IL PROGRAMMA MATLAB (ver 5.2)

Introduzione

In queste pagine si introduce in maniera molto breve il programma di simulazione MATLAB (una abbreviazione di MATrix LABoratory). La prima versione di questo programma risale agli inizi degli anni 80 ed è stata sviluppata come sottoprodotto dei progetti LINPACK ed EISPACK, da The MathWorks Inc. Qui si descriverà in particolare la versione 5.2 (per sapere quale versione di MATLAB si sta usando, digitare `ver` sulla linea di comandi o prompt del MATLAB che è rappresentato da un doppio simbolo di maggiore: »). Le principali differenze fra le versioni consistono, comunque, nel numero massimo di elementi per vettore, nella sintassi di alcuni comandi e nel linguaggio con cui è stato sviluppato il software. E' anche cambiata la filosofia del programma: mentre nelle prime versioni MATLAB appariva all'utente come un programma unico che eventualmente poteva chiamare procedure esterne, ora le funzionalità di calcolo, grafiche, ecc., del programma possono essere richiamate anche da programmi esterni.
All'utente medio MATLAB appare comunque come un ambiente integrato di calcolo, è infatti principalmente un programma interattivo di calcolo che permette di risolvere problemi numerici senza che sia necessario scrivere esplicitamente una procedura in un linguaggio di programmazione ad alto livello. MATLAB si basa sul calcolo matriciale, e si può pensare ad esso come ad un sistema efficace per accedere ad una libreria di procedure di calcolo numerico molto sofisticate, con in più la possibilità di rappresentare graficamente i risultati. Ha una interfaccia molto semplice, ed un singolo comando permette di risolvere problemi come l'inversione di una matrice o la soluzione di un sistema di equazioni differenziali. Inoltre, una successione di comandi può essere raccolta in un file (che deve avere estensione **.m**) ed essere eseguita invocando, dal prompt di comandi del programma MATLAB, il nome del file stesso; è poi possibile definire delle funzioni che forniscono dei valori in uscita, di modo che MATLAB risulta esso stesso programmabile e fornisce la possibilità di effettuare simulazioni e calcoli, anche molto complessi, in modo semplice e diretto.

Una guida completa a MATLAB è al di là degli scopi di queste note: si ricorda comunque che MATLAB fornisce un sistema di aiuto in linea a cui è molto conveniente ricorrere. Per richiamare l'help in linea di MATLAB sono disponibili tre diversi comandi:

`help`: mostra sulla linea di comando l'elenco delle procedure disponibili, per avere poi una descrizione di tali funzioni è necessario digitare `help` *nome_comando* (ad es. digitando `help exp` si ottengono informazioni sulla funzione `exp`, per il calcolo della funzione esponenziale, fornita da MATLAB e sulla sua sintassi);

`helpwin`: apre una finestra separata per navigare tra le procedure disponibili e le relative descrizioni (in questo caso per avere informazioni sulla funzione `exp` si seleziona il gruppo `elfun` (funzioni elementari) e quindi `exp`)

`helpdesk`: apre una finestra di ipertesto per la guida in linea e la risoluzione dei problemi.

Per una dimostrazione delle potenzialitá di MATLAB, si dia il comando `demo`.

N.B. Nel seguito verranno introdotte numerose funzioni di MATLAB, spesso senza molti dettagli. Per maggiori informazioni si consiglia sempre di vedere l'help ad esse associato (`help` *`nome_funzione`*)

Variabili in MATLAB

L'operazione di assegnazione di un valore ad una variabile avviene direttamente con l'operatore =
Il nome di una variabile può essere una qualsiasi combinazione di al massimo 19 caratteri alfanumerici minuscoli e maiuscoli (nota che **a≠A**). E' opportuno non utilizzare i nomi delle funzioni predefinite da MATLAB, ciò comporta l'annullamento del significato prestabilito da MATLAB per quella funzione e di conseguenza un messaggio di errore. **N.B.** Tutte le funzioni predefinite di MATLAB vanno richiamate con lettere minuscole (anche se sull'help compaiono in maiuscolo!).

