens plane rotation.
```

A.9 Data analysis and Fourier transforms.

A.9.1 Basic operations.

```
max         - Largest component.
min         - Smallest component.
mean        - Average or mean value.
median      - Median value.
std         - Standard deviation.
sort        - Sort in ascending order.
sortrows    - Sort rows in ascending order.
sum         - Sum of elements.
prod        - Product of elements.
hist        - Histogram.
trapz       - Trapezoidal numerical integration.
cumsum      - Cumulative sum of elements.
cumprod     - Cumulative product of elements.
cumtrapz    - Cumulative trapezoidal numerical integration.
```

A.9.2 Finite differences.

```
diff        - Difference and approximate derivative.
gradient    - Approximate gradient.
del2        - Discrete Laplacian.
```

A.9.3 Correlation.

```
corrcoef    - Correlation coefficients.
cov         - Covariance matrix.
subspace    - Angle between subspaces.
```


A.9.4 Filtering and convolution.

```
filter      - One-dimensional digital filter.
filter2     - Two-dimensional digital filter.
conv        - Convolution and polynomial multiplication.
conv2       - Two-dimensional convolution.
convn       - N-dimensional convolution.
deconv      - Deconvolution and polynomial division.
```

A.9.5 Fourier transforms.

```
fft         - Discrete Fourier transform.
fft2        - Two-dimensional discrete Fourier transform.
fftn        - N-dimensional discrete Fourier Transform.
ifft        - Inverse discrete Fourier transform.
ifft2       - Two-dimensional inverse discrete Fourier transform.
ifftn       - N-dimensional inverse discrete Fourier Transform.
fftshift    - Move zeroth lag to center of spectrum.
```

A.9.6 Sound and audio.

```
sound       - Play vector as sound.
soundsc     - Autoscale and play vector as sound.
speak       - Convert input string to speech (Macintosh only).
recordsound - Record sound (Macintosh only).
soundcap    - Sound capabilities (Macintosh only).
mu2lin      - Convert mu-law encoding to linear signal.
lin2mu      - Convert linear signal to mu-law encoding.
```

A.9.7 Audio file inport/export.

```
auwrite     - Write NeXT/SUN (".au") sound file.
auread      - Read NeXT/SUN (".au") sound file.
wavwrite    - Write Microsoft WAVE (".wav") sound file.
wavread     - Read Microsoft WAVE (".wav") sound file.
readsnd     - Read SND resources and files (Macintosh only).
writesnd    - Write SND resources and files (Macintosh only).
```

A.10 Interpolation and polynomials.

A.10.1 Data interpolation.

```
interp1     - 1-D interpolation (table lookup).
interp1q    - Quick 1-D linear interpolation.
interpft    - 1-D interpolation using FFT method.
interp2     - 2-D interpolation (table lookup).
interp3     - 3-D interpolation (table lookup).
interpn     - N-D interpolation (table lookup).
griddata    - Data gridding and surface fitting.
```

A.10.2 Spline interpolation.

```
spline      - Cubic spline interpolation.
ppval       - Evaluate piecewise polynomial.
```

A.10.3 Geometric analysis.

```
delaunay    - Delaunay triangulation.
dsearch     - Search Delaunay triagulation for nearest point.
tsearch     - Closest triangle search.
convhull    - Convex hull.
voronoi     - Voronoi diagram.
inpolygon   - True for points inside polygonal region.
rectint     - Rectangle intersection area.
polyarea    - Area of polygon.
```

A.10.4 Polynomials.

```
roots       - Find polynomial roots.
poly        - Convert roots to polynomial.
polyval     - Evaluate polynomial.
polyvalm    - Evaluate polynomial with matrix argument.
residue     - Partial-fraction expansion (residues).
polyfit     - Fit polynomial to data.
polyder     - Differentiate polynomial.
conv        - Multiply polynomials.
deconv      - Divide polynomials.
```

A.11 Function functions and ODE solvers.

A.11.1 Optimization and root finding.

```
fmin     - Minimize function of one variable.
fmins    - Minimize function of several variables.
fzero    - Find zero of function of one variable.
```

A.11.2 Numerical integration (quadrature).

```
quad     - Numerically evaluate integral, low order method.
quad8    - Numerically evaluate integral, higher order method.
dblquad  - Numerically evaluate double integral.
```

A.11.3 Plotting.

```
ezplot   - Easy to use function plotter.
fplot    - Plot function.
```


A.11.4 Inline function object.

```
inline   - Construct INLINE object.
argnames - Argument names.
formula  - Function formula.
char     - Convert INLINE object to character array.
```

A.11.5 Ordinary differential equation solvers.

```
(If unsure about stiffness, try ODE45 first, then ODE15S.)
  ode45    - Solve non-stiff differential equations, medium order method.
  ode23    - Solve non-stiff differential equations, low order method.
  ode113   - Solve non-stiff differential equations, variable order method.
  ode15s   - Solve stiff differential equations, variable order method.
  ode23s   - Solve stiff differential equations, low order method.
  odefile  - ODE file syntax.
```

A.11.6 ODE Option handling.

```
odeset   - Create/alter ODE OPTIONS structure.
odeget   - Get ODE OPTIONS parameters.
```

A.11.7 ODE output functions.

