



Introduzione a Matlab e Simulink

Matteo Sartini

matteo.sartini@unibo.it

<http://www-lar.deis.unibo.it/people/msartini>

Tel (051-20)93872



Cosa è Matlab?

- Matlab (MATrix LABoratory) è un ambiente di programmazione per applicazioni **scientifiche**, di **analisi numerica** e per la **simulazione** di sistemi dinamici.
- L'elemento di base di Matlab è la **matrice**.
- Matlab contiene:
 - Un vasto set di funzioni di base **general purpose**;
 - Possibilità di definire nuove funzioni
 - Estensioni application oriented (**toolboxes**);
 - es. Control System Toolbox
 - Un'interfaccia grafica interattiva per la modellazione e la simulazione di sistemi dinamici;
 - **Simulink**
 - Altri toolbox non di interesse nel corso di Controlli Automatici LA/LB (Signal Processing, Financial ecc...)



Perché Matlab?

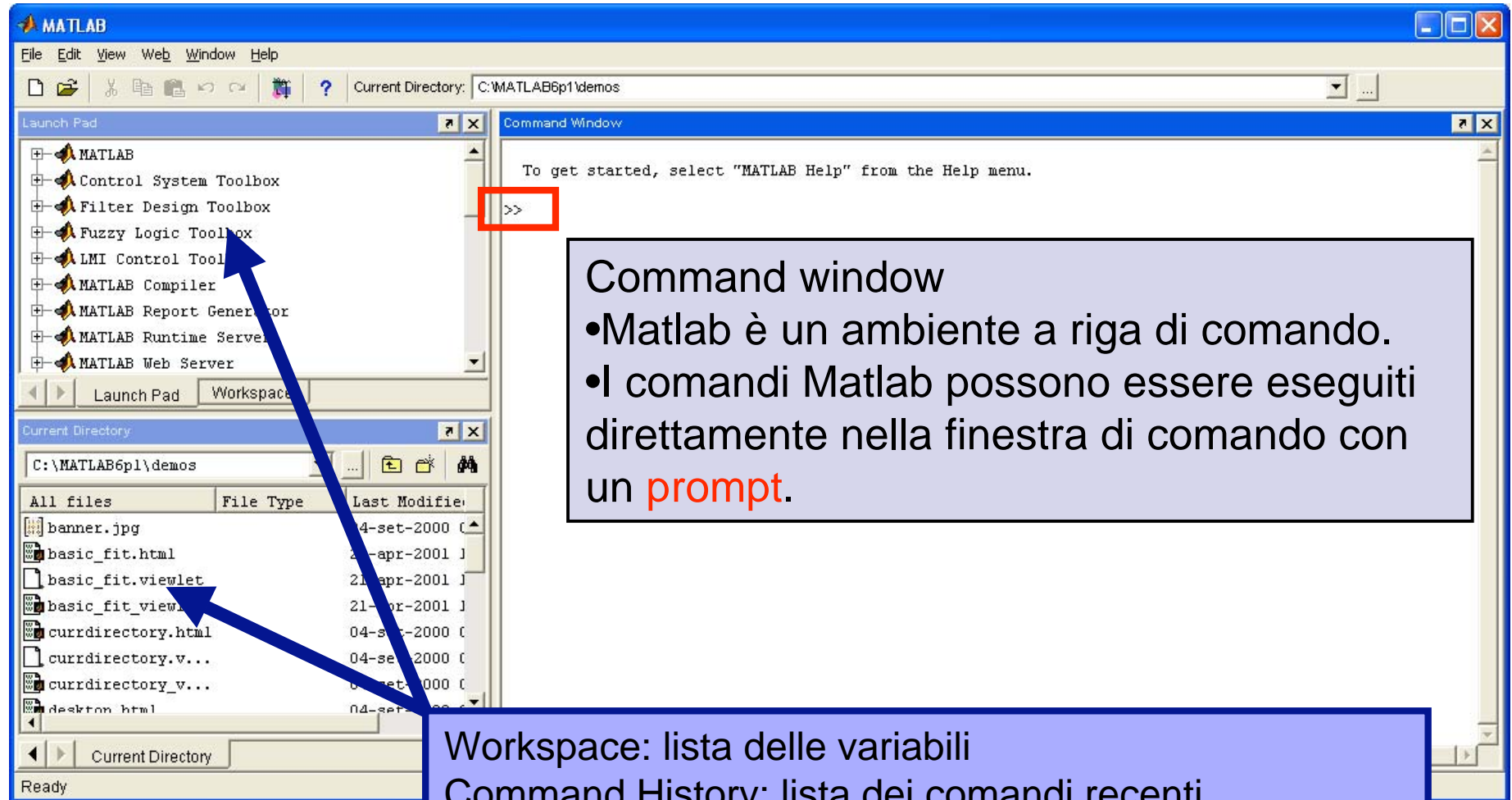
- Per il corso:

- strumento utile per la verifica personale dei concetti appresi, per la verifica degli esercizi e come approfondimento.
- programma utilizzato per sviluppare il **progetto di sistemi di controllo** (*Controlli Automatici LB*)
 - *Analisi dei sistemi (SISO, MIMO, lineari ecc...)*
 - *Sintesi dei controllori*

- Come ingegneri:

- ambiente di sviluppo software utilizzato nelle aziende per il progetto di sistemi di controllo, per la loro implementazione e sviluppo completo

L'interfaccia grafica



Command window

- Matlab è un ambiente a riga di comando.
- I comandi Matlab possono essere eseguiti direttamente nella finestra di comando con un **prompt**.

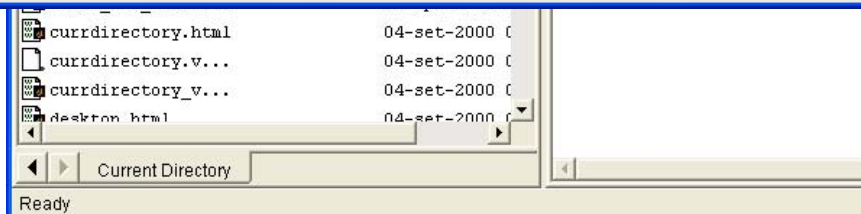
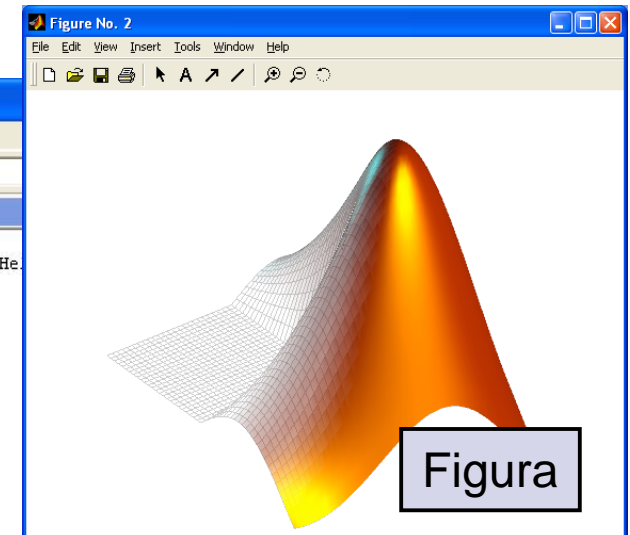
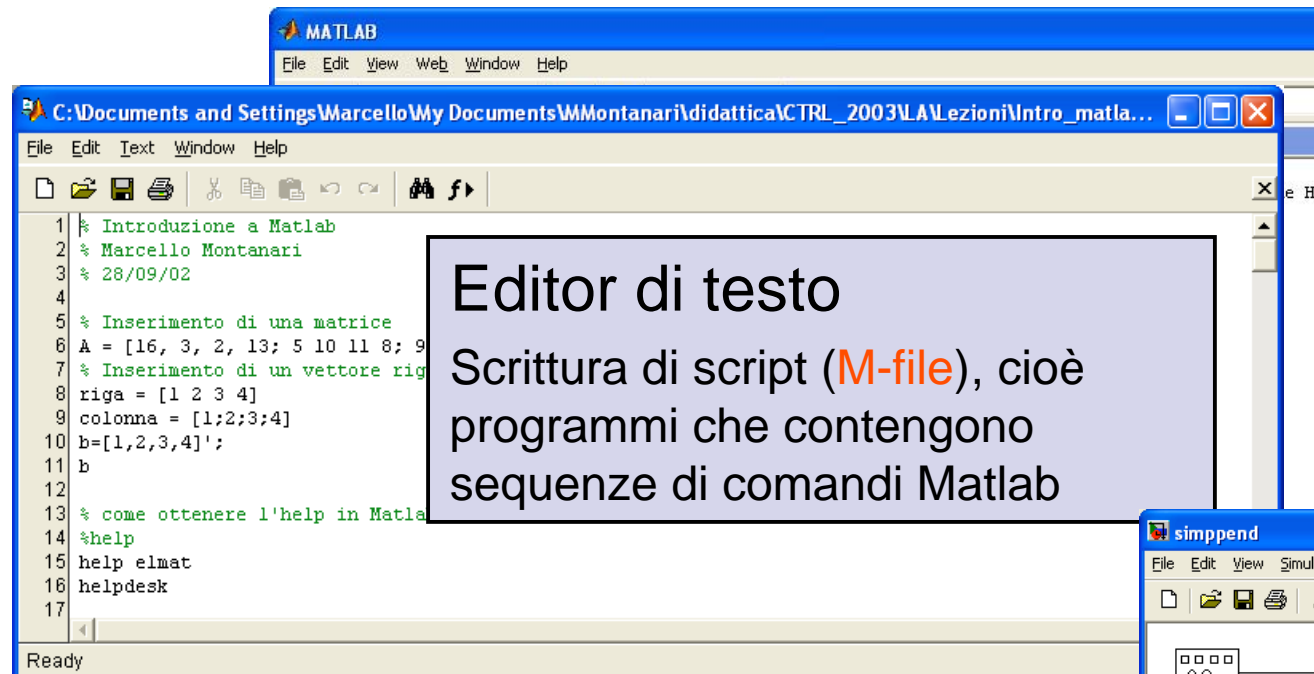
Workspace: lista delle variabili

