Introduzione a Matlab

Plinio Gatto

10 febbraio 2007

Plinio Gatto < pli@autistici.org >

Indice generale

Introduzione

Matlab è un ambiente di calcolo e un linguaggio di programmazione ad alto livello per lo sviluppo di algoritmi, analisi di dati e computazione numerica.
Questo documento è una semplice guida introduttiva all'uso di Matlab, verranno spiegati i comandi di base per l'utilizzo di variabili, matrici e funzioni. Inoltre sono presenti alcune note per lo studio delle funzioni di trasferimento e per la realizzazione di script.

Per chi non avesse la possibilità di utilizzare Matlab può usare Octave, un software simile a Matlab rilasciato sotto licenza GNU.

Questa guida è in fase di sviluppo, chiunque volesse collaborare o segnalare errori mi può contattare all'indirizzo email pli@autistici.org.

Variabili

Se vogliamo assegnare il valore 2 alla variabile a si utilizza il comando *a=2* seguito da invio, l'output sarà:

```
>> a=2

a =

     2

>>
```

L'assegnazione delle variabile è case sensitive, quindi $a \neq A$.

Nell'ambiente di lavoro sono già definite alcune variabili come eps o i:

```
>> eps

ans =

  2.2204e-016

>> i

ans =

        0 + 1.0000i

>>
```

Per visualizzare le variabili definite si utilizza l'istruzione *who*:

```
>> who

Your variables are:

a    ans

>>
```

La variabile ans contiene il risultato dell'ultimo comando chiamato.

Per cancellare una variabile si utilizza il comando *clear* seguito dal nome della variabile. Nel caso in cui la variabile che viene passata al comando clear è una variabile particolare come i, questa viene rigenerata. Se al comando clear non viene passata nessuna variabile, tutte le variabili definite verranno cancellate.

Se vogliamo avere delle informazioni su un particolare comando basta digitare *help* seguito dal nome del comando, ad esempio:

```
>> help who

 WHO    List current variables.
    WHO lists the variables in the current workspace.
    WHOS lists more information about each variable.
    WHO GLOBAL and WHOS GLOBAL list the variables in the global workspace.
    WHO -FILE FILENAME lists the variables in the specified .MAT file.

    WHO ... VAR1 VAR2 restricts the display to the variables specified.
    The wildcard character '*' can be used to display variables that
    match a pattern.  For instance, WHO A* finds all variables in the
    current workspace that start with A.

    Use the functional form of WHO, such as WHO('-file',FILE,V1,V2),
    when the filename or variable names are stored in strings.

    S = WHO(...) returns a cell array containing the names of the
    variables in the workspace or file. You must use the functional
    form of WHO when there is an output argument.

    See also WHOS.

>>
```

Se invece cerchiamo un comando che contiene una particolare parola possiamo utilizzare *lookfor* parola. Verrà visualizzato l'elenco dei comandi che contengono nella loro descrizione la parola cercata:

```
>> lookfor zero
ALL    True if all elements of a vector are nonzero.
ANY    True if any element of a vector is nonzero.
FIND   Find indices of nonzero elements.
ZEROS  Zeros array.
FIX    Round towards zero.
...
...
...
ZIZEROPAD  Zero pad from initial conditions.
>>
```

Matrici

È possibile definire delle matrici, ad esempio la definizione della matrice $A=\begin{bmatrix}2 & 3\\ 1 & 0\end{bmatrix}$ viene fatta in questo modo:

```
>> A=[2 3;1 0]

A =

     2     3
     1     0

>>
```

oppure:

```
>> A=[2 3
1 0]

A =

     2     3
     1     0

>>
```

Dopo la parentesi quadra vanno inseriti gli elementi della prima riga, per passare alla seconda riga si può andare a capo premendo invio oppure si può inserire un ; per dividere una riga da quella successiva. Gli elementi vanno separati da spazi oppure da virgole. Per inserire un numero decimale si utilizza il punto. Vediamo un altro esempio:

```
>> B=[2.3 4.5 0.1 ; 3 0 0 ]

B =

    2.3000    4.5000    0.1000
    3.0000         0         0

>>
```

Possiamo richiamare un singolo elemento di una matrice in questo modo:

```
>> B(1,2)

ans =

    4.5000

>>
```

Le operazioni tra le matrici si fanno utilizzando i comuni operatore + - * / , per trasporre la matrice si utilizza l'apice:

```
>> A'

ans =

     2     1
     3     0

>>
```

Gli operatori effettuano operazioni tra matrici, se vogliamo svolgere un'operazione tra una matrice e uno scalare, oppure elemento per elemento, si deve inserire un punto prima dell'operatore:

```
>> A^2

ans =

7     4
6     7

>> A.^2

ans =

1     4
9     1

>>
```

Per calcolare gli autovalori e gli autovettori di una matrice procediamo in questo modo:

```
>> [V,D]=eig(A)

V =

    0.6325   -0.6325
    0.7746    0.7746


D =

    3.4495         0
         0   -1.4495

>>
```

Il comando *eig()* fornisce una matrice contenente nelle colonne gli autovettori (nel nostro caso la matrice V) e una matrice che contiene i rispettivi autovalori sulla diagonale (nel nostro caso D).

