Introduzione all'utilizzo del software di calcolo MATLAB

Introduzione all'uso del software di calcolo MATLAB

Questa dispensa non è da considerarsi esaustiva nell'analisi delle funzionalità offerte dal software di calcolo MATLAB, ma costituisce una introduzione all'utilizzo delle routine e dei comandi che in tale pacchetto sono maggiormente utili per l'analisi delle prestazioni e la progettazione dei sistemi di controllo automatico. Per una conoscenza più approfondita si rimanda ai manuali di riferimento del software ed a testi specifici.

MATLAB (MATrix LABoratory) è un programma interattivo di calcolo scientifico e ingegneristico. È un linguaggio di programmazione ad alto livello per la soluzione numerica di problemi ed il tracciamento di grafici, ed è basato sull'utilizzo di dati di tipo matriciale (scalari, vettori, matrici a due o più dimensioni). Include diversi toolbox, ossia una collezione di file speciali detti m-file o file con estensione .m, che estendono le funzionalità del programma di base o nucleo. In particolare, siamo interessati all'uso del nucleo di MATLAB insieme al Control System Toolbox, che permette di analizzare e sintetizzare sistemi di controllo, e al pacchetto Simulink, che fornisce un'interfaccia grafica per tali sistemi. MATLAB funziona su diverse piattaforme, quali i personal computer, i sistemi Unix, i computer Macintosh, e indifferentemente dal sistema operativo l'interazione dell'utente con MATLAB è la stessa. Nel seguito faremo riferimento specificatamente alla versione MATLAB 4.2.

Una volta lanciato il file eseguibile (aprendo matlab.exe o cliccando sull'icona di MATLAB), si apre la finestra principale di MATLAB detta finestra di comando (MATLAB Command Window) nella quale appare un "prompt" (») che indica che il sistema è pronto e in attesa di comandi da interpretare. I comandi accettati in questo ambiente corrispondono a file con l'estensione .m, siano essi funzioni di MATLAB "built-in", incluse nelle directory previste dal percorso di ricerca (queste sono consultabili aprendo il file matlabrc.m), o sequenze di comandi che l'utente può costruire e salvare in propri programmi.

Per cominciare, si può eseguire uno dei comandi suggeriti all'apertura della finestra principale per iniziare a conoscere MATLAB (intro, demo, help help, ecc.). In questa dispensa sono riportati alcuni dei comandi più utili per l'analisi e la sintesi dei sistemi di controllo automatico. Essi sono impartiti nella finestra principale di MATLAB, secondo una sintassi consultabile con l'help in linea e che in genere prevede argomenti d'ingresso e argomenti d'uscita, questi ultimi opzionali:

```
» [au1, au2, …, aui, …]=nome_del_comando(ai1, ai2, … aii, …)
```

dove au_i è l'i-esimo argomento d'uscita e ai_i l'i-esimo argomento di ingresso. Per consultare l'aiuto fornito sulla sintassi e la funzione di ciascun comando, occorre digitare dopo il prompt:

```
» help nome_del_comando
```

Una sessione tipica di MATLAB utilizza diversi oggetti che permettono all'utente di interagire con il programma: le dichiarazioni con variabili, le matrici, i grafici, gli scripts.

1. Dichiarazioni e variabili

Le dichiarazioni sono nella forma

```
» Variabile=espressione
```

dove » è il prompt di MATLAB, che indica che il programma è pronto al calcolo. Dopo una dichiarazione è necessario premere il tasto invio o carriage return. L'uso del punto e virgola al termine del comando evita la scrittura del valore delle variabili dopo la dichiarazione. Nel seguito riportiamo alcune operazioni effettuate dalla finestra di comando di MATLAB, unitamente ai risultati.

```
» A=3;
» A=3

A =

     3
```

MATLAB può essere usato come un semplice calcolatore. Le operazioni possibili sono:

+ (addizione)
- (sottrazione)
* (moltiplicazione)
/ (divisione)
^ (elevamento a potenza)

L'uso delle parentesi permette di variare l'ordine di esecuzione delle stesse.

```
» 12.4/6.9

ans =

    1.7971

» (3+5)/4

ans =

    2
```

Gli operatori più importanti sono consultabili digitando

```
» help ops

  Operators and special characters.

  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

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

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

Special characters.
  colon      - Colon                               :
  paren      - Parentheses and subscripting      ( )
  paren      - Brackets                          [ ]
  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

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.
```

```
  Set operators.
    union      - Set union.
    unique     - Set unique.
    intersect  - Set intersection.
    setdiff    - Set difference.
    setxor     - Set exclusive-or.
    ismember   - True for set member.

  See also ARITH, RELOP, SLASH.
```