Command History: lista dei comandi recenti

Current Directory: lista dei file nella cartella corrente

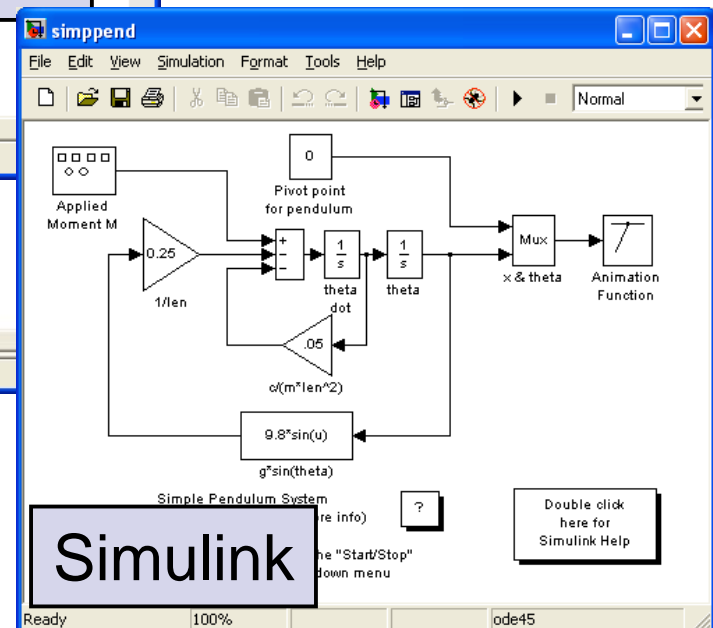
Launch Pad: accesso rapido a Help, Toolbox, Simulink

L'interfaccia grafica



interfaccia grafica a finestre:

- figure
- Schemi Simulink





Quali parti di Matlab ci interessano?

■ Matlab

- ☐ concetti base, uso delle matrici, vettori, ecc.
- ☐ la grafica con Matlab

■ Control System Toolbox

- ☐ definizione di sistemi dinamici (funzioni di trasferimento, sistemi in forma di stato, ecc.)
- ☐ strumenti per:
 - analisi di sistemi LTI (stabilità, diagrammi di Bode, luogo delle radici, ecc.)
 - progetto di regolatori

■ Simulink

- ☐ simulazione di sistemi dinamici (complessi)



L'help di Matlab

- 2 modi per ottenere l'help di una funzione:

- Help in linea:

- è sufficiente scrivere nella Command Window “**help** **<nomefunzione>**” per avere informazioni dettagliate sul funzionamento della funzione.

- Matlab Help Window

- Si esegue dal menu Help, contiene informazioni su tutte le funzionalità di Matlab, Simulink e i toolbox Help dettagliati in formato:

- Html (per accesso rapido)

- Pdf (per tutorial o approfondimenti)

- Da scaricare dal sito www.mathworks.com

Matlab come calcolatrice

- Valutare espressioni aritmetiche (al prompt dei comandi)

- Valutare $\sqrt{2} + 4 + \sin(0.2\pi) + e^2$

- ```
>> sqrt(2)+4+sin(0.2*pi)+exp(2)
```

- ```
ans =
```

- ```
13.3911
```

- Esiste un insieme di istruzioni matematiche (*help elfun, help elmat*)

- Aritmetiche

- Trigonometriche

- Esponenziali

- Per i numeri complessi

- Etc.

- N.B. Matlab è case-sensitive!!!



# Definizioni di variabili

- E' possibile definire variabili ed espressioni complesse:

```
>> a=4; b=2;
```

```
>> a*b
```

```
ans =
```

```
8
```

- Per cancellare una variabile (es. a):

```
>> clear a
```

- Per cancellare tutte le variabili

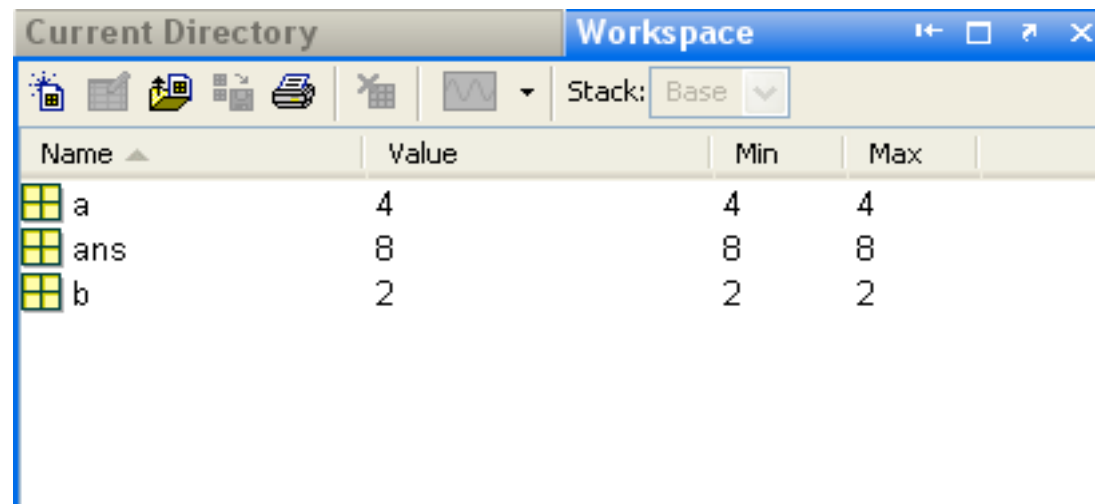
```
>> clear all
```

- Le matrici sono l'elemento di base in Matlab

- ☐ Ogni oggetto in Matlab è trattato come una matrice
- ☐ Gli scalari sono considerati casi particolari di matrici (matrici 1x1)
- ☐ Le matrici colonna (o riga) sono dette vettori (vectors)
- ☐ E' possibile definire, modificare, visualizzare e eseguire operazioni e funzioni su matrici

# Il workspace

- Ogni variabile definita in questo modo viene conservata in memoria nel workspace.
- Per workspace, si intende l'insieme di tutte le variabili



The screenshot shows the MATLAB Workspace window. The title bar is split into 'Current Directory' and 'Workspace'. The 'Workspace' tab is active, showing a list of variables. The 'Stack' dropdown is set to 'Base'. The table below lists the variables and their values.

| Name | Value | Min | Max |
|------|-------|-----|-----|
| a    | 4     | 4   | 4   |
| ans  | 8     | 8   | 8   |
| b    | 2     | 2   | 2   |

- Lista delle variabili del workspace: `who`
- Salvare il workspace (o alcune variabili): `save <file> <var1> <var2> ...`
- Caricare variabili salvate da disco: `load <file>`

# Inserimento di matrici

- Inserire una matrice 4x4

```
>> A = [16, 3, 2, 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

```
A =
```

```
16 3 2 13
 5 10 11 8
 9 6 7 12
 4 15 14 1
```

- Matlab mostra la matrice appena inserita
- Con “;” non viene visualizzato il risultato dell’istruzione a schermo
  - Utile nel caso di risultati “lunghi” (es. con matrici di dimensione molto elevata)

- Vettori riga e colonna:

```
riga = [1 2 3 4];
```

```
colonna = [1;2;3;4];
```

```
b=[1,2,3,4]';
```

# Operazioni matriciali

- in Matlab *tutte* le operazioni sono relative a matrici

- trasposta: `A'`

- Diagonale: `diag(A)`

- dimensioni di A

- Numero righe e colonne: `size(A)`

- Numero righe: `size(A,1)`

- per i vettori: `length(b)`

- alcune matrici utili ...

- Matrice di uno: `ones(m,n)`

- Matrice di zero: `zeros(m,n)`

- Matrice identità: `eye(n)`

- Matrice vuota: `X=[]`

# Come accedere agli elementi di una matrice

- singolo elemento

```
>> A(1,2)
```

```
3
```

```
>> A(end, end)
```

```
1
```

- una sottomatrice

```
>> A(1:3 , 2:4)
```

```
ans =
```

```
3 2 13
10 11 8
6 7 12
```

- selezionare la prima riga

```
>> A(1,:)
```

```
ans =
```

```
16 3 2 13
```

- selezionare la prima colonna

```
>> A(:,1)
```

```
ans =
```

```
16
5
9
4
```

- Sottomatrici (es. 2a e 4a riga)

```
>> A([2,4],:)
```

```
ans =
```

```
5 10 11 8
4 15 14 1
```

A =

```
16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1
```

- wildcard

Per accedere a intere righe o colonne di una matrice, si usa la wildcard ":"

# Come accedere agli elementi di una matrice

- singolo elemento

```
>> A(1,2)
```

```
3
```

```
>> A(end, end)
```

```
1
```

- una sottomatrice

```
>> A(1:3, 2:4)
```

```
ans =
```

```
3 2 13
```

```
10 11 8
```

```
6 7 12
```

- selezionare la prima riga

```
>> A(1,:)
```

```
ans =
```

```
16 3 2 13
```

- selezionare la prima colonna

```
>> A(:,1)
```

```
ans =
```

```
16
```

```
5
```

```
9
```

```
4
```

- Sottomatrici (es. 2a e 4a riga)

