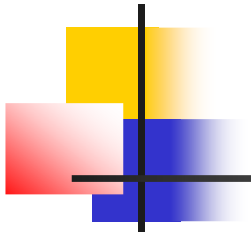
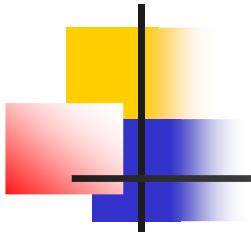


INTRODUZIONE ALL'USO DI MATLAB

Ing. Francesco Pierri
Ing. Alessandro Marino

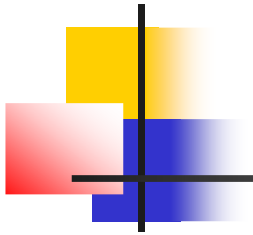


Che cosa è Matlab?

Matlab = MATrix LABoratory

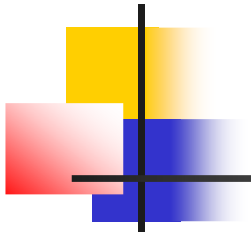
MATLAB è un sistema, basato sulle matrici, per la computazione tecnica.

Integra il calcolo, la visualizzazione e la programmazione in un ambiente di facile impiego in cui i problemi e le soluzioni sono espressi in notazione matematica familiare.



Vantaggi di Matlab

- Facilità e velocità di programmazione
- Potenza di calcolo
- Versatilità

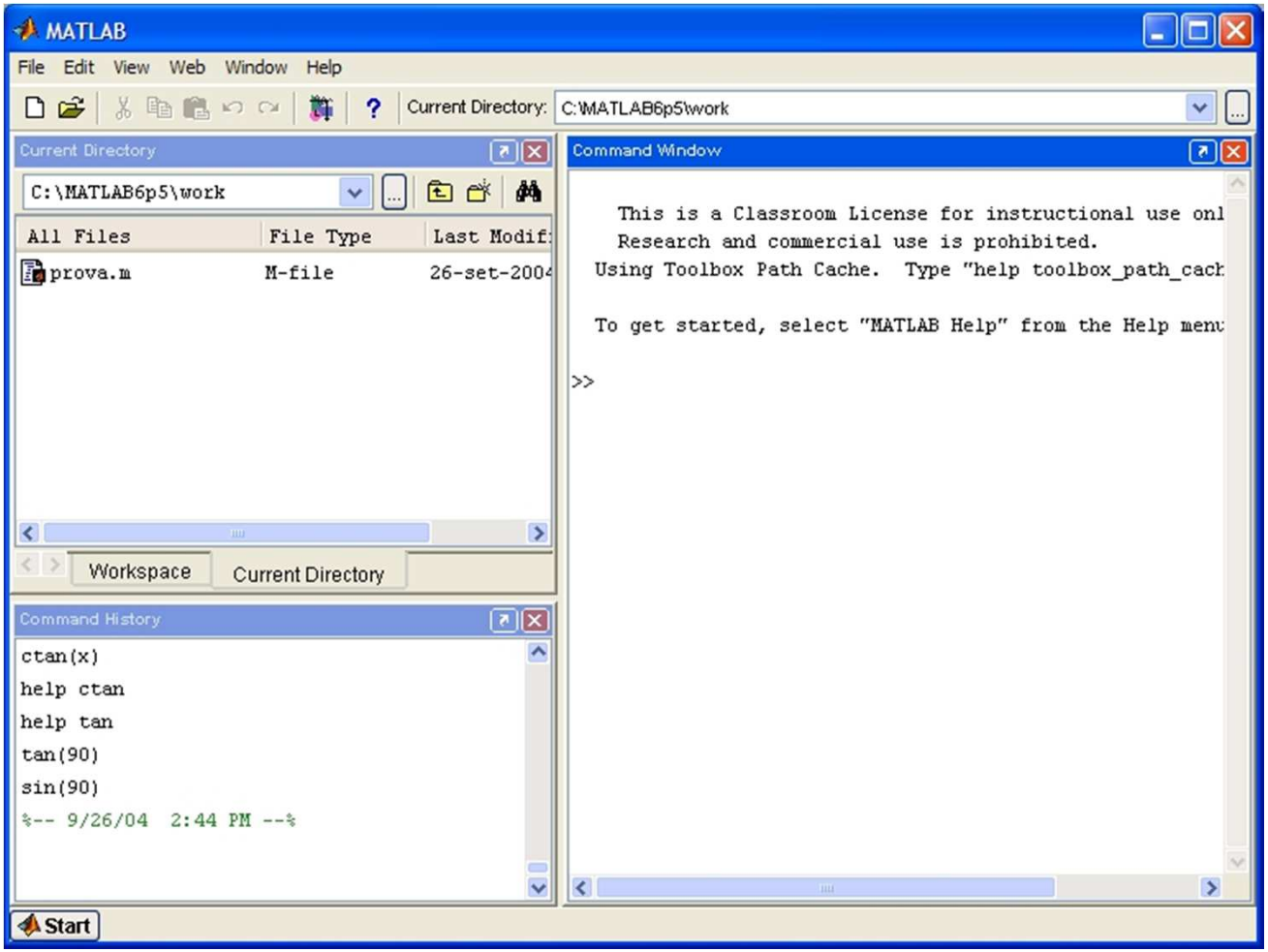


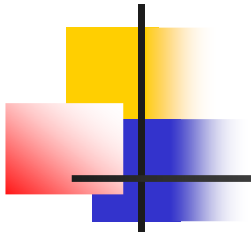
Toolboxes

I toolboxes sono collezioni complete di funzioni MATLAB (M-files) che estendono l'ambiente di MATLAB per risolvere particolari categorie di problemi. Alcuni toolboxes disponibili sono:

- Control System Toolbox
- Neural Network Toolbox
- Fuzzy Logic Toolbox
- Optimization Toolbox
- Statistic Toolbox
- Symbolic Math Toolbox

Iniziare con Matlab





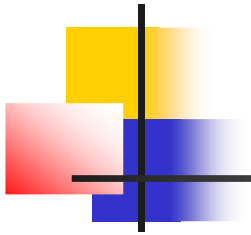
Le variabili

In Matlab non esistono dichiarazioni di tipo: il tipo di una variabile è stabilito al momento della sua definizione.

La variabile elementare in Matlab è una **matrice rettangolare** di numeri reali o complessi.

Scalari e vettori riga o colonna sono casi particolari di matrici.

E' possibile definire anche variabili di tipo carattere o stringa.



Definire una matrice 1\4

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

>> M =[1  2  3
       4  5  6
       7  8  9]

M =    1    2    3
       4    5    6
       7    8    9

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

M viene conservato come un oggetto nel workspace e può essere poi riutilizzato semplicemente digitando M nel command window

Definire una matrice 2\4

- Elemento di una matrice

```
>> M(2,1)

        ans =     4

>> M(4,1)

        ??? Index exceeds matrix dimensions.
```

- Riga o Colonna di una matrice

```
>> M(1,:)

    ans =     1     2     3

>> M(:,1)

    ans =   1

            4

            7
```

Definire una matrice 3\4

- Eliminare una riga o una colonna:

```
>> M=[1 2 3; 4 5 6; 7 8 9]
M =     1     2     3
        4     5     6
        7     8     9
>> M(2,:)=[]
M =     1     2     3
        7     8     9
>> M(:,2)=[]
M =     1     3
        7     9
```

Definire una matrice 4\4

•Concatenare due matrici:

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

>> N=[10 11 12];

>> O = [M; N]
M =     1     2     3
        4     5     6
        7     8     9
       10    11    12


>>N=[10; 11; 12];
>> O = [M N]
M =     1     2     3    10
        4     5     6    11
        7     8     9    12
```

Operazioni con le matrici 1\4

- \+ addizione

```
>> A=[1 1 1];
>> B=[2 2 2];
>> C=A+B
C =     3     3     3
```

- \- sottrazione
- * moltiplicazione

```
>> D=[3 4 5;6 7 8];
>> E=[1 3 2; 5 6 4 ; 8 3 1];
>> F=D*E
F =    63    48    27
             105    84    48
```

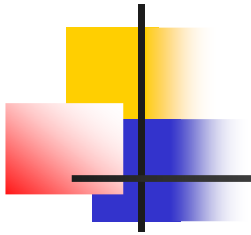
Operazioni con le matrici 2\4

- * Moltiplicazione

```
>> D=[3 4 5;6 7 8];
>> E=[1 3 2; 5 6 4 ; 8 3 1];
>> F=E*D
??? Error using ==> *
Inner matrix dimensions must agree.
```

- ^ Potenza A^2=A*A
- ‘ Trasposta

```
>> D
D =   3     4     5
            6     7     8
>> D‘
        ans =      3     6
                   4     7
                   5     8
```



Operazioni con le matrici 3\4

```
>> [2 2 2]*[3 3 3]
??? Error using ==> *
Inner matrix dimensions must agree.
>> [2 2 2].*[3 3 3]
ans=    6    6    6
>>  [2 2 2].^2
ans=    4    4    4
```

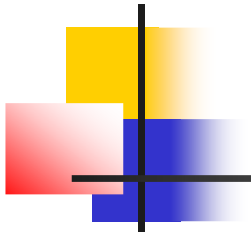
Operazioni con le matrici 4\4

- Operazioni con uno scalare (+, -, * ,/)

```
>> M = [2  2; 4 8] ;
>> M/2                >> M+1                >> M*2
        M= 1   1              M=3  3                M=4  4
           2   4                 5  9                   8  16
```

- \ divisione a sinistra
  / divisione a destra

  A quadrata ed invertibile;
  b e x vettori (riga o colonna) compatibili
  x=A\b=A^-1*b è la soluzione di A*x = b
  x=b/A è la soluzione di x*A = b

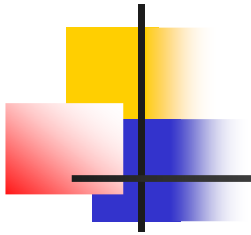


Il comando Format

Attraverso il comando format è possibile scegliere il formato numerico dei valori stampati sul video. Il comando regola solo come i numeri sono esposti, non come MATLAB li calcola o li salva.

```
>> 40/3
```

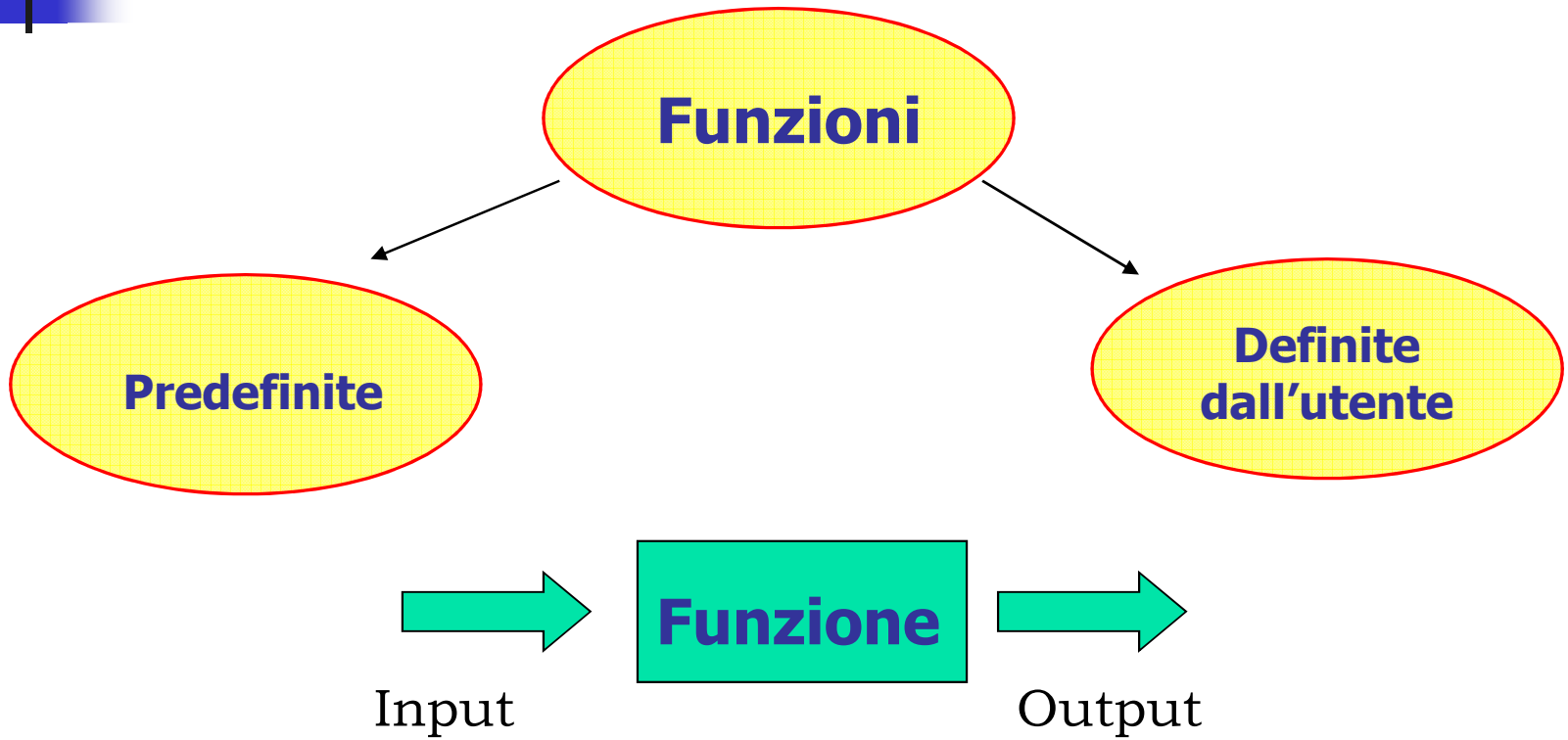
```
format short      13.3333
format short e    1.3333e+001
format short g    13.333
format long       13.3333333333333
format long e     1.33333333333333e+001
format long g     13.3333333333333
format bank       13.33
format rat        40/3
```



Alcuni comandi utili

- clear : cancella tutte le variabili nel workspace
- save NameFile : salva il workspace nel file NameFile.mat
- load NameFile : carica il file NameFile.mat nel workspace
- clc: pulisce lo schermo
- whos: variabili definite nel workspace

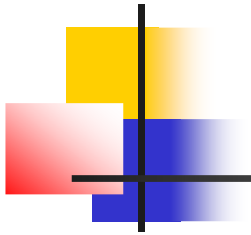
Introduzione alle funzioni



[output1, output2, ….]=nomefunzione(input1, input2, ….)

[A,B,C]=nomefunzione(d,e)

Funzioni per la generazione di matrici 1\2



- zeros (n,m)

Genera una matrice nxm composta tutta da 0

zeros(2,2)= 0 0

0 0

- ones (n,m)

Genera una matrice nxm composta tutta da 1

ones(2,2)= 1 1

1 1

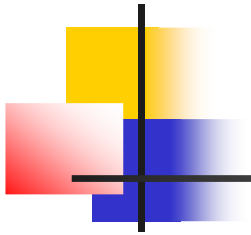
5*ones(2,2)= 5 5

5 5

- eye(n,m)

Genera la matrice identità nxm

Funzioni per la generazione di matrici 2\2



- rand (n,m)

Genera una matrice nxm composta numeri random distribuiti uniformemente

- randn (n,m)

Genera una matrice nxm composta numeri random secondo una distribuzione normale a media 0 e varianza 1.

- diag(vettore)

Crea una matrice diagonale con gli elementi del vettore lungo la diagonale

diag([1 1 1])

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Funzioni per la manipolazione di Matrici 1\3

- diag(M)

Estrae un vettore dalla matrice M contenente i valori sulla diagonale principale di M

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

diag(M) 1 5 9

- triu(M) (tril(M))

Estrae la matrice triangolare superiore (inferiore) da M

```
>> triu(M)     1     2     3          tril(M)   1     0     0
            0     5     6                      4     5     0
            0     0     9                      7     8     9
```

- [n,m]=size(M)

Restituisce le dimensioni della matrice M: n=3 m=3

Funzioni per la manipolazione di Matrici 2\3

- norm(M)

Calcola la norma di una matrice (o di un vettore)

norm(M,1), norm(M,'fro'), norm(M,inf)

- eig(M)

Calcola gli autovalori della matrice M

eig(M)= 16.1168 -1.1168 -0.0000

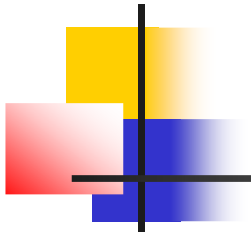
[D,L]=eig(M)

D = -0.2320 -0.7858 0.4082
    -0.5253 -0.0868 -0.8165
    -0.8187 0.6123 0.4082

L = 16.1168 0 0
    0 -1.1168 0
    0 0 0

M*D=D*L

Funzioni per la manipolazione di Matrici 3\3



- inv(M) = M^-1

Calcola l'inversa di una matrice M.

- det(M)

Calcola il determinante di una matrice M.

- rank(M)

Calcola il rango di una matrice M.

- null(M)

Calcola una base del nullo della matrice M

- trace(M)

Calcola la traccia della matrice M.

M=[1 4 6
4 5 7
7 8 9]

trace(M) = 15

Funzioni matematiche predefinite

- sqrt(x)

Calcola la radice quadrata di x

Se x è un vettore (matrice) restituisce un vettore (matrice) con la radice di ciascun elemento

sqrt([4 4 4])= 2 2 2

- round(x) : arrotondamento
- fix(x) : arrotondamento verso zero
- floor(x): arrotondamento verso meno infinito
- ceil(x): arrotondamento verso più infinito
- sign (x) : restituisce il segno di x (1 0 –1)
- abs(x) : valore assoluto di x
- exp(x) : e^x
- mean(x) : calcola il valor medio di un vettore x

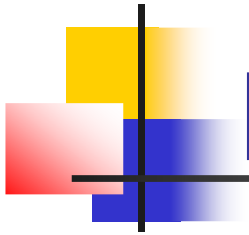
3.6 ⟶ 4

3.6 ⟶ 3

-3.6 ⟶ -4

-3.6 ⟶ -3

Funzioni matematiche predefinite



- max (x) e min(x) massimo e minimo di un vettore x
- log(x) logaritmo naturale di x
- log2(x), log10(x): logaritmi in base due e 10
- db(sqrt(2)/2) = -3.01db db2mag(-3.01) = 0.7071
- Numeri Complessi (i e j sono variabili predefinite)

>> z=2+3i

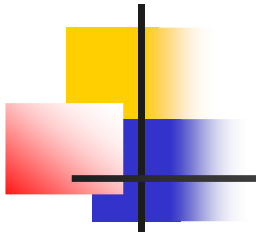
real(z) = 2 imag(z)= 3 abs(z) = 3.6056

conj(z) = 2 – 3i angle(z)= 0.98rad

- Funzioni trigonometriche

sin(x) cos(x) tan(x) sinh(x) cosh(x) tanh(x)

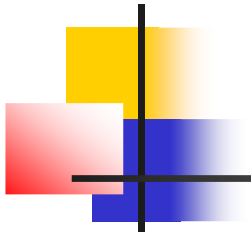
asin(y) acos(y) atan(y)



Funzioni matematiche predefinite

E' possibile richiamare l'help di matlab direttamente dal command window attraverso la sintassi:

```
>> help max
 MAX    Largest component.
    For vectors, MAX(X) is the largest element in X. For matrices,
    MAX(X) is a row vector containing the maximum element from each
    column. For N-D arrays, MAX(X) operates along the first
    non-singleton dimension.
    [Y,I] = MAX(X) returns the indices of the maximum values in vector I.
    If the values along the first non-singleton dimension contain more
    than one maximal element, the index of the first one is returned.
    MAX(X,Y) returns an array the same size as X and Y with the
    largest elements taken from X or Y.
    …
    See also MIN, MEDIAN, MEAN, SORT.
```



M-files

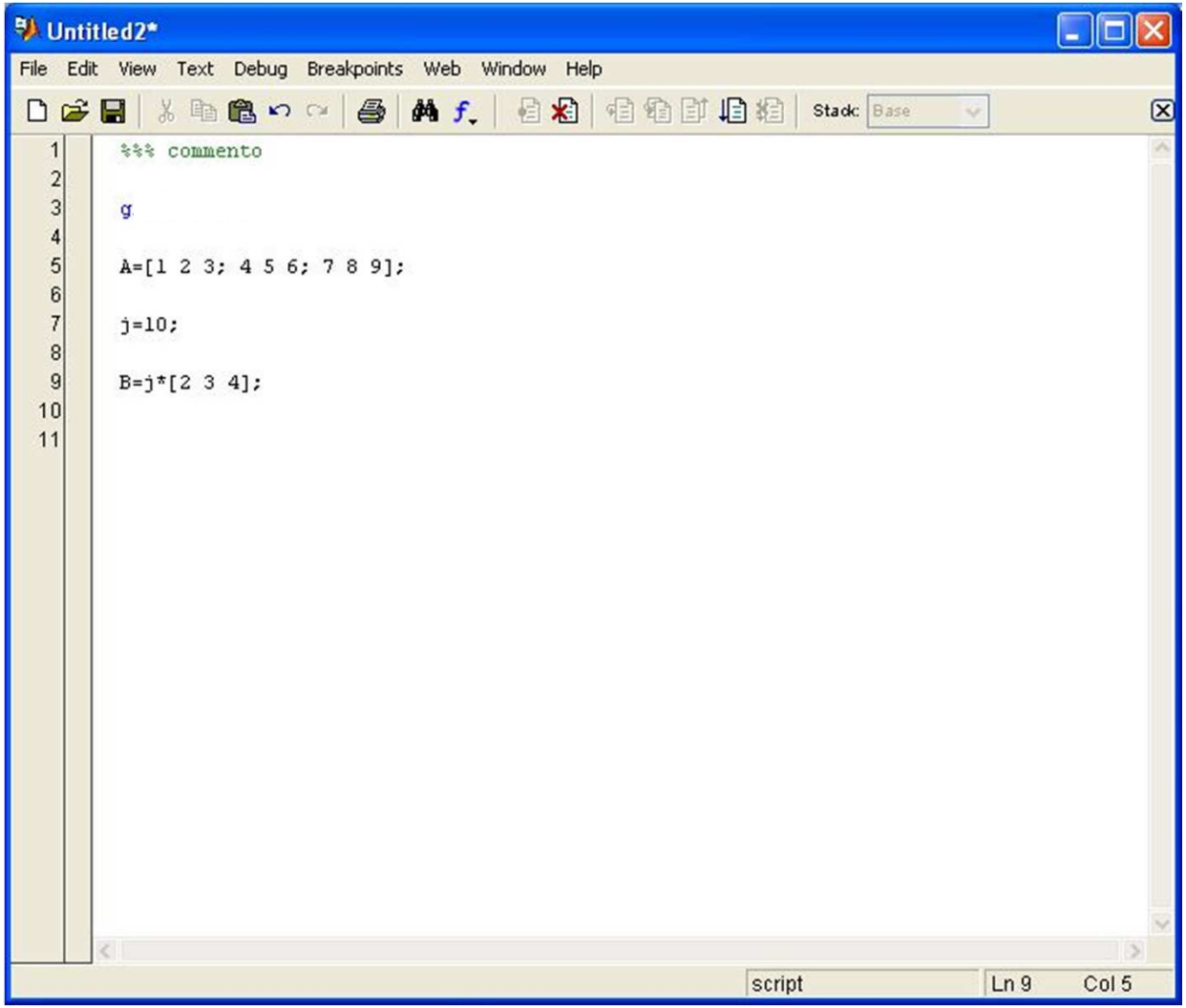
I Files che contengono codice MATLAB sono chiamati M-files. Dopo aver creato un M-file usando un qualsiasi editor di testo, tale file può essere usato come un comando od una funzione MATLAB. Ci sono due generi di M-file:

• Scripts che non accettano argomenti d'entrata o argomenti di uscita. Operano su dati nel workspace.
Per richiamare il file:

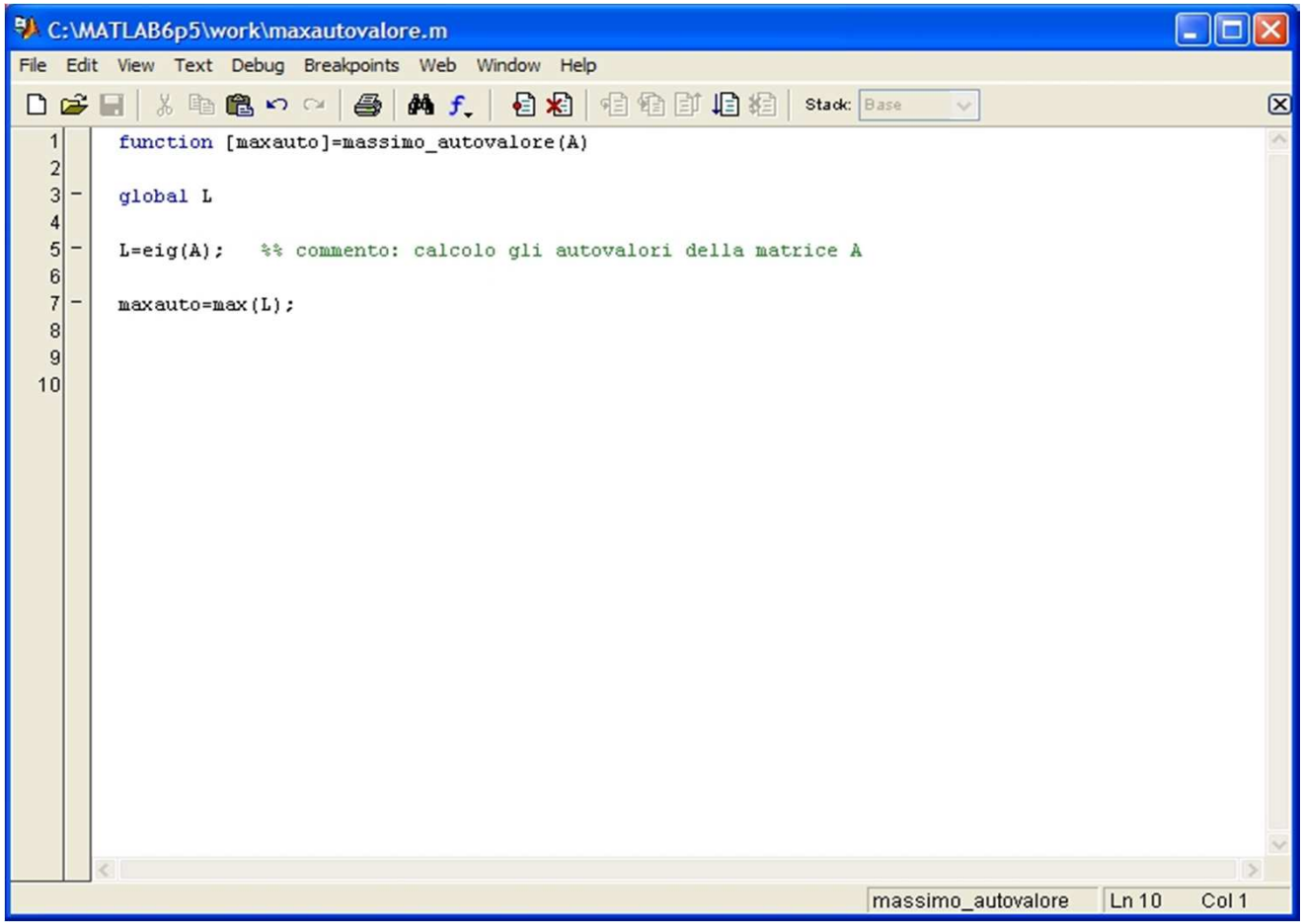
>> nomefile

• Functions che possono accettare argomenti d'entrata e argomenti di uscita.

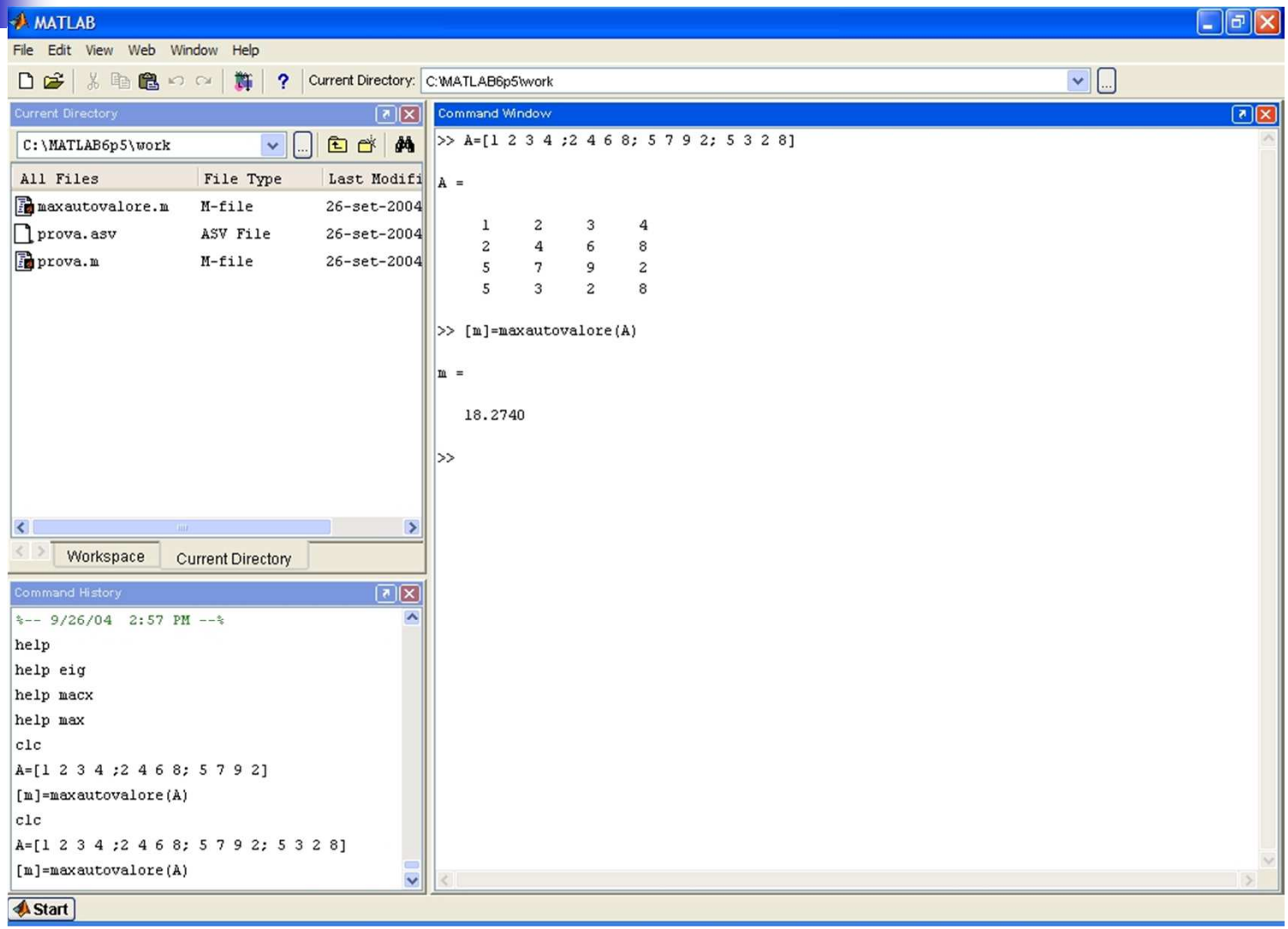
M-files


Untitled2*
File Edit View Text Debug Breakpoints Web Window Help
Stack: Base
%%% commento
g
A=[1 2 3; 4 5 6; 7 8 9];
j=10;
B=j*[2 3 4];
script
Ln 9 Col 5

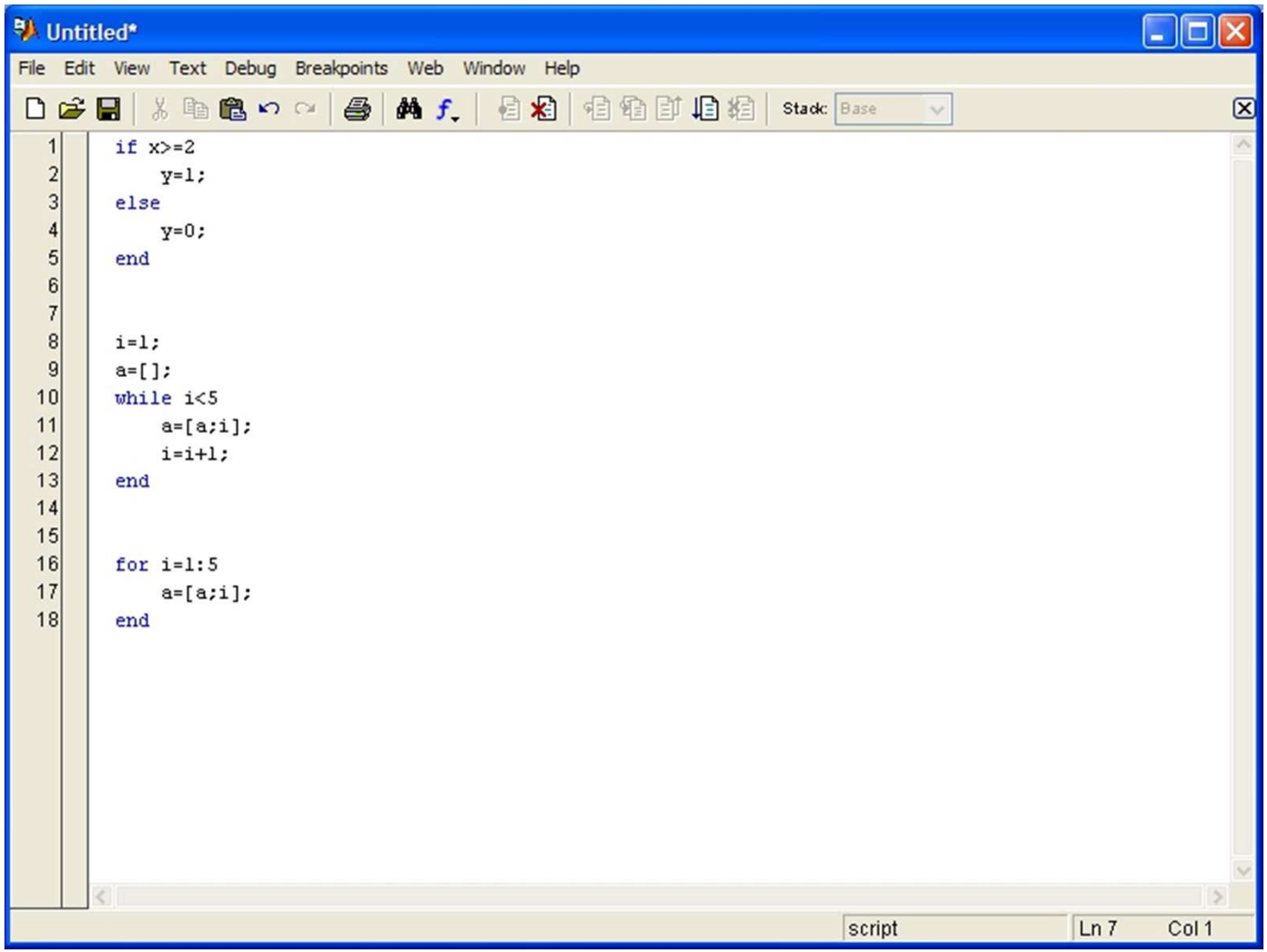
Funzioni definite dagli utenti

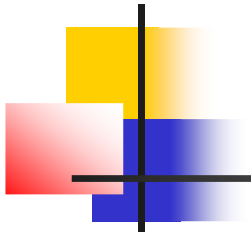


Funzioni definite dagli utenti



Funzioni definite dagli utenti





Polinomi

Un polinomio in Matlab è rappresentato con il vettore dei coefficienti

$s^3 + 3s^2 + 5s + 8$ p=[1 3 5 8]

$2s^3 + 5s + 8$ q=[2 0 5 8]

• funzione ***roots***

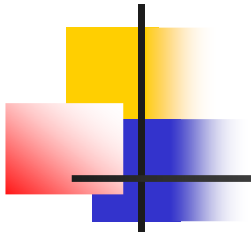
>>roots(p)

Calcola le radici del polinomio p

• funzione ***poly***

- poly(v) con v vettore calcola i coefficienti del polinomio le cui radici sono gli elementi di v (funzione inversa di roots).

- poly(A), con A matrice nxn, è un vettore i cui n+1 componenti sono i coefficienti del polinomio caratteristico della matrice A



Polinomi

Esempi:

```
>>p = [1 3 5 8];
>>v=roots(p)
      v =       -2.3283
                -0.3359 + 1.8230i
                -0.3359 - 1.8230i
>>poly(v)
ans =    1.0000    3.0000    5.0000    8.0000
>>A=[1 2 3; 4 5 6; 7 8 9];
>>poly(A)
ans =   1.0000  -15.0000  -18.0000   -0.0000
```

Il polinomio caratteristico di A è infatti: $\lambda^3-15\,\lambda^2-18\,\lambda$

Operazioni con i polinomi

• Calcolo del valore di un polinomio: ***polyval***

$s^3 + 3s^2 + 5s + 8$ p=[1 3 5 8];

>>polyval(p,2)

ans= 38

• Prodotto di due polinomi

$(s^3 + 3s^2 + 5s + 8) * (2s^3 + 5s + 8) = 2\,s^6 + 6\,s^5 + 15\,s^4 + 39\,s^3 + 49\,s^2 + 80\,s + 64$

>>conv(p,q)

ans= 2 6 15 39 49 80 64

• Divisione tra due polinomi

num=[1 3 5 8]; den=[3 5 8];

>>[ris,resto]=deconv(num,den)

ris = 0.33333 0.44444 ⟶ 0.33333s+0.44444

resto = 0 0 0.11111 4.4444 ⟶ 0.11111s+4.4444

Grafici con Matlab

Definizione di un vettore

>> v = 0:0.1:1

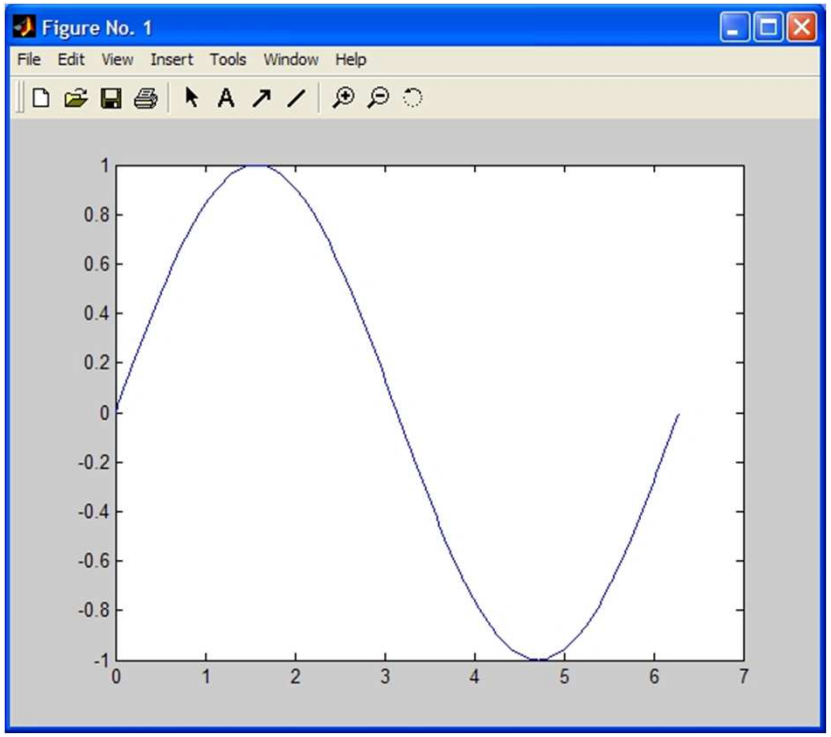
v = 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Funzione ***plot***

>>t = 0:pi/100:2*pi;

>>y = sin(t);

>>plot(t,y)



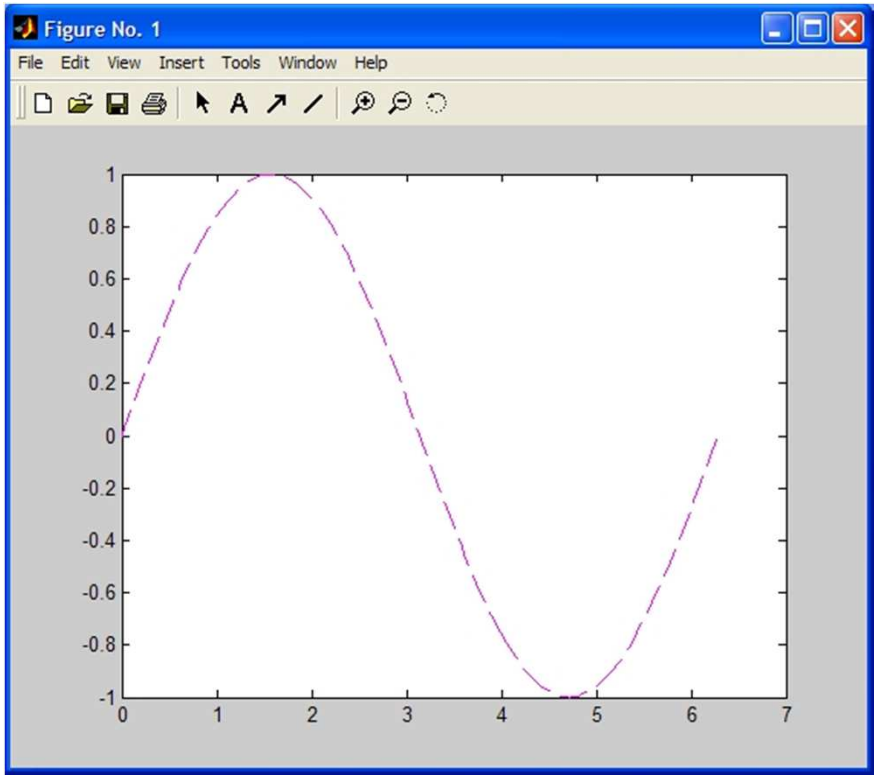
Grafici con Matlab

>>plot(t,y,'marcatore tipo-linea colore');

Marcatori: **'+', 'o', '*', 'x'** Tipo di linea: **'-', '--', ':', '-.', 'none'**

Colori: **'r', 'k', 'm', 'c', 'g', 'y', 'b'**

>> plot(t,y,'--m');

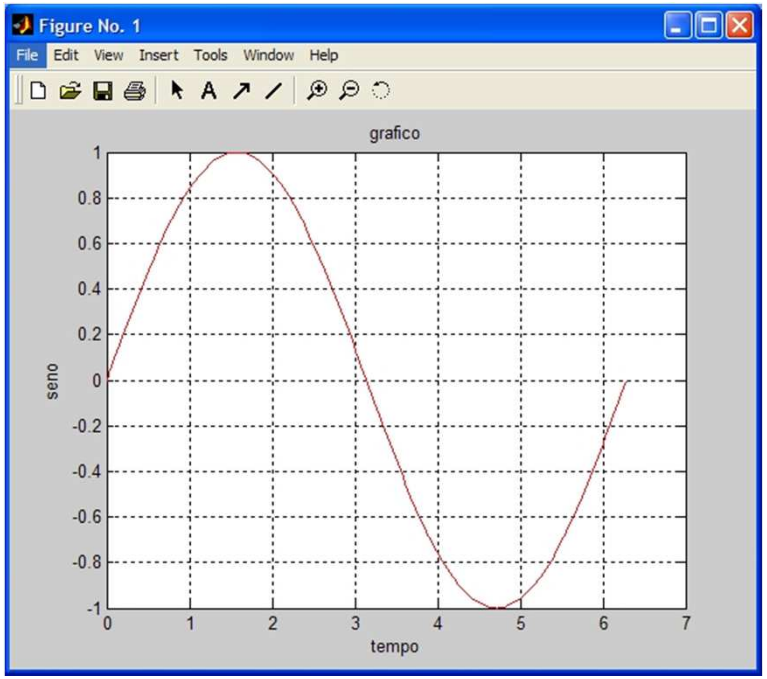


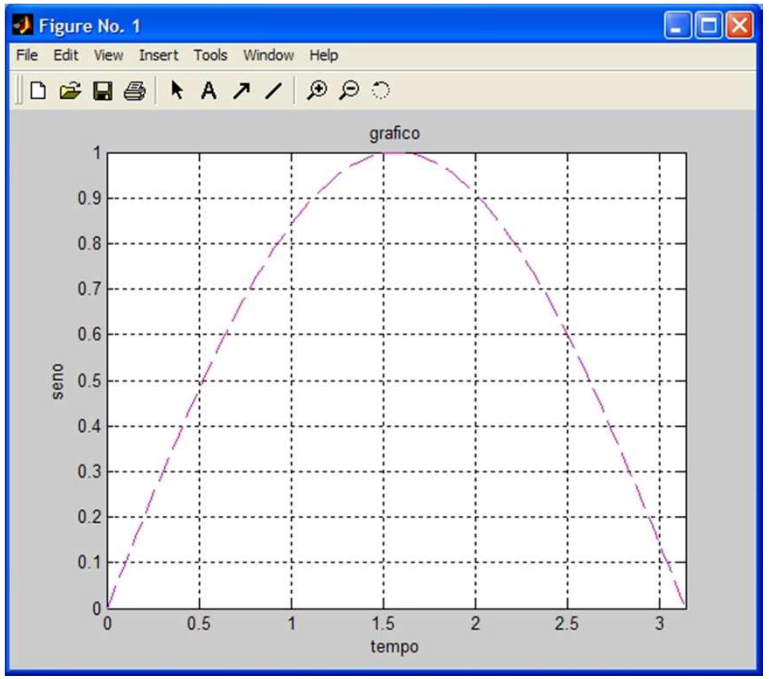
Grafici con Matlab

>>plot(t,y,'m'),grid,xlabel('tempo'),ylabel('seno'),title('grafico')

- Limiti degli assi

>>axis([0 3.14 0 1])



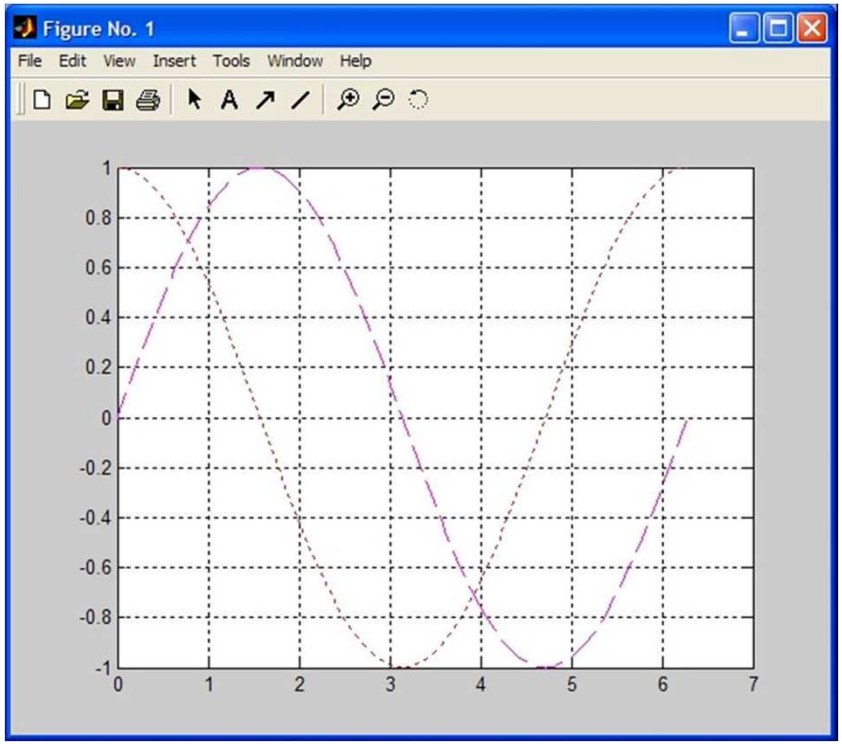


Grafici con Matlab

- Creazione di una nuova finestra: ***figure***

- Aggiunta di un tracciato ad un grafico già esistente: ***hold on***

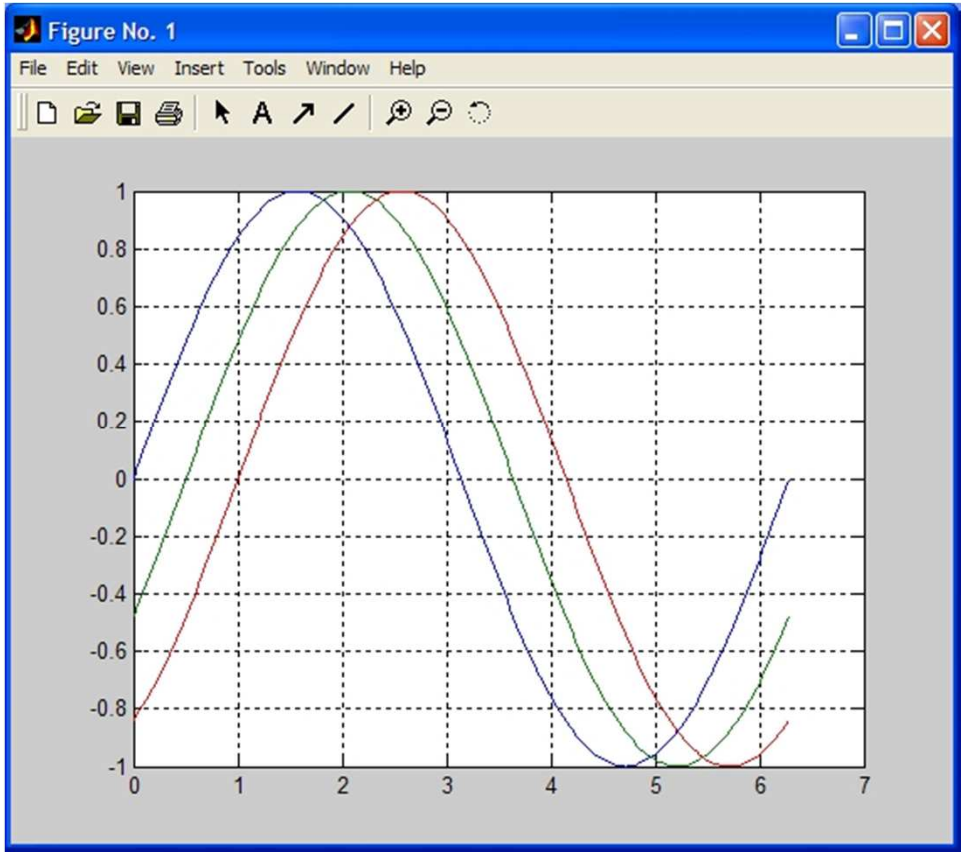
```
>>z=cos(t);
>>hold on
>>plot(t,z,':r')
```

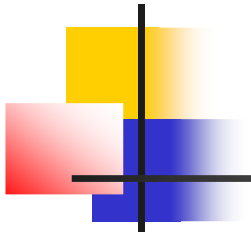


Grafici con Matlab

Grafici multipli:

```
>>t = 0:pi/100:2*pi;
>>y1 = sin(t);
>>y2 = sin(t-0.5);
>>y3 = sin(t-1);
>>plot(t,y1,t,y2,t,y3),grid
```



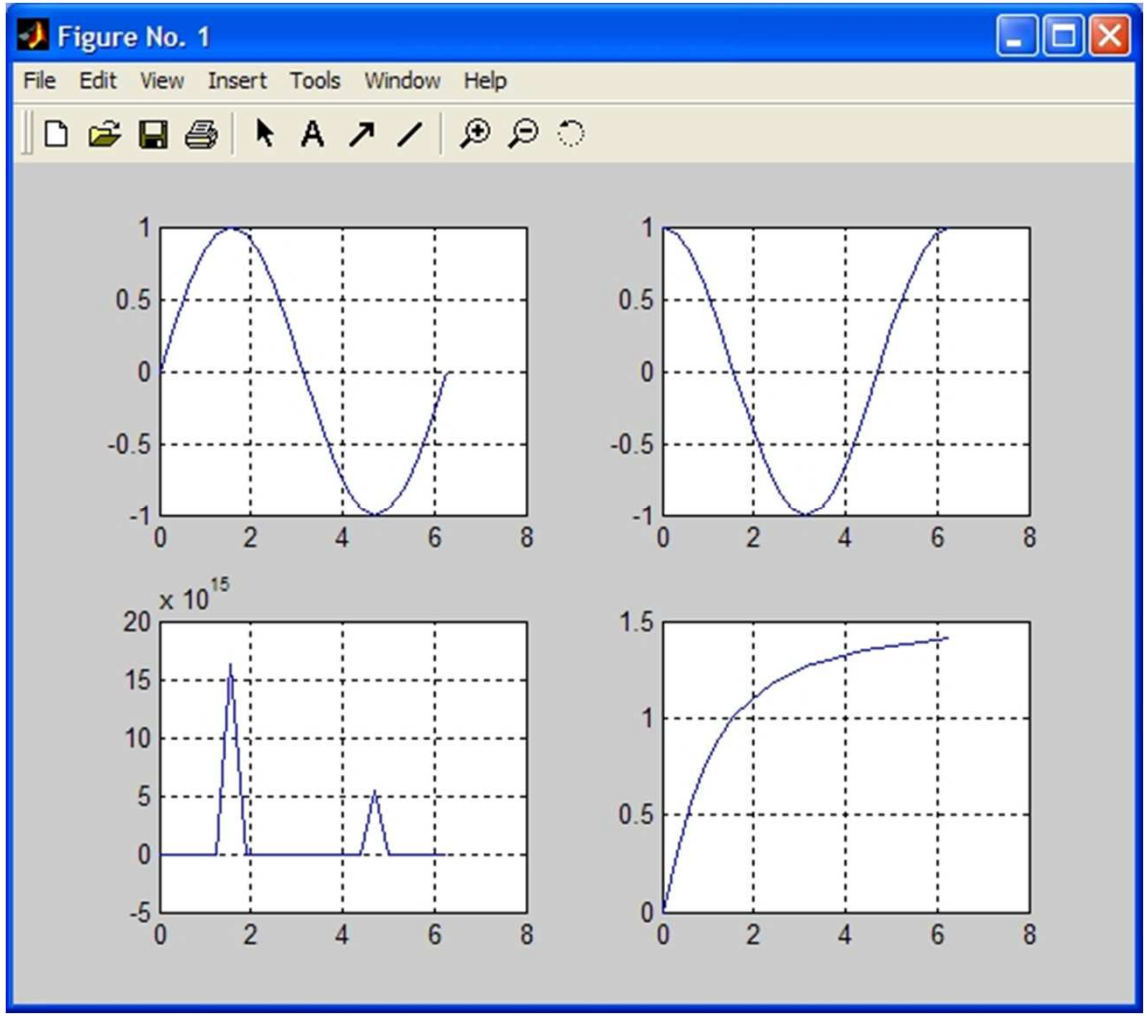


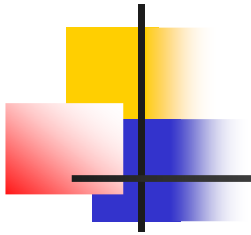
Grafici con Matlab

Funzione ***Subplot***