```
odeplot  - Time series ODE output function.
odephas2 - 2-D phase plane ODE output function.
odephas3 - 3-D phase plane ODE output function.
odeprint - Command window printing ODE output function.
```

A.12 Sparse matrices.

A.12.1 Elementary sparse matrices.

```
speye       - Sparse identity matrix.
sprand      - Sparse uniformly distributed random matrix.
sprandn     - Sparse normally distributed random matrix.
sprandsym   - Sparse random symmetric matrix.
spdiags     - Sparse matrix formed from diagonals.
```

A.12.2 Full to sparse conversion.

```
sparse      - Create sparse matrix.
full        - Convert sparse matrix to full matrix.
find        - Find indices of nonzero elements.
spconvert   - Import from sparse matrix external format.
```

A.12.3 Working with sparse matrices.

```
nnz         - Number of nonzero matrix elements.
nonzeros    - Nonzero matrix elements.
nzmax       - Amount of storage allocated for nonzero matrix elements.
spones      - Replace nonzero sparse matrix elements with ones.
spalloc     - Allocate space for sparse matrix.
issparse    - True for sparse matrix.
spfun       - Apply function to nonzero matrix elements.
spy         - Visualize sparsity pattern.
```

A.12.4 Reordering algorithms.

```
colmmd      - Column minimum degree permutation.
symmmd      - Symmetric minimum degree permutation.
symrcm      - Symmetric reverse Cuthill-McKee permutation.
colperm     - Column permutation.
randperm    - Random permutation.
dmperm      - Dulmage-Mendelsohn permutation.
```

A.12.5 Linear algebra.

```
eigs        - A few eigenvalues.
svds        - A few singular values.
luinc       - Incomplete LU factorization.
cholinc     - Incomplete Cholesky factorization.
normest     - Estimate the matrix 2-norm.
condest     - 1-norm condition number estimate.
sprank      - Structural rank.
```

A.12.6 Linear Equations (iterative methods).

```
pcg         - Preconditioned Conjugate Gradients Method.
bicg        - BiConjugate Gradients Method.
bicgstab    - BiConjugate Gradients Stabilized Method.
cgs         - Conjugate Gradients Squared Method.
gmres       - Generalized Minimum Residual Method.
qmr         - Quasi-Minimal Residual Method.
```

A.12.7 Operations on graphs (trees).

```
treelayout  - Lay out tree or forest.
treeplot    - Plot picture of tree.
etree       - Elimination tree.
etreeplot   - Plot elimination tree.
gplot       - Plot graph, as in "graph theory".
```


A.12.8 Miscellaneous.

```
symbfact    - Symbolic factorization analysis.
spparms     - Set parameters for sparse matrix routines.
spaugment   - Form least squares augmented system.
```

A.13 Two dimensional graphs.

A.13.1 Elementary X-Y graphs.

```
plot      - Linear plot.
loglog    - Log-log scale plot.
semilogx  - Semi-log scale plot.
semilogy  - Semi-log scale plot.
polar     - Polar coordinate plot.
plotyy    - Graphs with y tick labels on the left and right.
```

A.13.2 Axis control.

```
axis       - Control axis scaling and appearance.
zoom       - Zoom in and out on a 2-D plot.
grid       - Grid lines.
box        - Axis box.
hold       - Hold current graph.
axes       - Create axes in arbitrary positions.
subplot    - Create axes in tiled positions.
```

A.13.3 Graph annotation.

```
legend    - Graph legend.
title     - Graph title.
xlabel    - X-axis label.
ylabel    - Y-axis label.
text      - Text annotation.
gtext     - Place text with mouse.
```

A.13.4 Hardcopy and printing.

```
print      - Print graph or SIMULINK system; or save graph to M-file.
printopt   - Printer defaults.
orient     - Set paper orientation.
```

A.14 Three dimensional graphs.

A.14.1 Elementary 3-D plots.

```
plot3      - Plot lines and points in 3-D space.
mesh       - 3-D mesh surface.
surf       - 3-D colored surface.
```

```
fill3      - Filled 3-D polygons.
```

A.14.2 Color control.

```
colormap   - Color look-up table.
caxis      - Pseudocolor axis scaling.
shading    - Color shading mode.
hidden     - Mesh hidden line removal mode.
brighten   - Brighten or darken color map.
```

A.14.3 Lighting.

```
surfl      - 3-D shaded surface with lighting.
lighting   - Lighting mode.
material   - Material reflectance mode.
specular   - Specular reflectance.
diffuse    - Diffuse reflectance.
surfnorm   - Surface normals.
```

A.14.4 Color maps.

```
hsv        - Hue-saturation-value color map.
hot        - Black-red-yellow-white color map.
gray       - Linear gray-scale color map.
bone       - Gray-scale with tinge of blue color map.
copper     - Linear copper-tone color map.
pink       - Pastel shades of pink color map.
white      - All white color map.
flag       - Alternating red, white, blue, and black color map.
lines      - Color map with the line colors.
colorcube  - Enhanced color-cube color map.
jet        - Variant of HSV.
prism      - Prism color map.
cool       - Shades of cyan and magenta color map.
autumn     - Shades of red and yellow color map.
spring     - Shades of magenta and yellow color map.
winter     - Shades of blue and green color map.
summer     - Shades of green and yellow color map.
```

A.14.5 Axis control.