```
>> A([2,4],:)
```

```
ans =
```

```
5 10 11 8
```

```
4 15 14 1
```

A =

|    |    |    |    |
|----|----|----|----|
| 16 | 3  | 2  | 13 |
| 5  | 10 | 11 | 8  |
| 9  | 6  | 7  | 12 |
| 4  | 15 | 14 | 1  |

# Operazioni matriciali

- Operazioni su matrici:

- Operatori  $+$ ,  $-$ ,  $*$

- $*$  è il prodotto matriciale (righe per colonne)!

- $A[m \times n] * B[n \times p] = C[m \times p]$  (nell'esempio sotto non potrei fare  $b * a$ )

|                            |   |                             |   |                                                  |
|----------------------------|---|-----------------------------|---|--------------------------------------------------|
| $A =$<br>2 3<br>1 2<br>4 5 | * | $B =$<br>1 3 4 6<br>2 5 3 6 | = | $C =$<br>8 21 17 30<br>5 13 10 18<br>14 37 31 54 |
|----------------------------|---|-----------------------------|---|--------------------------------------------------|

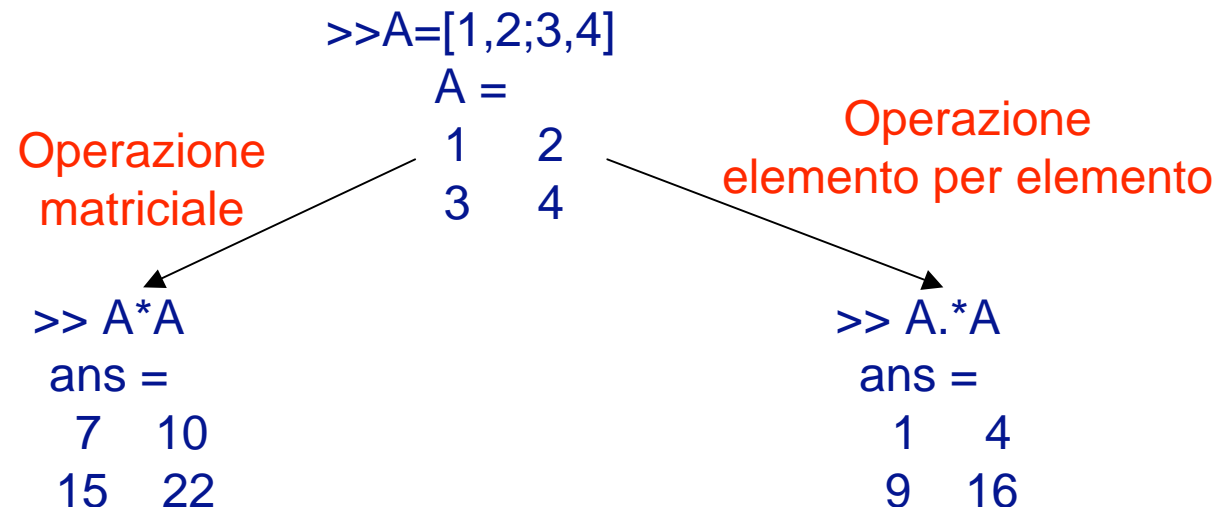
- Inversa di matrice: `inv(A)` (Per questa matrice A non esiste)
    - Se non è invertibile si può calcolare la pseudoinversa: `pinv(A)`
  - Determinante: `det(A)`
  - Autovalori: `eig(A)`

# Operazioni matriciali vs. operazioni elemento per elemento

- Normalmente le operazioni  $*$ ,  $/$  sono operazioni matriciali. Per eseguire operazioni elemento per elemento si antepone il punto  $(.)$  all'operatore

- L'operazione viene eseguita fra i singoli elementi delle matrici. Ci deve essere coerenza fra le dimensioni degli operandi.  
 $A[m*n] * B[m*n] = C[m*n]$

□ esempio  $.*$   $./$   $.^$





## I vettori


I vettori hanno due funzioni fondamentali in matlab:

- Rappresentazione dei polinomi (un polinomio è descritto dal vettore dei suoi coefficienti);
- Rappresentazione di segnali (un segnale è rappresentato mediante la sequenza dei valori che assume in un insieme di istanti di tempo, quindi mediante un vettore).

# Vettori con numeri reali a intervalli regolari

E' Possibile definire dei vettori con numeri a intervalli regolari:

■ `c=[1:2:8]`


  
 Valore iniziale
   
 Passo
   
 Valore finale

`c =`  
 1    3    5    7

■ `d=[1:5,2:2:10,2]`

`d =`

1    2    3    4    5    2    4    6    8    10    2

■ `e=[3.2:-.1:2.5]`

`e =`

3.2    3.1    3.0    2.9    2.8    2.7    2.6    2.5

■ Utile per

- ☐ la selezione degli elementi di una matrice
- ☐ la generazione di vettori "temporali" equispaziati



## Polinomi e loro operazioni

- Sono definiti come vettori: Es:  $s^4 + 3s^3 - 15s^2 - 2s + 9$

```
>>pol=[1 3 -15 -2 9]
```

```
pol=
```

```
1 3 -15 -2 9
```

- Calcolo delle radici:

```
>>roots(pol)
```

```
>>ans=
```

```
-5.5745
```

```
2.5836
```

```
-0.7951
```

```
0.7860
```



## Polinomi e loro operazioni

■ Valutazione in un punto:  $s^4 + 3s^3 - 15s^2 - 2s + 9$

```
>> polyval(pol,0)
```

```
>> ans=
```

9

Prodotto di polinomi (conv) Es:  $(s + 1)(s + 1) = s^2 + 2s + 1$

```
>>pol1=[1 1]; pol2=[1 1];
```

```
>>polprod=conv(pol1,pol2)
```

```
polprod=
```

1 2 1



## M-file

- Matlab è un linguaggio di programmazione e un ambiente di calcolo interattivo
- **M-file**: file contenente codice Matlab
- Vengono scritti mediante un qualsiasi editor di testo e eseguiti chiamandoli dalla linea di comando
  - In Matlab è disponibile un editor di testo (Edit)
- 2 tipi di M-file: *script* e *funzioni*



# Script e funzioni

- Gli **script** si usano per automatizzare le sequenze di comandi.
  - Quando viene eseguito uno script, l'esecuzione dei comandi è del tutto equivalente alla scrittura del codice con la tastiera.
    - Si utilizzano per evitare di riscrivere la stessa sequenza di comandi ripetutamente
  - Non hanno argomenti di input e output, tutte le variabili sono globali.
- Le **function** si usano per estendere le funzionalità di Matlab.
  - Normalmente generano una o più uscite (matriciali) dipendenti dai parametri in ingresso.
  - Le variabili sono locali alla funzione.

```
function [output]=nomefunction(input)
 istruzioni;
```

# Esempio di script e di funzione

## ■ Script

```
A1=[1 2 3];
```

```
A2=[3 4 5];
```

```
y=A1.*A2-A2.^3+log10(1+A2.^2)
```

```
y1=myfun(A1,A2)
```

## ■ Funzione

```
%Descrizione dell'operazione implementata da myfun
%In ingresso ci devono essere due vettori della stessa
% lunghezza
%y=myfun(x1,x2)=x1.*x2-x2.^3+log10(1+x2.^2)
y=x1.*x2-x2.^3+log10(1+x2.^2);
```

*commento*

*esegue lo script*

