Note Introduttive a MATLAB

MATLAB è un interprete in grado di eseguire istruzioni native (built-in) o contenute in files su disco (M-files). L'elemento fondamentale sul quale lavora MATLAB è la matrice, che può essere costituita anche da elementi complessi.

- Linee precedute dal segno di percentuale ('%') sono linee di commento.
- Il comando 'pause' genera una pausa durante l'esecuzione.

Introduzione di dati

- Introduzione diretta

Il nome della matrice deve essere seguito dal segno " = " e dalla lista dei valori racchiusa tra parentesi quadre.

```
% matrice quadrata (3,3)
>> A = [1 2 3; 0 3 1; 9 7 2]
A =
     1     2     3
     0     3     1
     9     7     2

% matrice riga (1,3)
>> r = [7,8,9] oppure r = [7 8 9]
r =
     7     8     9

% matrice colonna (3,1)
>> c = [7;8;9]
c =
     7
     8
     9

% composizione di due matrici
>> B = [A;r]
B =
     1     2     3
     0     3     1
     9     7     2
     7     8     9

% composizione di due matrici
>> C = [A,c]
C =
     1     2     3     7
     0     3     1     8
     9     7     2     9

% generazione di un vettore di elementi: da 1 a 3.4 con passo 0.5
>> a2=1:0.5:3.4
a2 =
    1.0000    1.5000    2.0000    2.5000    3.0000
```

si noti che l'ultimo elemento del vettore è 3.0 anziché 3.5 che risulta maggiore del limite superiore assegnato (3.4)

```
% matrici come sequenza di vettori colonna separati da :
>> D=[1 3 5 7; 2:2:8; 1 2 4 9]

D =
     1     3     5     7
     2     4     6     8
     1     2     4     9

% oppure come sequenza di vettori colonna separati da , (o spazi)
>> bc=[1; 4; 3; 5; 3; 5];
>> B=[bc,2*bc 3*bc]
B =
     1     2     3
     4     8    12
     3     6     9
     5    10    15
     3     6     9
     5    10    15
```

- Generazione di una matrice

```
% matrice quadrata di zeri
>> M0 = zeros(2,2)
M0 =
     0     0
     0           0

% matrice quadrata di 1
>> I = ones(2,2)
I =
     1     1
     1     1

% matrice Identità
>> E = eye(2,2)
E =
     1     0
     0     1

% matrice quadrata di numeri casuali
>> R = rand(2,2)
R =
    0.2190    0.6789
    0.0470    0.6793

% i numeri complessi non ricevono alcun trattamento speciale
>> C=ones(3,3) + i * A(:,1:3)
C =
   1.0000 + 1.0000i   1.0000 + 2.0000i   1.0000 + 3.0000i
   1.0000             1.0000 + 3.0000i   1.0000 + 1.0000i
   1.0000 + 9.0000i   1.0000 + 7.0000i   1.0000 + 2.0000i
```

1. 'i' rappresenta sqrt(-1) così come 'j'
2. sottomatrici possono essere estratte dalle matrici usando gli indici. In questo caso la parte immaginaria di C è una sottomatrice di A che consiste delle prime 3 colonne di A.

Il primo indice in un array indica sempre le righe, mentre il secondo le colonne. Gli indici possono scalari oppure vettori.
Esempio: A(:,1:3), A(1:3,1:3), e A([1 2 3],[1 2 3]) danno lo stesso risultato

```
ans =
      1     2     3
      0     3     1
      9     7     2
```

- Gestione della memoria

```
% cancella la matrice R dalla memoria
>> clear R

% cancella tutte le variabili
>> clear

% cancella lo schermo
>> clc

% mostra lo stato del Workspace
>> whos     oppure      who
```

- Operazioni sulle matrici

```
% matrice trasposta
>> A = [1 2 3; 0 3 1; 9 7 2];
>> T = A'
T =
     1     0     9
     2     3     7
     3     1     2

% matrice quadrata (2,2) complessa
>> C = [1+i 2-i; 3+2*i 4];
>> T = C'
T =
   1.0000 - 1.0000i   3.0000 - 2.0000i
   2.0000 + 1.0000i   4.0000
```