Esempio:

```
» exp(2)      ↵  (questo simbolo indica l'invio)
ans =
    7.3891        (la funzione exp restituisce e^2)
» exp=1;      ↵  (assegno ad una variabile di nome exp il valore 1)
» exp(2)      ↵  (exp non è più una funzione ma una variabile di valore 1)
???  Index exceeds matrix dimensions.
»
```

Nota: se non si mette il punto e virgola (**;**) alla fine dell'istruzione, MATLAB fa l'eco, cioè si ottiene:

```
» a=1    ↵
a =
     1
»
```

La possibilità di sopprimere l'eco (terminando le istruzioni con ;) è particolarmente utile quando si scrivono programmi che contengono sequenze di comandi, oppure quando si lavora con matrici particolarmente grandi.

`Who` restituisce l'elenco delle variabili in memoria
`whos` fornisce informazioni più dettagliate per ogni variabile
`clear` ripulisce dalla memoria le variabili e le funzioni (vedi `help clear` per maggiori dettagli)

Ci sono delle variabili pre-memorizzate, le più usate sono:

eps	rappresenta l'accuratezza
inf	infinito (risultato di una divisione per 0)
NaN	Not A Number (risultato di 0/0)
pi	pi-greco
i	unità immaginaria (anche indicata con j)

Ad una variabile può anche essere assegnato il risultato di una espressione. Se in un'espressione si vuole andare a capo è necessario mettere "..." prima di procedere sulla nuova riga. Gli operatori hanno logicamente le solite priorità e le parentesi si usano anche qui per cambiare le priorità.

Matrici in MATLAB

MATLAB lavora essenzialmente con un unico tipo di dati: le matrici. Queste sono vettori bidimensionali, in generale con *m* righe e *n* colonne, i cui singoli elementi sono variabili reali oppure complesse. Una matrice può essere introdotta in quattro diversi modi:
- introducendo esplicitamente i suoi valori mediante un comando di assegnazione
- come risultato di comandi o funzioni predefiniti
- come risultato di un programma (m-file)
- caricata da file esterni (file di dati che hanno estensione **.mat**)

L'assegnazione diretta dal prompt dei comandi è:

```
» a=[1 2 3; 4 5 6];  ↵
»
```

Abbiamo così creato la matrice `a` di dimensioni 2×3.
Quindi gli elementi della matrice sono racchiusi tra parentesi quadre, gli elementi della stessa riga sono poi separati da uno spazio, ma si può anche usare una virgola; gli elementi di una riga della matrice da quelli della riga successiva sono separati da un punto e virgola o da un invio (cioè assegnazione su righe diverse).
Ad es. si poteva anche scrivere:

```
» a=[1,2,3   ↵ (Nota: non ritorna il prompt)
4 5 6]
a =
     1     2     3
     4     5     6
»
```

La stessa matrice `a` poteva anche essere creata tramite un programma. Si poteva infatti, ad esempio, creare un file **genera.m** costituito dalle seguenti righe:

```
a=[1 2 3;
   4 5 6]
```

Quindi per creare la matrice è sufficiente digitare dal prompt dei comandi il nome del file senza estensione:

```
» genera  ↵
```

A questo punto, dato che nel file non abbiamo finito l'istruzione con ";" si ottiene

```
a =
     1     2     3
     4     5     6
»
```

Se ci fosse stato il ";" MATLAB avrebbe restituito il solo prompt e per vedere la matrice sarebbe stato necessario digitare il suo nome (`a`).

Nota: per i nomi dei file vale lo stesso che per il nome delle variabili (cioè al massimo 19 caratteri alfanumerici minuscoli e maiuscoli).

Per salvare dati in un file l'istruzione è

```
save nome_file variabile1 variabile2 ...
```

Così nel file *nome_file.mat* (nota: l'estensione è automaticamente **.mat**) si salvano le variabili *variabile1, variabile2*, ...e il formato di salvataggio può essere scelto (vedi `help save`). L'istruzione per il recupero dei dati è poi

```
load nome_file
```

Elementi delle matrici

Gli elementi di una matrice possono essere direttamente numeri ma anche una qualsiasi espressione di MATLAB
Ad es.

```
» x=[-1.3 sqrt(3) (1+2+3)*4/5] ↵
x =
   -1.3000    1.7321    4.8000
»
```

Nota: I dati possono essere visualizzati in diversi formati. Per default si hanno 4 decimali in virgola fissa, per avere altre visualizzazioni si deve usare la funzione `format` (vedi `help format`)