```
1 function y = myfun1(x1,x2)
2 %Descrizione dell'operazione implementata da myfun
3 %In ingresso ci devono essere due vettori della stessa lunghezza
4 %y=myfun(x1,x2)=x1.*x2-x2.^3+log10(1+x2.^2)
5
6 y=x1.*x2-x2.^3+log10(1+x2.^2);
7
```



# Programmare in Matlab

- Matlab è un linguaggio di programmazione
- Esistono comandi per il controllo di flusso:
  - `if...elseif...else...end`
  - `while...end`
  - `for`
  - `switch`
  - `break`

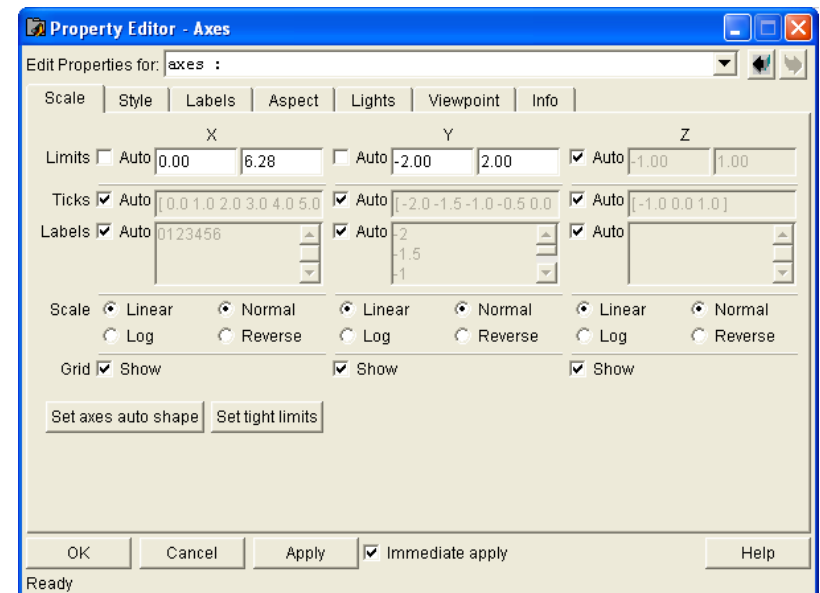
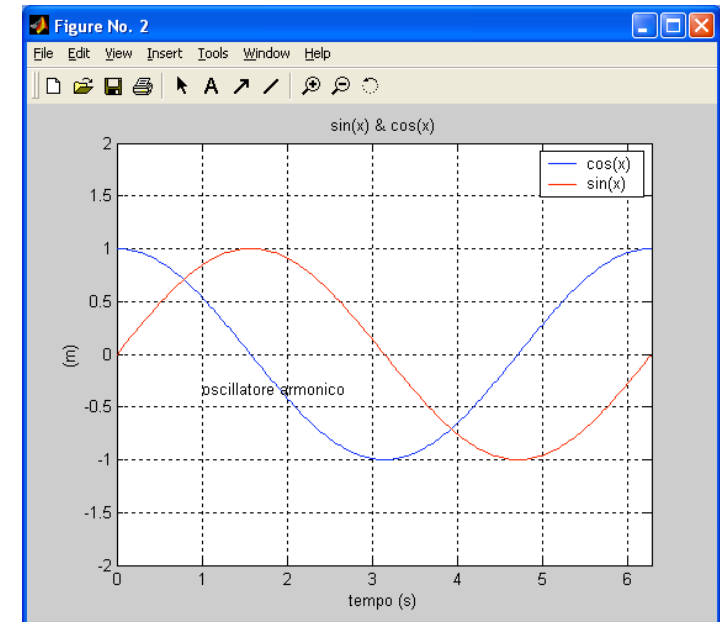


# La grafica in Matlab

■ *v. File matlab “grafica.m”*

# La grafica in Matlab

- I grafici vengono visualizzati in *figure*
- Il comando di base per la grafica è **plot(y)** – visualizza gli elementi del vettore y rispetto agli indici del vettore stesso  
**plot(x,y)** – visualizza il vettore y vs. il vettore x
- E' possibile modificare in modo interattivo l'aspetto dei grafici
  - mediante il *Plot Editing Mode*
  - mediante riga di comando





## Generare i dati

- Per visualizzare una qualsiasi funzione  $y=f(x)$  in Matlab, è SEMPRE necessario creare i vettori  $x$  e  $y$  nel dominio di interesse:  
(N.b. Matlab è un programma per l'analisi numerica, non simbolica!)
  - % un oscillatore armonico  
 $t=[0:\pi/100:2*\pi];$   
 $x=\cos(t);$   
 $y=\sin(t);$
  - E' importante selezionare la "risoluzione" lungo l'asse  $x$  sufficientemente elevata

# Scegliere la finestra

- **figure(n)** specifica su quale figura lavorare

`figure(1)`

`plot(t,x)`

% per disegnare entrambe le funzioni

`plot(t,x,t,y)`

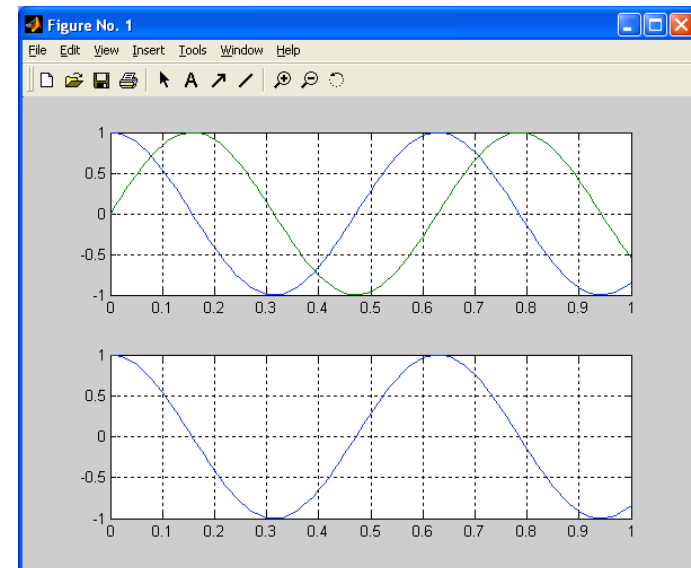
- **subplot** permette di suddividere la finestra in più grafici, per visualizzare contemporaneamente diversi segnali

□ `figure(1)`

`subplot(211),...`

`subplot(212),...`

- **hold on** mantiene il grafico presente nella figura
- **clf** - pulisce la figura corrente





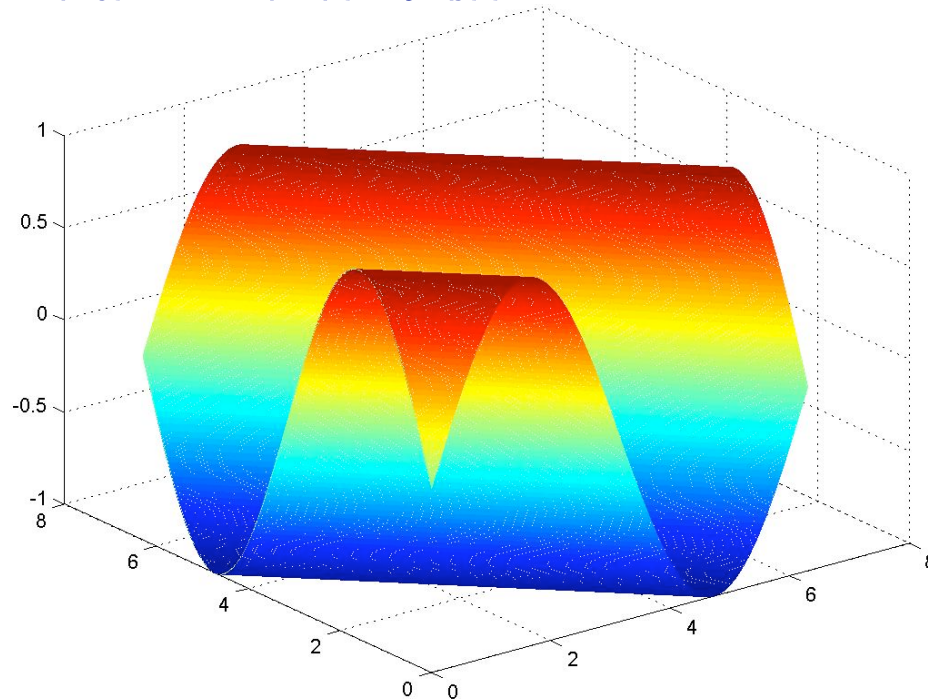
# Tracciare il grafico

- Esistono diversi comandi per rappresentare i dati:
- **plot** grafico 2-D con scala lineare lungo entrambi gli assi
- **loglog** grafico con scale logaritmiche per entrambi gli assi
- **semilogx** grafico con scala logaritmica per l'asse x e lineare per l'asse y
- **semilogy** grafico con scala logaritmica per l'asse y e lineare per l'asse x
- La struttura dei comandi è ***plot(x1,y1,x2,y2,...)***
- E' possibile specificare il colore e il tipo di linea dei grafici
  - v. help plot

## Grafici 3-D

- Possibilità di tracciare grafici tridimensionali (mesh):

