INTRODUZIONE A MATLAB

9

Il *Matlab*, prodotto dalla *Mathworks* Inc. è un programma per l'elaborazione di dati numerici e la presentazione grafica dei risultati. Questo programma è utilizzato estensivamente per l'analisi di sistemi e per il progetto di controllori. Queste note presentano alcune caratteristiche di base del programma.

AVVIO

Lanciare la finestra di MATLAB cliccando sulla relativa icona.

FINESTRA DI COMANDO

E' la finestra che si utilizza per comunicare con l'interprete di comandi di MATLAB. Quando l'interprete e' pronto per accettare comandi, e' visibile il *prompt* >> e di possono digitare istruzioni che vengono terminate premendo il tasto ENTER.

L'aspetto principale del programma è la semplicità concettuale con cui vengono rappresentati i dati. I dati vengono introdotti nel programma in maniera molto semplice, mediante assegnamento.

Ad esempio, con l'istruzione:

```
>> a =4
```

definiamo la variabile a assegnandole il valore 4 .Occorre notare che il programma ribadisce il risultato della istruzione precedente, visualizzandolo sullo schermo.

Il programma non richiede definizioni particolari di tipo durante l'inizializzazione di variabili, ma il tipo viene assegnato automaticamente in funzione del dato inserito.

Ad esempio, l'istruzione:

```
>> b =4+5i
```

definisce ed inizializza la variabile *b* al valore complesso 4+5i.

A seguito dell'assegnazione, il programma esegue l'*echo* del dato introdotto:

```
b =
        4.0000 + 5.0000i
```

INSERIRE MATRICI E VETTORI

MATLAB lavora sempre con matrici. Gli scalari sono trattati come matrici 1x1. I vettori sono matrici con una sola riga o una sola colonna. MATLAB alloca automaticamente le matrici in memoria pertanto non e' necessario dichiararne le dimensioni (al contrario del Fortran per esempio).

Le matrici si inseriscono come segue:

- separando gli elementi con virgole o spazi
- separando le righe con punto e virgola
- delimitando la matrice tra parentesi quadre

Ad esempio, l'istruzione:

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

produce il risultato:

```
A =
     1     2     3
     4     5     6
     7     8     9
```

La matrice è disponibile in una memoria "locale" di MATLAB denominata *workspace* per uso successivo.

Si noti inoltre che il programma è *case sensistive*, distingue, cioè fra maiuscole e minuscole.

I vettori sono particolari matrici con 1 colonna e n righe (oppure n colonne ed 1 riga), introducibili in modo analogo a quanto fatto per le matrici.

Ad esempio, l'istruzione:
```
>> v = [1; 2; 56]
```
produce il risultato:
```
v =
   1
   2
   56
```

L'operazione di **trasposizione** (sia di vettori che di matrici) è l'apice (').
Ad esempio, l'istruzione:
```
>> v'
```
produce il risultato:
```
ans =
   1    2       56
```

dove *ans* è l'abbreviazione di *answer*, ovvero "risposta ", vale a dire la variabile che contiene il risultato della elaborazione richiesta. Volendo conservare tale risultato si può scrivere:
```
>> vt = v';
```

Il punto e virgola alla fine della linea viene usato per sopprimere l'*echo*.

Se però il vettore (o la matrice) è complesso l'operatore "apice" esegue la trasposta coniugata.
La semplice trasposizione si ottiene con l'operatore "punto apice" (.'):
```
>> v.'
```

ELEMENTI DI MATRICI
Gli elementi di matrici possono essere espressioni di MATLAB
Ad esempio, l'istruzione:
```
x=[-1.3 sqrt(3) (1+2+3)*4/5]
```
produce il risultato:
```
x =
      -1.3000      1.7321      4.8000
```

I singoli elementi possono essere individuati mediante indici racchiusi tra parentesi tonde.
Ad esempio, l'istruzione:
```
x(5)= abs(x(1))
```
produce il risultato:
```
   x = -1.3000      1.7321      4.8000      0      1.3000
```

Come si vede la dimensione di x viene automaticamente incrementata in modo da inglobare il nuovo elemento e gli elementi indefiniti (in questo caso il quarto) viene posto uguale a zero.

Le matrici possono essere costruite a partire da matrici di dimensioni minori.
Ad esempio, l'istruzione:
```
r=[10 11 12];
A=[A; r]
```
produce il risultato:
```
A =
      1     2     3
      4     5     6
      7     8     9
      10    11    12
```

USO DEI DUE PUNTI

L'operatore due punti è molto importante e di uso frequente in MATLAB.
Può essere usato per creare vettori, per isolare righe o colonne di matrici e per controllare i loops.