➔ Si noti che la trasposizione implica la coniugazione.

```
% somma di matrici di uguali dimensioni
>> B = A';
>> C = A + B
C =
     2     2    12
     2     6     8
    12     8     4

% prodotto di matrici con uguali dimensioni interne
>> C = A * B
```

```
C =
    14     9    29
     9    10    23
    29    23   134


% moltiplicazione per uno scalare
>> C = 3 * A
C =
     3     6     9
     0     9     3
    27    21     6


% Divisione destra
>> r = [1 -1 2];
>> X = A/r



X =
    0.8333
   -0.1667
    1.0000
➔ soluzione di X * A = r


% Divisione sinistra
>> c = r';
>> X = A\c
X =
    0.5000
   -0.5000
    0.5000
➔ soluzione di A * X = c

% Calcolo dell'inversa di una matrice
>> A1 = inv(A)
A1 =
    0.0156   -0.2656    0.1094
   -0.1406    0.3906    0.0156
    0.4219   -0.1719   -0.0469

>> A * A1
ans =
    1.0000    0.0000         0
         0    1.0000         0
    0.0000    0.0000    1.0000
➔ prodotto di A per la sua inversa = matrice identità


% ELEMENTI DI UNA MATRICE:
% i primi due elementi della 3° colonna
>> A(1:2,3)
ans =
     3
     1
```

```
% la seconda riga
>> A(2,:)
ans =
     0     3     1


% il vettore con gli elementi sulla diagonale principale
>> D = diag(A)
D =
     1
     3
     2

% il vettore con gli elementi sulla diagonale secondaria
>> D = diag(A,2)
D =
     3

% la matrice triangolare superiore
>> T = triu(A)
T =
     1     2     3
     0     3     1
     0     0     2


% la matrice triangolare inferiore
>> T = tril(A)

T =
     1     0     0
     0     3     0
     9     7     2

% determinante
>> d = det(A)
d =
   -64

$ rango
>> B = [A; 1 0 2];                % matrice (4,3)
>> r = rank(A)
r =
     3

% massimo valore di un vettore
>> x = A(:,1);
>> m = max(x)
m =
     9

% massimo valore di una matrice
>> M = max(A)
M =
     9     7     3
```
➔ fornisce un vettore riga contenente il massimo valore per ciascuna colonna

```
% somma degli elementi di un vettore
>> s = sum(x)
```

```
s =
    10
```

- Caricamento da M-file

Si consiglia vivamente di fare uso di *macro* (file editabili con estensione ***.m***) che contengono la sequenza dei comandi *Matlab* per la definizione di ogni operazione complessa. Un M-file è un file di testo, con suffisso .m, che contiene istruzioni MATLAB.

```
File  NEWMAT.M

N = [1 2; 0 3]

NI = inv(N)
```

```
" newmat

N =

   1   2
   0   3

NI =

    1.0000   -0.6667
         0    0.3333
```

Il file così definito sarà memorizzato come "NEWMAT.M" e quindi richiamabile per l'esecuzione direttamente dall'ambiente di lavoro introducendo nella riga di comando semplicemente il nome del file. Una matrice può essere generata caricandola da un file generato in precedenza da MATLAB stesso. Il file ascii, leggibile quindi da un programma di elaborazione in altro linguaggio.

```
" J = [2 6; 1 0];
" save Jmat J
" clear J
" J
"
```

```
" load Jmat
" J

J =

   2   6
   1   0
```

```
" J = [4 2; 5 7];
" save Jmat.dat J /ascii
"
```

```
Jmat.dat:

4.0000000e+000    2.0000000e+000
5.0000000e+000
7.0000000e+000
```

Dalla linea di comando e' possibile eseguire comandi DOS come "dir", "cd". E' quindi possibile salvare i files creati in un proprio direttorio di lavoro.

```
" cd a:
" dir

mat_env

" cd mat_env
" dir

.      ..      newmat.m
```

```
" clear
" A = [1 2, 3 4];
" clear
" A = [1 2; 3 4];
" B = eye(2,2);
" whos
        Name      Size      Elements    Bytes    Density   Complex

          A       2 by 2        4          32       Full       No
          B       2 by 2        4          32       Full       No

Grand total is 8 elements using 64 bytes
```