```
t = 0:pi/100:2*pi;
y = sin(t);
subplot(2,2,1),  plot(t,y),grid
y1=cos(t);
subplot(2,2,2), plot(t,y1),grid
y2=tan(t);
subplot(2,2,3), plot(t,y2),grid
y3=atan(t);
subplot(2,2,4), plot(t,y3),grid
```

Grafici con Matlab





Grafici su scala logaritmica

Per ottenere grafici su scala logaritmica o semi logaritmica si possono usare le seguenti funzioni, in sostituzione di plot:

- semilogy(t,y)

Si usa una scala logaritmica (base 10) per l'asse delle ordinate

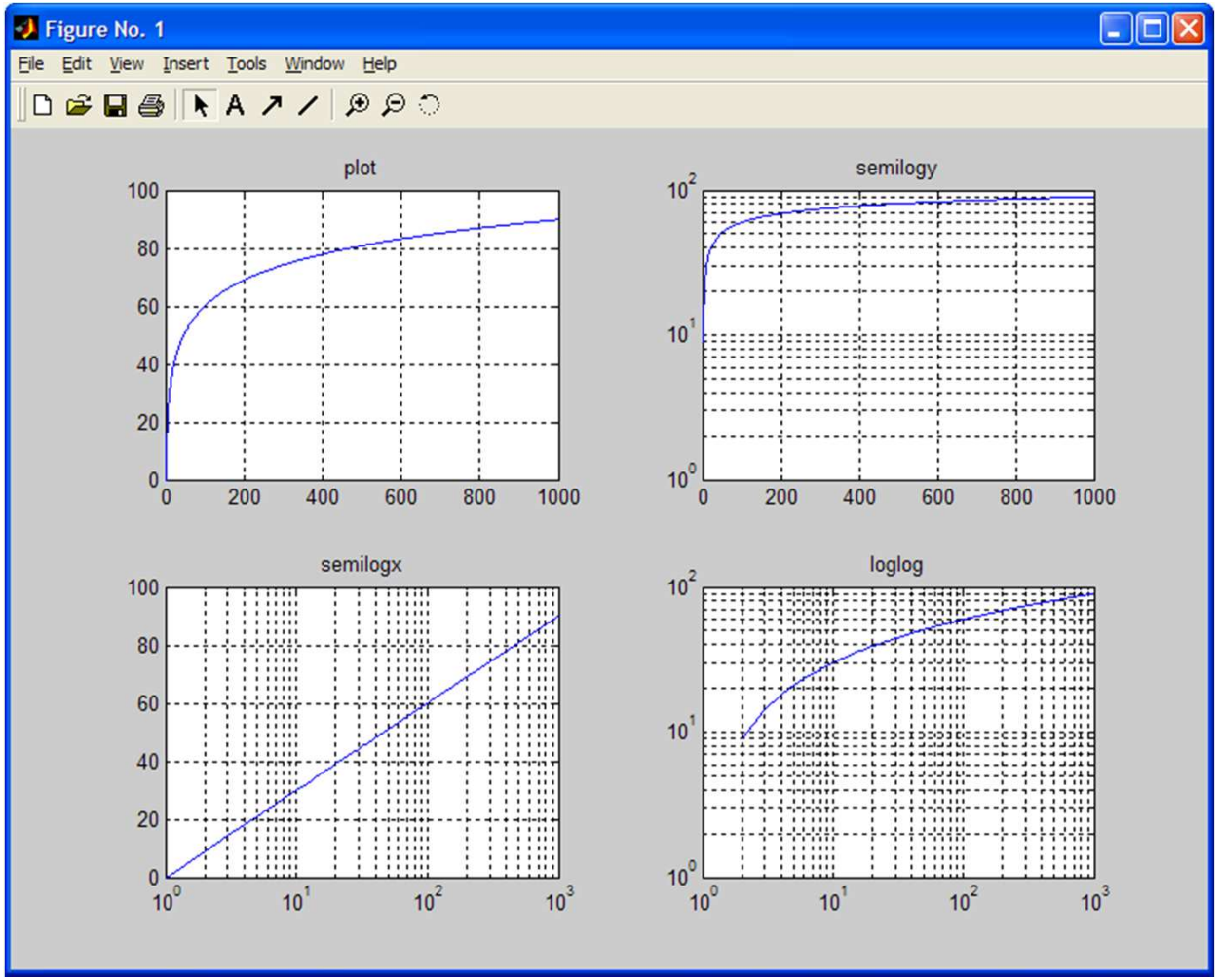
- semilogx(t,y)

Si usa una scala logaritmica (base 10) per l'asse delle ascisse

- loglog(t,y)

Si usa una scala logaritmica (base 10) per l'asse delle ordinate

Grafici su scala logaritmica



Control System Toolbox

Offre una serie di strumenti per l'analisi di sistemi dinamici lineari e stazionari (LTI), sia in tempo continuo che in tempo discreto. Sono supportati sia sistemi SISO (un ingresso una uscita) che sistemi MIMO (più ingressi e più uscite). Un sistema LTI in Matlab può essere specificato in termini di:

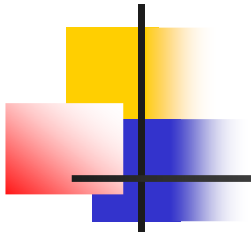
- Spazio degli Stati (State Space)

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

- Funzione di Trasferimento

$$G(s) = \frac{NUM(s)}{DEN(s)}$$

- Zeri-Poli-Guadagno $G(s) = k' \frac{(s - z_1)(s - z_2)....(s - z_m)}{(s - p_1)(s - p_2)....(s - p_n)}$

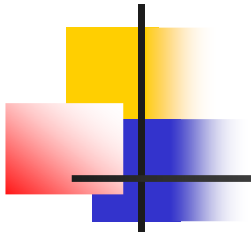


Control System Toolbox

A seconda del tipo di rappresentazione un modello LTI può consistere in una semplice coppia numeratore-denominatore, in quattro matrici e in vettori di poli e zeri. Il Control Sjstem Toolbox consente di creare degli “oggetti” (LTI objects) che rappresentano il sistema, indipendentemente dal tipo di modello adottato.

E’ possibile infatti passare da un tipo di rappresentazione all’altra senza perdita di informazioni.

Rappresentazione dei modelli LTI in Matlab



- modelli State Space (***ss***)

>>sys_ss=ss(A,B,C,D)

Esempio

A=[-4 -3.25; 4 0]; B=[1; 0] C=[1 0]; D=[0];

```
a =
       x1     x2
  x1  -4   -3.25
  x2   4      0

b=
       u1
  x1   1
  x2   0

c =
       x1     x2
  y1    1      0

d=
       u1
  y1    0
```

Continuous-time model.

Rappresentazione dei modelli LTI in Matlab

- modelli Function Transfer (***tf***)

>>sys_tf=tf(num,den)

num= [1 0]; den=[1 4 13]; $G(s) = \frac{s}{s^2 + 4s + 13}$

E' possibile rappresentare un modello tf direttamente come una funzione razionale nella variabile s:

>> s=tf('s')

>> sys_tf= s/(s^2+4*s+13)

Rappresentazione dei modelli LTI in Matlab

- modelli Zero-Pole-Gain (***zpk***)

>>sys_zpk=zpk(z,p,k);

z=[0];

p=[-2+3i -2-3i];

k=1;

$$G(s) = \frac{s}{(s+2-3i)(s+2+3i)}$$

Se non ci sono zeri (poli) si definisce un vettore vuoto z=[] (p=[]).

Anche in questo caso è possibile rappresentare un modello zpk direttamente come una funzione razionale in 's'

>> s=zpk('s')

>> sys_zpk=k*(s-z(1))/((s-p(1))*(s-p(2)))

Passaggio da una rappresentazione all'altra

```
>> sys1_tf=tf(sys_ss)

Transfer function:
      s
--------------
s^2 + 4 s + 13

>> sys1_ss=ss(sys_tf)

a =
      x1    x2
  x1  -4   -3.25
  x2   4     0

b=
      u1
x1     1
x2     0

c =
     x1  x2
  y1   1   0

d=
      u1
  y1  0

Continuous-time model.
```

Operazioni tra funzioni di trasferimento

In Matlab è possibile fare operazioni tra funzioni di trasferimento usando i comuni operatori matematici.

$$C(s)=\frac{1}{s} \qquad P(s)=\frac{2s+3}{4s^2+5s+1} \qquad H(s)=3$$

```
>> G=C*P  or G=series(C, P)

 Transfer function:

     2 s + 3
-----------------
4 s^3 + 5 s^2 + s
```

```
>> F=H*G or G=series(H, G)

 Transfer function:

     6 s + 9
-----------------
4 s^3 + 5 s^2 + s
```

```
>> G = C + P or G = parallel(C, P)

    6 s^2 + 8 s + 1
  -----------------
4 s^3 + 5 s^2 + s
```

Operazioni tra funzioni di trasferimento

In Matlab è possibile fare operazioni tra funzioni di trasferimento relativamente ai sistemi in retroazione

$$C(s)=\frac{1}{s} \qquad P(s)=\frac{2s+3}{4s^2+5s+1} \qquad H(s)=3$$

```
>> SYS = feedback(sys1,sys2)
```

```
  u --->O---->[ SYS1 ]----+---> y
        |                 |          y = SYS * u
        +-----[ SYS2 ]<---+
```

```
>> SYS = feedback(series(C,P), H)     or      >> SYS = minreal(G/(1+G*H))
```

```
 Transfer function:
        0.5 s + 0.75
-----------------------------
s^3 + 1.25 s^2 + 1.75 s + 2.25
```

Ritardi di tempo

E' possibile specificare un ritardo di tempo.

```
>>w = tf(1, [1 4 13])

Transfer function:
      1
--------------
s^2 + 4 s + 13


>>w.iodelay = 2

Transfer function:
                        1
exp(-2*s) * ------------------
                s^2 + 4 s + 13
```

Lo stesso vale per funzioni di trasferimento definite mediante zpk.
N.B. Funzioni di trasferimento con ritardo sono incompatibili con il comando feedback()

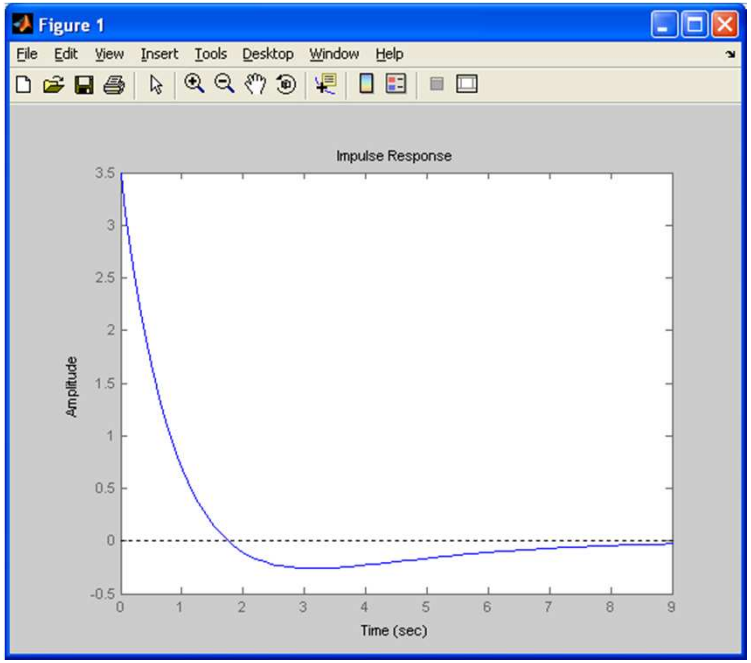
Risposta dei sistemi dinamici

Consideriamo il sistema con funzione di trasferimento $G(s) = \frac{7s+1}{2s^2+3s+1}$

- Risposta ad un impulso

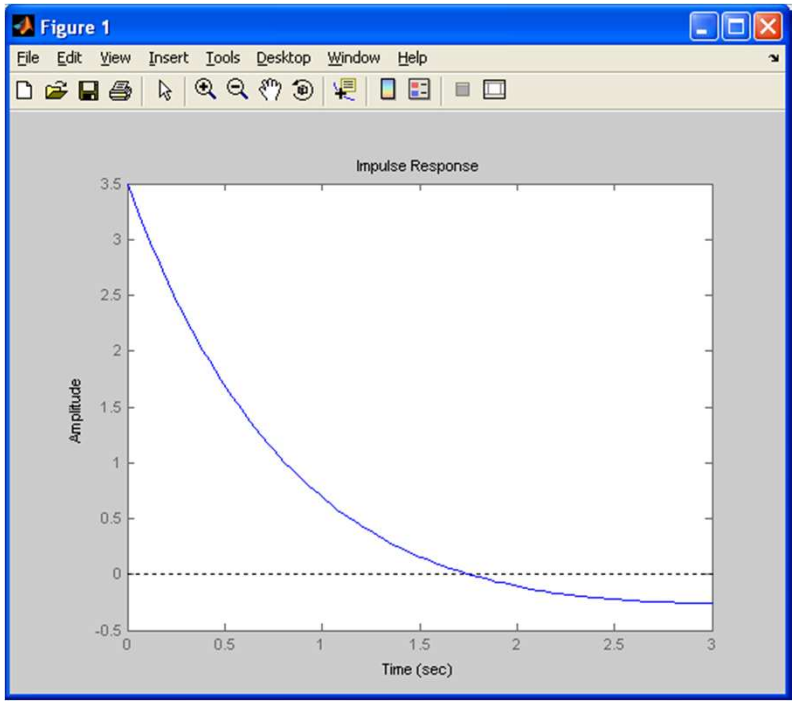
Si usa la funzione **impulse** che richiede come argomento di ingresso un modello LTI (rappresentato in una qualsiasi forma)

>>impulse(sys)



Risposta dei sistemi dinamici

>>impulse(sys,3)



>>[y,t]=impulse(sys)

Genera i vettori dell'uscita e del tempo di simulazione senza visualizzare alcun grafico.

Risposta dei sistemi dinamici

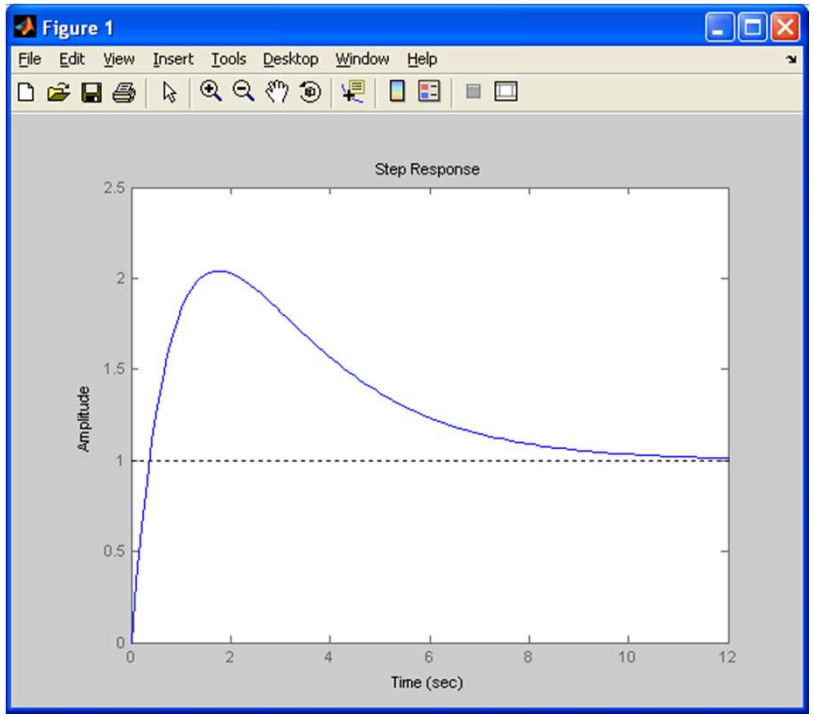
Consideriamo il sistema con funzione di trasferimento

$$G(s) = \frac{7s+1}{2s^2+3s+1}$$

- Risposta allo scalino unitario

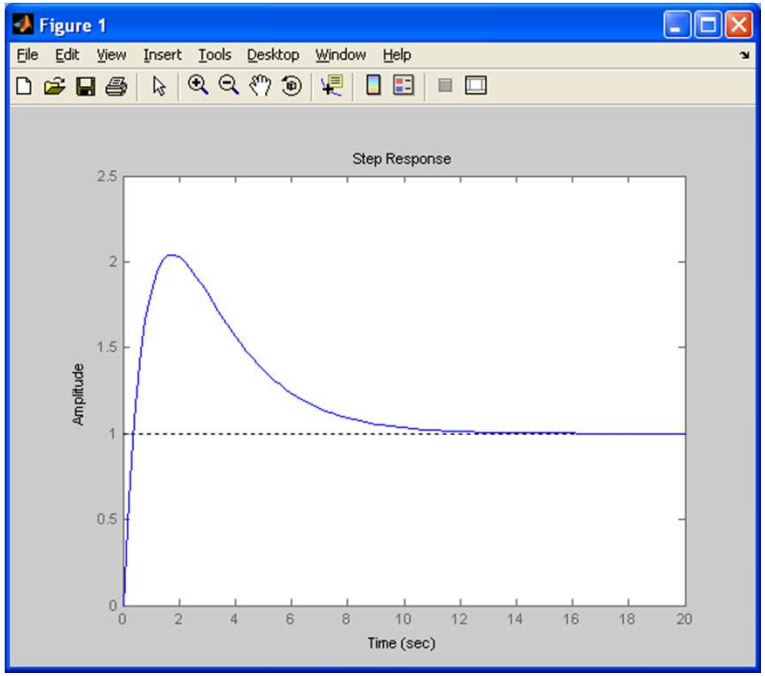
Si usa la funzione **step**

>>step(sys)



Risposta dei sistemi dinamici

>>step(sys,20)



>>[y,t]=step(sys)

Genera i vettori dell'uscita e del tempo di simulazione senza visualizzare alcun grafico.

Risposta dei sistemi dinamici

Consideriamo il sistema con funzione di trasferimento $G(s) = \frac{7s+1}{2s^2+3s+1}$

- Risposta ad un generico segnale di ingresso

Si usa la funzione **lsim**, che richiede come argomenti di ingresso il sistema e il segnale di ingresso descritto da una coppia di vettori u e t

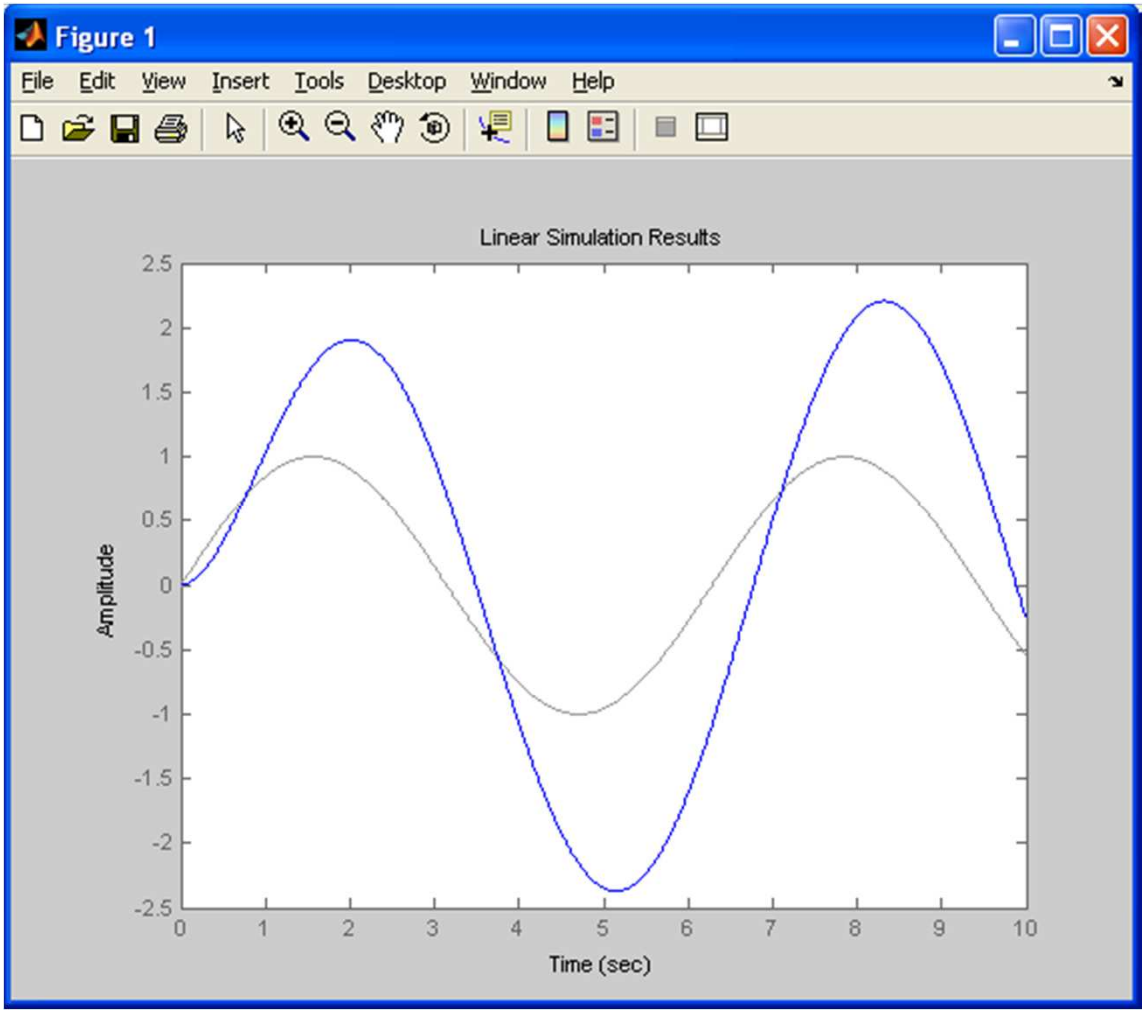
```
>> t = 0:0.01:10;  u = sin(t);  lsim(sys,u,t)
```

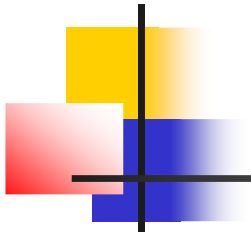
Calcola la risposta del sistema ad un ingresso sinusoidale di durata 10 secondi.

Se il sistema è espresso in forma di spazio degli stati è possibile definire uno stato iniziale x0 da cui far iniziare la simulazione.