Vi sono poi molte funzioni elementari disponibili. Ne seguono alcune.

```
» help elfun

  Elementary math functions.

  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.

  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.
Complex.
    abs         - Absolute value.
    angle       - Phase angle.
    complex     - Construct complex data from real and imaginary
                  parts.
    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.

  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.
```

MATLAB è un software case sensitive, che distingue cioè i caratteri in maiuscolo da quelli in minuscolo, quindi le variabili M e m non sono uguali.

```
» M=2

M =

     2

» m=3

m =

     3

» who

Your variables are:

M         m
```

MATLAB ha alcune variabili predefinite, quali pi (pari al valore numerico π), Inf (pari a $+\infty$), Nan (Not a number, risultato di operazioni non definite, come 0/0), i e j (entrambe pari a $\sqrt{-1}$). Le variabili predefinite possono essere riscritte, anche se non è consigliabile, quindi i può essere ad esempio utilizzato come indice in un ciclo di programmazione.

```
» 0/0
Warning: Divide by zero.

ans =
```

```
   NaN

» who

Your variables are:

A         M         ans       m

» WHO
??? Undefined variable or capitalized internal function WHO; Caps
Lock may be on.

» Who
??? Undefined variable or capitalized internal function Who; Caps
Lock may be on.

» whos
  Name      Size          Bytes  Class

  A         1x1               8  double array
  M         1x1               8  double array
  ans       1x1               8  double array
  m         1x1               8  double array

Grand total is 4 elements using 32 bytes

» clear A
» who

Your variables are:

M         ans       m

» clear all
» who
```

Il formato predefinito delle variabili è decimale con quattro cifre dopo la virgola, ma con il comando 'format' si possono utilizzare più cifre significative o rendere il formato di tipo esponenziale, ossia basato sulle potenze.

```
» pi

ans =

    3.1416

» format long; pi

ans =

   3.14159265358979
```

```
» format long e; pi

ans =

    3.141592653589793e+000

» format short e; pi

ans =

  3.1416e+000

» format short; pi

ans =

    3.1416
```

MATLAB è in grado di trattare numeri complessi per mezzo delle variabili predefinite i e j. Vediamo in seguito alcuni esempi di definizione di numeri e matrici complessi validi.

```
» z1=3+4*i

z1 = 3.0000 + 4.0000i

» z2=-5+6j

z2 = -5.0000 + 6.0000i

» z3=-5+8*j

z3 = -5.0000 + 8.0000i

» 0.001*exp(2*i)

ans =

 -4.1615e-004 +9.0930e-004i
```

2. Matrici

MATLAB è l'acronimo di Matrix Laboratory, dunque l'elemento fondamentale di calcolo è la matrice. Vettori e scalari sono infatti solo delle matrici particolari. Vediamo di seguito come eseguire la dichiarazione delle matrici, nonché alcune delle operazioni basilari tra esse.

```
» A=[1 -4*j sqrt(2);log(-1) sin(pi/2) cos(pi/3); asin(0.5)
acos(0.8) exp(0.8)]

A =
```

```
   1.0000                  0 - 4.0000i   1.4142
        0 + 3.1416i   1.0000             0.5000
   0.5236             0.6435             2.2255

» det(A)
ans =

 -26.8037 + 1.8118i

» inv(A)

ans =

  -0.0707 - 0.0048i  -0.0114 - 0.3329i   0.0475 + 0.0778i
  -0.0273 + 0.2590i  -0.0552 - 0.0037i   0.0297 - 0.1637i
   0.0245 - 0.0738i   0.0186 + 0.0794i   0.4296 + 0.0290i

» B=[1 0 0;0 1 0;0 0 1]

B =

     1     0     0
     0     1     0
     0     0     1

» A+b
??? Undefined function or variable 'b'.
» A+B

ans =

   2.0000                  0 - 4.0000i   1.4142
        0 + 3.1416i   2.0000             0.5000
   0.5236             0.6435             3.2255

» c=[1;2;0]

c =

     1
     2
     0

» A*c

ans =

   1.0000 - 8.0000i
   2.0000 + 3.1416i
   1.8106
```

```
» A'
ans =

   1.0000                  0 - 3.1416i   0.5236
        0 + 4.0000i   1.0000             0.6435
   1.4142             0.5000             2.2255

» d=[0; 2;3]

d =

     0
     2
     3

» c'*d

ans =

     4

» c*d'

ans =

     0     2     3
     0     4     6
     0     0     0
```