Esempio: creare un vettore di punti equispaziati

1:4	produce il risultato	1	2	3	4	
1:3:8	produce il risultato	1	4	7		

In generale

j:k	produce il risultato	j	j+1	j+2	...	k
j:i:k	produce il risultato	j	j+i	j+2i	...	k

I due punti e le matrici

A(:,j)	è la j-ma colonna di A					
A(k,:)	è la k-ma riga di A					
A(:,:)	coincide con A					
A(j,k)	è l'elemento (j,k)					
A(:,j:k)	è il vettore	A(:,j)	A(:,j+1)	A(:,j+2)	...	A(:,k)

ISTRUZIONI, VARIABILI E ESPRESSIONI

Le istruzioni di MATLAB sono frequentemente nella forma: *variabile=espressione*
o semplicemente: *espressione*

Espressioni comprendono variabili, operatori, funzioni e caratteri speciali.
Una espressione, una volta valutata, produce una matrice.
Di *default* la matrice è visualizzata sulla finestra di comando di MATLAB, ma l'output sullo schermo può essere soppresso terminando l'espressione con il punto e virgola (;).
Se il nome della variabile e il segno = sono omessi, MATLAB crea automaticamente la matrice *ans*.
Ad esempio, l'istruzione:
1900/81
produce il risultato:
ans =
23.4568

MATLAB distingue le minuscole dalle maiuscole.
Pertanto le variabili *name*, *Name* e *NAME* sono differenti.

Matrici aventi le stesse dimensioni possono essere combinate usando gli operatori + e -

La **moltiplicazione di matrici** (aventi dimensioni congruenti) si effettua usando l'operatore *
C=A*B

Matlab prevede due simboli per la **divisione**: / e \.
Supponendo che A sia una matrice quadrata e non singolare, con:
A = [10 37 64; 13 39 61; 22 61 100]
B = [1 2 3]'

l'istruzione:
>> X =B '/A

fornisce come risultato
X=
-0.0833 0 0.0833
che `e soluzione di X *A = B'.
Infatti: >> E = X *A
E =
1.0000 2.0000 3.0000
coincide con B '.

Mentre la divisione X = A \ B ,
>> X = A \ B
X =
0.1593
0.1324
-0.0858
è soluzione di B = A * X .
Infatti: >> A * X
ans =
1.0000
2.0000
3.0000
coincide con B .

L'**inversa** di una matrice si effettua come segue: inv(B)

Per valutare una operazione "**termine per termine**", per esempio moltiplicare ogni termine del vettore A=[1 3 2] per ogni termine del vettore B=[-1 2 1] in modo da ottenere il risultato C=[-1 6 2], si usa l'operatore .*
Ad esempio:
C=A.*B

WORKSPACE E HELP
In ogni momento si può visualizzare quali sono le variabili contenute nel Workspace (una sorta di memoria locale di MATLAB).
Si usa il comando *who* oppure *whos* (per avere informazioni più dettagliate).

Esiste un help in linea. Sul *prompt* di MATLAB è sufficiente digitare *help* con il nome del comando specifico. Se non si conosce il nome del comando per eseguire una certa operazione si può tentare con l'istruzione *lookfor*.

Ad esempio: help fft
Visualizza l'help del comando che esegue la trasformata di Fourier

mentre: lookfor interpolation
è un esempio di ricerca di funzioni per eseguire l'interpolazione

COMANDI UTILI E ISTRUZIONI DOS-LIKE
Per verificare il contenuto della directory di lavoro, basta digitare il comando DOS-like
>> dir
e verrà mostrato il contenuto della directory di lavoro.
Per visualizzare la directory corrente di lavoro, si utilizza il comando
>> pwd

Le istruzioni DOS più utilizzate in ambiente Matlab sono:

dir	elenca i files contenuti nella directory corrente.
type *filename*	visualizza il contenuto del file *filename* .
delete filename	elimina il file *filename* dalla directory corrente.
↑ (freccia sù della tastiera)	richiama le istruzioni digitate in precedenza.
! *command*	invia l'istruzione *command* al sistema operativo.