- Rappresentazione di polinomi

La rappresentazione di un *polinomio* avviene definendo un vettore riga contenente i coefficienti in ordine decrescente a partire dal termine di grado massimo. Ad esempio il polinomio di terzo ordine $p(s)=3s^3+2s^2+3s+4$ si definisce come:

```
>> p=[3 2 3 4];

% Calcolo delle radici del polinomio i cui coefficienti sono gli elementi del
vettore p. Se p ha n+1 componenti, il polinomio è:
      p(1)*x^n + ... + p(n)*x + p(n+1)

>> r=roots(p)
r =
   0.1667 + 1.1426i
   0.1667 - 1.1426i
  -1.0000

% Se A è una matrice N*N, POLY(A) genera un vettore con N+1 elementi che sono i
coefficienti del polinomio caratteristico, DET(s*EYE(A) - A). Se V è un vettore,
POLY(V) genera un vettore i cui elementi sono i coefficienti del polinomio le
cui radici sono gli elementi di V.
>> A=[1 2;3 4];
>> p=poly(A)
p =
    1.0000   -5.0000   -2.0000
➔ polinomio caratteristico di A (s^2-5*s-2)
>> r=roots(p)


r =
    5.3723
   -0.3723
➔ si ottengono tutti gli autovalori di A (analogo al comando EIG(A)).

>> pp = poly(r)
pp =
    1.0000   -5.0000   -2.0000

% Se p è un vettore di N+1 elementi (coefficienti di un certo polinomio), allora
y = POLYVAL(p,x) restituisce il valore del polinomio valutato in x.
>> y = POLYVAL(pp,1)
y =
    -6

% Se p e q sono due vettori che contengono i coefficienti di due polinomi p(x) e
q(x), il comando 'CONV' esegue il prodotto tra i due polinomi.
>> p = [1 2 3];
>> q = [4 5 6];
>> c = conv(p,q)
c =
     4    13    28    27    18

% divisione tra due polinomi
>> [q,r] = deconv(c,p)
q =
     4     5     6
r =
     0     0     0     0     0
```

➔ si noti che **q** è il polinomio *quoziente*, mentre **r** è il polinomio *resto*.

```
%       SVILUPPO               s + 1          2       -1
%          IN               ------------  = ------ + -----
%    FRATTI SEMPLICI        s^2 + 5s + 6    s + 3    s + 2

>> N = [1 1];
>> D = [1 5 6];

% calcolo dei residui
>> [R,P,K] = residue(N,D)
R =
     2
    -1
P =
    -3
    -2
K =
     []

% Procedimento inverso
>> [N,D] = residue(R,P,K)
N =
     1     1
D =
     1     5     6
```

- For, while, if

```
Controllo del flusso:

>> x = []; for i = 1:n, x=[x,i^2 ], end

oppure

>> x = [];
>>        for i = 1:n
>>            x = [x,i^2 ]
>>        end

produce il vettore (n=3)
x =
     1     4     9


% Dato un numero a, calcolo il più numero non negativo tale che:
>> n = 0;
>>       while  2^n < a
>>            n = n + 1;
>>       end
>>       n
(se a=5)
n =
     3
```

- Rappresentazione grafica

```
% grafici a due dimensioni
>> x = [1 4 7 3];
>> plot(x, '+'), grid

% genera una sinusoide ad ampiezza crescente
>> t=0:720;
>> for ii=1:721,
>>    x(ii)=t(ii)*sin(pi/180*t(ii));
>> end
>> plot(t,x)

% visualizzazione nel piano complesso
>> x = [i 2+2*i 3-i -2+2*i]
>> plot(x, '+'), grid

% vettori ascisse-ordinate
>> x = [1 2 3 4 5 6 7 8]
>> y = 2*x
>> plot(x,y,'+'), grid

% esempio: funzione SIN(x)
>> time = [0:0.1:20];
>> omega = pi/2;
>> x = sin(omega * time);
>> plot(time,x), grid, title('funzione SIN(x)')
```

- Introduzione ai comandi per l'analisi di sistemi dinamici

La generica funzione di trasferimento (fdt) ***G(s)*** viene rappresentata mediante i vettori **num** e **den** contenenti rispettivamente i coefficienti del numeratore e del denominatore.

```
% Generazione di un vettore riga y contenente N punti equispaziati linearmente
nell'intervallo [x1,x2]
>> y=linspace(0,10,100);
```

➔ Un analogo risultato lo si ottiene con il comando

```
>> y=0:.1:10;
```

❑ [NUMc,DENc] = CLOOP(NUM,DEN,SIGN)

Calcola numeratore NUMc e denominatore DENc della fdt del sistema singolo ingresso singola uscita in catena chiusa ottenuto con contro-reazione unitaria e con segno SIGN.