```
>>x=(0:0.01:2*pi);y=(0:0.01:2*pi);
>>for i=1:length(x)
 for j=1:length(y)
 z(i,j)=sin(x(i)+y(j));
 end
end
>>mesh(x,y,z)
```



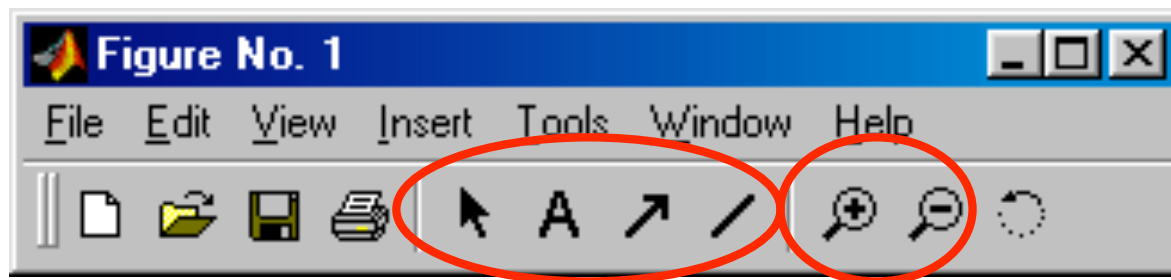


# Gestione degli assi

- **axis ( [XMIN XMAX YMIN YMAX] )**
    - ☐ imposta la scala degli assi
  - **grid on / grid off**
    - ☐ abilita e disabilita la griglia
  - **title**
    - ☐ inserisce il titolo
  - **xlabel, ylabel**
    - ☐ inserisce le etichette negli assi, p.e. per specificare le unità di misura
  - **legend**
    - ☐ inserisce la legenda
- 
- Queste caratteristiche si impostano anche in “Menu, Axes Properties”

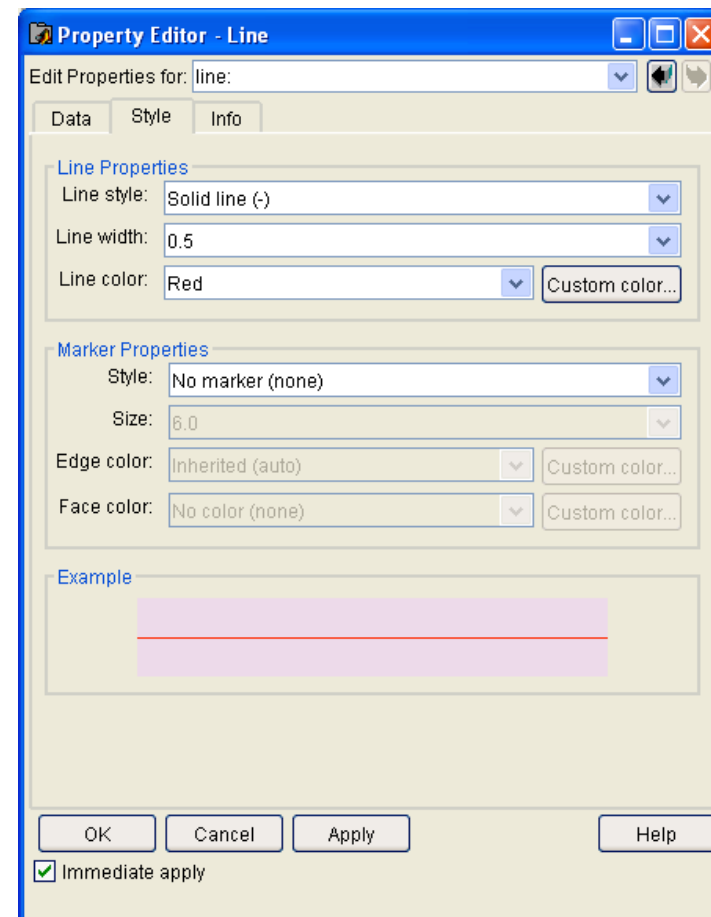
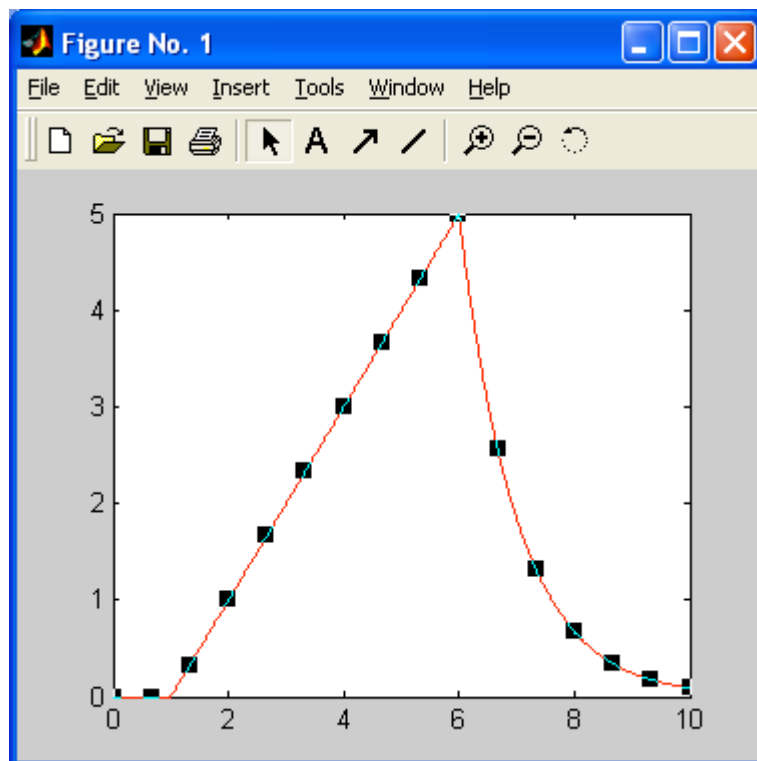
# Plot Editing Mode

- Interfaccia user-friendly interattiva per modificare le proprietà delle figure, delle linee, ecc. (Plot Editing Mode)
  - modificare le proprietà delle linee, degli assi, delle griglie
    - Menu “Edit”, “Figure Properties”, “Axes Properties”, “Current Object Properties”
  - zoom
  - aggiungere linee, frecce e testo alle figure



# Plot Editing Mode

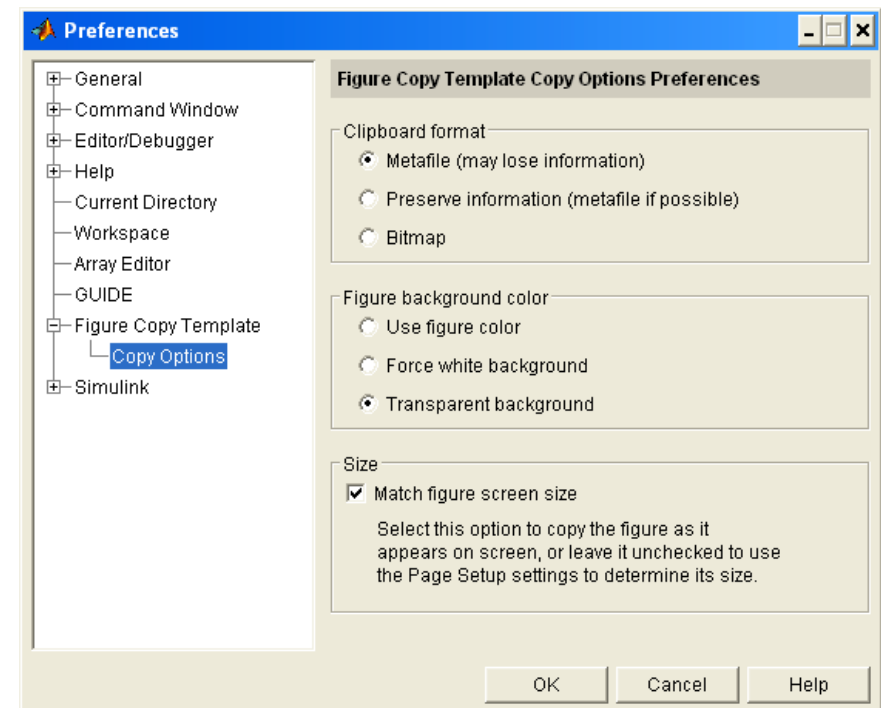
- E' possibile modificare le caratteristiche di una linea (es. colore, spessore, stile), selezionando la linea e andando nel menu "Edit, Current Object Properties"



# Esportare i grafici

## ■ Esportazione in Word/Powerpoint

- In Matlab, “Edit, Copy Figure” per copiare la figura negli appunti di Windows
- In Matlab, “Edit, Copy Options” per modificare le impostazioni
  - Scegliere “Metafile” come “clipboard format”
- In Word/Powerpoint, incollare con “Modifica, Incolla speciale” e selezionare “Enhanced Metafile”





# Control System Toolbox

■ *v. File matlab "ConTBox.m"*



## Il Control System Toolbox

- Il **Control System Toolbox** mette a disposizione una serie di strumenti per la modellazione, l'analisi e il controllo di sistemi dinamici (come f.d.t. o in forma di stato)
- E' costituito da una collezione di comandi scritti come *M-file*, che permettono di
  - inserire un sistema LTI in vari modi
    - come funzione di trasferimento
    - in forma di stato
  - manipolare sistemi dinamici
  - analizzare risposte temporali e frequenziali (diagrammi di Bode, Nyquist)
  - progettare un controllore con varie tecniche (luogo delle radici, progetto frequenziale, ecc.)



## Il Control System Toolbox

- E' lo strumento utilizzato nel corso di Controlli Automatici
- In particolare si considerano:
  - Sistema LTI definito
    - come f.d.t. in forma  $N(s)/D(s)$  e in forma poli/zeri
    - in forma di stato (definendo le matrici A, B, C, D)
  - Proprietà di una f.d.t. (poli, zeri, guadagno, ecc.)
  - Risposte temporali
  - Risposte frequenziali
  - Luogo delle radici
  - ecc.

# Inserimento di una f.d.t. (1° metodo)

## ■ Inserire

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_0} = \frac{s + 1}{s^2 + 2s + 3}$$

```
Num=[1 1]; den=[1 2 3];
```

```
G=tf(num,den)
```

Transfer function:

$s + 1$

-----

$s^2 + 2 s + 3$

→ Inseriti come polinomi  
(Vedi lucidi precedenti)

## Inserimento di una f.d.t. (2° metodo)

### ■ Inserire

$$G(s) = k \frac{\prod_i (s - z_i)}{\prod_i (s - p_i)}, \quad z_i, p_i \in \mathbf{C}$$
$$= 20 \frac{(s + 2)(s - 4)}{(s + 3 + j)(s + 3 - j)}$$

$$s^2 + 2\delta\omega_n s + \omega_n^2 \rightarrow s_i = -\delta\omega_n \pm j\omega_n \sqrt{1 - \delta^2}$$

```
>> k=20;Z=[-2,4];P=[-3+i, -3-i];G=zpk(Z,P,k)
```

Zero/pole/gain:

20 (s+2) (s-4)

-----  
(s^2 + 6s + 10)

## Inserimento di una f.d.t. (3° metodo)

- In modo più intuitivo, definire la variabile s come **s=tf('s')**
- Inserire la f.d.t. come funzione razionale fratta in s

```
>> G3 = (s + 160)/(s^3 + 12 *s^2 + 30 *s + 100)
```

Transfer function:

**s + 160**

-----

**s^3 + 12 s^2 + 30 s + 100**

# Inserimento di un sistema in forma di stato

## ■ Inserire

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

```
A=[0,1 ; -2,-1];
```

```
B=[0;1];
```

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

```
D=0;
```

```
S=ss(A,B,C,D)
```

```
a =
```

```
 x1 x2
```

```
 x1 0 1
```

```
 x2 -2 -1
```

```
b =
```

```
 u1
```

```
 x1 0
```

```
 x2 1
```

```
c =
```

```
 x1 x2
```

```
 y1 1 0
```

```
d =
```

```
 u1
```

```
 y1 0
```

Continuous-time model.

# Rappresentazione di un sistema dinamico

- le rappresentazioni con **ss**, **tf** o **zpk** sono equivalenti. Si può passare dall'una all'altra indifferentemente

```
>> zpk(G3)
```

Zero/pole/gain:

(s+160)

-----  
(s+10) (s^2 + 2s + 10)

```
>> tf(G3)
```

Transfer function:

s + 160

-----  
s^3 + 12 s^2 + 30 s + 100

```
>> ss(G3)
```

a =

|    | x1  | x2     | x3      |
|----|-----|--------|---------|
| x1 | -12 | -1.875 | -0.7813 |
| x2 | 16  | 0      | 0       |
| x3 | 0   | 8      | 0       |

b =

|    | u1 |
|----|----|
| x1 | 1  |
| x2 | 0  |
| x3 | 0  |

c =

|    | x1 | x2     | x3   |
|----|----|--------|------|
| y1 | 0  | 0.0625 | 1.25 |

d =

|    | u1 |
|----|----|
| y1 | 0  |

Continuous-time model.



## Proprietà delle f.d.t.

- Con **tfdata** si estraggono il numeratore e il denominatore della f.d.t.

```
[num,den] = tfdata(G1,'v')
```

- Con **zpkdata** si estraggono gli zeri, i poli e la costante di trasferimento della f.d.t.