Disegniamo una funzione: sin(t)

Per disegnare l'andamento di una funzione come sin(t) possiamo costruire un vettore che contiene i valori di t e poi creare un vettore y che contiene il seno di ogni elemento del vettore t. A questo punto possiamo plottare la funzione. Vediamo come procedere:

Costruiamo un vettore t che contiene gli elementi da 0 a 10 con passo 0.01:

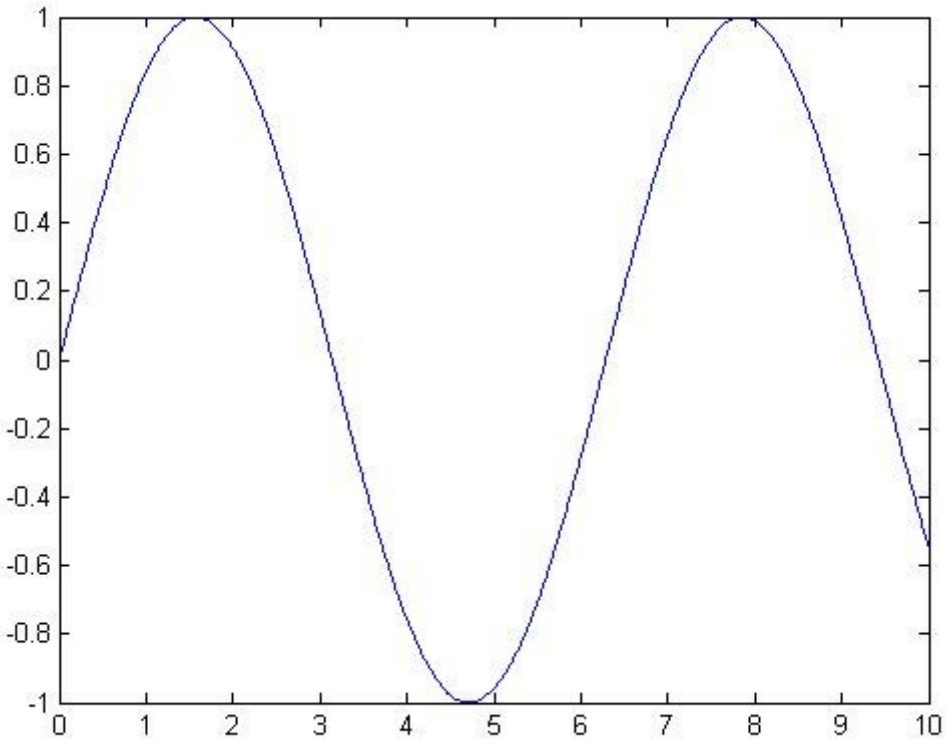
```
>> t=0:0.01:10

t =

Columns 1 through 5

            0    0.0100    0.0200    0.0300    0.0400

  Columns 6 through 10

    0.0500    0.0600    0.0700    0.0800    0.0900
...
...
...

  Columns 996 through 1000

    9.9500    9.9600    9.9700    9.9800    9.9900

  Column 1001

   10.0000

>>
```

Verranno elencati tutti gli elementi contenuti nel vettore t.

Creiamo ora il vettore y che contiene il seno di ogni elemento del vettore t:

```
>> y=sin(t)

y =

  Columns 1 through 5

          0    0.0100    0.0200    0.0300    0.0400

  Columns 6 through 10

    0.0500    0.0600    0.0699    0.0799    0.0899
...
...
...

  Columns 996 through 1000

   -0.5014   -0.5100   -0.5186   -0.5271   -0.5356

  Column 1001

   -0.5440

>>
```

A questo punto chiamando il comando *plot(t,y)* possiamo disegnare la funzione.