La variabile `x` dichiarata (e contemporaneamente assegnata) nell'esempio precedente può essere vista come una matrice 1×3 o, analogamente, come un vettore riga di 3 elementi. Ogni elemento di x può essere letto e scritto indipendentemente dagli altri. L'accesso avviene tramite selezione con indice (**Nota**: gli indici partono sempre da **1**). Quindi scrivendo sulla linea comando `x(1,1)` (se vista come matrice) o equivalentemente `x(1)` (se visto come vettore riga) ottengo -1.3. Nelle matrici il primo indice indica la *riga* il secondo la *colonna*. La risposta ad una richiesta di valore di una certa variabile o di una funzione viene sempre memorizzata nella variabile `ans` (answer=risposta), quindi `ans` in generale è la risposta di MATLAB quando manca l'assegnazione ad una variabile.
La variabile `ans` può essere molto utile anche come variabile intermedia
Es.:

```
» x(2)     ↵
ans =
    1.7321
» ans*ans ↵
ans =
    3.0000
»
```

In MATLAB si ha l'allocazione dinamica delle variabili e quindi il dimensionamento dei vettori o delle matrici avviene automaticamente
Es:

```
» x=[-1 2 3];     ↵
» x(5)=abs(x(1)) ↵
x =
    -1     2     3     0     1
» x(2,1)=3        ↵
x =
    -1     2     3     0     1
     3     0     0     0     0
»
```

(il vettore ha assunto dimensione 5, con uno zero come quarto elemento)

(il vettore diviene una matrice 2×5, dove non ci sono assegnazioni esplicite si ha l'aggiunta automatica di zeri)

Nota: si consiglia di utilizzare questa possibilità con somma cautela

Nota: per rendere più veloci le simulazioni talvolta può essere conveniente predisporre all'inizio delle iterazioni vettori nulli della lunghezza massima necessaria (vedi `help zeros`, simile è anche la funzione `ones` che pone tutti gli elementi uguali a 1).

`size` fornisce la dimensione della matrice
`end` indica sempre l'ultimo elemento di un vettore
`find` ritorna gli indici degli elementi non nulli di un vettore

Una matrice si può creare accostando più matrici piccole, e analogamente una matrice si può estrarre da una più grande

```
» A=[1 2 3; 4 5 6]; ↵
» B=[7 8 9];        ↵
» A=[A;B]           ↵
A =
     1     2     3
     4     5     6
     7     8     9
» C=A(2,:)   ↵      (assegna a C la seconda riga (2) e tutte le colonne (:) di A)
C =
     4     5     6
»
```

I due punti (**:**) sono un operatore con diversi significati:
- <u>Generazione di sequenze</u> con vari passi (per default 1)

```
» t=(1:10)         ↵      (assegna a t i numeri da 1 a 10)
t =
     1     2     3     4     5     6     7     8     9    10
» t=(1:0.1:1.5)    ↵      (assegna a t i numeri da 1 a 1.5 con passo 0.1)
t =
    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000
»
```

- <u>Selezione</u> di un gruppo di elementi:

```
» a=t(2:4)    ↵      (assegna ad a gli elementi di t dal secondo al quarto)
a =
    1.1000    1.2000    1.3000
» b=t(1,:)    ↵      (assegna a b tutte le colonne di t)
b =
    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000
»
```

Nota: `x=[]` crea una matrice x vuota
`A(:,[2 5])=[]` cancella da A le colonne 2 e 5
`A([1:3],:)=[]` cancella da A le righe dalla 1 alla 3

Le variabili complesse sono gestite in maniera molto intuitiva. Il numero immaginario è indicato con **i** o con **j** e non necessita del simbolo *, quindi il comando
`a=[3+2i  4]`
crea un vettore con due elementi complessi (**Nota**: `2i≡2j` ma `i2` non ha nessun significato!) Tutte le funzioni MATLAB accettano variabili di tipo complesso.