❑ [Wn,Z] = DAMP(A)

Restituisce i vettori w_n e Z contenenti le pulsazioni naturali e gli smorzamenti delle radici di A. Se A è un vettore riga allora contiene i coefficienti di un polinomio della fdt. Se A è un vettore colonna allora contiene le radici.

❑ K = DCGAIN(NUM,DEN)

Calcola il guadagno stazionario del sistema continuo avente fdt G(s)=NUM(s)/DEN(s) e dove NUM e DEN contengono i coefficienti dei polinomi in potenze discendenti di *s*.

❑ IMPULSE(NUM,DEN)
Traccia la risposta all'impulso della fdt G(s) = NUM(s)/DEN(s).

❑ PRINTSYS(NUM,DEN,'s')
Stampa sullo schermo la fdt nella forma razionale, rapporto di due polinomi nella variabile s.

❑ [Y,X] = STEP(NUM,DEN,T)
Calcola la risposta al gradino di G(s) = NUM(s)/DEN(s).

❑ [NUM,DEN] = ZP2TF(Z,P,K)
Conversione dalla forma zeri-poli alla fdt.

❑ [Z,p,k] = TF2ZP(NUM,den)
Conversione dalla fdt alla forma zeri-poli:

$$H(s) = K \frac{(s-z1)(s-z2)...(s-zn)}{(s-p1)(s-p2)...(s-pn)}$$

▪ **Esempio: RISPOSTA LIBERA Di UN SISTEMA CONTINUO**

```
% SISTEMA DI EQUAZIONI DIFFERENZIALI
                .
%               X = A * X + B * U
%
%               Y = C * X + D * U


% CASO 1 : autovalori reali negativi
>> A = [ -1 1;3 -10];
>> B = [1;1];
>> C = [0 1];
>> D = [0];

% POLINOMIO CARATTERISTICO
>> P = poly(A)
P =
     1    11     7

% AUTOVALORI DEL POLINOMIO CARATTERISTICO
>> E = eig(A)
E =
   -0.6782
  -10.3218
>> plot(E, [0 0], '*'), grid

% RISPOSTA LIBERA
>> X0 = [10 100]; (condizioni iniziali)
>> initial(A, B, C, D, X0)
➔ disegna la risposta libera del sistema


% CASO 2 : autovalori reali positivi
>> A = [ 0 10;30  2];
>> B = [1;1];
>> C = [0 1];
```

```
>> D = [0];

% POLINOMIO CARATTERISTICO
>> P = poly(A)
P =
     1    -2  -300

% AUTOVALORI DEL POLINOMIO CARATTERISTICO
>> E = eig(A)
E =
  -16.3494
   18.3494
>> plot(E, [0 0], '*'), grid

% RISPOSTA LIBERA
>> X0 = [10 100];
>> initial(A, B, C, D, X0)


% CASO 3 : autovalori complessi coniugati a parte reale negativa
>> A = [-10 10;-30  2];
>> B = [1;1];
>> C = [0 1];
>> D = [0];

% POLINOMIO CARATTERISTICO
>> P = poly(A)
P =
    1.0000    8.0000  280.0000

>> Omega = sqrt(P(3))
Omega =
   16.7332
>> Sigma = P(2) / (2.0 * Omega)
Sigma =
    0.2390
% AUTOVALORI DEL POLINOMIO CARATTERISTICO
>> E = eig(A)
E =
  -4.0000 +16.2481i
  -4.0000 -16.2481i
>> plot(E, '*'), grid

% RISPOSTA LIBERA
>> X0 = [10 100];
>> initial(A, B, C, D, X0)


% CASO 4 : autovalori complessi coniugati a parte reale positiva
>> A = [ 10 10;-30  2];
>> B = [1;1];
>> C = [0 1];
>> D = [0];

% POLINOMIO CARATTERISTICO
>> P = poly(A)
P =
     1   -12   320
```

```
>> Omega = sqrt(P(3))
Omega =
   17.8885
>> Sigma = P(2) / (2.0 * Omega)
Sigma =
   -0.3354

% AUTOVALORI DEL POLINOMIO CARATTERISTICO
>> E = eig(A)
E =
   6.0000 +16.8523i
   6.0000 -16.8523i
>> plot(E, '*'), grid

% RISPOSTA LIBERA
>> X0 = [10 100];
>> initial(A, B, C, D, X0)
```