```
>> t = 0:0.01:10;  u = sin(t);  lsim(sys,u,t,x0)
```

Risposta dei sistemi dinamici





Poli e Zeri di un sistema

•funzione ***pole***

>>pole(sys)

Calcola i poli del sistema

• funzione ***zero***

>>zero(sys)

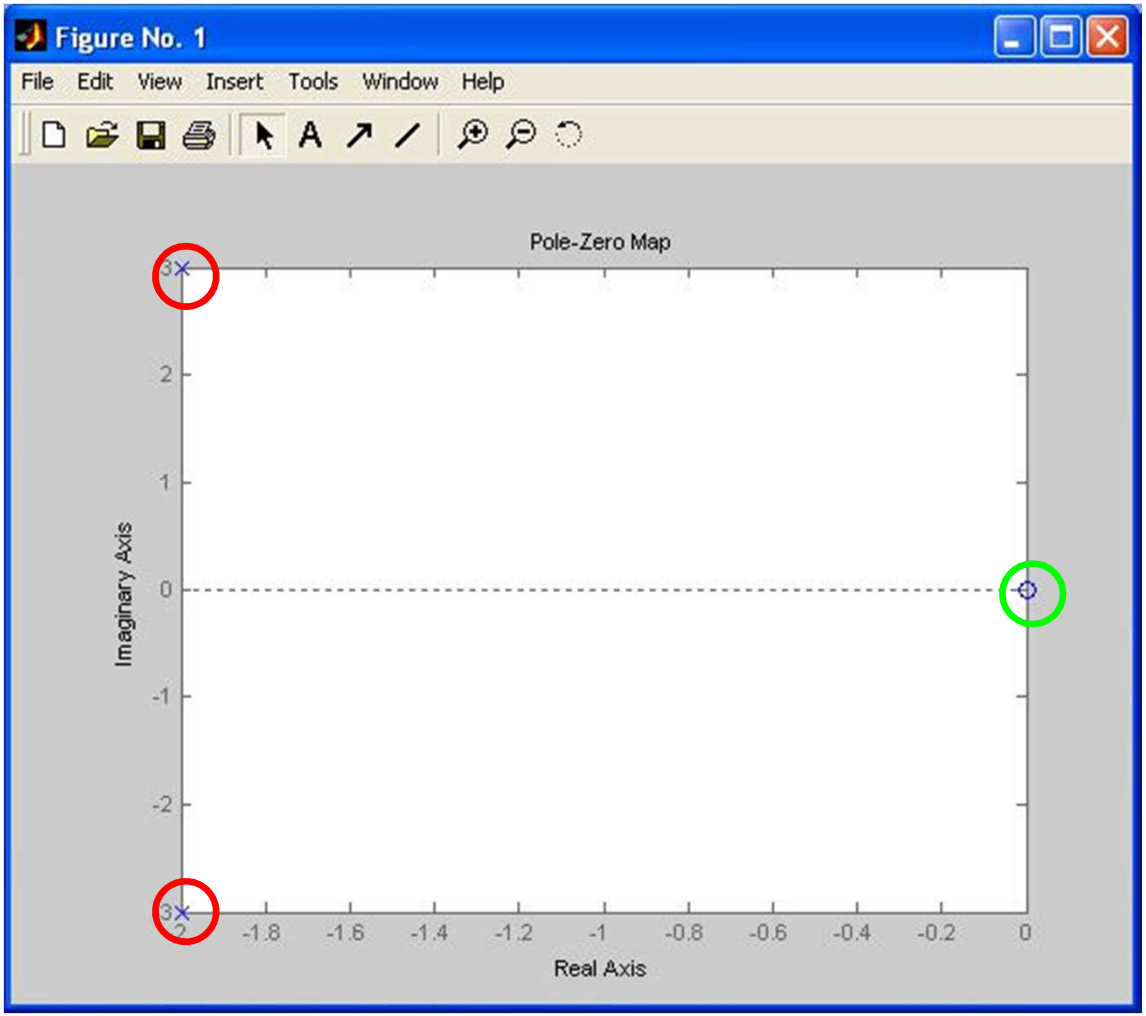
Calcola gli zeri del sistema

• funzione ***pzmap***

>>pzmap(sys) disegna la mappa poli-zeri del sistema

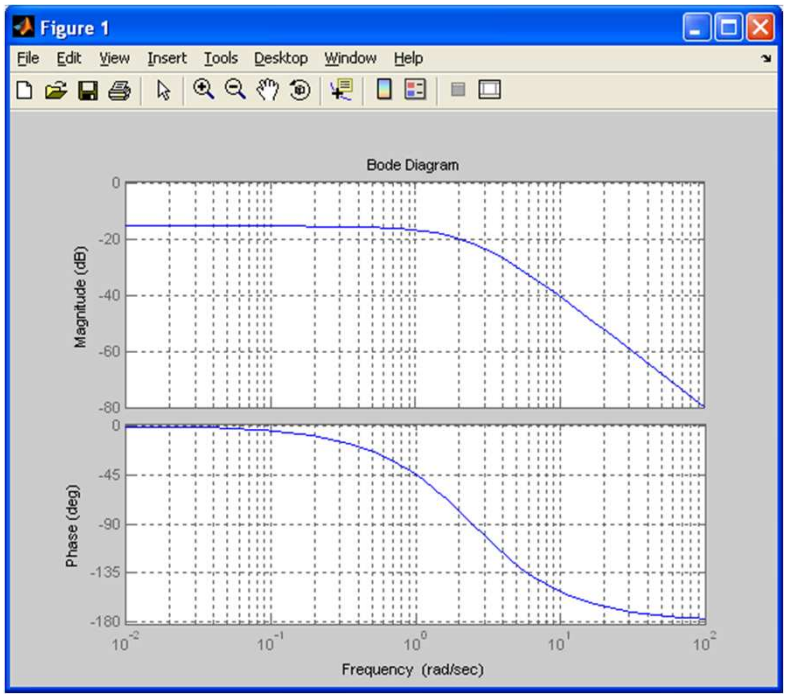
>>[p,z]=pzmap(sys) salva nei vettori p e z i poli e gli zeri del sistema

Poli e Zeri di un sistema



Diagrammi di Bode, Nyquist, Nichols

>> bode(sys)



>> [mag,phase]=bode(sys)

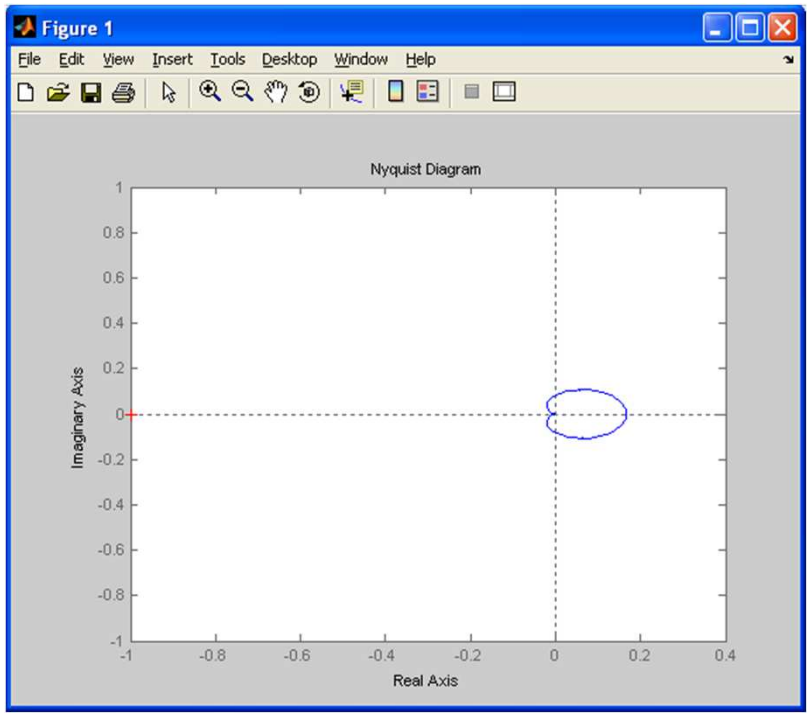
conserva in mag e phase i valori del modulo (valore naturale) e della fase. Non viene visualizzato alcun grafico

>> [mag,phase]=bode(sys,w)

Calcola i valori di modulo e fase in corrispondenza della frequenza w, o del vettore di frequenze w

Diagrammi di Bode, Nyquist, Nichols

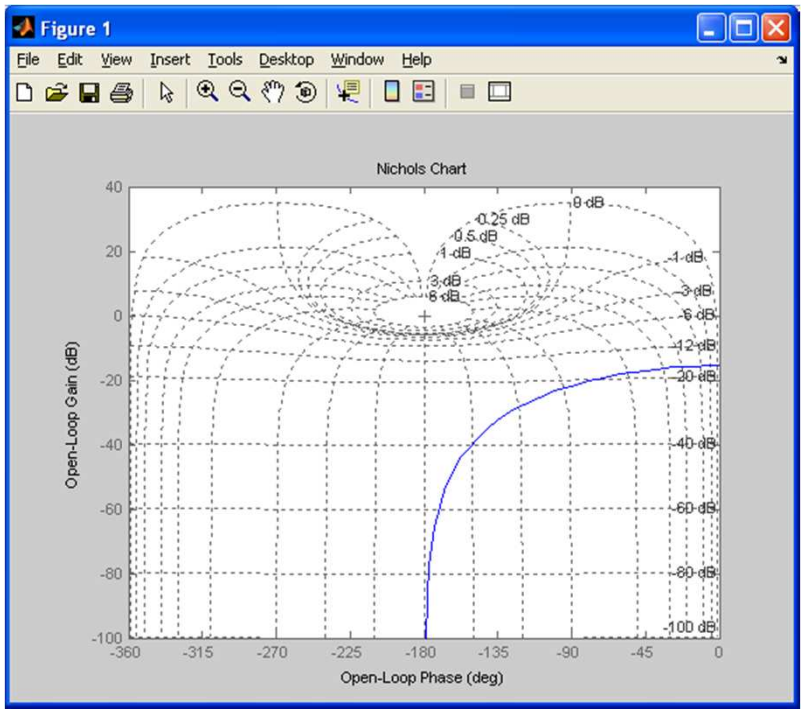
>>nyquist(sys)



>>[RE,IM]=nyquist(sys)
calcola il diagramma di Nyquist e salva in RE e IM i vettori della parte reale e della parte immaginaria della risposta in frequenza rispettivamente.Non viene visualizzato alcun grafico

Diagrammi di Bode, Nyquist, Nichols

>>nichols(sys)



>> [mag,phase]=nichols(sys)

conserva in M e F i valori del modulo (valore naturale) e della fase (in gradi). Non viene visualizzato alcun grafico.

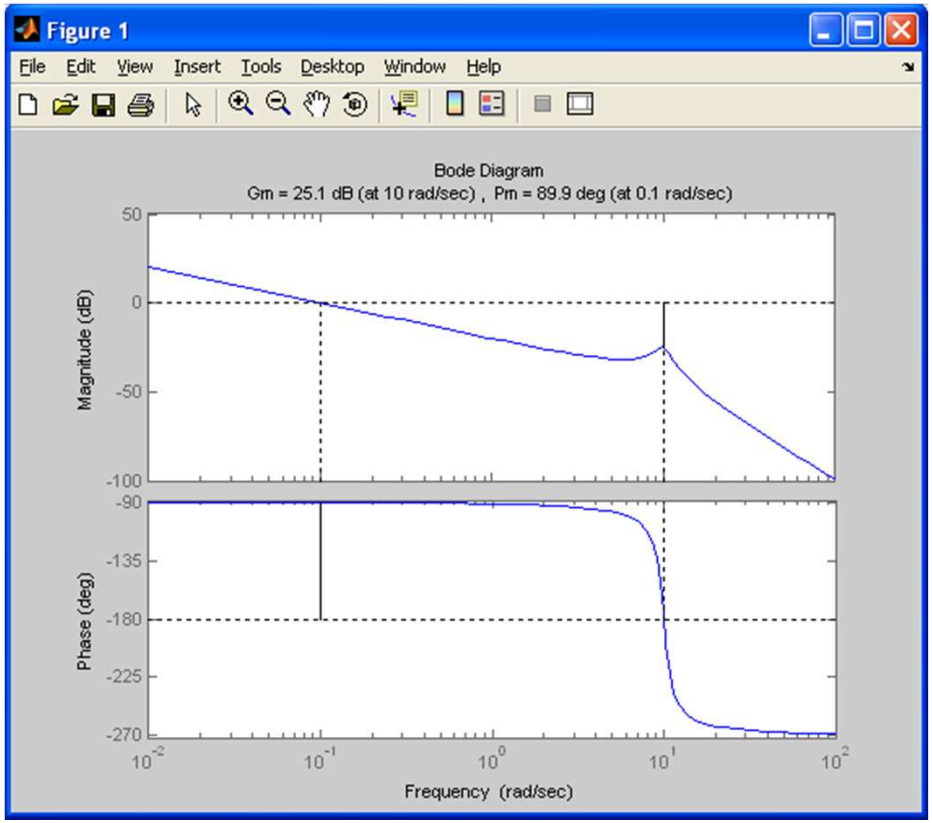
>> [mag,phase]=nichols(sys,w)

Calcola i valori di modulo e fase in corrispondenza della frequenza w, o del vettore di frequenze w

Margini di guadagno e di fase

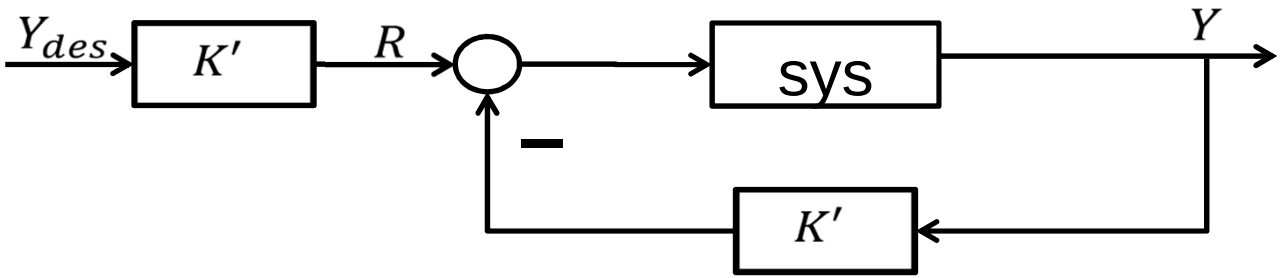
La funzione **margin** calcola i margini di fase e di guadagno e li visualizza sul diagramma di bode

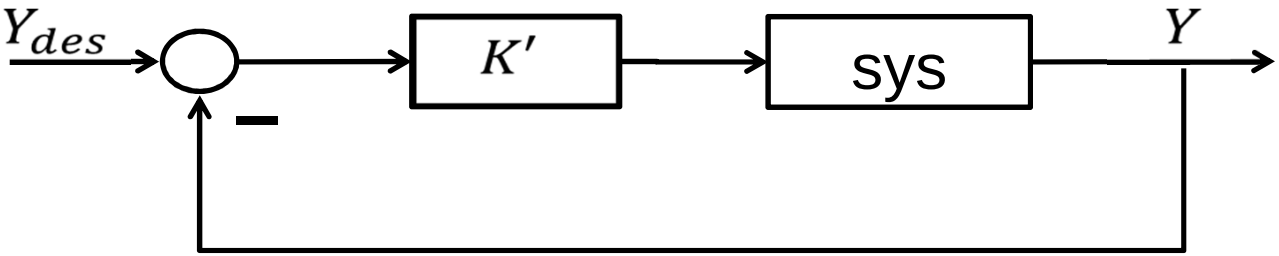
>> margin(sys)



Luogo delle radici

La funzione **rlocus** calcola evisualizza il luogo delle radici per $K' \in [0, +\infty)$ per il sistema in retroazione seguente





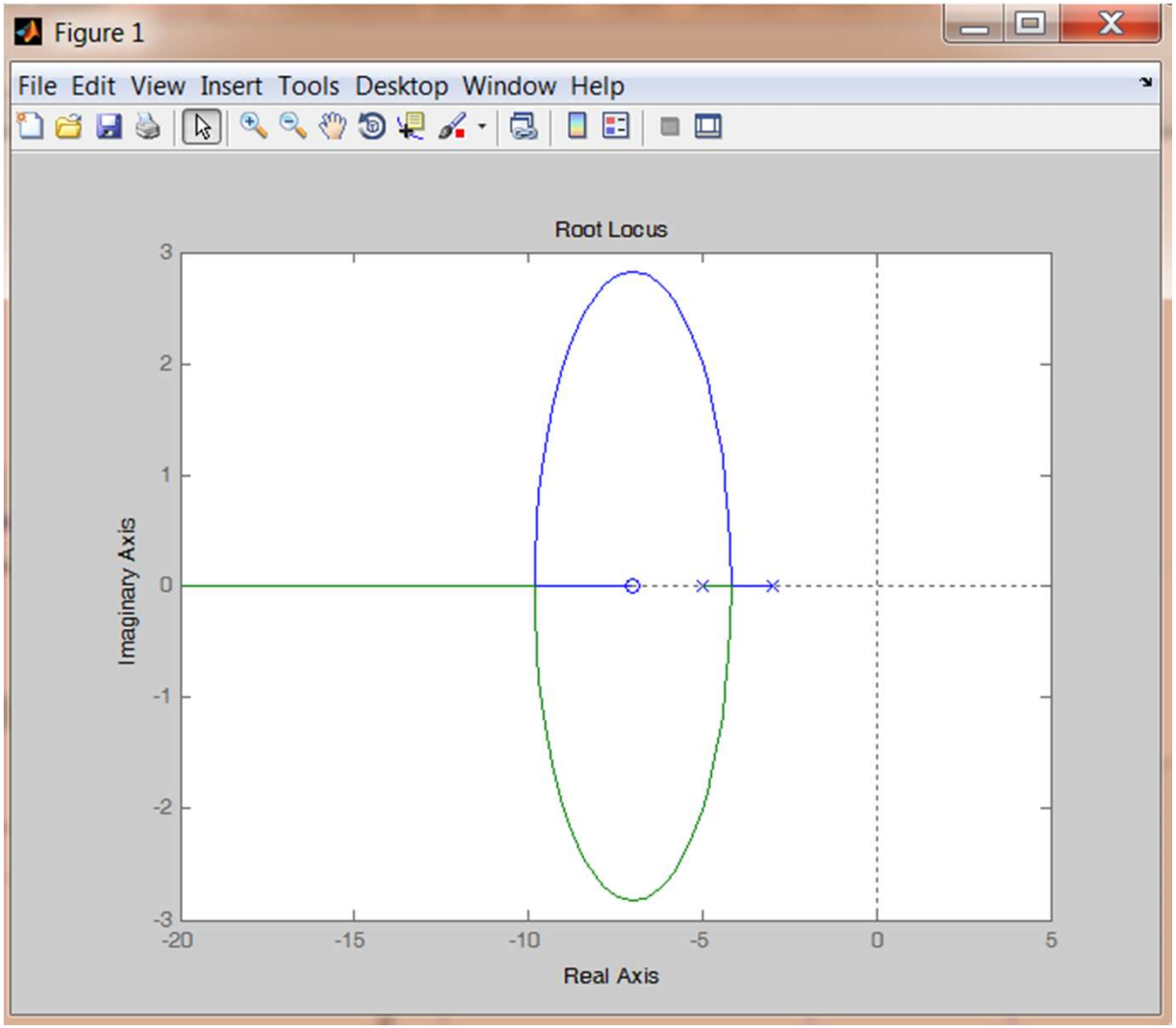
Luogo delle radici

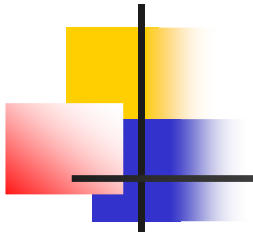
Transfer function:

$$\frac{s + 7}{s^2 + 8s + 15}$$

Sintassi