E' opportuno non usare i o j come variabili (ad esempio contatori) per non creare ambiguità

Operazioni fra matrici

Le seguenti operazioni fra matrici sono disponibili in MATLAB:

+	addizione
-	sottrazione
*	moltiplicazione
\	divisione a sinistra (1\4=4)
/	divisione a destra (1/4=0.25)
^	elevazione a potenza
'	trasposto o coniugato (è l'apice o apostrofo non l'accento)

Addizione e sottrazione.
Possono essere effettuate fra elementi della stessa dimensione

```
» A=[1 2; 3 4];   ↵
» size(A)         ↵  (size fornisce la dimensione della matrice)
ans =
     2     2
» B=[5 6; 7 8];   ↵
» size(B)         ↵
ans =
     2     2
» A+B             ↵
ans =
     6     8
    10    12
» size(ans)       ↵
ans =
     2     2
» A-B             ↵
ans =
    -4    -4
    -4    -4
»
```

Si possono fare addizioni e sottrazioni fra una matrice (o un vettore) ed uno scalare

```
» x=[1 2 3]; ↵
» y=x+1       ↵  (somma 1 a tutti gli elementi di x)
y =
     2     3     4
»
```

Moltiplicazione e divisione
Possono essere *operazioni matriciali* del tipo:
matrice matrice
matrice vettore
matrice scalare

Ad esempio, date le matrici `A` e `B` (con A invertibile, cioè quadrata e non singolare)

`x=A\B` x = soluzione di Ax=B, cioè $x=A^{-1}B$

`x=B/A` x = soluzione di xA=B, cioè $x=BA^{-1}$ (nel caso di matrice A rettangolare, MATLAB fornisce la soluzione del sistema ai *minimi quadrati*).

Nota: l'inversa di una matrice A si ottiene con `inv(A)` (sempre se A è invertibile altrimenti MATLAB fornisce un messaggio di errore), il determinante di A si ottiene con `det(A)`.

Ovviamente per svolgere le precedenti operazioni le dimensioni delle matrici e dei vettori devono essere corrette.

Si possono anche effettuare operazioni sulle matrici *elemento per elemento*. In tal caso gli operatori devono essere preceduti dal simbolo "**.**" (non ci deve essere spazio tra punto e operatore). Per esempio, il comando:

```
[1 2].*[1 3]
```

fornisce il vettore [1 6] (nota che in questo caso l'omissione del punto darebbe luogo ad un messaggio di errore, dato che la dimensione delle matrici non è compatibile con l'operazione di moltiplicazione matriciale).

Nota: Anche l'addizione e la sottrazione possono essere precedute dal punto ma il risultato non cambia.

Elevazione a potenza

Sia A una matrice quadrata e p uno scalare (non necessariamente intero)

`A^p` fornisce la p-esima potenza di A, cioè

se p è intero → A*A*A*.....*A (p volte)
se p non è intero → V*D.*p/V (dove V=matrice autovettori, D=insieme autovalori)

Trasposto o coniugato

L'apice fornisce la matrice trasposta (logicamente il trasposto di un vettore riga è un vettore colonna)

```
» x=[1 2 3]; ↵
» y=x'       ↵
y =
     1
     2
     3
» size(x)    ↵
ans =
     1     3
» size(y)    ↵
ans =
     3     1
»
```

Se la matrice è complessa, l'operazione di trasposizione fornisce la matrice *coniugata trasposta* (se si vuole ottenere la sola matrice trasposta di Z occorre usare `conj(Z')`)

```
» Z=[1+i 2; 3+2i 4] ↵
Z =
   1.0000+ 1.0000i   2.0000
   3.0000+ 2.0000i   4.0000
» Y=Z'              ↵
Y =
   1.0000- 1.0000i   3.0000- 2.0000i
   2.0000            4.0000
» W=conj(Z')        ↵
W =
   1.0000+ 1.0000i   3.0000+ 2.0000i
   2.0000            4.0000
»
```

Le funzioni in MATLAB possono operare o sui singoli elementi delle matrici o sulla matrice nella sua globalità. Quindi della stessa funzione ci sono spesso due versioni differenti.
Esempio:

```
» A=[1 2; 4 8];   ↵
» sqrt(A)         ↵  (ritorna la radice quadrata di ogni singolo elemento di A)
ans =
    1.0000    1.4142
    2.0000    2.8284
» sqrtm(A)        ↵  (ritorna la radice quadrata della matrice A)
ans =
    0.3333    0.6667
    1.3333    2.6667
»
```

`help sqrt` oltre a spiegare la funzione `sqrt` fornisce l'elenco di tutte le funzioni ad essa legate.