```
axis       - Control axis scaling and appearance.
zoom       - Zoom in and out on a 2-D plot.
grid       - Grid lines.
box        - Axis box.
hold       - Hold current graph.
axes       - Create axes in arbitrary positions.
subplot    - Create axes in tiled positions.
```


A.14.6 Viewpoint control.

```
view       - 3-D graph viewpoint specification.
viewmtx    - View transformation matrix.
rotate3d   - Interactively rotate view of 3-D plot.
```

A.14.7 Graph annotation.

```
title      - Graph title.
xlabel     - X-axis label.
ylabel     - Y-axis label.
zlabel     - Z-axis label.
colorbar   - Display color bar (color scale).
text       - Text annotation.
gtext      - Mouse placement of text.
```

A.14.8 Hardcopy and printing.

```
print      - Print graph or SIMULINK system; or save graph to M-file.
printopt   - Printer defaults.
orient     - Set paper orientation.
```

A.15 Specialized graphs.

A.15.1 Specialized 2-D graphs.

```
area       - Filled area plot.
bar        - Bar graph.
barh       - Horizontal bar graph.
bar3       - 3-D bar graph.
bar3h      - Horizontal 3-D bar graph.
comet      - Comet-like trajectory.
errorbar   - Error bar plot.
ezplot     - Easy to use function plotter.
feather    - Feather plot.
fill       - Filled 2-D polygons.
fplot      - Plot function.
hist       - Histogram.
pareto     - Pareto chart.
pie        - Pie chart.
pie3       - 3-D pie chart.
plotmatrix - Scatter plot matrix.
ribbon     - Draw 2-D lines as ribbons in 3-D.
stem       - Discrete sequence or "stem" plot.
stairs     - Stairstep plot.
```

A.15.2 Contour and 2-1/2 D graphs.

```
contour    - Contour plot.
```

```
contourf   - Filled contour plot.
contour3   - 3-D Contour plot.
clabel     - Contour plot elevation labels.
pcolor     - Pseudocolor (checkerboard) plot.
quiver     - Quiver plot.
voronoi    - Voronoi diagram.
```

A.15.3 Specialized 3-D graphs.

```
comet3     - 3-D comet-like trajectories.
meshc      - Combination mesh/contour plot.
meshz      - 3-D mesh with curtain.
stem3      - 3-D stem plot.
quiver3    - 3-D quiver plot.
slice      - Volumetric slice plot.
surfc      - Combination surf/contour plot.
trisurf    - Triangular surface plot.
trimesh    - Triangular mesh plot.
waterfall  - Waterfall plot.
```

A.15.4 Images display and file I/O.

```
image      - Display image.
imagesc    - Scale data and display as image.
colormap   - Color look-up table.
gray       - Linear gray-scale color map.
contrast   - Gray scale color map to enhance image contrast.
brighten   - Brighten or darken color map.
colorbar   - Display color bar (color scale).
imread     - Read image from graphics file.
imwrite    - Write image to graphics file.
imfinfo    - Information about graphics file.
```

A.15.5 Movies and animation.

```
capture    - Screen capture of current figure.
moviein    - Initialize movie frame memory.
getframe   - Get movie frame.
movie      - Play recorded movie frames.
qtwrite    - Translate movie into QuickTime format (Macintosh only).
rotate     - Rotate object about specified orgin and direction.
frame2im   - Convert movie frame to indexed image.
im2frame   - Convert index image into movie format.
```

A.15.6 Color related functions.

```
spinmap    - Spin color map.
rgbplot    - Plot color map.
```



```
colstyle   - Parse color and style from string.
```

A.15.7 Solid modeling.

```
cylinder   - Generate cylinder.
sphere     - Generate sphere.
patch      - Create patch.
```

A.16 Handle Graphics.

A.16.1 Figure window creation and control.

```
figure     - Create figure window.
gcf        - Get handle to current figure.
clf        - Clear current figure.
shg        - Show graph window.
close      - Close figure.
refresh    - Refresh figure.
```

A.16.2 Axis creation and control.

```
subplot    - Create axes in tiled positions.
axes       - Create axes in arbitrary positions.
gca        - Get handle to current axes.
cla        - Clear current axes.
axis       - Control axis scaling and appearance.
box        - Axis box.
caxis      - Control pseudocolor axis scaling.
hold       - Hold current graph.
ishold     - Return hold state.
```

A.16.3 Handle Graphics objects.

```
figure     - Create figure window.
axes       - Create axes.
line       - Create line.
text       - Create text.
patch      - Create patch.
surface    - Create surface.
image      - Create image.
light      - Create light.
uicontrol  - Create user interface control.
uimenu     - Create user interface menu.
```

A.16.4 Handle Graphics operations.

```
set        - Set object properties.
get        - Get object properties.
reset      - Reset object properties.
```

```
delete     - Delete object.
gco        - Get handle to current object.
gcbo       - Get handle to current callback object.
gcbf       - Get handle to current callback figure.
drawnow    - Flush pending graphics events.
findobj    - Find objects with specified property values.
copyobj    - Make copy of graphics object and its children.
```

A.16.5 Hardcopy and printing.