Oltre alle solite operazioni tra matrici, esistono anche le operazioni dette di array, che operano sulle matrici elemento per elemento.

```
» e=[1;2;3];f=[-6;7;10];
» e.*f

ans =

    -6
    14
    30

» e.^2

ans =

     1
     4
     9

» A.^2

ans =
```

```
    1.0000  -16.0000    2.0000
   -9.8696    1.0000    0.2500
    0.2742    0.4141    4.9530

» A^2

ans =

  14.3069             0.9100 - 8.0000i   4.5616 - 2.0000i
   0.2618 + 6.2832i  13.8881             1.6128 + 4.4429i
   1.6889 + 2.0216i   2.0756 - 2.0944i   6.0153
```

È possibile definire anche matrici ad elementi complessi.

```
» G=[1 2;3 4]+i*[5 6;7 8]

G =

   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 7.0000i   4.0000 + 8.0000i

» H=[1+5*i 2+6*i;3+7*i 4+8*i]

H =

   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 7.0000i   4.0000 + 8.0000i
```

MATLAB permette di indirizzare gli elementi di una matrice in modo piuttosto flessibile. Vediamo alcuni esempi di indirizzamento di elementi di una matrice di numeri casuali, generata attraverso la funzione 'rand'.

```
» A=rand(7,7)

A =

0.5751    0.6831    0.1901    0.0841    0.7275    0.2731    0.2393
0.4514    0.0928    0.5869    0.4544    0.4784    0.2548    0.0498
0.0439    0.0353    0.0576    0.4418    0.5548    0.8656    0.0784
0.0272    0.6124    0.3676    0.3533    0.1210    0.2324    0.6408
0.3127    0.6085    0.6315    0.1536    0.4508    0.8049    0.1909
0.0129    0.0158    0.7176    0.6756    0.7159    0.9084    0.8439
0.3840    0.0164    0.6927    0.6992    0.8928    0.2319    0.1739

» A(1:5,3)

ans =

    0.1901
    0.5869
    0.0576
```

```
    0.3676
    0.6315

» A(1:5,:)

ans =

0.5751    0.6831    0.1901    0.0841    0.7275    0.2731    0.2393
0.4514    0.0928    0.5869    0.4544    0.4784    0.2548    0.0498
0.0439    0.0353    0.0576    0.4418    0.5548    0.8656    0.0784
0.0272    0.6124    0.3676    0.3533    0.1210    0.2324    0.6408
0.3127    0.6085    0.6315    0.1536    0.4508    0.8049    0.1909

» A(1:5,4:7)

ans =

    0.0841    0.7275    0.2731    0.2393
    0.4544    0.4784    0.2548    0.0498
    0.4418    0.5548    0.8656    0.0784
    0.3533    0.1210    0.2324    0.6408
    0.1536    0.4508    0.8049    0.1909

» A(3,4)

ans =

    0.4418

» A(:,[3 5 7])

ans =

    0.1901    0.7275    0.2393
    0.5869    0.4784    0.0498
    0.0576    0.5548    0.0784
    0.3676    0.1210    0.6408
    0.6315    0.4508    0.1909
    0.7176    0.7159    0.8439
    0.6927    0.8928    0.1739
```

Esistono varie matrici di utilità, tra cui le più utili sono 'zeros' (matrice tutta nulla), 'ones' (matrice con elementi tutti unitari), 'eye' (matrice identità) e 'rand' (matrice di elementi casuali nell'intervallo [0, 1]).

```
» zeros(5)

ans =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
     0     0     0     0     0
     0     0     0     0     0

» ones(3,4)

ans =

     1     1     1     1
     1     1     1     1
     1     1     1     1

» eye(2,2)

ans =

     1     0
     0     1

» rand(3)

ans =

    0.9616    0.5485    0.0493
    0.0589    0.2618    0.5711
    0.3603    0.5973    0.7009
```

Da ricordare in questo paragrafo è che i polinomi vengono trattati come dei vettori, definiti dai coefficienti del polinomio stesso in ordine decrescente di potenza, fino al termine noto. In particolare, la funzione 'roots' restituisce le radici di un polinomio, mentre 'poly' costruisce i coefficienti di un polinomio a partire dalle sue radici. La funzione 'polyval' determina il valore di un polinomio in un punto, mentre le funzioni 'conv' e 'deconv' permettono di eseguire tra polinomi le operazioni di moltiplicazione e divisione, quest'ultima restituendo un quoziente q e un resto r.

```
» p=[1 0 -6 3]

p =

     1     0    -6     3

» r=roots(p)

r =

   -2.6691
    2.1451
    0.5240

» pp=poly(r)

pp =

    1.0000    0.0000   -6.0000    3.0000
```

```
» valore=polyval(p,2)

valore =

    -1

» a=[1,2,3]

a =

     1     2     3

» b=[4 5 6]

b =

     4     5     6

» c=conv(a,b)

c =

     4    13    28    27    18

» [q,r]=deconv(c,a)

q =

     4     5     6

r =

     0     0     0     0     0
```