Strutture di controllo

Matlab rende disponibili le tre funzioni di controllo for, if, while, vediamo alcuni esempi:

if: se a è maggiore di 2 allora b=5 altrimenti b=3

```
a =

     1

>> if a>2 b=5; else b=3; end
>> b

b =

     3

>>
```

while: inserisce nel vettore c numeri da 1 a 10

```
>> k=1;c=0;
>> while (k<=10); c(k)=k; k=k+1; end
>> c

c =

     1     2     3     4     5     6     7     8     9    10

>>
```

for: inserisce nel vettore c numeri da 1 a 10

```
>> m=10

m =

    10

>> for i=1:m c(i)=i; end
>> c

c =

     1     2     3     4     5     6     7     8     9    10

>>
```

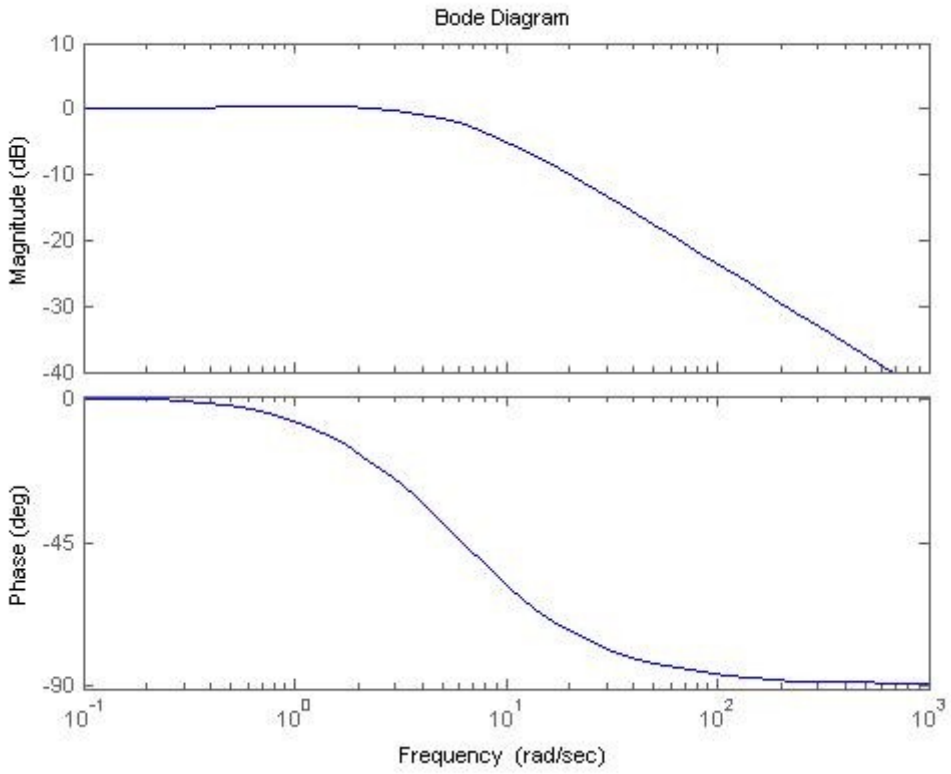
Funzioni di trasferimento

Le funzioni di trasferimento possono essere utilizzate per valutare la risposta di un sistema ad un ingresso assegnato. Possiamo definire una funzione di trasferimento con il comando *tf()*, ad esempio:

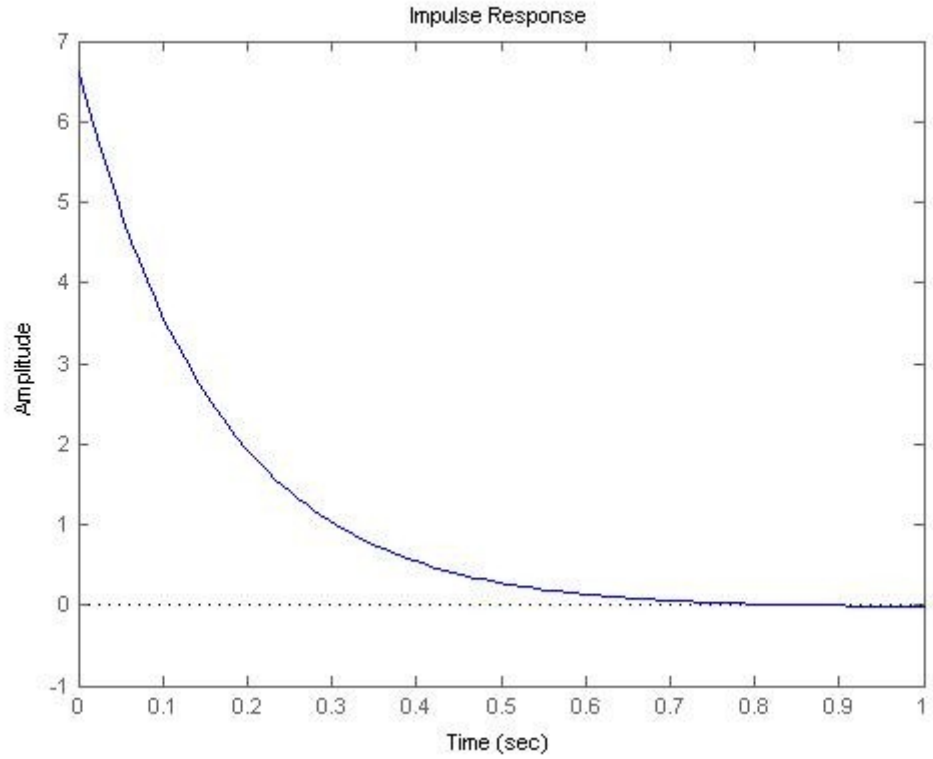
```
>> w=tf([2 1],[0.3 2 1])

Transfer function:
     2 s + 1
-----------------
0.3 s^2 + 2 s + 1

>> bode(w)
>>
```