```
print      - Print graph or SIMULINK system; or save graph to M-file.
printopt   - Printer defaults.
orient     - Set paper orientation.
```

A.16.6 Utilities.

```
closereq   - Figure close request function.
newplot    - M-file preamble for NextPlot property.
ishandle   - True for graphics handles.
```

A.17 Graphical user interface tools.

A.17.1 GUI functions.

```
uicontrol  - Create user interface control.
uimenu     - Create user interface menu.
ginput     - Graphical input from mouse.
dragrect   - Drag XOR rectangles with mouse.
rbbox      - Rubberband box.
selectmoveresize   - Interactively select, move, resize, or copy objects.
waitforbuttonpress - Wait for key/buttonpress over figure.
waitfor    - Block execution and wait for event.
uiwait     - Block execution and wait for resume.
uiresume   - Resume execution of blocked M-file.
```

A.17.2 GUI design tools.

```
guide       - Design GUI.
align       - Align uicontrols and axes.
cbedit      - Edit callback.
menuedit    - Edit menu.
propedit    - Edit property.
```

A.17.3 Dialog boxes.

```
dialog     - Create dialog figure.
axlimdlg   - Axes limits dialog box.
errordlg   - Error dialog box.
helpdlg    - Help dialog box.
```

```
inputdlg   - Input dialog box.
listdlg    - List selection dialog box.
menu       - Generate menu of choices for user input.
msgbox     - Message box.
questdlg   - Question dialog box.
warndlg    - Warning dialog box.
uigetfile  - Standard open file dialog box.
uiputfile  - Standard save file dialog box.
uisetcolor - Color selection dialog box.
uisetfont  - Font selection dialog box.
pagedlg    - Page position dialog box.
printdlg   - Print dialog box.
waitbar    - Display wait bar.
```

A.17.4 Menu utilities.

```
makemenu   - Create menu structure.
menubar    - Computer dependent default setting for MenuBar property.
umtoggle   - Toggle "checked" status of uimenu object.
winmenu    - Create submenu for "Window" menu item.
```

A.17.5 Toolbar button group utilities.

```
btngroup   - Create toolbar button group.
btnstate   - Query state of toolbar button group.
btnpress   - Button press manager for toolbar button group.
btndown    - Depress button in toolbar button group.
btnup      - Raise button in toolbar button group.
```

A.17.6 User-defined figure/axes property utilities.

```
clruprop   - Clear user-defined property.
getuprop   - Get value of user-defined property.
setuprop   - Set user-defined property.
```

A.17.7 Miscellaneous utilities.

```
allchild   - Get all object children.
hidegui    - Hide/unhide GUI.
edtext     - Interactive editing of axes text objects.
getstatus  - Get status text string in figure.
setstatus  - Set status text string in figure.
popupstr   - Get popup menu selection string.
remapfig   - Transform figure objects positions.
setptr     - Set figure pointer.
getptr     - Get figure pointer.
overobj    - Get handle of object the pointer is over.
```

A.18 Character strings.

A.18.1 General.

```
char        - Create character array (string).
double      - Convert string to numeric character codes.
cellstr     - Create cell array of strings from character array.
blanks      - String of blanks.
deblank     - Remove trailing blanks.
eval        - Execute string with MATLAB expression.
```

A.18.2 String tests.

```
ischar      - True for character array (string).
iscellstr   - True for cell array of strings.
isletter    - True for letters of the alphabet.
isspace     - True for white space characters.
```

A.18.3 String operations.

```
strcat      - Concatenate strings.
strvcat     - Vertically concatenate strings.
strcmp      - Compare strings.
strncmp     - Compare first N characters of strings.
findstr     - Find one string within another.
strjust     - Justify character array.
strmatch    - Find possible matches for string.
strrep      - Replace string with another.
strtok      - Find token in string.
upper       - Convert string to uppercase.
lower       - Convert string to lowercase.
```

A.18.4 String to number conversion.

```
num2str     - Convert number to string.
int2str     - Convert integer to string.
mat2str     - Convert matrix to evalable string.
str2num     - Convert string to number.
sprintf     - Write formatted data to string.
sscanf      - Read string under format control.
```

A.18.5 Base number conversion.

```
hex2num     - Convert IEEE hexadecimal to double precision number.
hex2dec     - Convert hexadecimal string to decimal integer.
dec2hex     - Convert decimal integer to hexadecimal string.
bin2dec     - Convert binary string to decimal integer.
dec2bin     - Convert decimal integer to binary string.
base2dec    - Convert base B string to decimal integer.
```



```
dec2base    - Convert decimal integer to base B string.
```

A.19 File input/output.

A.19.1 File opening and closing.

```
fopen       - Open file.
fclose      - Close file.
```

A.19.2 Binary file I/O.

```
fread       - Read binary data from file.
fwrite      - Write binary data to file.
```

A.19.3 Formatted file I/O.

```
fscanf      - Read formatted data from file.
fprintf     - Write formatted data to file.
fgetl       - Read line from file, discard newline character.
fgets       - Read line from file, keep newline character.
input       - Prompt for user input.
```

A.19.4 String conversion.

```
sprintf     - Write formatted data to string.
sscanf      - Read string under format control.
```

A.19.5 File positioning.