```
>> rlocus(sys)
>> rlocus(sys,k')
```





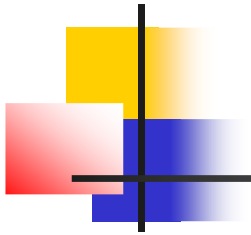
Altre funzioni importanti

>> bandwidth(sys) calcola la banda passante del sistema

>> dcgain(sys) calcola il guadagno statico del sistema

>> ltiview(sys) interfaccia grafica che permette di visualizzare bode, nychols, nyquist, step, etc..

>> sisotool(sys) interfaccia grafica per la progettazione dei sistemi di controllo



Riferimenti bibliografici

- Guida Matlab
- Manuale Matlab a cura di Giuseppe Ciaburro

  http://www.ciaburro.it/indmat/doc.htm
- Roberto Bucher

  Introduzione a Matlab
  Introduzione a Simulink
  Matlab e Simulink nella regolazione automatica

  Disponibili al sito: www.dti.supsi.ch/~bucher/
- Tutorial for Control System toolbox

  http://techteach.no/publications/control_system_toolbox/
- Manuale sintetico per l'uso del Control System Toolbox di Matlab

  di Alessandro Melis e Pierluigi Muntoni, disponibile al sito:

  www.diee.unica.it/~giua/ANSIS/matlab_ansis.pdf

DIPARTIMENTO DI INGEGNERIA E FISICA
DELL'AMBIENTE
UNIVERSITA' DELLA BASILICATA

INTRODUZIONE ALL'USO DI MATLAB: SIMULINK

Francesco Pierri

Simulink

Simulink è un pacchetto software integrato in ambiente Matlab per modellare, simulare e analizzare sistemi dinamici, sia lineari che non lineari, a tempo continuo o a tempo discreto.

Per la modellazione Simulink fornisce una interfaccia grafica per costruire il modello come uno schema a blocchi.

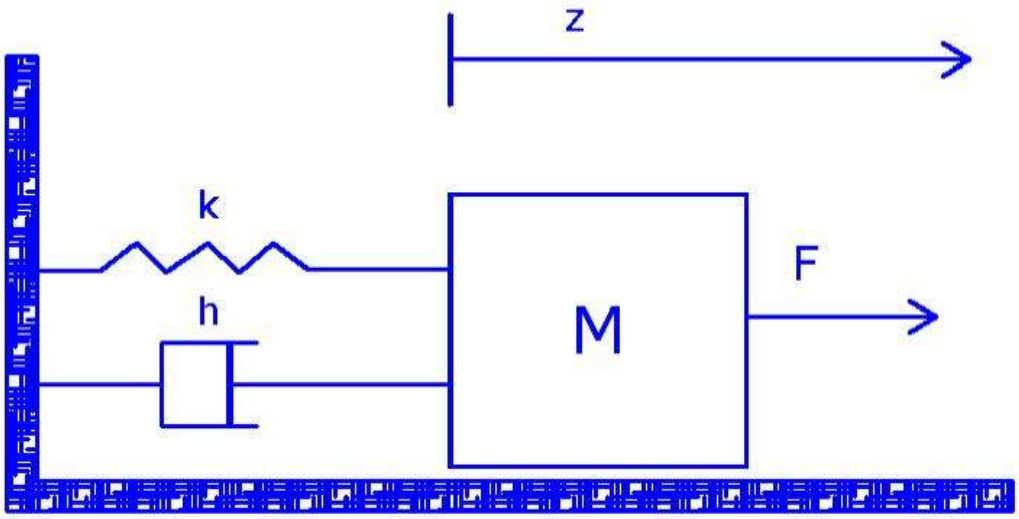
E' inclusa una libreria di blocchi ed inoltre è possibile creare nuovi blocchi definiti dall'utente.

Dopo aver definito un modello è possibile simularlo, scegliendo tra diversi metodi di integrazione. Per la simulazione è possibile usare il menù del Simulink o usare linee di comando dal *command window* di Matlab.

I risultati della simulazione possono essere salvati nel workspace di Matlab per la visualizzazione e l'analisi.

Un Esempio di Sistema

Sistema massa-molla-smorzatore



Un Esempio di Sistema

Sistema massa-molla-smorzatore

$$M\ddot{z}(t) = -kz(t) - h\dot{z}(t) + F(t)$$