- Funzioni in MatLab

```
Esempio:    function  [mean, stdev] = stat(x)
            % Media e deviazione standard
            % Per ciascun vettore x, stat(x) restituisce due vettori
            % riga contenenti la media e la deviazione di ciascuna colonna
            [m  n] = size(x);
            if m == 1
                  m = n;     % caso di un vettore riga
            end
            mean = sum(x)/m;
            stdev = sqrt(sum(x.^ 2)/m - mean.^2);

Salvato questo file in 'stat.m', eseguire un comando MATLAB:
            [xm, xd] = stat(x)
```

- Formato dell'output

```
Tutti i calcoli in Matlab sono eseguiti in doppia precisione.

 format short     fixed point con 4 posti decimali (default)
 format long      fixed point con 14 posti decimali
 format short e   scientific notation con 4 posti decimali
 format long e    scientific notation con 15 posti decimali
```

- Esempio: calcolo di autovalori, autovettori e di `exp(At)`

```
Per calcolare autovalori ed autovettori di una matrice quadrata:
>> A = [ 10 10;-30  2];
>> [V,L]=eig(A)
V =
   0.4865 - 0.1155i   0.4865 + 0.1155i
        0 + 0.8660i        0 - 0.8660i
L =
   6.0000 +16.8523i        0
```

```
        0             6.0000 -16.8523i
```
dove V è la matrice dei vettori normalizzati di A, L è una matrice sulla cui diagonale compaiono gli autovalori di A.

- Esempio 1:

```
>> A = [ 4 -2 2; -1 3 1; 1 -1 5];
```

Applicando il comando [V,L]=eig(A) si ottengono gli autovalori 2,4,6 e gli autovettori [1 1 0]' [0 1 1]' [1 0 1]'. Attenzione che gli autovettori compaiono nella forma normalizzata: [0.7071 0.7071 0.0000]' etc., cioè [1 1 0]' diviso per la sua norma, la radice quadrata di 2.
La matrice V è la matrice modale, con T=inv(V). Ora si può verificare che inv(V)*A*V produce la matrice diagonale L.

- Esempio 2:

```
>> B=[3 -1 1; 2 0 2; -3 3 -1]
```

Questa matrice ha autovalori -2, 2, 2 e tre autovettori linearmente indipendenti. Per verificare questo è sufficiente controllare il rango della rank(V). La risposta dovrebbe essere 3. Mentre:

```
>> inv(V)*B*V
ans =
   -2.0000    0.0000    0.0000
    0.0000    2.0000    0.0000
    0.0000    0.0000    2.0000
```

fornisce la forma diagonale di A.

- Esempio 3:

```
>> C=[0 1 0; 0 0 1; 0 0 0]
```

Questa matrice ha tre autovalori uguali a 0, ed un solo autovettore linearmente indipendente. Infatti:

```
>> [V,L]=eig(C)
V =
    1.0000   -1.0000    1.0000
         0    0.0000    0.0000
         0         0    0.0000
L =
     0     0     0
     0     0     0
     0     0     0

>> rank(V)
ans =
     1
```

La matrice V ha rango 1, quindi non è invertibile. Allora la matrice C non può essere equivalente ad una forma canonica diagonale attraverso una trasformazione di similarità.