3. Grafici

I grafici giocano un ruolo importante nella analisi e sintesi dei sistemi di controllo. Il comando principale per la rappresentazione di grafici bidimensionali in MATLAB è plot(x,y), dove x e y sono vettori della stessa dimensione.

```
» help plot

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

PLOT(Y) plots the columns of Y versus their index.
If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
```

```
In all other uses of PLOT, the imaginary part is ignored.
Various line types, plot symbols and colors may be obtained with
PLOT(X,Y,S) where S is a character string made from one element
from any or all the following 3 colunms:

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

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus
at each data point; PLOT(X,Y,'bd') plots blue diamond at each data
point but does not draw any line.

PLOT(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots defined by
the (X,Y,S) triples, where the X's and Y's are vectors or matrices
and the S's are strings.

For example, PLOT(X,Y,'y-',X,Y,'go') plots the data twice, with a
solid yellow line interpolating green circles at the data points.

The PLOT command, if no color is specified, makes automatic use of
the colors specified by the axes ColorOrder property.  The default
ColorOrder is listed in the table above for color systems where
the default is yellow for one line, and for multiple lines, to
cycle through the first six colors in the table. For monochrome
systems, PLOT cycles over the axes LineStyleOrder property.

PLOT returns a column vector of handles to LINE objects, one
handle per line.

The X,Y pairs, or X,Y,S triples, can be followed by
parameter/value pairs to specify additional properties of the
lines.

See also SEMILOGX, SEMILOGY,
LOGLOG, GRID, CLF, CLC, TITLE,
XLABEL, YLABEL, AXIS, AXES,
HOLD, COLORDEF, LEGEND, and
SUBPLOT.
```

Vediamo di seguito tre esempi di uso della procedura grafica plot. Nel primo si traccia una parte della funzione seno (figura 1), nel secondo si traccia ancora una funzione trigonometrica (figura 2), nel terzo si confronta questo grafico con la funzione seno (figura 3).

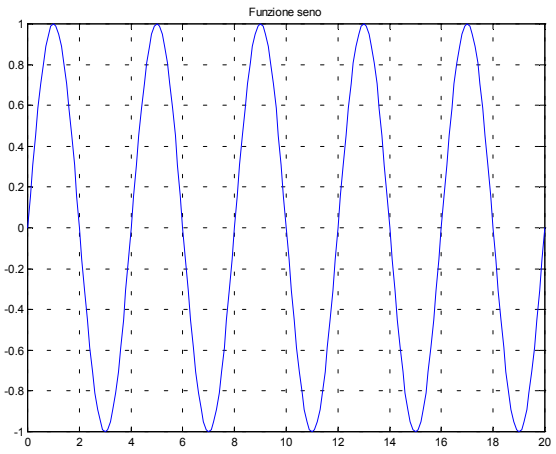


Figura 1. Grafico della funzione seno.

```
» tempo=[0:0.1:20];
» omega=pi/2;
» y=sin(omega*tempo);
» plot(tempo,y),grid,title('Funzione seno')
```

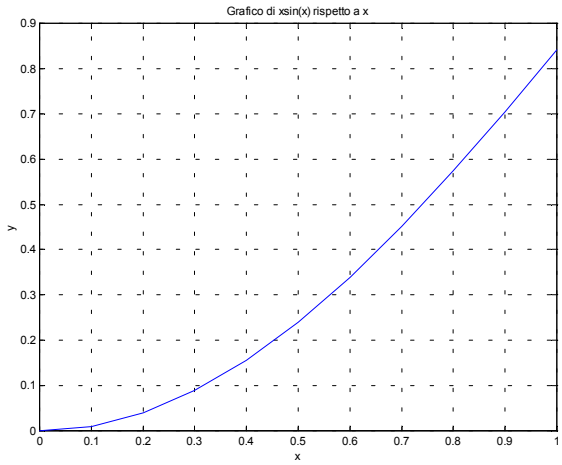


Figura 2. Grafico della funzione xsin(x).