```
ferror      - Inquire file error status.
feof        - Test for end-of-file.
fseek       - Set file position indicator.
ftell       - Get file position indicator.
frewind     - Rewind file.
```

A.19.6 File name handling

```
matlabroot  - Root directory of MATLAB installation.
filesep     - Directory separator for this platform.
pathsep     - Path separator for this platform.
mexext      - MEX filename extension for this platform.
fullfile    - Build full filename from parts.
partialpath - Partial pathnames.
tempdir     - Get temporary directory.
tempname    - Get temporary file.
```

A.19.7 File import/export functions.

```
load         - Load workspace from MAT-file.
save         - Save workspace to MAT-file.
dlmread      - Read ASCII delimited file.
dlmwrite     - Write ASCII delimited file.
wk1read      - Read spreadsheet (WK1) file.
wk1write     - Write spreadsheet (WK1) file.
```

A.19.8 Image file import/export.

```
imread       - Read image from graphics file.
imwrite      - Write image to graphics file.
imfinfo      - Return information about graphics file.
```

A.19.9 Audio file import/export.

```
auwrite      - Write NeXT/SUN (".au") sound file.
auread       - Read NeXT/SUN (".au") sound file.
wavwrite     - Write Microsoft WAVE (".wav") sound file.
wavread      - Read Microsoft WAVE (".wav") sound file.
```

A.19.10 Command window I/O

```
clc          - Clear command window.
home         - Send cursor home.
disp         - Display array.
input        - Prompt for user input.
pause        - Wait for user response.
```

A.20 Time and dates.

A.20.1 Current date and time.

```
now          - Current date and time as date number.
date         - Current date as date string.
clock        - Current date and time as date vector.
```

A.20.2 Basic functions.

```
datenum      - Serial date number.
datestr      - String representation of date.
datevec      - Date components.
```

A.20.3 Date functions.

```
calendar     - Calendar.
weekday      - Day of week.
eomday       - End of month.
datetick     - Date formatted tick labels.
```


A.20.4 Timing functions.

```
cputime     - CPU time in seconds.
tic, toc    - Stopwatch timer.
etime       - Elapsed time.
pause       - Wait in seconds.
```

A.21 Data types and structures.

A.21.1 Data types (classes)

```
double      - Convert to double precision.
sparse      - Create sparse matrix.
char        - Create character array (string).
cell        - Create cell array.
struct      - Create or convert to structure array.
uint8       - Convert to unsigned 8-bit integer.
inline      - Construct INLINE object.
```

A.21.2 Multi-dimensional array functions.

```
cat         - Concatenate arrays.
ndims       - Number of dimensions.
ndgrid      - Generate arrays for N-D functions and interpolation.
permute     - Permute array dimensions.
ipermute    - Inverse permute array dimensions.
shiftdim    - Shift dimensions.
squeeze     - Remove singleton dimensions.
```

A.21.3 Cell array functions.

```
cell        - Create cell array.
celldisp    - Display cell array contents.
cellplot    - Display graphical depiction of cell array.
num2cell    - Convert numeric array into cell array.
deal        - Deal inputs to outputs.
cell2struct - Convert cell array into structure array.
struct2cell - Convert structure array into cell array.
iscell      - True for cell array.
```

A.21.4 Structure functions.

```
struct      - Create or convert to structure array.
fieldnames  - Get structure field names.
getfield    - Get structure field contents.
setfield    - Set structure field contents.
rmfield     - Remove structure field.
isfield     - True if field is in structure array.
isstruct    - True for structures.
```

A.21.5 Object oriented programming functions.

```
class       - Create object or return object class.
struct      - Convert object to structure array.
methods     - Display class method names.
isa         - True if object is a given class.
isobject    - True for objects.
inferiorto  - Inferior class relationship.
superiorto  - Superior class relationship.
```

A.21.6 Overloadable operators.

```
minus       - Overloadable method for a-b.
plus        - Overloadable method for a+b.
times       - Overloadable method for a.*b.
mtimes      - Overloadable method for a*b.
mldivide    - Overloadable method for a\b.
mrdivide    - Overloadable method for a/b.
rdivide     - Overloadable method for a./b.
ldivide     - Overloadable method for a.\b.
power       - Overloadable method for a.^b.
mpower      - Overloadable method for a^b.
uminus      - Overloadable method for -a.
uplus       - Overloadable method for +a.
horzcat     - Overloadable method for [a b].
vertcat     - Overloadable method for [a;b].
le          - Overloadable method for a<=b.
lt          - Overloadable method for a<b.
gt          - Overloadable method for a>b.
ge          - Overloadable method for a>=b.
eq          - Overloadable method for a==b.
ne          - Overloadable method for a~=b.
not         - Overloadable method for ~a.
and         - Overloadable method for a&b.
or          - Overloadable method for a|b.
subsasgn    - Overloadable method for a(i)=b, a{i}=b, and a.field=b.
subsref     - Overloadable method for a(i), a{i}, and a.field.
colon       - Overloadable method for a:b.
transpose   - Overloadable method for a.
ctranspose  - Overloadable method for a
subsindex   - Overloadable method for x(a).
```

A.22 Dynamic data exchange (DDE).

A.22.1 DDE Client Functions.