```
[z,p,k] = zpkdata(G1,'v')
```

- Con **ssdata** si estraggono le matrici del sistema scritto in forma di stato (in una base nello spazio degli stati scelta da Matlab)

```
[A,B,C,D] = ssdata(SYS)
```



## Proprietà delle f.d.t.

### ■ Altre proprietà delle f.d.t.

|           |                                                    |
|-----------|----------------------------------------------------|
| damp      | pulsazione naturale e coeff. smorz. di poli e zeri |
| dcgain    | guadagno statico                                   |
| pole, eig | poli                                               |
| zero      | zeri                                               |
| pzmap     | grafico di poli e zeri nel piano complesso         |

# Risposte temporali

- Comandi per generare risposte temporali

**step** Risposta al gradino unitario

**impulse** Risposta all'impulso di Dirac

**Initial** Risposta al movimento libero

**Isim** Risposta a un ingresso generico (definito mediante vettore dell'asse dei tempi e vettore dell'ingresso)

Grafico:

**step(S)**

Memorizza i dati:

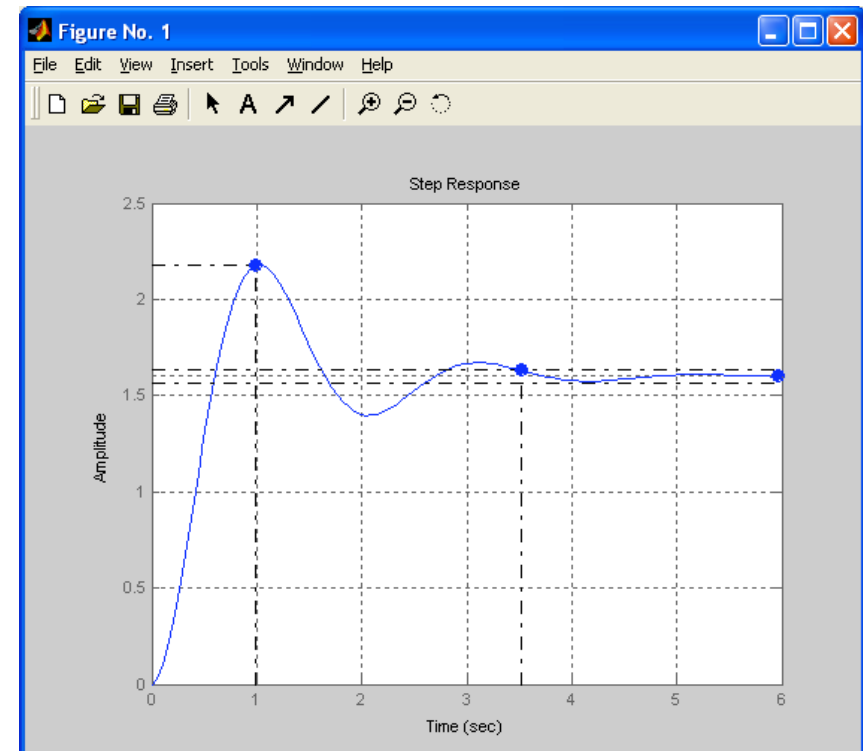
**[T,Y]=step(S);**

**plot(T,Y)**

- Caratteristiche della risposta al gradino:

- tempo di assestamento,
- tempo di salita,
- sovraelongazione, ecc.

*(Valutabili dalla finestra della risposta – pulsante destro del mouse)*



# Risposte temporali

- Risposta a un ingresso generico
  - Successivamente si vedrà come realizzarla con Simulink

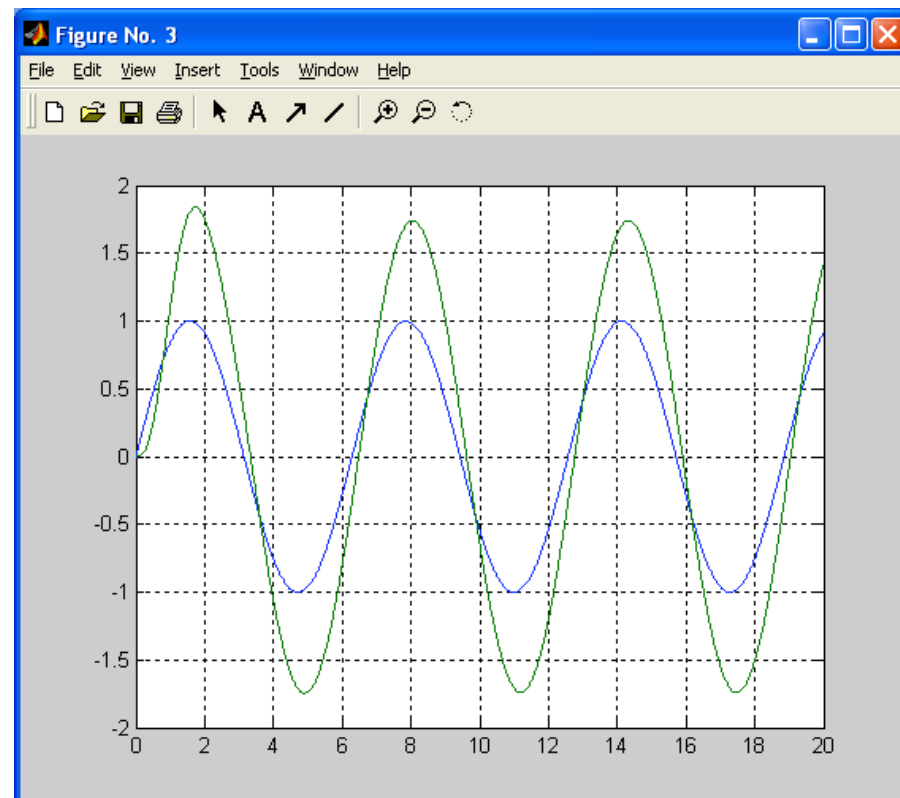
```
G1=tf(20*[1 8],[1 12 30 100]);
```