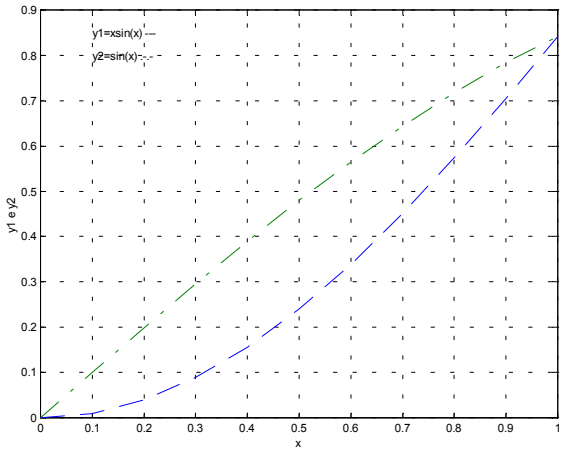


Figura 3. Grafico di senx e xsin(x).

```
» x=[0:0.1:1]'

x =

         0
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000
    0.6000
    0.7000
    0.8000
    0.9000
    1.0000

» y=x.*sin(x);
» plot(x,y)
» grid
» xlabel('x')
» ylabel('y')
» title('Grafico di xsin(x) rispetto a x')

» x=[0:0.1:1]';
» y1=x.*sin(x);y2=sin(x);
» plot(x,y1,'--',x,y2,'-.')
» xlabel('x'),ylabel('y1 e y2')
» grid
» text(0.1,0.85,'y1=xsin(x)---')
» text(0.1,0.8,'y2=sin(x) .-.-')
```

4. Scripts

I modi di interagire con MATLAB fino a questo punto illustrati sfruttano tutti il prompt, ossia vengono impartiti dalla cosiddetta 'command window', o finestra di comando. Se i comandi devono però essere ripetuti più volte o a distanza di tempo, è utile realizzare degli scripts, ossia degli elenchi di dichiarazioni e operazioni salvati in un file con estensione .m. Gli scripts sono dei file di testo in formato ASCII e sono creati e modificati con un editor di testo, che può anche essere quello fornito con lo stesso MATLAB.

Quando lo script è invocato dalla command line di MATLAB, esso viene eseguito e opera sul workspace, quindi anche sulle variabili in esso già eventualmente presenti.

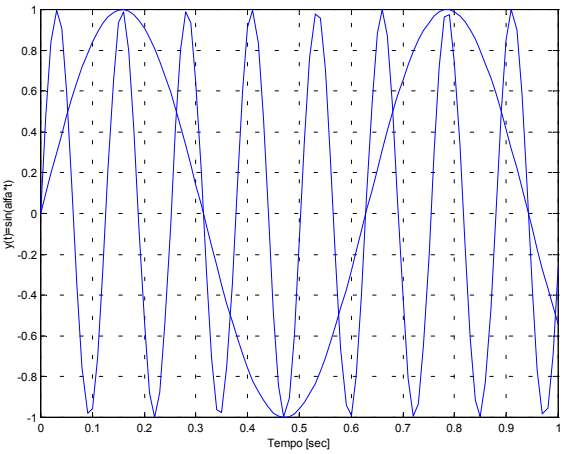
I commenti negli scripts sono preceduti dal simbolo %.

Vediamo di seguito un esempio di script, chiamato plotdata.m, e del suo uso dalla finestra di comando.

```
% Questo è uno script per disegnare la funzione y=sin(alfa*t).
%
% Il valore di alfa deve esistere nel workspace
% prima di invocare lo script.
%
t=[0:0.01:1];
y=sin(alfa*t);
plot(t,y)
xlabel('Tempo [sec]')
ylabel('y(t)=sin(alfa*t)')
grid
```

Invocando con diversi valori di alfa lo script plotdata.m dalla finestra di comando si ottengono i risultati in figura.

```
» alfa=10;plotdata
» hold
» alfa=50;plotdata
```



5. Funzioni

La potenza di MATLAB risiede nelle sue numerosissime funzioni. Alcune di esse sono built-in (ossia presenti nel nucleo di MATLAB), altre sono disponibili in pacchetti (toolbox) di m-files. Per esempio, le funzioni di interesse principale per i controlli automatici sono racchiuse nel Control toolbox. Infine, l'utente può aggiungere le proprie funzioni scrivendole sotto forma di m-file.

Una funzione può prevedere uno o più parametri di ingresso, che non vengono modificati da MATLAB, e uno o più parametri di uscita. Per esempio, la funzione MATLAB 'log' prevede un ingresso e una uscita, la funzione 'conv' due ingressi e una uscita, la funzione 'min' prevede uno o più ingressi e una o più uscite.