Ci sono poi funzioni (tipo `max, min, sum, prod, mean, std`,...), che hanno significati diversi a seconda che l'argomento sia un vettore, una matrice o un vettore n-dimensionale. Vedi l'help in linea per i dettagli nei vari casi.

Nota: Il risultato fornito da una funzione può cambiare anche in base a come la funzione viene richiamata. E' quindi sempre opportuno guardare l'help in linea delle singole funzioni.
Esempio:

```
» x=[1 2 3 4]; ↵
» max(x)        ↵     (fornisce solo il valore dell'elemento massimo del vettore x)
ans =
     4
» [y,n]=max(x) ↵     (y=valore massimo di x, n=sua posizione)
y =
     4
n =
     4
```

Operatori relazionali e logici

I seguenti operatori di relazione sono definiti in MATLAB:

<	minore
>	maggiore
<	minore uguale
>=	maggiore uguale
==	uguale
~=	diverso

Operano una comparazione elemento per elemento. Il risultato è

Numero ≠ 0	se il test è TRUE
0	se il test è FALSE

Si noti la differenza fra il simbolo usato per l'assegnazione (=) ed il simbolo usato per il confronto (==).

Gli operatori logici definiti in MATLAB sono:

&	AND
|	OR
~	NOT

Consentono di combinare più operatori relazionali. Le variabili su cui operano vanno considerate

TRUE	se diverse da 0
FALSE	se uguali a 0

i risultati sono, ancora una volta,

Numero ≠ 0	se TRUE
0	se FALSE

Strutture di controllo

In MATLAB è possibile utilizzare strutture di controllo analoghe a quelle che sono disponibili nei linguaggi di programmazione, cioè `if, for, while`. Per mostrarne la sintassi vediamo alcuni esempi.

Struttura if.
Il comando:

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

fornisce, nella variabile `x`, il valore della funzione gradino a tempi continui, calcolata in `t`.

Struttura for.
La sequenza di comandi:

```
[m n]=size(a);
for i=1:m          (ciclo da 1 a m)
   for j=1:n       (ciclo da 1 a n)
     c(i,j)=2*a(i,j);
   end             (ogni ciclo deve essere chiuso con un end)
end
c
```

creano e visualizzano la matrice `c` ottenuta moltiplicando per 2 gli elementi della matrice `a`, come se si fosse scritto `c=2*a`.
Se si vuole fare un ciclo con passo diverso da 1 si deve specificare il valore del passo:

```
for i=1:2:m          (incrementa i di 2 ad ogni iterazione)
```

Struttura while.
La sequenza di comandi:

```
sum=1;
add=x;
while abs(add)>=1e-3
   sum=sum+add;
   add=x*add;
end
```

calcola la somma della serie geometrica $\sum_{k=0}^{\infty} x^k$, $|x|<1$, fermandosi quando il valore assoluto del termine corrente è minore di 10^{-3}.

Nota: per uscire da un ciclo senza terminarlo bisogna usare il comando `break`

Visualizzazione grafica e stampa

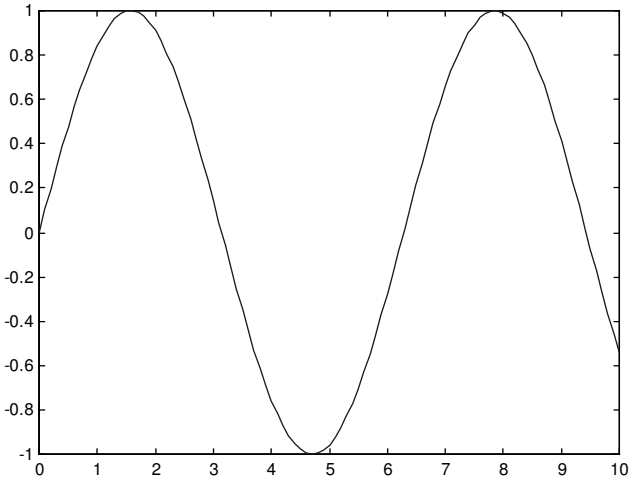
MATLAB può produrre sia grafici di funzioni monodimensionali che curve di livello e grafici di funzioni a più dimensioni. Per una dimostrazione delle potenzialità grafiche di MATLAB, si può dare il comando `demo`.
Di seguito si descrivono a grandi linee i comandi più utilizzati per la grafica.