- Esempio 4:

```
Calcolo della matrice esponenziale exp(At) per un dato valore di t.
>> A = [ 4 -2 2; -1 3 1; 1 -1 5];
>> Et1=expm(A)          % t=1
Et1 =
  205.4089 -198.0199  198.0199
  -23.6045   30.9936   23.6045
  174.4153 -174.4153  229.0135

>> Et2=expm(A*2)        % t=2



Et2 =
  1.0e+004 *
    8.1405   -8.1350    8.1350
   -0.1463    0.1518    0.1463
    7.9887   -7.9887    8.2868
```

- Simulazione di sistemi lineari

Il comando Matlab ‘lsim’ produce la soluzione numerica dell’equazione di stato di un sistema date le condizioni iniziali e il segnale d’ingresso. Per vedere come funziona il comando:

```
>>help lsim
```

Bisogna definire le matrici A,B,C,D della rappresentazione in variabili di stato, la scala temporale T, il controllo U e la condizione iniziale x0. Definiamo T

```
>> T = 0:0.01:9.99;
```

Questo produce una sequenza di campioni temporali da 0, con incremento 0.01 (1/100 sec). Usiamo 9.99 anzichè 10 per avere esattamente 1000 elementi in T, e non 1001.

Il segnale di controllo in ingresso può essere facilmente definito come una sequenza di campioni (tempo-discreto) nell’intervallo da 0 a 9.99 . Ad esempio (si usi plot(T,u) dopo ogni istruzione per disegnare l’andamento)

```
>> u=sin(pi*T);

>> u=ones(size(T));

>> u=zeros(size(T));

>> u=exp(T);

>> u=exp(-3*T).*cos(2*pi*T);

>> u=sign(sin(pi/5*T));

>> u=1/2*(ones(size(T))-sign(sin(pi/5*T)));
```

```
>> U=u';
```

Se esistono più segnali d’ingresso allora si definiscono u1 e u2 e poi

```
>> U=[u1;u2]';
```

- Esempio 1:

Simuliamo il sistema:

```
>> A = [ 0 omega;-omega 0]
>> B = [ 0;1 ]
>> C = [ 1 1]
>> D = [ 0 ]
```

per omega=pi, con ingresso u(t) = 1 per tutti t > 0 e x0 = [1 0]'. Definiamo

```
>> u = ones(size(T));
>> U=u';
>> [Y,X]=lsim(A,B,C,D,U,T,x0);
>> plot(T,Y)

>> y=1/omega*(1+(omega-1)*(cos(omega*T)-sin(omega*T)));
>> plot(T,Y,'b',T,y,'r')
```

La curva rossa è la soluzione analitica mentre quella blu è quella calcolata con lsim’ (dovrebbero coincidere).

Ora poniamo U=0*U ed eseguiamo ‘lsim’. Dovremmo ottenere Y come cos(pi*T) - sin(pi*T) (la soluzione zero-input).

- Esempio 2:

```
>> A=[ - 1 1; -9 -1];
>> B=[0 1]';
>> C=[ 1 1]';
>> D=[0];

>> x0=[1 0]';
>> u=zeros(size(T));

>> u(1)=100;
```

Questa definizione di u produce un’approssimazione dell’impulso delta(t). u(1) lsim’ come un impulso di ampiezza 100 e larghezza 1/100. Si risolva l’esercizio confrontando le soluzione con quella ottenuta analiticamente.

- Esempio 3:

```
>> A = [0 1 0; 0 0 1; -2 -4 -3];
>> B = [1 0; 0 1; -1 1];
>> C = [0 1 -1; 1 2 1];
>> D = [ 0 0 ; 0 0];
```

```
>> x0=[1 0 0 ]';
>> u1=ones(size(T));
>> u2=u1;

>> U=[u1;u2]';
```

Si determini la traiettoria dello stato. Si confronti con la soluzione analitica:

```
>> x1 = (-2 +7/2*exp(T)-1/2*cos(T)-3/2*sin(T)).*exp(-T);
>> x2 = (2 -exp(T)-cos(T)+2*sin(T))*exp(-T);
>> x3 = (-2 -exp(T)+3*cos(T)-sin(T))*exp(-T);
```

Sia Xa=[x1;x2;x3] e Ya=C*Xa. Si confronti Ya con Y.

Nota: per confrontare le due funzioni (sequenze) si può calcolare la massima differenza tra:

```
>> max(x1-X(:,1)')
```

etc., per le due rimanenti variabili di stato.