ISTRUZIONI PER LA GRAFICA

Esempio

t=[0:0.01:10];	crea un vettore [0, 0.01, 0.02, ..., 10]
ys=sin(2*pi*t);	crea un'onda sinusoidale con frequenza di 1 Hz
plot(t,y);	plotta l'onda sinusoidale
yc=cos(2*pi*2*t);	crea un coseno avente frequenza pari a 2 Hz
plot(t,ys,t,yc)	plotta le due onde contemporaneamente
axis([0 10 -1.5 1.5])	forza gli estremi degli assi
title('Onda sinusoidale')	scrive il titolo della figura
xlabel('Tempo [s]')	scrive la label delle ascisse

Esempio

f=[0:0.01:3];	crea un vettore "di frequenze"
z=0.01;	fattore di smorzamento
H=1.0./sqrt((1-f.^2).^2+(2*z*f).^2);	e' la risposta in frequenza di un sistema SDOF
plot(f,H)	plotta il grafico in scala lineare
semilogy(f,H)	plotta le ordinate in scala logaritmica
loglog(f,H)	plotta entrambi gli assi in scala logaritmica

OPERAZIONI MATRICIALI

det(A)	calcola il determinante di A
inv(A)	calcola l'inversa di A
trace(A)	somma gli elementi sulla diagonale
eig(A)	calcola gli autovalori di A
rank(A)	calcola il rango di A

VARIABILI COMPLESSE

sqrt(-9)	MATLAB può valutare questa espressione
z1=2+3*j	definisce un numero complesso
z2=-2-9*i	MATLAB usa indifferentemente i o j
abs(z1)	valuta il modulo
angle(z1)	valuta la fase
real(z1)	parte reale
imag(z1)	parte immaginaria

SCRIPT FILES

Invece di inserire i comandi dal prompt di MATLAB è possibile eseguire una serie di comandi scritti in files. Questi files sono chiamati m-files perche' hanno estensione .m
Ci sono due tipi di m-files: gli *script* files e i *function* files.
Gli script files contengono una serie di comandi per l'interprete di MATLAB.
Un set di comandi può essere inserito in un file di testo (ascii) usando un editor (ad esempio il "blocco note" di Windows; le versioni 5.0 e successive di MATLAB hanno un editor dedicato).

Esempio: se lo script file si chiama *pippo.m* i comandi che esso contiene si eseguono digitando il nome del file senza estensione (*pippo*) dal prompt di MATLAB.

Le istruzioni contenute in uno *script* file lavorano sulle variabili contenute nello *workspace* globale.
Tutte le variabili utilizzate dallo *script* file rimangono disponibili una volta terminata l'esecuzione.

FUNCTION FILES

Sono m-files che iniziano con la parola *function*.
Consentono, in ambiente Matlab, la definizione di funzione simili a quelle previste nei linguaggi di programmazione standard.
Le variabili vengono passate per valore.
Differiscono dagli script files perché tutti gli argomenti sono passati alla funzione attraverso una lista di parametri e tutte le variabili contenute nella funzione sono locali.

Esempio

```
function y=mean(x)
% Calcola il valor medio.
% Per i vettori, mean(x) restituisce il valor medio.
% Per le matrici, mean(x) restituisce un vettore riga contenente il valor medio di ogni colonna.
[m,n]=size(x);
if m==1
   m=n;
end
y=sum(x)/m;
return
```

La prima linea dichiara il nome della funzione e gli argomenti in input e output.
Le successive tre righe documentano il file (sono commenti) e sono mostrate quando si digita help *mean* dal *prompt* di MATLAB. Questa ultima caratteristica e' molto utile perché consente di ottenere automaticamente un help in linea per le funzioni nuove.

Una *function* differisce da uno script perché lavora su variabili locali e per il fatto che non accede alle variabili globali.
Nell'esempio, le variabili *m*, *n* e *y* sono locali e non esisteranno nel *Workspace* dopo che la funzione *mean* ha terminato il proprio compito. Al contrario, le variabili negli *script files* sono disponibili nel *Workspace* ad esecuzione dello script file conclusa.

Esempio

```
z=1:99;          vettore di interi da 1 a 99
mz=mean(z);      istruzione per calcolare il valor medio di z e attribuirlo alla variabile mz.
```

Il vettore viene passato alla funzione *mean* dove diventa la variabile locale *x*.
Non e' necessario che i vettori *z* e *x* abbiano lo stesso nome.

ISTRUZIONI DI CONTROLLO

for	ripetizione di un insieme di istruzioni per un numero predeterminato di iterazioni. Deve terminare con end.
while	ripetizione di un insieme di istruzioni fino a quando una condizione rimane era. Deve terminare con end .
if	istruzione condizionale. Deve terminare con end. Si possono utilizzare anche *else* e *elseif.*
break	interruzione di un ciclo.