$$y(t) = z(t); \qquad u(t) = F(t);$$

$$x(t) = (z(t) \quad \dot{z}(t))^T$$

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = -\dfrac{k}{M} x_1(t) - \dfrac{h}{M} x_2(t) + \dfrac{1}{M} u(t) \\ y(t) = x_1(t) \end{cases}$$

Un Esempio di Sistema

$$A = \begin{bmatrix} 0 & 1 \\ -k/M & -h/M \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1/M \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad D = 0.$$

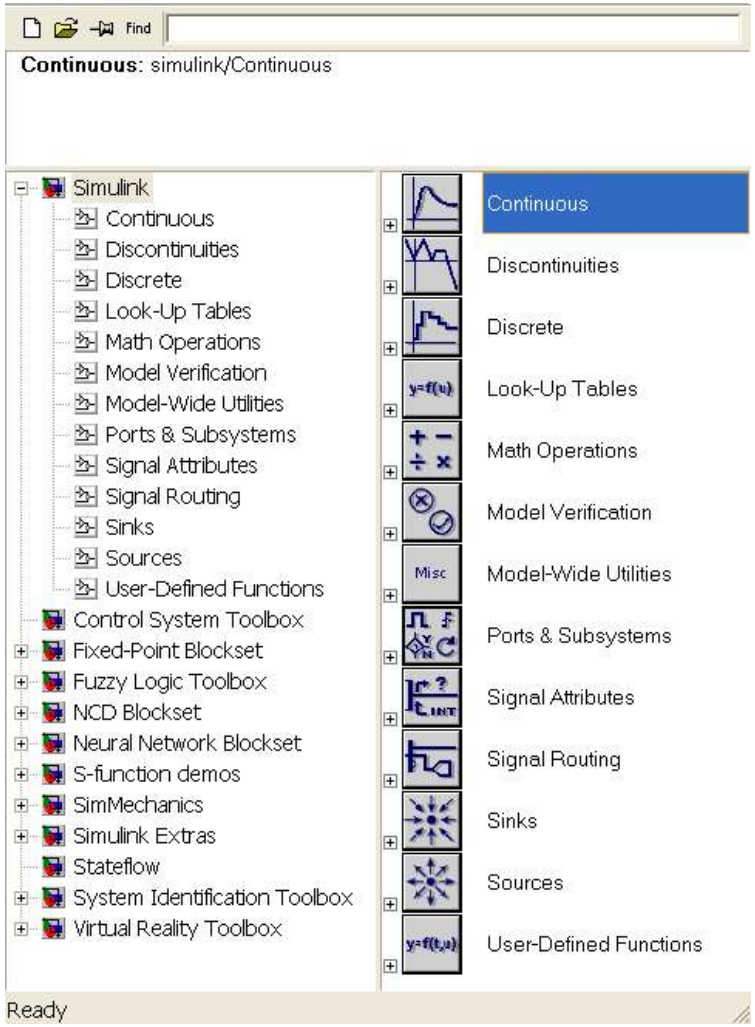
$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

Un Esempio di Sistema

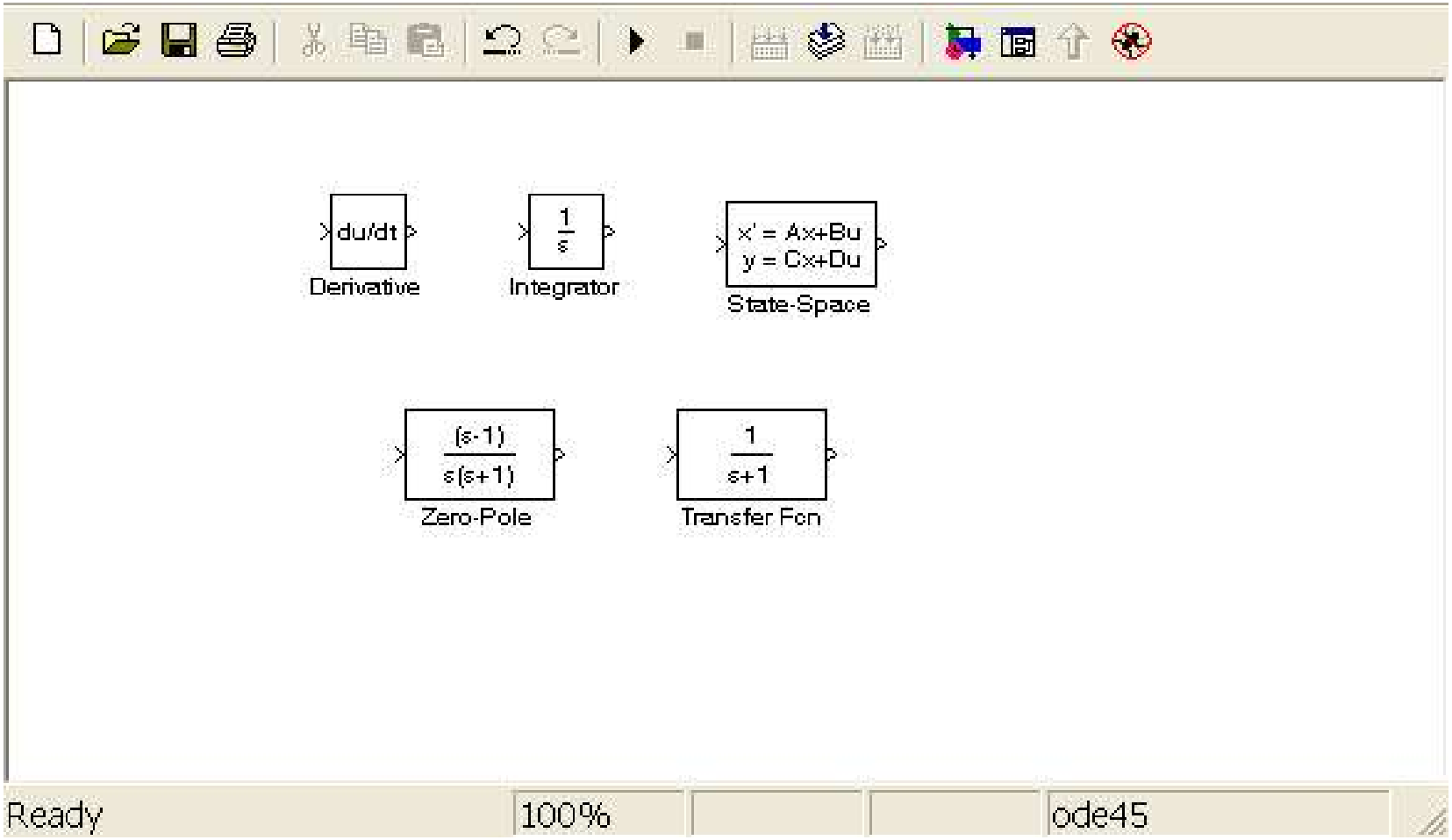
$$G(s) = \frac{1/M}{s^2 + sh/M + k/M}$$

$$G(s) = \frac{1}{M} \frac{1}{\left(s - \frac{h}{2M} + \sqrt{\frac{h^2}{4M^2} - \frac{k}{M}}\right)\left(s - \frac{h}{2M} - \sqrt{\frac{h^2}{4M^2} - \frac{k}{M}}\right)}$$

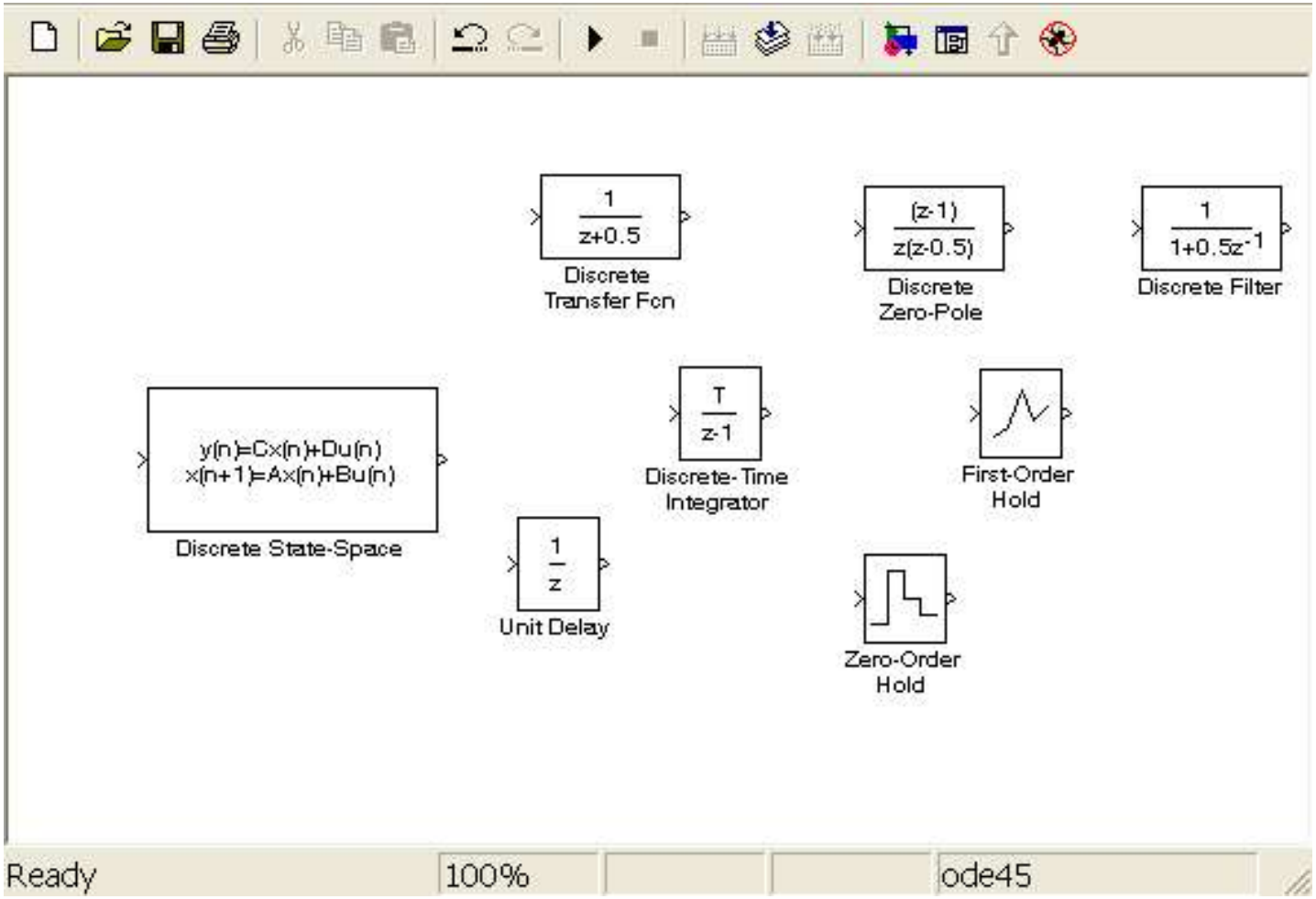
Libreria di Simulink



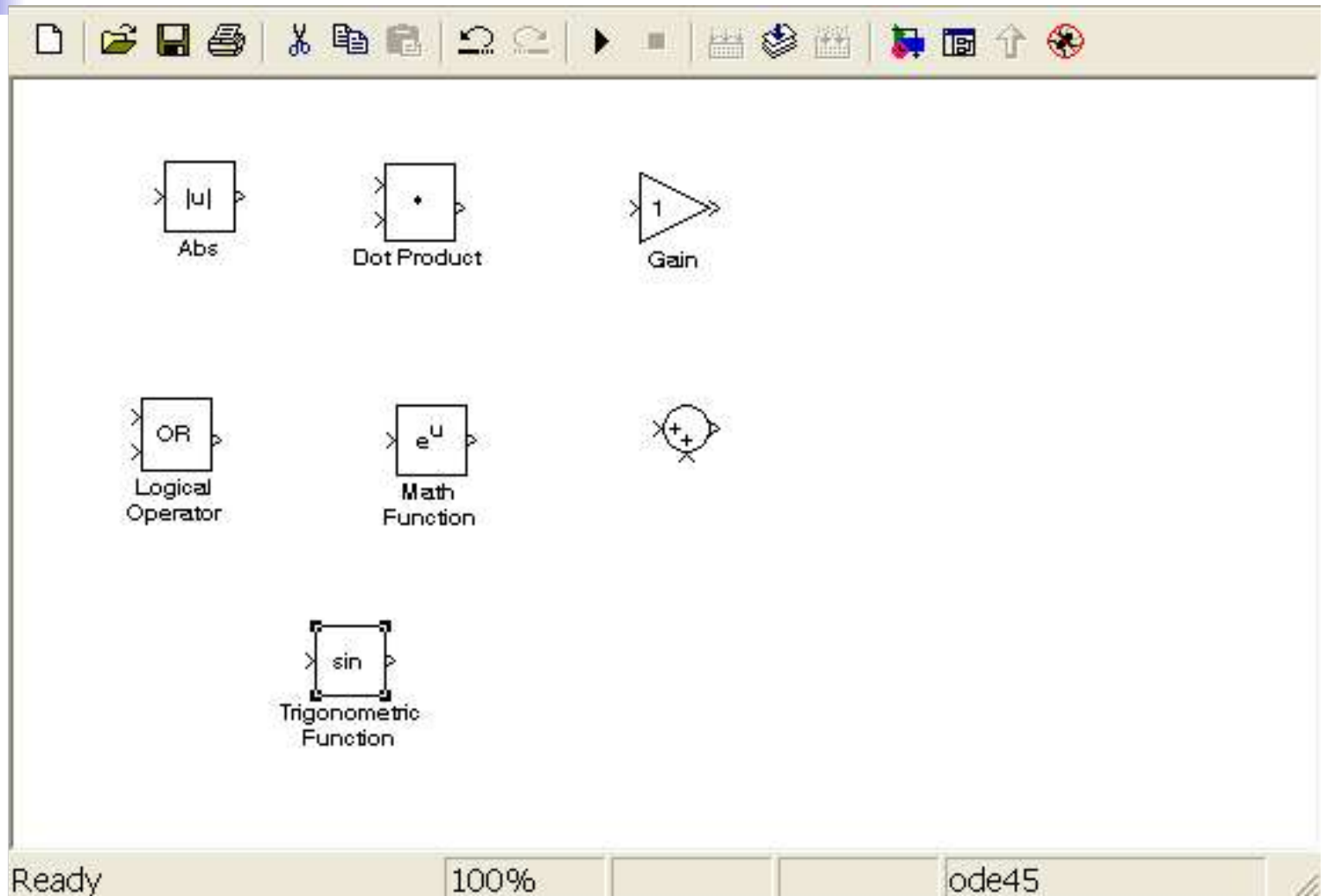
Libreria di Simulink: Continous



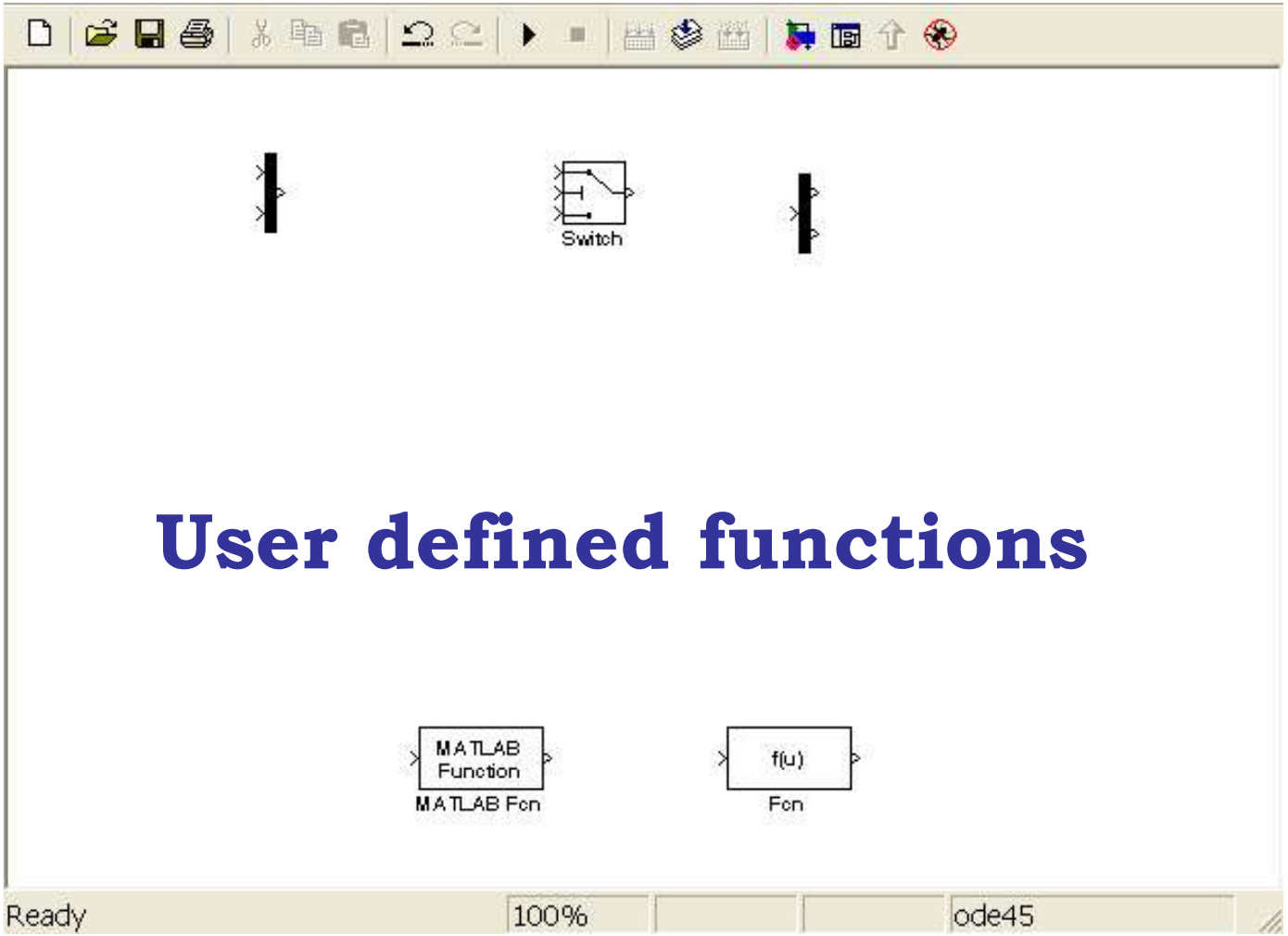
Libreria di Simulink: Discrete



Libreria di Simulink: Math Operations

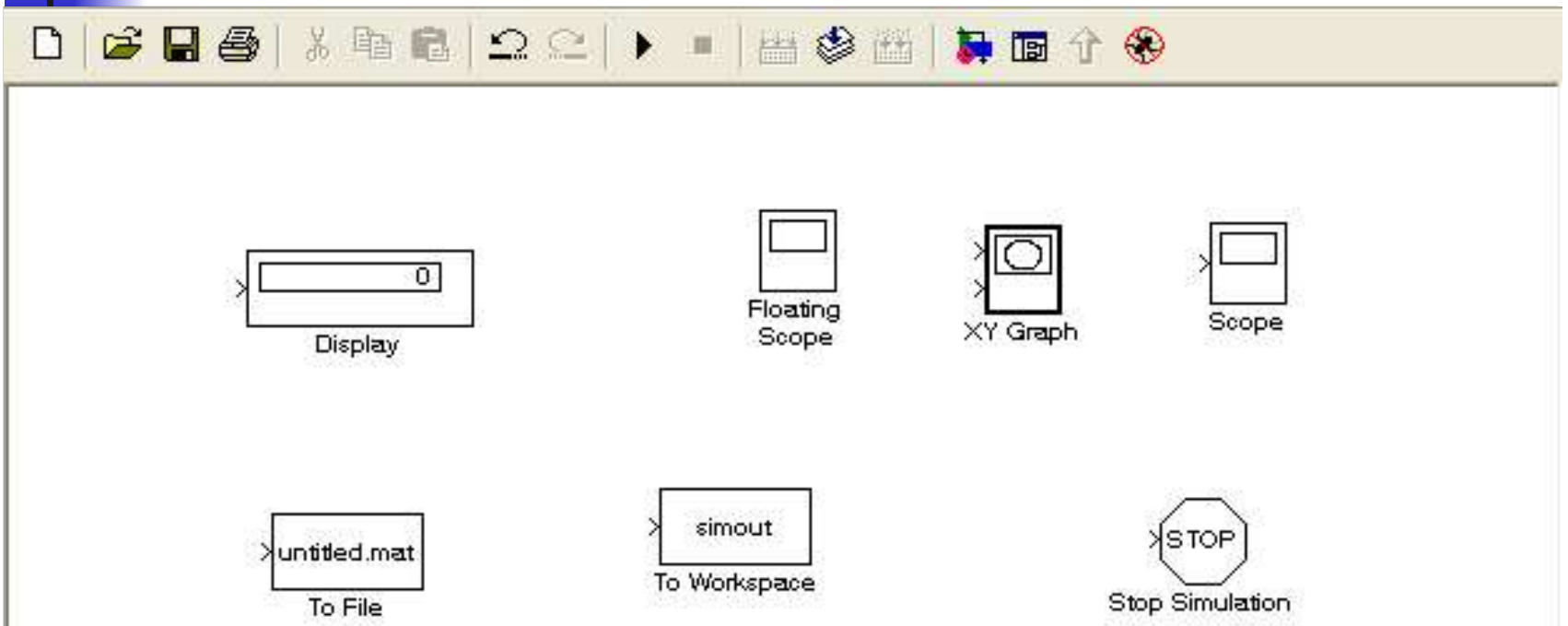


Libreria di Simulink: Signal Routing

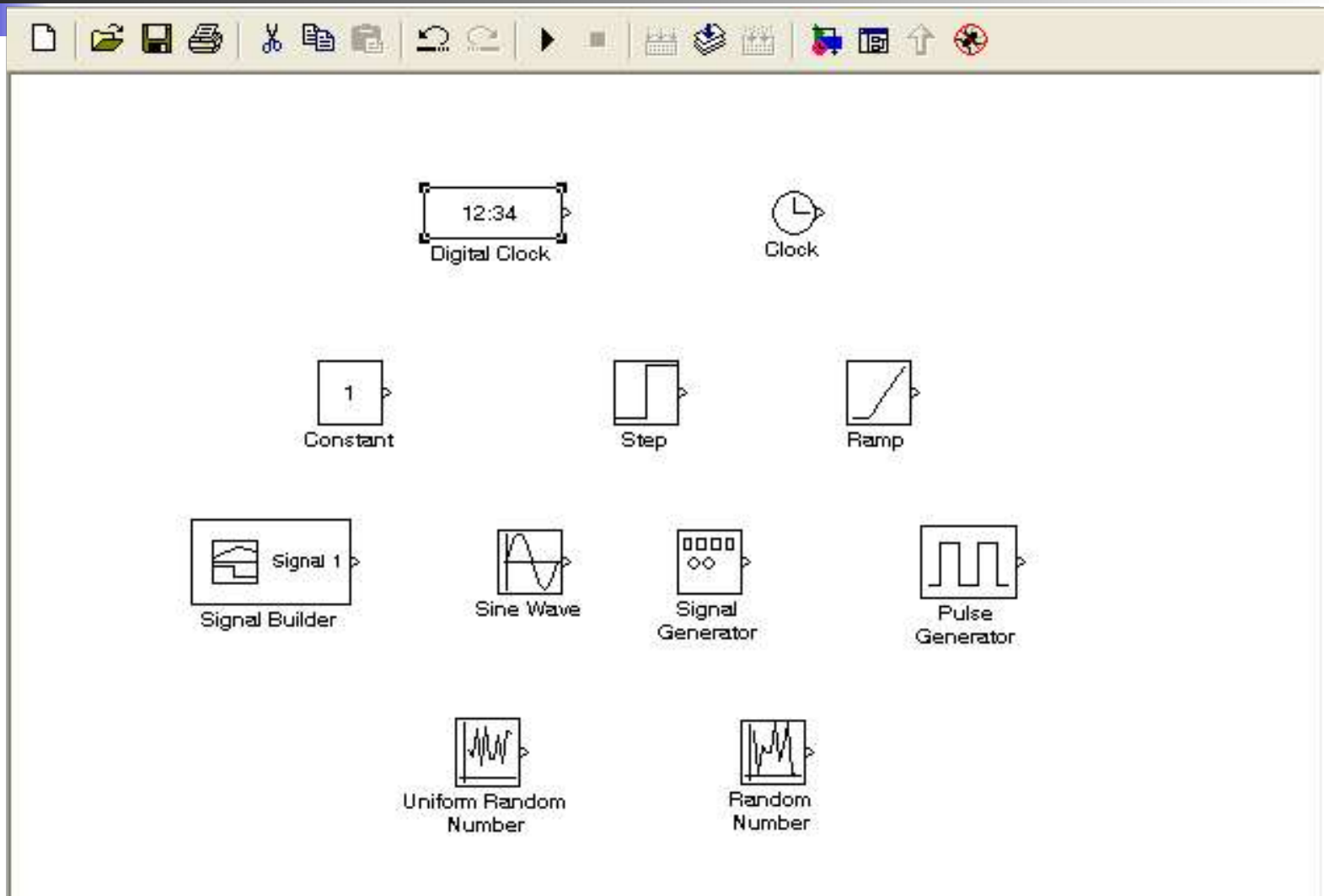


User defined functions

Libreria di Simulink: Sinks

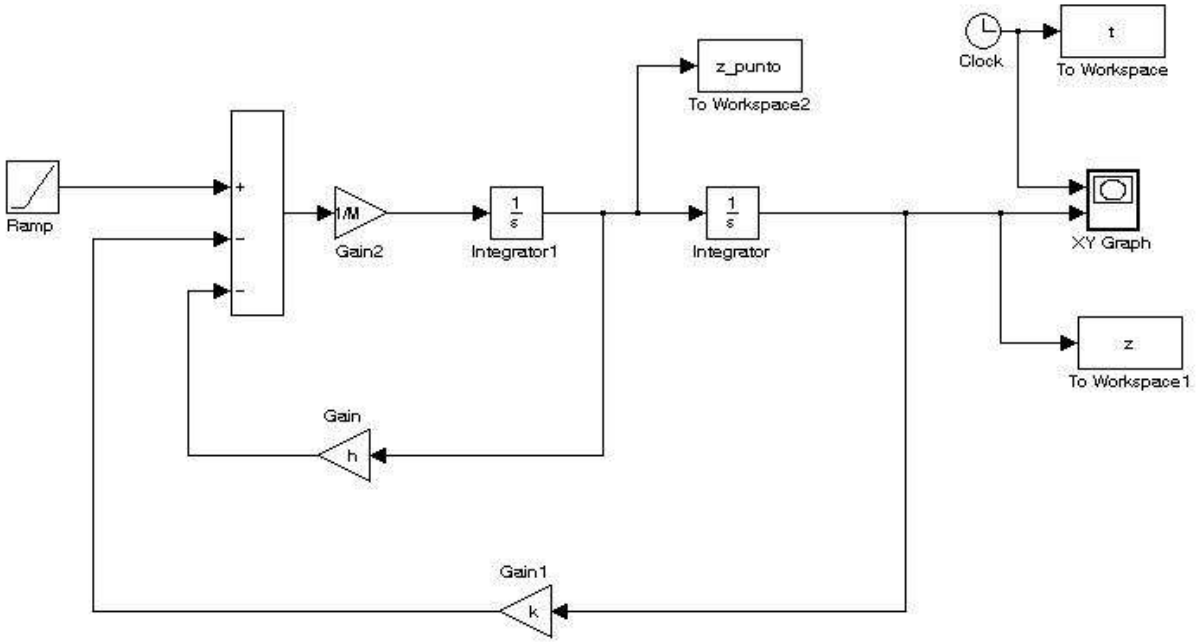


Libreria di Simulink: Sources



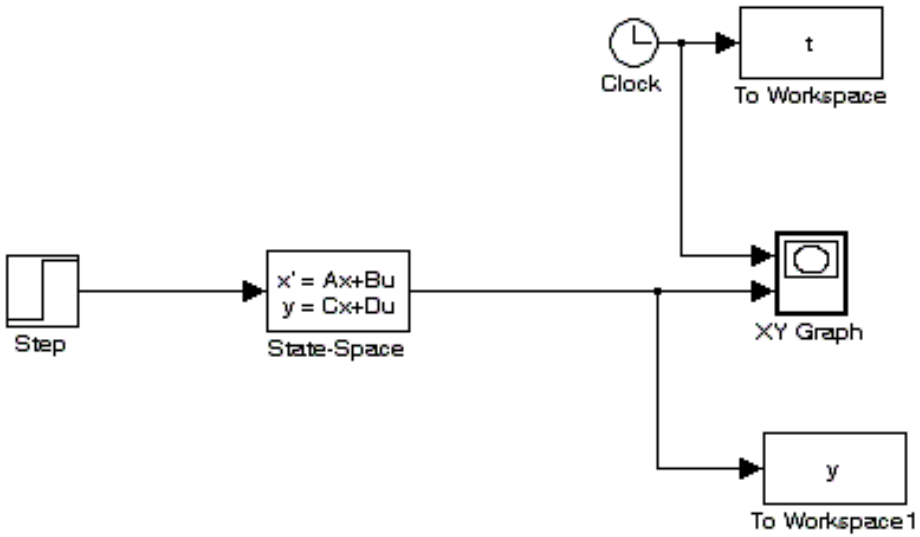
Creazione di un modello

$$\ddot{z}(t) = \frac{1}{M}\left(-h\dot{z}(t) - kz(t) + F(t)\right)$$



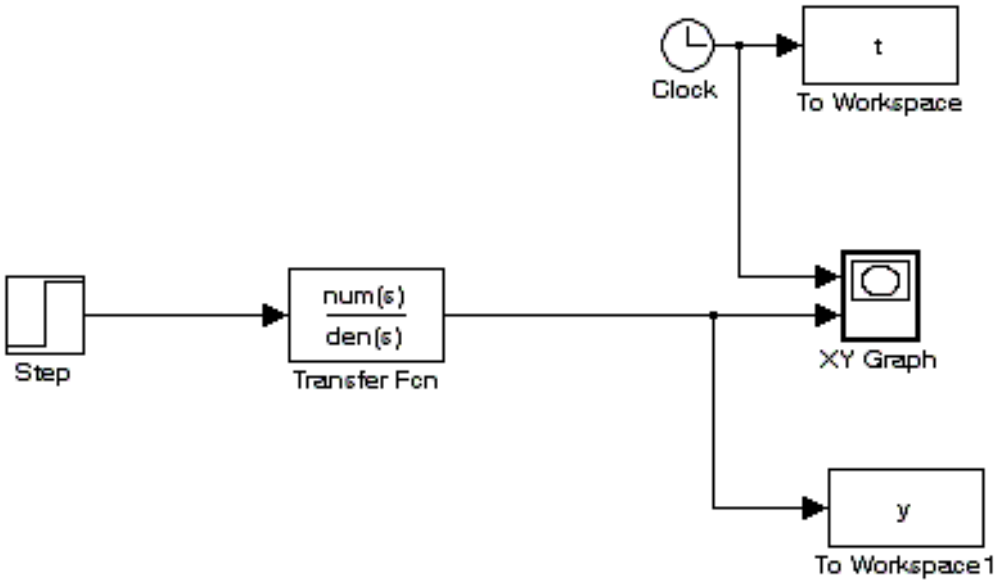
Creazione di un modello LTI

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$



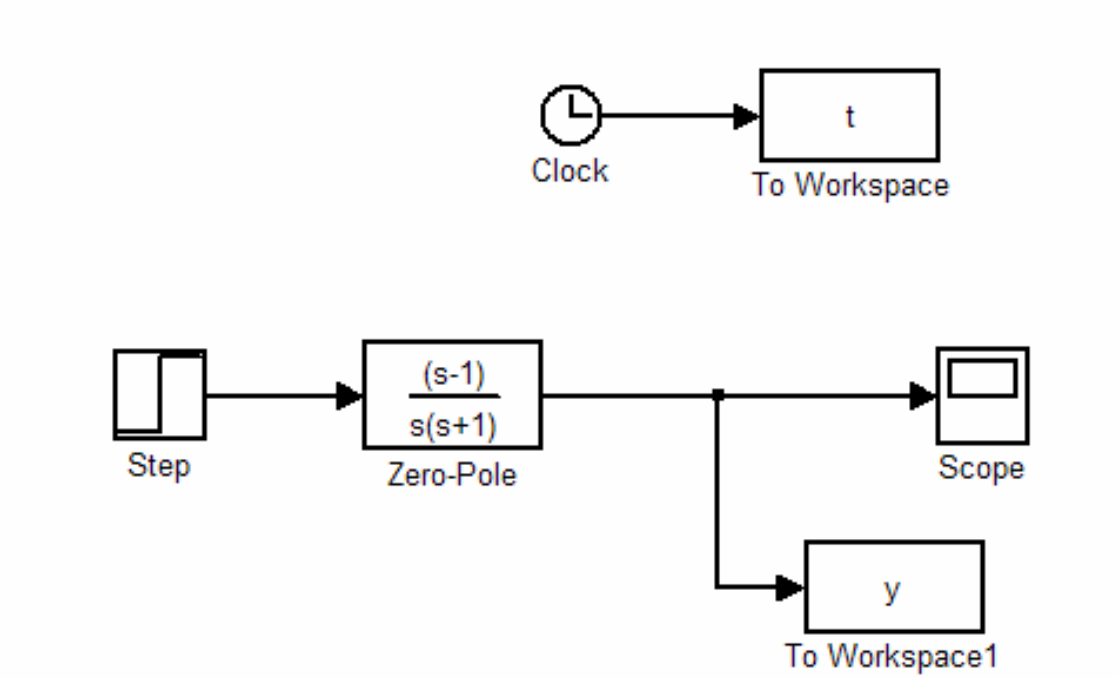
Creazione di un modello LTI

$$G(s) = \frac{1/M}{s^2 + sh/M + k/M}$$



Creazione di un modello LTI

$$G(s) = \frac{1}{M} \frac{1}{\left(s - \frac{h}{2M} + \sqrt{\frac{h^2}{4M^2} - \frac{k}{M}}\right)\left(s - \frac{h}{2M} - \sqrt{\frac{h^2}{4M^2} - \frac{k}{M}}\right)}$$

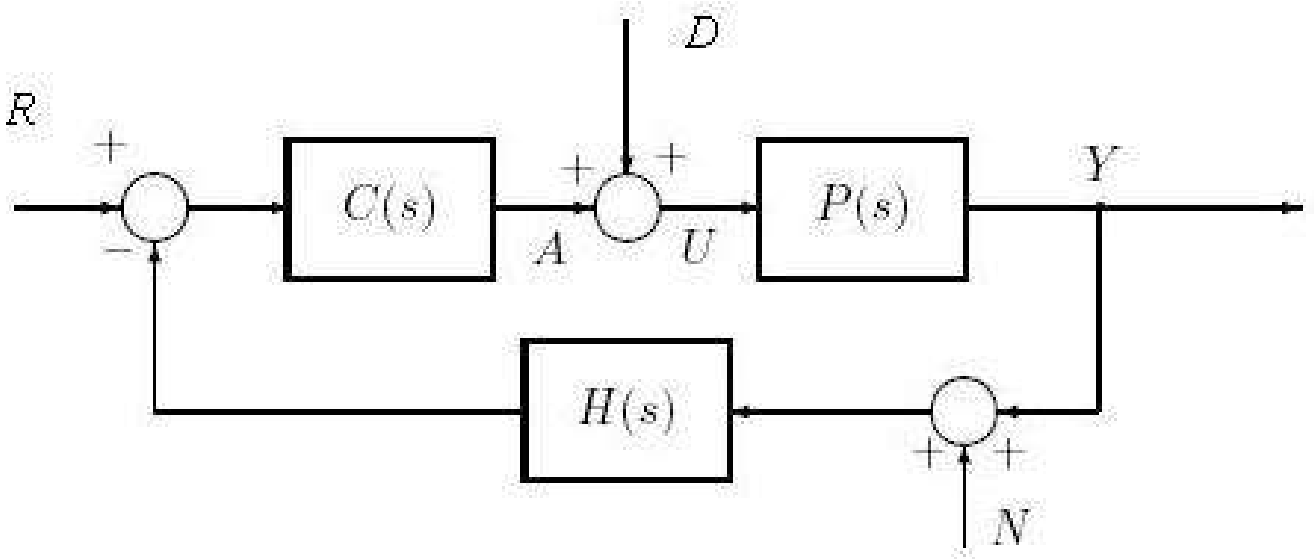


Schema a retroazione

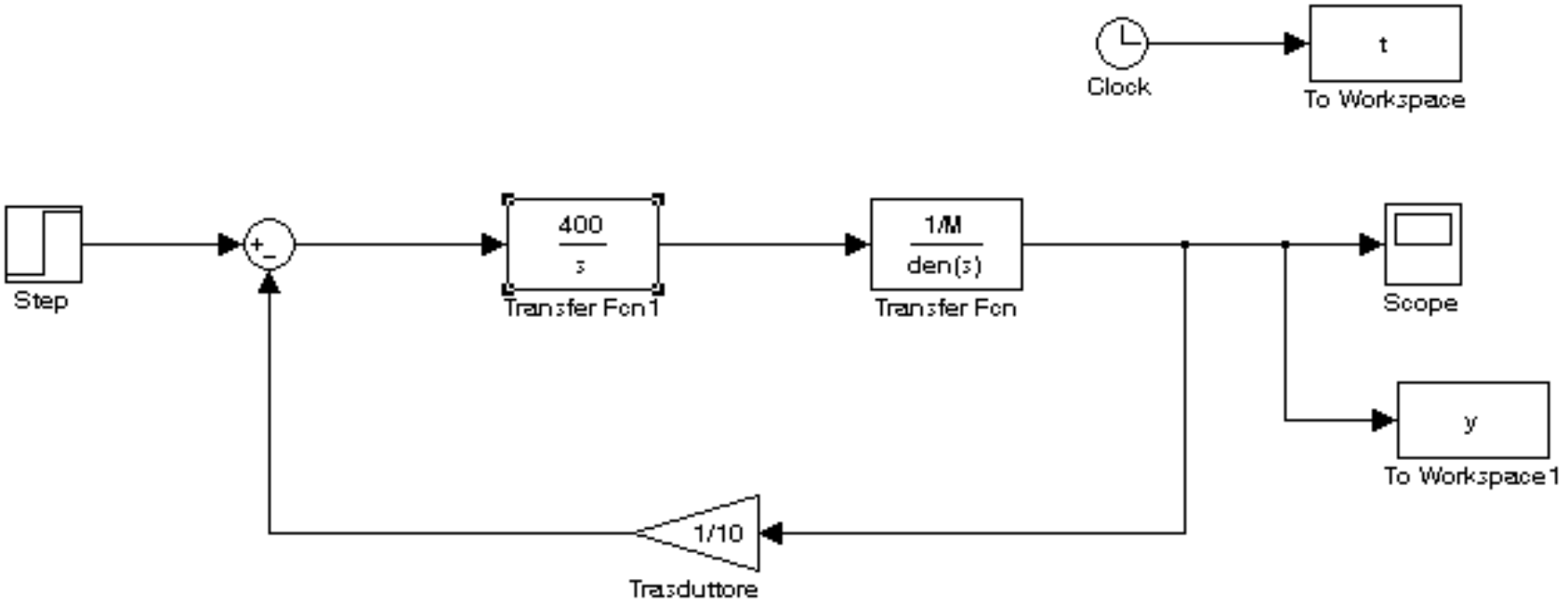
Specifiche: ad un riferimento a gradino unitario corrisponda un'uscita desiderata di ampiezza 10 e l'errore assoluto a regime sia nullo