```
ddeadv   - Set up advisory link.
ddeexec  - Send string for execution.
```



```
ddeinit  - Initiate DDE conversation.
ddepoke  - Send data to application.
ddereq   - Request data from application.
ddeterm  - Terminate DDE conversation.
ddeunadv - Release advisory link.
```

A.23 Examples and demonstrations.

A.23.1 MATLAB/Introduction.

```
demo        - Browse demos for MATLAB, Toolboxes, and SIMULINK
```

A.23.2 MATLAB/Matrices.

```
intro       - Introduction to basic matrix operations in MATLAB.
inverter    - Demonstrate the inversion of a matrix.
buckydem    - Connectivity graph of the Buckminster Fuller geodesic dome.
sparsity    - Demonstrate effect of sparsity orderings.
matmanip    - Introduction to matrix manipulation.
eigmovie    - Symmetric eigenvalue movie.
rrefmovie   - Computation of Reduced Row Echelon Form.
delsqdemo   - Finite difference Laplacian on various domains.
sepdemo     - Separators for a finite element mesh.
airfoil     - Display sparse matrix from NASA airfoil.
```

A.23.3 MATLAB/Numerics.

```
funfuns     - Demonstrate functions that operate on other functions.
fitdemo     - Nonlinear curve fit with simplex algorithm.
sunspots    - FFT: the answer is 11.08, what is the question?
e2pi        - 2D visual solutions: Which is greater, e^pi or pi^e?
bench       - MATLAB Benchmark.
fftdemo     - Use of the fast finite Fourier transform.
census      - Try to predict the US population in the year 2000.
spline2d    - Demonstrate GINPUT and SPLINE in two dimensions.
lotkademo   - An example of ordinary differential equation solution.
quaddemo    - Adaptive quadrature.
zerodemo    - Zerofinding with fzero.
fplotdemo   - Plot a function.
quake       - Loma Prieta Earthquake.
```

A.23.4 MATLAB/Visualization.

```
graf2d      - 2D Plots: Demonstrate XY plots in MATLAB.
graf2d2     - 3D Plots: Demonstrate XYZ plots in MATLAB.
grafcplx    - Demonstrate complex function plots in MATLAB.
lorenz      - Plot the orbit around the Lorenz chaotic attractor.
imageext    - Image colormaps: changing and rotating colormaps.
```

```
xpklein     - Klein bottle demo.
vibes       - Vibration movie: Vibrating L-shaped membrane.
xpsound     - Visualizing sound: Demonstrate MATLABs sound capability.
imagedemo   - Demonstrate MATLABs image capability.
penny       - Several views of the penny data.
earthmap    - View Earths topography.
xfourier    - Graphic demo of Fourier series expansion.
colormenu   - Select color map.
cplxdemo    - Maps of functions of a complex variable.
```

A.23.5 MATLAB/Language.

```
xplang      - Introduction to the MATLAB language.
hndlgraf    - Demonstrate Handle Graphics for line plots.
graf3d      - Demonstrate Handle Graphics for surface plots.
hndlaxis    - Demonstrate Handle Graphics for axes.
```

A.23.6 MATLAB/ODE Suite.

```
odedemo     - Demo for the ODE suite integrators.
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).
ballode     - Equations of motion for a bouncing ball used by BALLDEMO.
besslode    - Bessels equation of order 0 used by BESSLDEMO.
brussode    - Stiff problem modelling a chemical reaction (Brusselator).
buiode      - Stiff problem with analytical solution due to Bui.
chm6ode     - Stiff problem CHM6 from Enright and Hull.
chm7ode     - Stiff problem CHM7 from Enright and Hull.
chm9ode     - Stiff problem CHM9 from Enright and Hull.
d1ode       - Stiff problem, nonlinear with real eigenvalues (D1 of EHL).
fem1ode     - Stiff problem with a time-dependent mass matrix.
fem2ode     - Stiff problem with a time-independent mass matrix.
gearode     - Stiff problem due to Gear as quoted by van der Houwen.
hb1ode      - Stiff problem 1 of Hindmarsh and Byrne.
hb2ode      - Stiff problem 2 of Hindmarsh and Byrne.
hb3ode      - Stiff problem 3 of Hindmarsh and Byrne.
orbitode    - Restricted 3 body problem used by ORBITDEMO.
orbt2ode    - Non-stiff problem D5 of Hull et al.
rigidode    - Euler equations of a rigid body without external forces.
sticode     - A spring-driven mass stuck to surface, used by STICDEMO.
vdpode      - Parameterizable van der Pol equation (stiff for large mu).
```

A.23.7 Extras/Gallery.

```
knot        - Tube surrounding a three-dimensional knot.
quivdemo    - Demonstrate the quiver function.
klein1      - Construct a Klein bottle.
```



```
cruller     - Construct cruller.
tori4       - Hoops: Construct four linked tori.
spharm2     - Construct spherical surface harmonic.
modes       - Plot 12 modes of the L-shaped membrane.
logo        - Display the MATLAB L-shaped membrane logo.
```

A.23.8 Extras/Games.

```
xpbombs     - Minesweeper game.
life        - Conways Game of Life.
bblwrap     - Bubblewrap.
soma        - Soma cube
```

A.23.9 Extras/Miscellaneous.