Un m-file che individua una funzione, anche detto function-file, inizia con una istruzione che contiene la parola function. Nella stessa riga vengono dichiarati uno o più parametri di uscita, il nome della function e uno o più parametri di ingresso.

Una function differisce da uno script per il fatto che le istruzioni contenute in un function-file lavorano su variabili locali, e non hanno accesso alle variabili definite nel workspace globale. Le variabili che devono essere manipolate dalle istruzioni contenute in una funzione debbono quindi essere passate come argomenti di ingresso, mentre le variabili che devono rimanere accessibili nel workspace al termine dell'esecuzione della function devono essere restituite in uscita.

Riportiamo nel seguito un esempio di function-file che realizza la somma di due matrici.

```
function y=somma(a,b)
%funzione somma di due matrici a e b
[m1,n1]=size(a);
[m2,n2]=size(b);
if(m1~=m2)|(n1~=n2)
   'Errore: le matrici non hanno le stesse dimensioni'
   return
end
y=a+b;
```

Riportiamo inoltre i risultati ottenuti nella Command Window utilizzando tale funzione.

```
» A=[1 2;3 4]

A =

     1     2
     3     4

» B=[5 6;7 8]

B =

     5     6
     7     8

» somma(A,B)

ans =

     6     8
    10    12

» C=[1 2]

C =

     1     2

» somma(B,C)

ans =

Errore: le matrici non hanno le stesse dimensioni
```

6. Elenco dei principali comandi di MATLAB.

Riportiamo nel seguito alcuni dei principali comandi di MATLAB e del Control System Toolbox.

6.1. Alcuni comandi di ausilio

Nome	Funzione
`who`	elenca le variabili immagazzinate da MATLAB nello spazio di lavoro
`whos`	elenca le variabili correnti fornendo una serie di informazioni
`%`	consente di inserire linee di commenti, non interpretate come comandi
`..`	consente di continuare il comando sulla linea successiva
`what`	elenca i file .m nella directory di lavoro corrente
`what cartella`	elenca i file .m nella directory 'cartella'
`help`	fornisce un aiuto su directory o files:
`help comando`	fornisce aiuto sulla sintassi e funzione del comando
`dir`	elenca i file nella directory corrente
`cd cartella`	imposta la directory 'cartella' come corrente
`chdir cartella`	imposta la directory 'cartella' come corrente
↑	richiama le istruzioni precedenti
p↑	richiama le istruzioni precedenti che cominciano con la lettera p
`type nomefile`	lista il contenuto del file 'nomefile'; se non si fornisce un'estensione, questa viene assunta uguale a .m per default
`edit nomefile`	apre il file 'nomefile' nell'editor del MATLAB; se non si fornisce un'estensione, questa viene assunta uguale a .m per default
`! comando`	Esegue il comando DOS 'comando' dalla Command Window.

6.2. Comandi per le principali operazioni elementari

Operatore	Operazione	Esempio	Risultato
*	moltiplicazione	`2*3`	`ans = 6`
/	divisione destra	`5/2`	`ans = 2.5000`
\	divisione sinistra	`5\2`	`ans = 0.4000`
+	somma	`[1]+[2]`	`ans = 2`
-	sottrazione	`[1]-[2]`	`ans = -1`

6.3 Operazioni su vettori e matrici

Operatore	Sintassi	Funzione
'	`v', A'`	trasposizione del vettore v e della matrice A (se A è complessa si ottiene la trasposta coniugata di A)
*	`s*v, s*A`	prodotto di scalare s per vettore v o matrice A
*	`x*y`	prodotto scalare tra un vettore riga x ed uno colonna y

`*`	`A*B`	prodotto tra matrici A e B (numero di colonne di A deve essere uguale al numero di righe di B)
`'*`	`x'*y`	prodotto scalare tra due vettori colonna x ed y
`.*`	`x.*y`	prodotto, elemento per elemento, tra gli elementi di vettori equidimensionali
`/`	`A/B`	divisione destra tra le matrici A e B, equivale ad `A*inv(B)`
`\`	`A\B`	divisione sinistra tra le matrici A e B, equivale ad `inv(A)*B`
`^`	`s^p`	elevamento di uno scalare s alla potenza p
`.^`	`x.^p`	elevamento alla potenza p, elemento per elemento, di un vettore x
`:`	`A(r,:)`	individuazione della riga r (1≤r≤m) della matrice A ad m righe
	`A(:,c)`	individuazione della colonna c (1≤c≤n) della matrice A ad n colonne
	`x=vi:p:vf`	vettore costituito da tutti i numeri che vanno (in senso crescente o decrescente) dal valore iniziale vi a quello finale vf con passo p (positivo o negativo)
`inv`	`inv(A)`	inversa della matrice A
`conj`	`conj(A)`	coniugata della matrice o vettore A
`length`	`length(x)`	dimensione del vettore x
`norm`	`norm(x)`	norma del vettore x
`eye`	`eye(n)`	costruzione della matrice identità di ordine n
`eig`	`[x,D]=eig(A)`	autovalori ed autovettori della matrice A
`ones`	`ones(1,L)`	vettore riga di uno, di dimensione L
`zeros`	`zeros(1,L)`	vettore riga di zeri, di dimensione L