Il comando fondamentale per la grafica bidimensionale è `plot`

`plot(t,x)` apre una finestra con il grafico di `x` in funzione di `t` in scala lineare, se `x` e `t` sono due vettori della stessa dimensione

Ad esempio la seguente sequenza di istruzioni genera una nuova finestra con il grafico riportato a destra:

```
» t=0:0.1:10;
» plot(t,sin(t))
»
```



La funzione plot ha diversi significati a seconda degli argomenti:

`plot(x,y)` disegna y in funzione di x;

`plot(y)` disegna y in funzione dei suoi indici; se y è un numero complesso è equivalente a plot(real(y),imag(y)

`plot(x,[y1;y2])` disegna sullo stesso grafico sia y1 che y2 in funzione di x (**nota:** x ,y1 ,y2 devono avere la stessa dimensione)

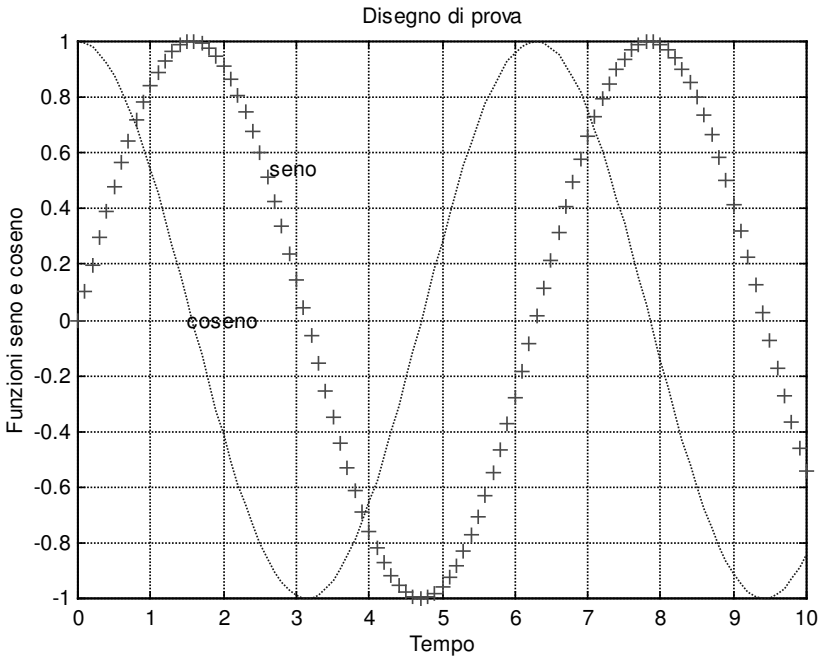
`plot(x1,y1,x2,y2)` disegna sullo stesso grafico y1 in funzione di x1 e y2 in funzione di x2 (**nota:** x1 e y1 devono avere la stessa dimensione come x2 con y2, ma non x1 e x2)

Si possono aggiungere griglie (`grid on`) e si possono usare diversi simboli e colori per le varie tracce (terzo argomento della funzione `plot` memorizzato in una stringa, `plot(x, y, 's')`). E' possibile dare un titolo al grafico (`title`), e agli assi (`xlabel, ylabel`), definire le caratteristiche degli assi (`axis` (regola le proprietà di base, cioè scala e aspetto), `axes` (gestisce tutte le proprietà dell'oggetto asse della figura)), posizionare del testo all'interno del grafico (`gtext, text`), risalire alle coordinate di un punto del grafico (`ginput`), colorare delle aree del grafico (`patch`), aggiungere delle linee sul grafico (`line`). Vedere i rispettivi `help` per i dettagli.
Se si vuole aggiungere un grafico su uno già esistente è indispensabile eseguire prima il comando `hold on` (si disabilita con `hold off`), se si vogliono vedere due grafici su due finestre grafiche separate è necessario aprire una nuova finestra con il comando `figure` (per chiuderla si usa `close`). Se invece si vogliono vedere più grafici sulla stessa finestra ma non sovrapposti si deve usare il comando `subplot`. Ad esempio con `subplot(2,3,1)` si divide la videata grafica in 2×3 finestrelle e si abilita la prima.

Per richiamare l'ultima finestra grafica si usa `shg` e per cancellarla `clf`.