```
truss       - Animation of a bending bridge truss.
travel      - Traveling salesman problem.
spinner     - Colorful lines spinning through space.
xpquad      - Superquadrics plotting demonstration.
codec       - Alphabet transposition coder/decoder.
xphide      - Visual perception of objects in motion.
makevase    - Generate and plot a surface of revolution.
wrldtrv     - Great circle flight routes around the globe.
logospin    - Movie of The MathWorks logo spinning.
crulspin    - Spinning cruller movie.
```

A.23.10 General Demo/Helper functions.

```
cmdlnwin    - An Demo gateway routine for playing command line demos.
cmdlnbgn    - Set up for command line demos.
cmdlnend    - clean up after command line demos.
finddemo    - Finds demos available for individual toolboxes.
helpfun     - Utility function for displaying help text conveniently.
pltmat      - Display a matrix in a figure window.
```

A.23.11 MATLAB/Helper functions.

```
bucky       - The graph of the Buckminster Fuller geodesic dome.
peaks       - A sample function of two variables.
membrane    - Generate MathWorkss logo.
```

Appendice B

Proprietà degli oggetti di Matlab

B.1 root

```
Language
CurrentFigure
Diary: [ on | off ]
DiaryFile
Echo: [ on | off ]
ErrorMessage
Format: [ short | long | shortE | longE | shortG | longG | hex |
          bank | + | rational ]
FormatSpacing: [ loose | compact ]
PointerLocation
Profile: [ on | off ]
ProfileFile
ProfileCount
ProfileInterval
ScreenDepth
ShowHiddenHandles: [ on | {off} ]
Units: [ inches | centimeters | normalized | points | pixels ]
AutomaticFileUpdates: [ on | off ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
```



```
Visible: [ {on} | off ]
```

B.2 figure

```
BackingStore: [ {on} | off ]
CloseRequestFcn
Color
Colormap
CurrentAxes
CurrentObject
CurrentPoint
Dithermap
DithermapMode: [ auto | {manual} ]
IntegerHandle: [ {on} | off ]
InvertHardcopy: [ {on} | off ]
KeyPressFcn
MenuBar: [ none | {figure} ]
MinColormap
Name
NextPlot: [ {add} | replace | replacechildren ]
NumberTitle: [ {on} | off ]
PaperUnits: [ {inches} | centimeters | normalized | points ]
PaperOrientation: [ {portrait} | landscape ]
PaperPosition
PaperPositionMode: [ auto | {manual} ]
PaperType: [ {usletter} | uslegal | a3 | a4letter | a5 | b4 | tabloid ]
Pointer: [ crosshair | fullcrosshair | {arrow} | ibeam | watch |
           topl | topr | botl | botr | left | top | right |
                   bottom | circle | cross | fleur | custom ]
PointerShapeCData
PointerShapeHotSpot
Position
Renderer: [ {painters} | zbuffer ]
RendererMode: [ {auto} | manual ]
Resize: [ {on} | off ]
ResizeFcn
ShareColors: [ {on} | off ]
Units: [ inches | centimeters | normalized | points | {pixels} ]
WindowButtonDownFcn
WindowButtonMotionFcn
WindowButtonUpFcn
WindowStyle: [ {normal} | modal ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
```

```
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.3 axes

```
AmbientLightColor
Box: [ on | {off} ]
CameraPosition
CameraPositionMode: [ {auto} | manual ]
CameraTarget
CameraTargetMode: [ {auto} | manual ]
CameraUpVector
CameraUpVectorMode: [ {auto} | manual ]
CameraViewAngle
CameraViewAngleMode: [ {auto} | manual ]
CLim
CLimMode: [ {auto} | manual ]
Color
ColorOrder
DataAspectRatio
DataAspectRatioMode: [ {auto} | manual ]
DrawMode: [ {normal} | fast ]
FontAngle: [ {normal} | italic | oblique ]
FontName
FontSize
FontUnits: [ inches | centimeters | normalized | {points} | pixels ]
FontWeight: [ light | {normal} | demi | bold ]
GridLineStyle: [ - | -- | {:} | -. | none ]
Layer: [ top | {bottom} ]
LineStyleOrder
LineWidth
MinorGridLineStyle: [ - | -- | {:} | -. | none ]
NextPlot: [ add | {replace} | replacechildren ]
PlotBoxAspectRatio
PlotBoxAspectRatioMode: [ {auto} | manual ]
Projection: [ {orthographic} | perspective ]
Position
TickLength
TickDir: [ {in} | out ]
```



```
TickDirMode: [ {auto} | manual ]
Title
Units: [ inches | centimeters | {normalized} | points | pixels ]
View
XColor
XDir: [ {normal} | reverse ]
XGrid: [ on | {off} ]
XLabel
XAxisLocation: [ top | {bottom} ]
XLim
XLimMode: [ {auto} | manual ]
XScale: [ {linear} | log ]
XTick
XTickLabel
XTickLabelMode: [ {auto} | manual ]
XTickMode: [ {auto} | manual ]
YColor
YDir: [ {normal} | reverse ]
YGrid: [ on | {off} ]
YLabel
YAxisLocation: [ {left} | right ]
YLim
YLimMode: [ {auto} | manual ]
YScale: [ {linear} | log ]
YTick
YTickLabel
YTickLabelMode: [ {auto} | manual ]
YTickMode: [ {auto} | manual ]
ZColor
ZDir: [ {normal} | reverse ]
ZGrid: [ on | {off} ]
ZLabel
ZLim
ZLimMode: [ {auto} | manual ]
ZScale: [ {linear} | log ]
ZTick
ZTickLabel
ZTickLabelMode: [ {auto} | manual ]
ZTickMode: [ {auto} | manual ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
```

```
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.4 line