6.4 Comandi per il tracciamento di grafici

Nome	Sintassi	Funzione
`plot`	`plot(x,y)`	grafico di y in funzione di x (vettori equidimensionali)
`xlabel`	`xlabel('x')`	etichetta sull'asse delle ascisse
`ylabel`	`ylabel('y')`	etichetta sull'asse delle ordinate
`title`	`title('Nome')`	scrittura del titolo del grafico
`grid`	`grid`	sovrapposizione di griglie al grafico corrente
`text`	`text(x,y,'tt')`	testo a partire dal punto di coordinate x e y
`semilogx`	`semilogx(x,y)`	grafico semilogaritmico, ossia logaritmico sull'asse orizzontale di rappresentazione del vettore x e lineare sull'asse verticale di rappresentazione del vettore y
`semilogy`	`semilogy(x,y)`	grafico semilogaritmico, lineare sull'asse orizzontale di rappresentazione del vettore x e logaritmico sull'asse verticale di rappresentazione del vettore y
`loglog`	`loglog(x,y)`	grafico su scala bilogaritmica
`subplot`	`subplot(mnp)`	suddivisione della finestra grafica in m*n parti secondo una matrice m x n ed uso della parte p (corrispondente al p-esimo degli elementi ordinati per righe della

		matrice) per il grafico specificato nel comando successivo
`hold`	`hold on`	trattiene nella finestra corrente il grafico precedentemente tracciato in modo da poterne sovrapporre un altro
	`hold off`	disabilita la funzione di trattenuta del grafico
	`hold`	cambia lo stato della funzione da on a off o viceversa
`zoom`	`zoom on`	evidenzia una regione della finestra grafica
	`zoom off`	disabilita la funzione di zoom
	`zoom`	cambia lo stato della funzione da on a off e viceversa
`axis`	`axis([`x_1 x_2 y_1 y_2`])`	in un grafico stabilisce la scala per le ascisse x tra x_1 e x_2 e quella per le ordinate y tra y_1 e y_2

6.5 Comandi per il trattamento di polinomi

Nome	Sintassi	Funzione
`poly`	`poly(r)`	determinazione dei coefficienti delle potenze decrescenti nel polinomio le cui radici sono date dal vettore r
	`poly(A)`	determinazione degli N+1 coefficienti del polinomio caratteristico della matrice A
`roots`	`roots(p)`	determinazione delle radici del polinomio i cui coefficienti sono contenuti nel vettore p
`conv`	`conv(`p_1, p_2`)`	prodotto di convoluzione tra i due polinomi p_1 e p_2, non necessariamente di uguale dimensione
`deconv`	`deconv(`p_1, p_2`)`	divisione di convoluzione tra i polinomi p_1 e p_2

6.6 Comandi per la rappresentazione di funzioni di trasferimento o sistemi in variabili di stato

Nome	Sintassi	Funzione
`tf2zp`	`[z,p,k]=tf2zp(n,d)`	determinazione degli zeri, dei poli e della costante di guadagno della funzione di trasferimento espressa tramite il vettore n dei coefficienti del numeratore e il vettore d dei coefficienti del denominatore
`zp2tf`	`[n,d]=zp2tf(z,p,k)`	determinazione dei coefficienti dei polinomi n a numeratore e d a denominatore della funzione di trasferimento fattorizzata con gli zeri di z, i poli di p e la costante k di guadagno statico
`pzmap`	`pzmap(n,d)`	mappa poli-zeri della funzione di trasferimento con numeratore n e denominatore d
	`pzmap(p,z)`	mappa poli-zeri della funzione di trasferimento con i poli precisati dal vettore p e gli zeri dal vettore z
	`[p,z]=pzmap(n,d)`	determinazione di zeri e poli senza rappresentare la mappa
`residue`	`[r,p,k]=residue(n,d)`	determinazione dei residui in r, dei poli in p e delle costanti dei termini impulsivi in k (quando m≥n, con m