```
t = 0:0.01:20;
```

```
u = sin(t);
```

```
[y,t_y]=lsim(G1,u,t);
```

```
plot(t,u,t_y,y)
```



# Analisi frequenziale

## ■ Diagrammi di Bode

□ `bode(G1)`

## ■ Margini di fase e ampiezza

□ `margin(G1)`

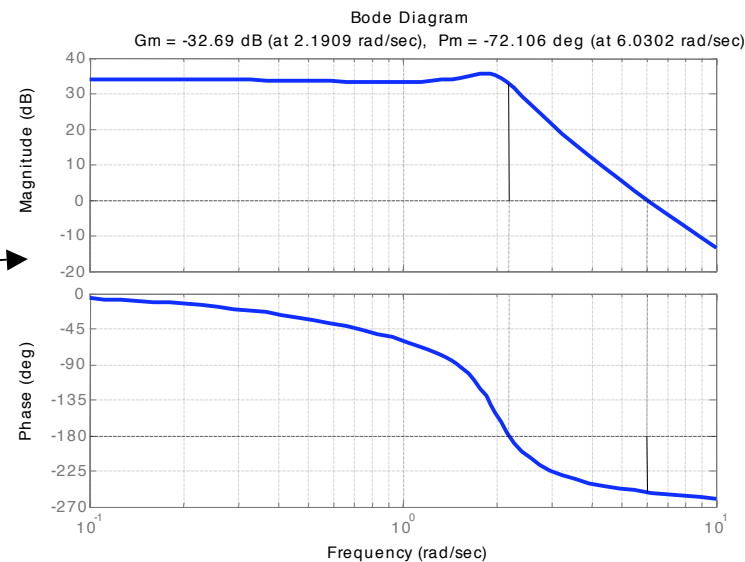
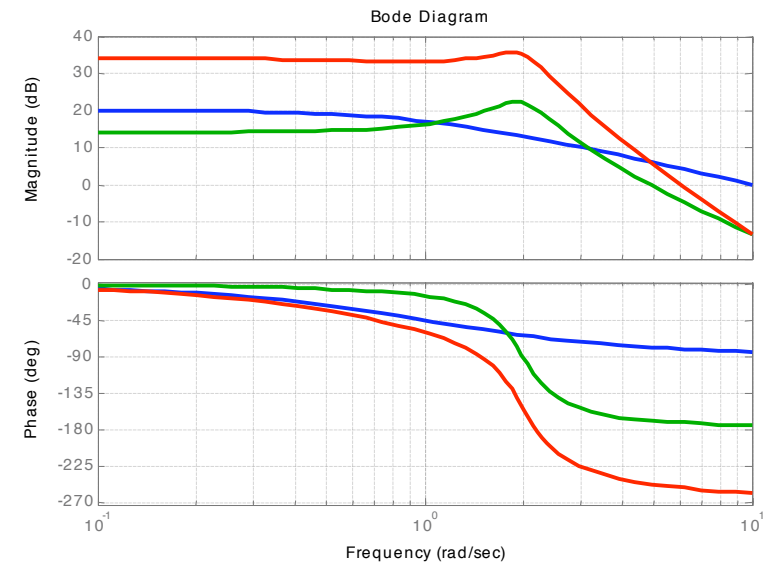
$$G1=10/(s+1)$$

$$G2=20/(s^2+0.8*s+4)$$

$$L=G1*G2$$

`bode(G1,G2,L)`

`margin(L)`

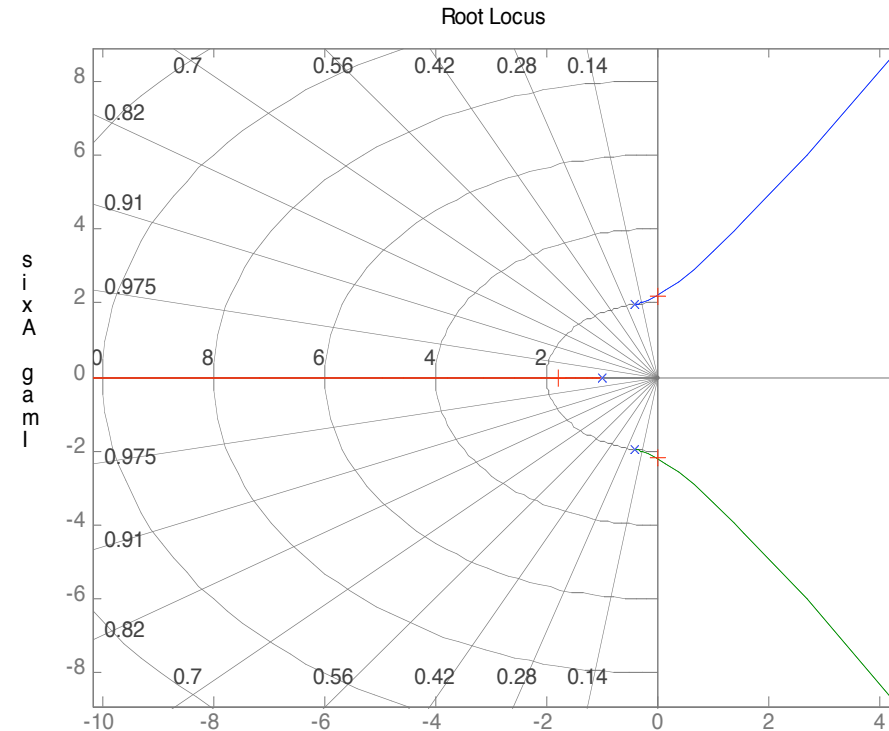


# Luogo delle radici

## ■ rlocus

□ rlocus(L)

□ sgrid



Dopo aver disegnato il luogo delle radici con rlocus, per trovare il guadagno del regolatore K in retroazione che corrisponde a una certa posizione dei poli, si usa:

**rlocfind(L)**

In modo interattivo, si clicca sul luogo delle radici per indicare la posizione del polo desiderato (es. poli c.c. con parte reale nulla). Matlab restituisce la posizione dei poli e il guadagno:

Select a point in the graphics window

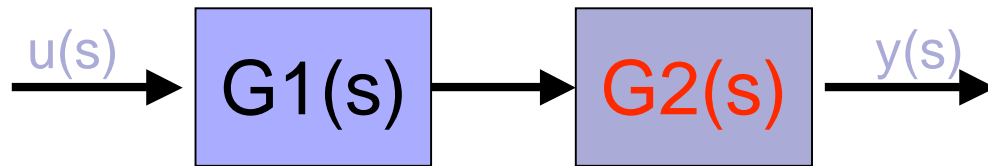
selected\_point =

0.0073 + 2.1804i

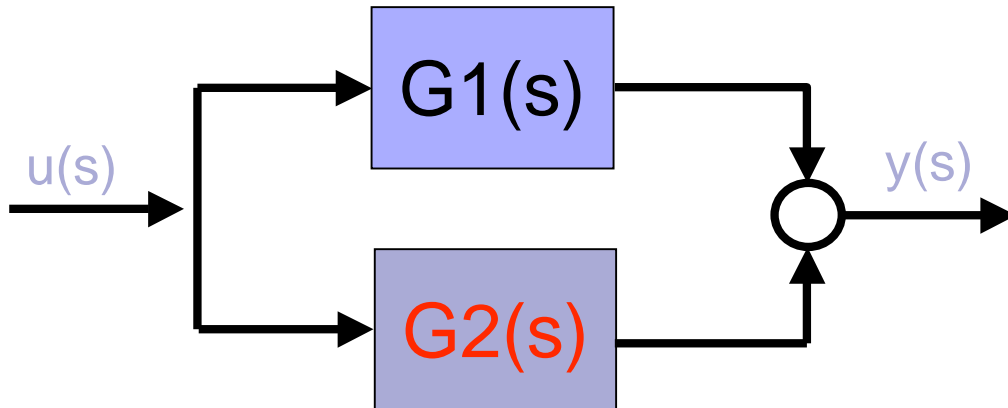
ans =

0.0231

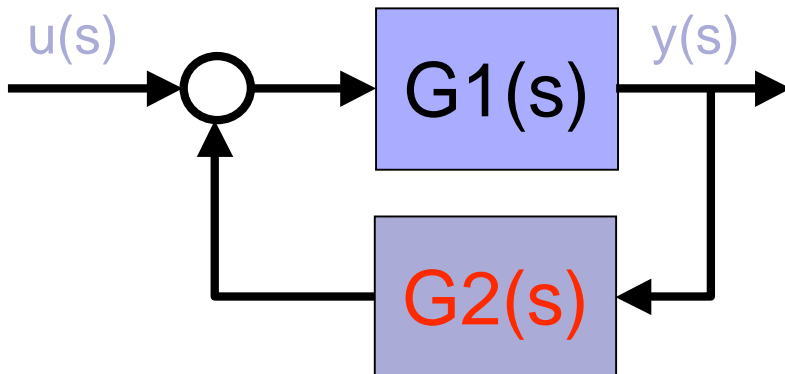
## Interconnessione di sistemi



**Serie:**  
 **$G_{\text{series}} = G1 * G2$**



**Parallelo:**  
 **$G_{\text{par}} = G1 + G2$**

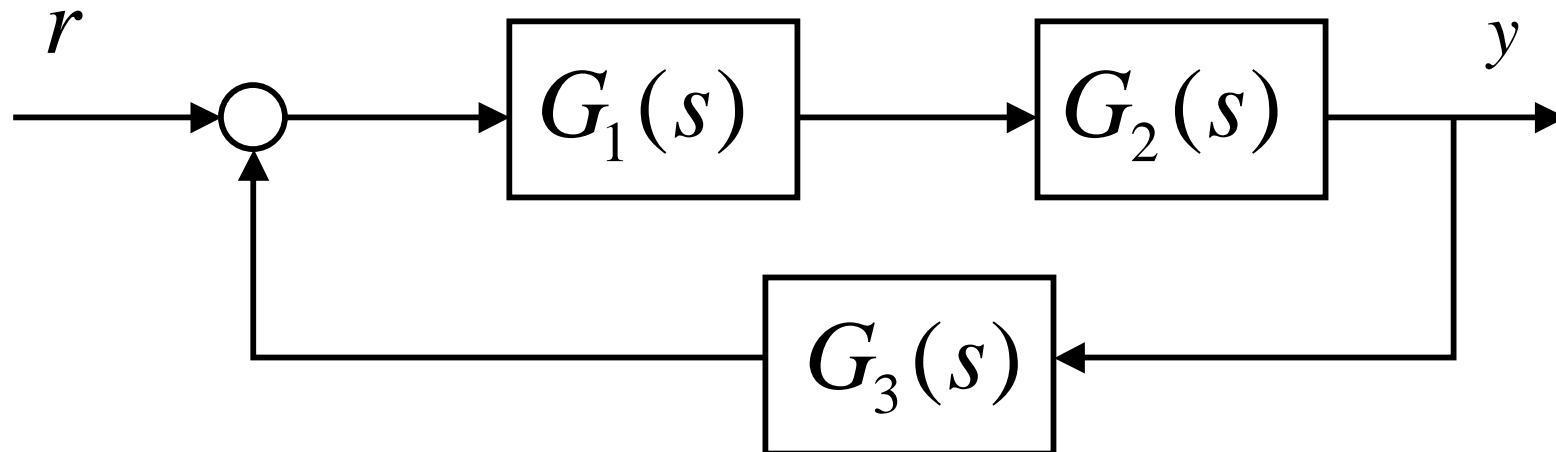


**Feedback:**  
 **$G_{\text{tot}} = \text{feedback}(G1, G2)$**

## Esempio di interconnessione

- Calcolare la f.d.t.  $y(s)/r(s)$

$$G_1(s) = \frac{1}{s+1}; \quad G_2(s) = \frac{4}{s^2 + 0.8s + 4}; \quad G_3(s) = \frac{2}{s+2};$$





## Esempio di interconnessione...