k=100; h=80; M=10;

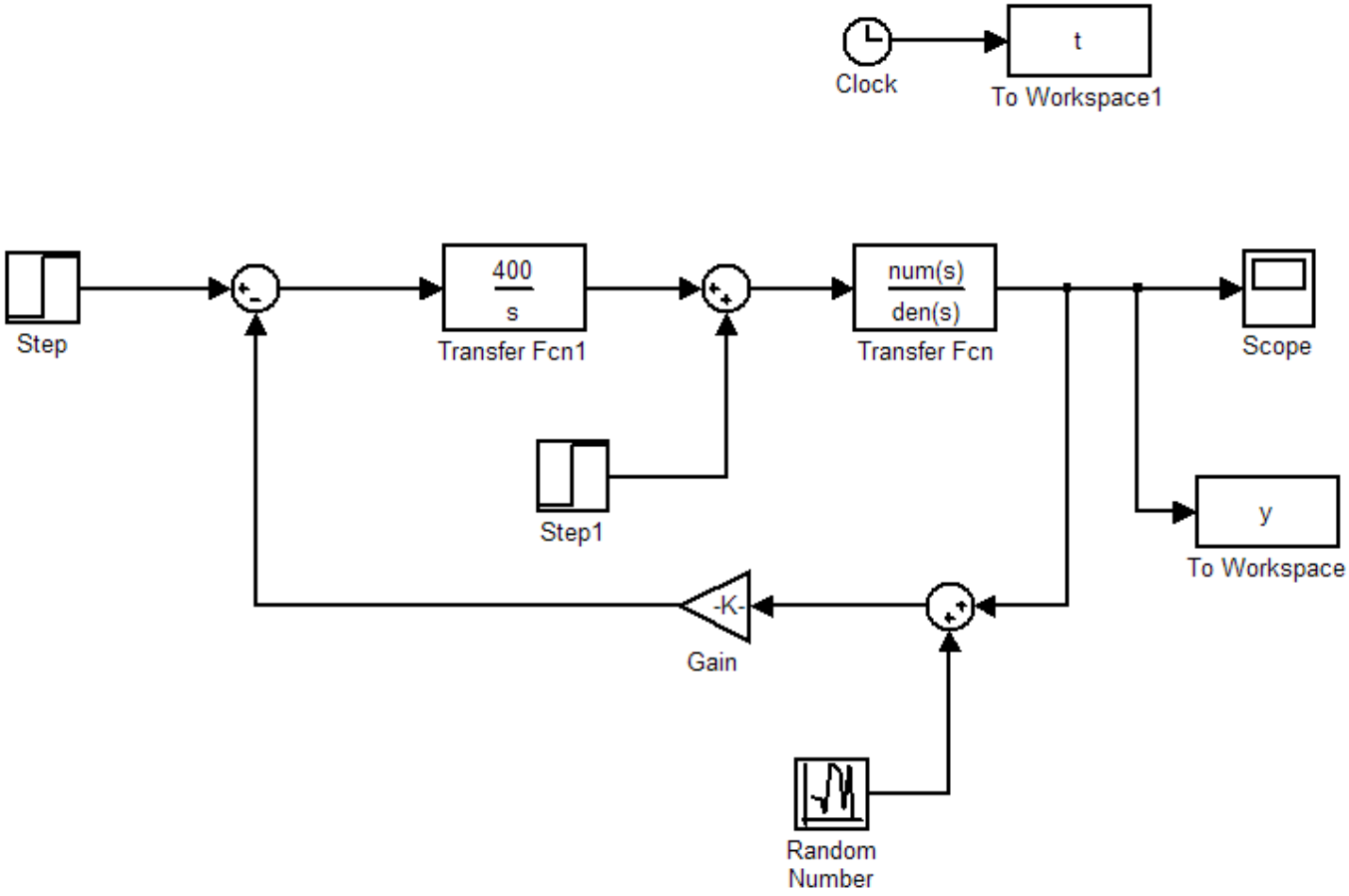
H=1/10; C(s)=Kc/s; 0<Kc<8000



Schema a retroazione

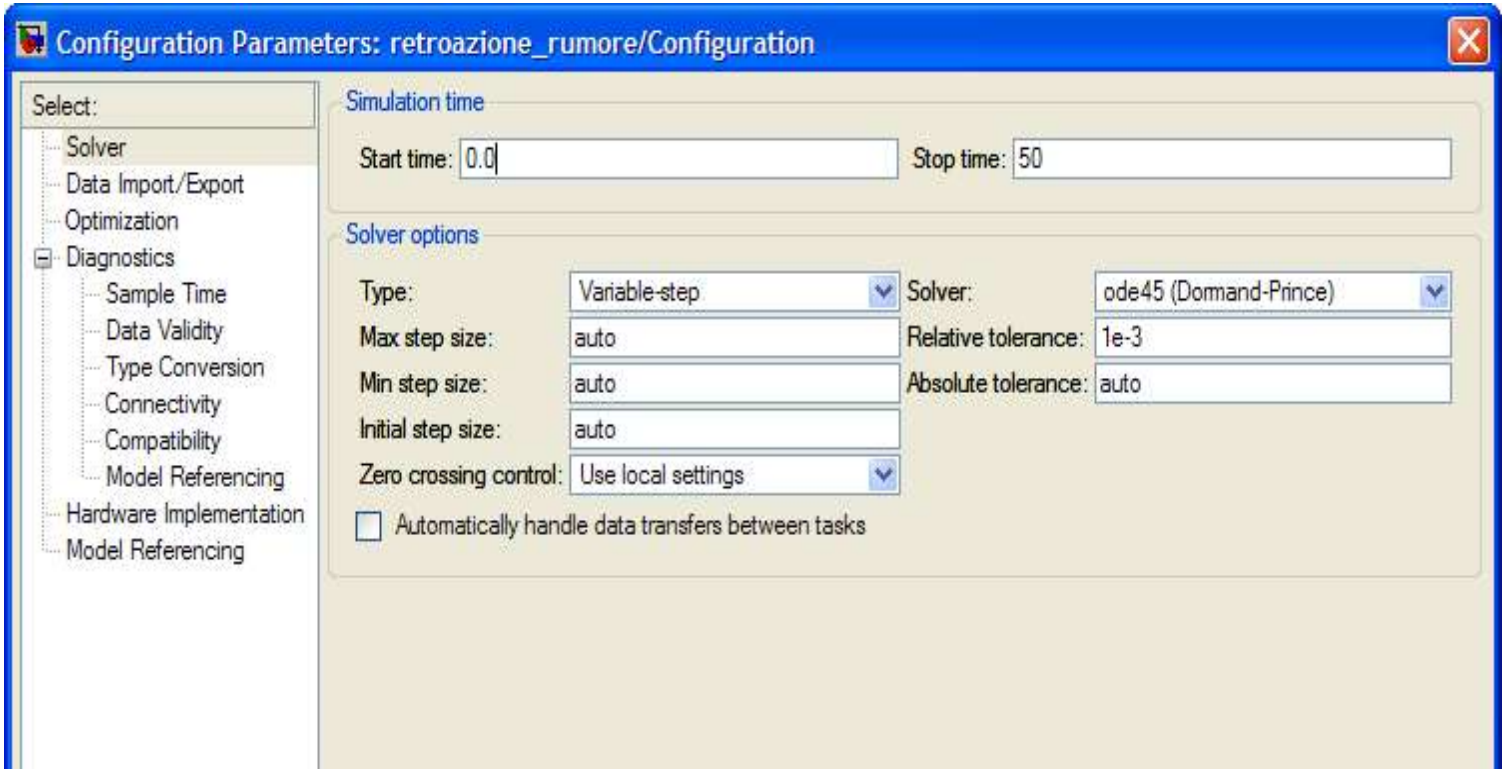


Schema a retroazione con disturbo e rumore di misura



Parametri di Simulazione

E' possibile settare i parametri di simulazione scegliendo **Configuration Parameters** (o solo Parameters per versioni precedenti alla 7) dal menù **Simulation**.



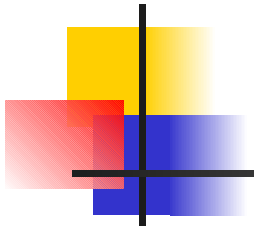
Parametri di Simulazione

Simulation Time

Si possono scegliere l'istante iniziale e finale della simulazione.

Il tempo di simulazione e il tempo "reale" non sono la stessa cosa. Per esempio eseguire una simulazione di 10 secondi in genere non richiederà 10 secondi.

La somma del tempo necessario a eseguire una simulazione dipende da molti fattori: la complessità del modello, il passo del metodo di risoluzione, la velocità del computer.



Parametri di Simulazione

Metodi di integrazione

La simulazione di un modello in Simulink richiede l'utilizzo di un metodo di integrazione numerica. Simulink mette a disposizione numerosi metodi.

- metodi a passo fisso
- metodi a passo variabile

Per i metodi a passo fisso è possibile scegliere il passo, per quelli a passo variabile determinare il passo minimo e il passo massimo.

Parametri di Simulazione

Tolleranza

Il metodo di risoluzione monitorizza l'errore ad ogni passo. Durante ogni step viene calcolato il valore dello stato e determinato l'errore locale di stima. Allora l'errore commesso è comparato con quello accettabile che è funzione della tolleranza relativa e di quella assoluta.

- La tolleranza relativa rappresenta l'accuratezza con cui è valutato lo stato del sistema ad ogni passo. Di default è 1e-3, ciò vuol dire che lo stato è calcolato con un errore massimo dell'0.1%.
- La tolleranza assoluta è un valore di soglia per l'errore.

Riferimenti bibligrafici

- Guida Matlab

- Roberto Bucher
  Introduzione a Simulink
  Matlab e Simulink nella regolazione automatica

  Disponibili al sito: www.dti.supsi.ch/~bucher/