La funzione tf() prende in ingresso un vettore con i coefficienti del numeratore e uno con i coefficienti del denominatore.
Il comando *bode(w)* ci permette di visualizzare il diagramma di Bode della funzione di trasferimento, inoltre per valutare la risposta all'impulso possiamo utilizzare il comando *impulse(w),* mentre per valutare la risposta al gradino possiamo utilizzare il comando *step(w).*

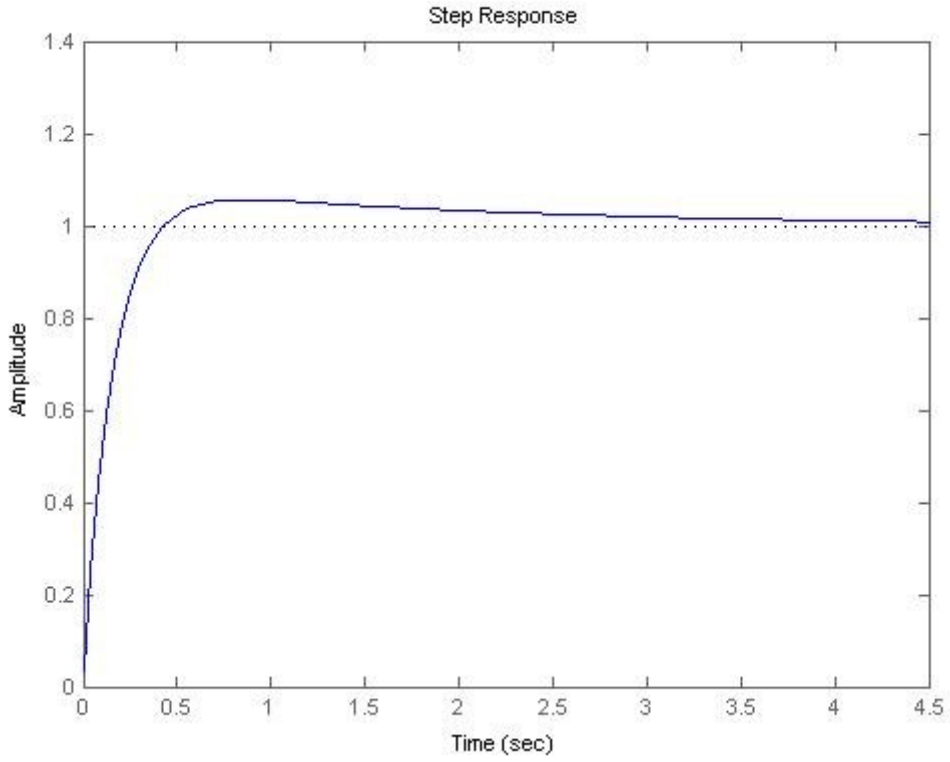
Output del comando bode(w):



Output del comando impulse(w):



Output del comando step(w):



Script

Matlab ci da la possibilità di scrivere degli script in modo tale da non dover ripetere i comandi per eseguire un certo tipo di lavoro. Ad esempio per plottare la funzione sin(t) si può scrivere, con un qualsiasi editor di testo, uno script con le istruzioni necessarie per plottare la funzione:

```
% Il simbolo di percentuale serve per inserire dei commenti all'interno
% dello script.
% Inoltre, se inserite un ; alla fine di ogni istruzione, viene
% disabilitato l'output

close all;        % Chiude tutte le finestre
clear all;        % Elimina tutte le variabili

t=0:0.01:10;      % Crea il vettore per le ascisse
y=sin(t);         % Crea il vettore con i valori della funzione
plot(t,y);        % Disegna la funzione
```

Una volta scritto lo script è sufficiente salvarlo con estensione .m cambiare la *Current Directory* di Matlab in modo tale da posizionarsi nella directory dove c'è lo script e richiamare il nome con il quale è stato salvato (senza l'estensione).
Ad esempio se lo script è stato salvato come plotsin.m è sufficiente eseguire il comando plotsin in Matlab.