```
Color
EraseMode: [ {normal} | background | xor | none ]
LineStyle: [ {-} | -- | : | -. | none ]
LineWidth
Marker: [ + | o | * | . | x | square | diamond | v | ^ | > | < |
          pentagram | hexagram | {none} ]
MarkerSize
MarkerEdgeColor: [ none | {auto} ] -or- a ColorSpec.
MarkerFaceColor: [ {none} | auto ] -or- a ColorSpec.
XData
YData
ZData

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.5 patch

```
CData
CDataMapping: [ direct | {scaled} ]
FaceVertexCData
EdgeColor: [ none | flat | interp ] -or- {a ColorSpec}.
EraseMode: [ {normal} | background | xor | none ]
FaceColor: [ none | flat | interp ] -or- {a ColorSpec}.
Faces
```



```
LineStyle: [ {-} | -- | : | -. | none ]
LineWidth
Marker: [ + | o | * | . | x | square | diamond | v | ^ | > | < |
          pentagram | hexagram | {none} ]
MarkerEdgeColor: [ none | {auto} | flat ] -or- a ColorSpec.
MarkerFaceColor: [ {none} | auto | flat ] -or- a ColorSpec.
MarkerSize
Vertices
XData
YData
ZData
FaceLighting: [ none | {flat} | gouraud | phong ]
EdgeLighting: [ {none} | flat | gouraud | phong ]
BackFaceLighting: [ unlit | lit | {reverselit} ]
AmbientStrength
DiffuseStrength
SpecularStrength
SpecularExponent
SpecularColorReflectance
VertexNormals
NormalMode: [ {auto} | manual ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.6 surface

```
CData
CDataMapping: [ direct | {scaled} ]
EdgeColor: [ none | flat | interp ] -or- {a ColorSpec}.
EraseMode: [ {normal} | background | xor | none ]
FaceColor: [ none | {flat} | interp | texturemap ] -or- a ColorSpec.
LineStyle: [ {-} | -- | : | -. | none ]
LineWidth
Marker: [ + | o | * | . | x | square | diamond | v | ^ | > | < |
```

```
          pentagram | hexagram | {none} ]
MarkerEdgeColor: [ none | {auto} | flat ] -or- a ColorSpec.
MarkerFaceColor: [ {none} | auto | flat ] -or- a ColorSpec.
MarkerSize
MeshStyle: [ {both} | row | column ]
XData
YData
ZData
FaceLighting: [ none | {flat} | gouraud | phong ]
EdgeLighting: [ {none} | flat | gouraud | phong ]
BackFaceLighting: [ unlit | lit | {reverselit} ]
AmbientStrength
DiffuseStrength
SpecularStrength
SpecularExponent
SpecularColorReflectance
VertexNormals
NormalMode: [ {auto} | manual ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.7 image

```
CData
CDataMapping: [ {direct} | scaled ]
EraseMode: [ {normal} | background | xor | none ]
XData
YData

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
```



```
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.8 text

```
Color
EraseMode: [ {normal} | background | xor | none ]
Editing: [ on | off ]
FontAngle: [ {normal} | italic | oblique ]
FontName
FontSize
FontUnits: [ inches | centimeters | normalized | {points} | pixels ]
FontWeight: [ light | {normal} | demi | bold ]
HorizontalAlignment: [ {left} | center | right ]
Position
Rotation
String
Units: [ inches | centimeters | normalized | points | pixels | {data} ]
Interpreter: [ {tex} | none ]
VerticalAlignment: [ top | cap | {middle} | baseline | bottom ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.9 light

```
Position
Color
```

```
Style: [ {infinite} | local ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.10 uicontrol

```
BackgroundColor
Callback
Enable: [ {on} | off | inactive ]
FontAngle: [ {normal} | italic | oblique ]
FontName
FontSize
FontUnits: [ inches | centimeters | normalized | {points} | pixels ]
FontWeight: [ light | {normal} | demi | bold ]
ForegroundColor
HorizontalAlignment: [ left | {center} | right ]
ListboxTop
Max
Min
Position
String
Style: [ {pushbutton} | radiobutton | checkbox | edit | text |
          slider | frame | listbox | popupmenu ]
SliderStep
Units: [ inches | centimeters | normalized | points | {pixels} ]
Value

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
```



```
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```

B.11 uimenu

```
Accelerator
Callback
Checked: [ on | {off} ]
Enable: [ {on} | off ]
ForegroundColor
Label
Position
Separator: [ on | {off} ]

ButtonDownFcn
Children
Clipping: [ {on} | off ]
CreateFcn
DeleteFcn
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UserData
Visible: [ {on} | off ]
```