Esempio:

```
» plot(t,sin(t),'+r',t,cos(t),':')   (disegna il primo grafico con crocette (+) rosse
                                      (r) e il secondo con linea tratteggiata (:))
» grid on                            (inserisce la griglia)
» title('Disegno di prova')          (inserisce un titolo)
» ylabel('Funzioni seno e coseno')   (assegna un'etichetta all'asse y)
» xlabel('Tempo')                    (assegna un'etichetta all'asse x)
» gtext('seno')                      (permette di introdurre la scritta seno in un punto a piacere
                                      del grafico)
» text(1.5,0,'coseno')               (introduce la scritta coseno alle coordinate 1.5, 0)
»
```



Se si vogliono visualizzare i dati come campioni discreti si usa al posto di `plot` la funzione `stem`, mentre per avere un'interpolazione a gradini dei punti si deve usare la funzione `stairs`.

Oltre alla scala lineare è possibile utilizzare:

doppia scala logaritmica	`loglog`
scala logaritmica su asse x	`semilogx`
scala logaritmica su asse y	`semilogy`
diagrammi polari	`polar`

Si possono stampare sia grafici MATLAB che finestre SIMULINK (programma che verrà descritto più avanti). Il comando per stampare è `print` che deve poi essere utilizzato con i parametri opportuni (vedi `help print`). Si può anche stampare su un file contenente i comandi necessari per la successiva stampa.

Funzioni di input interattivo:

`input` restituisce all'utente il prompt per introdurre dati. La sintassi è:
`y=input('string')` dove string può essere una qualsiasi sequenza. Dopo questa riga di comandi ritorna il prompt di MATLAB e il numero scritto da tastiera viene memorizzato in y. Se si vuole introdurre una sequenza di caratteri è necessario specificare 's' (`input(string,'s')`)

`Keyboard` richiama la tastiera dall'interno di un m-file. Quindi interrompe l'esecuzione dell'm-file, e ritorna al prompt del MATLAB. Lo stato speciale viene indicato dal fatto che il prompt è "K»". In tale stato si possono valutare e cambiare le variabili. Si esce digitando RETURN e schiacciando il tasto di invio. Il controllo torna quindi all'm-file.

`pause` interrompe l'esecuzione fino a che l'utente non preme un tasto. (`pause(n)`: la interromape per n secondi)

`menu` genera una finestra di menu di scelte per l'utente. Ad esempio:
`K = menu('Choose a color','Red','Blue','Green')`
genera una finestra con tre bottoni (red, blue, green), e restituisce alla variabile K un valore equivalente al numero del bottone cliccato. Si possono inserire al massimo 32 valori.

`disp(x)` mostra il messaggio contenuto nella stringa x

Funzioni di utilità varie

`error(message)` in presenza di un errore termina l'esecuzione ritornando all'input da tastiera dopo aver mostrato il messaggio

`nargchk` verifica il numero di ingressi ad una funzione, ritornando un errore se non è adeguato

`return` termina l'esecuzione e torna all'input da tastiera

`num2str, int2str, str2num` convertono da numeri a stringhe e viceversa

`fopen, fprintf,...` funzioni di input ed output da file a basso livello, che sono in tutto simili alle equivalenti C

`exist` verifica se una variabile/funzione/file esiste

`any` ritorna true se almeno uno degli elementi della variabile di input e` true

`all` ritorna true se tutti gli elementi della variabile di input sono true

M-files e funzioni esterne

Come accennato nell'introduzione, è possibile raggruppare una sequenza di comandi MATLAB in un file esterno che deve avere estensione `.m`. Tale file può essere creato con un editor di testi qualsiasi oppure si può utilizzare l'editor fornito dallo stesso MATLAB che si richiama con il comando `edit`. Con `edit nome_file` si apre e si può modificare un file esistente contenuto nella directory di lavoro.