		grado del numeratore num e n grado del denominatore den) a partire da num e den
	`[n,d]=residue(r,p,k)`	determinazione di numeratore e denominatore, noti i residui, i poli e le costanti dei termini impulsivi
`tf2ss`	`[A,B,C,D]=tf2ss(n,d)`	determinazione di una rappresentazione in variabili di stato nella forma canonica di controllo dalla funzione di trasferimento rappresentata dal numeratore n e dal denominatore d
`ss2tf`	`[n,d]=ss2tf(A,B,C,D,i)`	determinazione del numeratore e del denominatore della funzione di trasferimento dall'i-esimo ingresso

6.7 Comandi per la determinazione delle risposte dei sistemi

Nome	Sintassi	Funzione
`step`	`step(n,d)`	risposta al gradino unitario
	`step(n,d,t)`	risposta al gradino unitario nell'intervallo t
	`step(A,B,C,D,i,t)`	risposta all'ingresso i a gradino unitario
`impulse`	`impulse(n,d)`	risposta all'impulso
	`impulse(n,d,t)`	risposta all'impulso nell'intervallo t
	`impulse (A,B,C,D,i,t)`	risposta all'ingresso i-esimo a impulso
`initial`	`initial(A,B,C,D,t,x0)`	risposta alle condizioni iniziali nell'intervallo t
`lsim`	`lsim(n,d,u,t)`	risposta all'ingresso u nell'intervallo t
	`lsim(A,B,C,D,u,t,x0)`	risposta a condizioni iniziali x0 e ad ingresso u

6.8 Comandi per il tracciamento dei diagrammi di Bode

Nome	Sintassi	Funzione
`bode`	`bode(n,d)`	diagrammi di Bode della funzione di trasferimento avente numeratore n e denominatore d
	bode(n,d,ω)	diagrammi di Bode nell'intervallo di frequenze ω
	`[a,f]=bode(n,d,w)`	assegnazione delle ampiezze e delle fasi valutate per le frequenze definite in ω ai vettori a e f
	`dB=20*log10(a)`	determinazione delle ampiezze in dB
`fbode`	`fbode(n,d)`	diagramma di Bode più veloce ma meno accurato
`logspace`	`w=logspace(`e_1`,`e_2`,N)`	determinazione di N punti equispaziati su scala logaritmica tra 10^{e1} e 10^{e2} (50 se N non è specificato)
`margin`	`[G,P,`w_p`,`w_c`]=margin(n,d)`	determinazione di margini e frequenze di cross-over
	`margin(n,d)`	rappresentazione sul diagramma di Bode dei margini di ampiezza (G) e fase (P), con le frequenze di cross-over del guadagno (ω_c) e della fase (ω_p)

6.9 Comandi per i luoghi delle radici, diagrammi di Nyquist e di Nichols

Nome	Sintassi	Funzione

`rlocus`	`rlocus(n,d)`	luogo delle radici della funzione di trasferimento avente numeratore con i coefficienti nel vettore n e denominatore con i coefficienti nel vettore d
	`rlocus(n,d,k)`	luogo delle radici per i valori del guadagno specificati nel vettore k
`rlocfind`	`[k,p]=rlocfind(n,d)`	selezione di un punto del luogo ed automatica determinazione del guadagno e dei poli in anello chiuso corrispondenti, posti rispettivamente in k e nel vettore p
`sgrid`	`sgrid(d,`w_n`)`	tracciamento nel piano s di due semirette dall'origine, corrispondenti al fattore di smorzamento δ, e di una semicirconferenza corrispondente alla frequenza naturale ω_n
	`sgrid`	griglia di semirette e semicirconferenze nel piano s
`nyquist`	`nyquist(n,d)`	diagramma di Nyquist della funzione di trasferimento con numeratore n e denominatore d
`nichols`	`nichols(n,d)`	diagramma di Nichols della funzione di trasferimento avente numeratore n e denominatore d

7 Bibliografia

A. Cavallo, R. Setola, F. Vasca, Guida operativa a MATLAB, Simulink e Control Toolbox, 1994, Liguori.
H. Saadat, Computational Aids in Control Systems Using MATLAB, 1993, Mc Graw-Hill.
M. Tibaldi, Note introduttive a MATLAB e Control System Toolbox, 1993, Progetto Leonardo.
N.S. Nise, Control Systems Engineering, seconda edizione, 1995, Benjamin Cummings.