Esempio
```
for i=1:m,
   for j=1:n,
        A(i,j)=1/(i+j-1);
   end
end
```

Esempio
```
while i < 100
   i=i+1;
end
```

Esempio
```
if a>max
   max=a;
end
```

Gli operatori relazionali sono:

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

Nota: l'operatore per testare l'uguaglianza è == (non =).

ISTRUZIONI LOAD E SAVE

Si utilizzano per caricare (salvare) dati da (su) files esterni.
Di seguito si riporta un estratto dell'help on line di MATLAB per queste due istruzioni.

LOAD

LOAD Load workspace variables from disk.
LOAD FNAME retrieves the variables from the MAT-file 'fname.mat'.
LOAD FNAME -ASCII or LOAD FNAME -MAT can be used to force LOAD to treat the file as either an ASCII file or a MAT file.

SAVE

SAVE Save workspace variables to disk.
SAVE fname saves all workspace variables to the binary "MAT-file" named fname.mat. The data may be retrieved with LOAD. Omitting the filename causes SAVE to use the default filename "matlab.mat".

SAVE fname X saves only X.
SAVE fname X Y Z saves X, Y, and Z.
SAVE fname X Y Z -ASCII uses 8-digit ASCII form instead of binary.
SAVE fname X Y Z -ASCII -DOUBLE uses 16-digit ASCII form.
SAVE fname X Y Z -ASCII -DOUBLE -TABS delimits with tabs.

```
close all
clear
%Ampiezza
m=20;           %massa
F0=300;         %ampiezza forza
k=200000;      %rigidezza
omega_nat=sqrt(k/m);     %pulsazione naturale
freq=[0:0.1:30];          %vettore delle frequenze
omega=freq.*2*pi;        %vettore delle pulsazioni
r=omega./omega_nat;


%%%%FRF
figure
    for zita=0.05:0.005:0.15
    Aux1=F0/(m*omega_nat^2);
    Aux2=(1-r.^2).^2+(2*zita*r).^2;
    X0=Aux1./sqrt(Aux2);              %ampiezza della risposta
    H=X0/F0;  %FRF
    Ascissa=r.^2;
    plot(Ascissa,H),hold on,grid,title('FRF')
    end
```

```
%%eccitazione della massa sospesa
close all
clear
%Ampiezza
m=20;            %massa
F0=300;          %ampiezza forza
k=200000;       %rigidezza
omega_nat=sqrt(k/m);     %pulsazione naturale
freq=[0:0.1:30];          %vettore delle frequenze
omega=freq.*2*pi;        %vettore delle pulsazioni
r=omega./omega_nat;

figure
for zita=0.1:0.1:0.6

    Aux1=F0/(m*omega_nat^2);
    Aux2=(1-r.^2).^2+(2*zita*r).^2;
    X0=Aux1./sqrt(Aux2);              %ampiezza della risposta
    T0=(X0*m*(omega_nat^2)).*sqrt(1+(2*zita.*r).^2);
    tau=T0./F0;
    Ascissa=r.^2;
    plot(Ascissa,tau),grid,title('tau'),hold on,grid
end
```

```
%%%Forza eccitatrice con ampiezza variabile

close all
clear

m=20;              %massa
F0=300;            %ampiezza forza
k=200000;         %rigidezza
omega_nat=sqrt(k/m);     %pulsazione naturale
freq=[0:0.1:30];          %vettore delle frequenze
omega=freq.*2*pi;         %vettore delle pulsazioni
r=omega./omega_nat;

figure
    for zita=0.05:0.005:0.15
    %for zita=0.1:0.1:1
    A=2;
    Aux3=(1-r.^2).^2+(2*zita*r).^2;
    Aux4=(A*r.^2)/m;
    X0=Aux4./sqrt(Aux3);
    Ordinata=X0*m/A;
    Ascissa=r.^2;
    uno=ones(1,size(Ascissa,2));
    plot(Ascissa,Ordinata),grid,title('Risp.ecc.variabile'),hold on
    plot(Ascissa,uno,'r')
    end
```

```
%%integrazione con ode
t0=0;
tf=6;
y0=0;
[t,y]=ode23('risolvo',[t0 tf],y0)

figure
plot(t,y,'-o')
```

```
function yp=risolvo(t,y)
A=60;
B=5;
C=-0.02;
yp=A+B.*y+C*y.^2;
```