N.B.:

`pwd`	mostra la directory in cui si sta lavorando
`cd`	cambia directory
`ls` o `dir`	mostra il contenuto di una directory
`what` *dirname*	elenco dei files .m, .mat, contenuti nella directory *dirname*
`which` *filename*	localizza la directory in cui e' contenuto il file *filename*
`type` *filename*	visualizza il contenuto del file *filename*
`lookfor` *string*	trova le funzioni/comandi in cui la stringa *string* compare nella prima riga del commento

Per eseguire poi la sequenza delle istruzioni contenute nel file `nome_file.m` sarà quindi sufficiente digitare comando `nome_file` (senza estensione) dalla linea di comandi MATLAB. E' spesso opportuno far precedere i comandi di un M-file dal comando `clear` che annulla lo spazio delle variabili create fino a quel momento in MATLAB: questo affinché i risultati od i nomi di variabili precedentemente utilizzati non interferiscano con quelli usati nel file `.m`. E' inoltre conveniente aggiungere dei commenti nel file: a questo scopo, si può far precedere una riga dal carattere %, che forza MATLAB a non interpretare i caratteri successivi. Se un comando si estende oltre la lunghezza di una riga, esso può essere continuato nella riga seguente interrompendo con tre caratteri di interpunzione (...)
Un file .m può contenere richiami a funzioni predefinite in MATLAB ma anche definite dall'utente stesso. E' cioè possibile aggiungere nuove funzioni al vocabolario di MATLAB. I comandi relativi ad una funzione devono essere contenuti in un file esterno, ancora con estensione `.m`. Tale nome definisce il nome della nuova funzione. La prima riga del file deve contenere la definizione della sintassi della nuova funzione, e quindi lista dei parametri della funzione (che possono essere sia scalari che matrici) e la parola chiave `function`. Ad esempio, volendo scrivere una funzione che calcola la media e la deviazione standard di un vettore, si può creare il file `stats.m`:

```
% STATS calcola media e deviazione standard di un vettore

function [mean,stdev] = stats(x)
n = length(x);              (la funzione length ritorna la lunghezza del vettore)
mean = sum(x) / n;          (la funzione sum esegue la somma degli elementi di x)
stdev = sqrt(sum((x - mean).^2)/n);
```

Le variabile contenute nel corpo della funzione, e che non compaiono nella lista dei parametri, sono tutte *locali* alla funzione stessa, e possono pertanto avere lo stesso nome di variabili definite in altre procedure senza che si abbiano pericolose interferenze. Si possono dichiarare delle variabili come *globali* con `global` (per vedere quali sono le variabili globali: `who global`).
Per calcolare, quindi, media e deviazione standard di un vettore `a` e porre il risultato nelle variabili `media` e `stdv` è sufficiente digitare il comando

```
[media,stdv]=stats(a)
```

Si possono avere un qualsiasi numero di argomenti di uscita e di ingresso: questi ultimi vengono passati alla funzione *per valore* e non possono essere modificati al suo interno. Se la funzione ha un

solo argomento di uscita, le parentesi quadre possono essere omesse. La funzione stats precedente poteva anche essere definita così

```
function y = stats(x)
n = length(x);
y(1) = sum(x) / n;
y(2) = sqrt(sum((x - mean).^2)/n);
```

In tal caso darebbe come risultato un vettore con i valori desiderati.
Per procedure che lavorano nello spazio di lavoro globale vedere `script`.
Se si definisce una nuova funzione con la parola chiave `function` dentro il corpo di un'altra funzione, tale sottofunzione sarà visibile solo dalla funzione in esame. Ad esempio la funzione avg è una sottofunzione dentro il file `stats.m` e solo qui è visibile.

```
function [mean,stdev] = stats(x)
n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);

function mean = avg(x,n)
mean = sum(x)/n;
```

Normalmente le funzioni terminano quando raggiungono la fine dei comandi contenuti nel file. Per forzare un'uscita da una funzione si può usare `return`.

Nota: se le prime righe di un file, contenente una funzione da noi dichiarata, sono dei commenti, esse vengono visualizzate invocando dalla linea di comandi MATLAB `help nome_file`. Ad esempio il comando `help stats` ritorna:

```
STATS calcola media e deviazione standard di un vettore
»
```

Librerie o Toolbox disponibili in commercio

Sono state create e sono disponibili in commercio delle librerie, chiamate "toolbox", contenenti funzioni MATLAB molto complesse per la soluzione di svariati problemi nel campo del controllo, dell'identificazione, ecc.
Sono ad esempio disponibili i seguenti toolbox:

`control`	per controllo
`robust`	per controllo robusto
`signal`	per elaborazione numerica di segnali
`ident`	per identificazione di sistemi
`optim`	per ottimizzazione (interpolazione, ecc.)
`nnet`	per reti neurali

Il comando `help nome_toolbox` fornisce informazioni sul corrispondente toolbox.