```
>> s=tf('s');
>> G1=1/(s+1); G2=4/(s^2+0.8*s+4);
G3=2/(s+2);
>> fdt_dir=series(G1,G2);
>> Gtot=feedback(fdt_dir,G3)
```

Transfer function:

$$4 s + 8$$

-----

$$s^4 + 3.8 s^3 + 8.4 s^2 + 13.6 s + 16$$

# Esempio di interconnessione

- $\text{zpk}(\text{G}_{\text{tot}})$

$$4 (s+2)$$

$$\frac{4 (s+2)}{(s^2 + 3.707s + 4.5) (s^2 + 0.09294s + 3.556)}$$

- Oppure:

$$\text{G}_{\text{tot1}} = \text{zpk}(\text{G1} * \text{G2} / (1 + \text{G1} * \text{G2} * \text{G3}))$$

$$4 (s+2) (s+1) (s^2 + 0.8s + 4)$$

$$\frac{4 (s+2) (s+1) (s^2 + 0.8s + 4)}{(s+1) (s^2 + 3.707s + 4.5) (s^2 + 0.09294s + 3.556) (s^2 + 0.8s + 4)}$$

**ATTENZIONE: Per cancellare poli/zeri utilizzare minreal:**

$$\text{G}_{\text{tot1}} = \text{zpk}(\text{minreal}(\text{G1} * \text{G2} / (1 + \text{G1} * \text{G2} * \text{G3})))$$

$$4 (s+2)$$

$$\frac{4 (s+2)}{(s^2 + 3.707s + 4.5) (s^2 + 0.09294s + 3.556)}$$

# Sistemi con ritardo finito

Per inserire un ritardo finito di 1.0s

- `s=tf('s');G=1/(s+1)`

- `get(G)`

num: {1}

den: {1}

Variable: 's'

Ts: 0

ioDelay: 0

InputDelay: 0

OutputDelay: 0

InputName: {''}

OutputName: {''}

InputGroup: {0x2 cell}

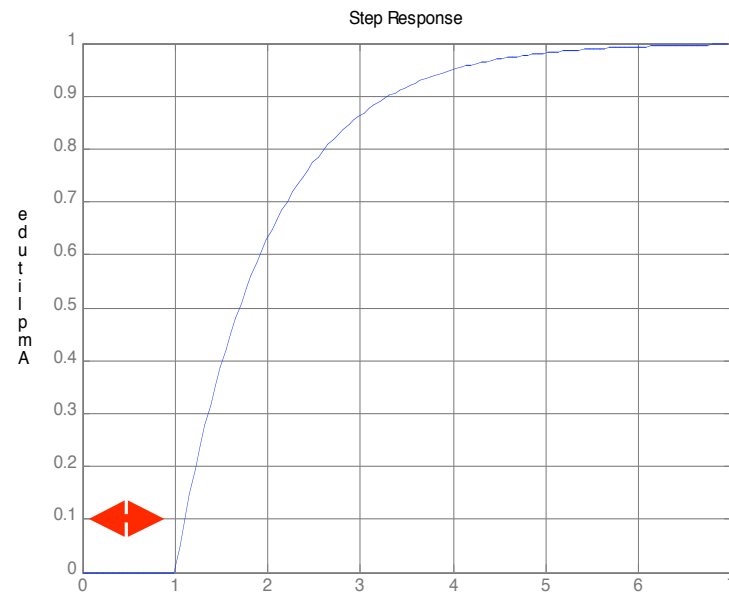
OutputGroup: {0x2 cell}

Notes: {}

UserData: []

- `set(G,'InputDelay',1);`

- `step(G)`





# Simulink

■ *v. file .mdl*



## Che cos'è?

- Simulink è un programma per la modellazione, la simulazione e l'analisi di sistemi dinamici descritti da equazioni differenziali, mediante funzioni di trasferimento, sistemi in forma di stato, ecc.
  - Permette la simulazione di sistemi complessi
    - di ordine elevato
    - con nonlinearità
    - segnali di riferimento complessi



## Come funziona?

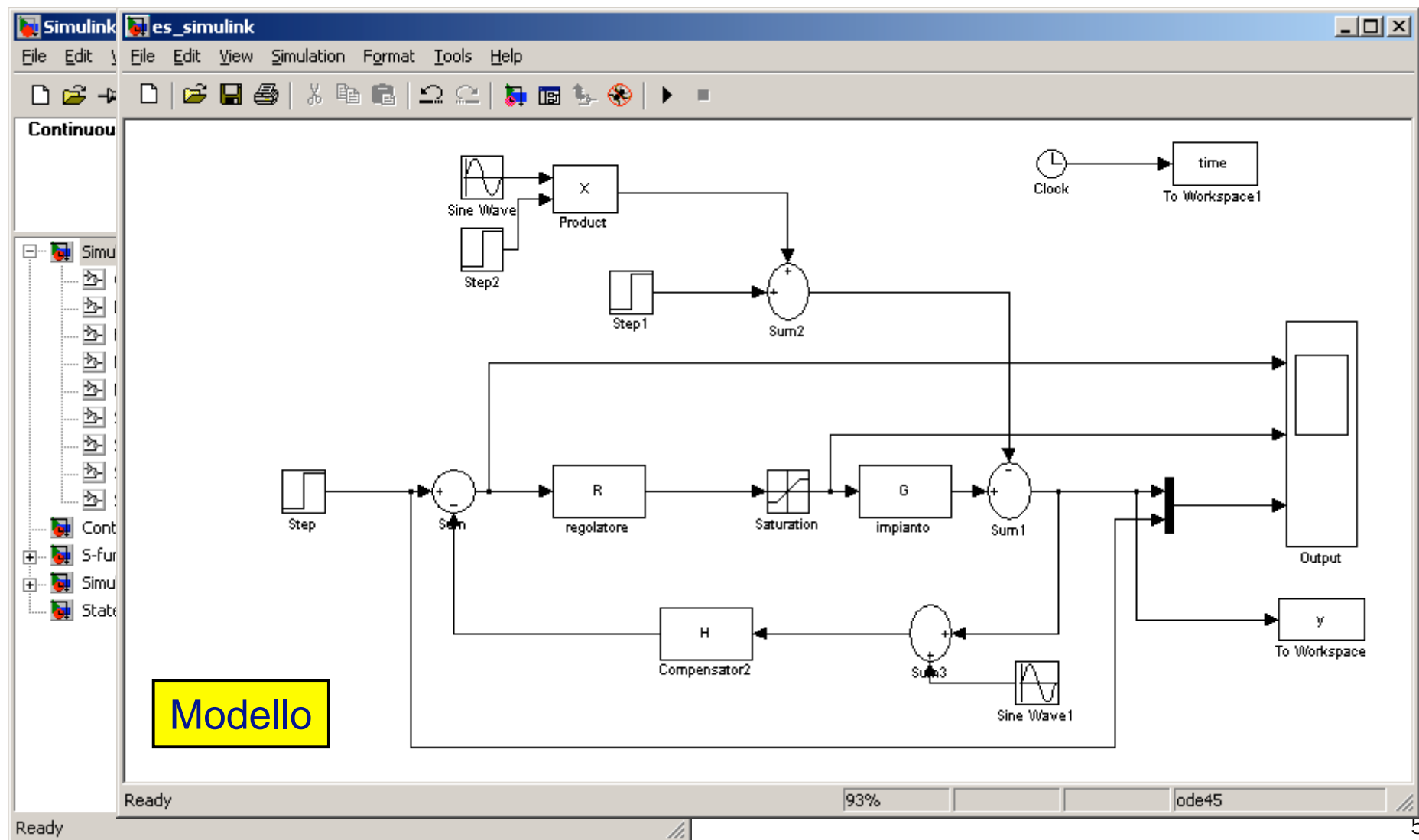
- La simulazione avviene in due passi:
  - Descrizione del sistema mediante schema a blocchi
    - Simulink contiene una libreria di blocchi che descrivono elementi algebrici e dinamici elementari
      - Funzioni di trasferimento, ecc
      - Guadagno, saturazioni, funzioni matematiche
    - L'utente compone lo schema a blocchi del sistema da simulare mediante l'interconnessione dei blocchi elementari
  - Simulazione del sistema
    - integrazione numerica delle equazioni differenziali, in un certo intervallo di tempo



## Interazione Matlab Simulink

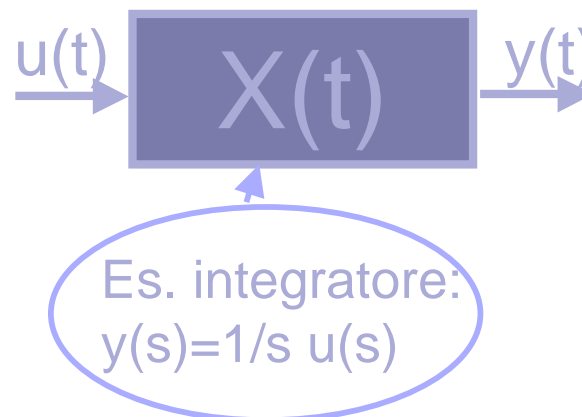
- Simulink interagisce con Matlab mediante il workspace: i modelli simulink possono contenere variabili del workspace.
- Allo stesso modo il risultato delle simulazioni può essere esportato nel workspace e analizzato con Matlab.
- Digitando 'simulink' dal prompt di Matlab si apre la libreria dei modelli.
- Da qui è possibile creare un nuovo modello (foglio bianco) e comporre il sistema da simulare mediante i diversi blocchi.

# Simulink



# Diagrammi a blocchi

- Interfaccia grafica:
- Un sistema è rappresentato come un **diagramma a blocchi**, cioè come blocchi interconnessi da linee
- Ogni blocco rappresenta un sistema elementare:
  - sistema dinamico o algebrico
  - È caratterizzato da ingressi e uscite (una o più), e eventualmente da stati interni (per i sistemi dinamici)
  - I blocchi possono essere parametrizzati (attraverso finestre)
    - Es. guadagno, poli/zeri di fdt, ecc.



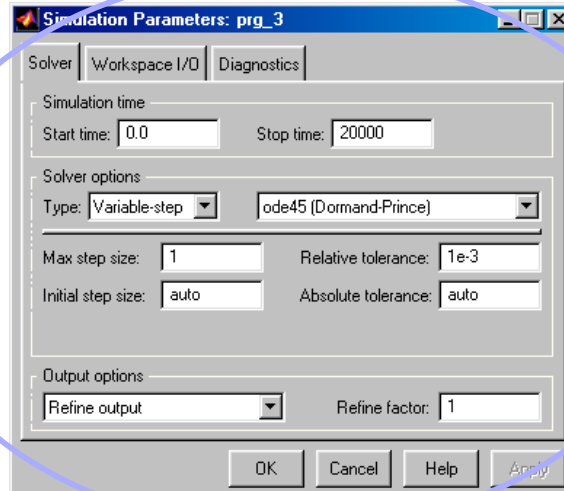
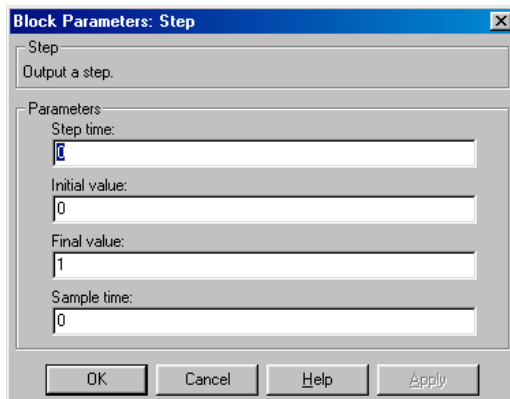
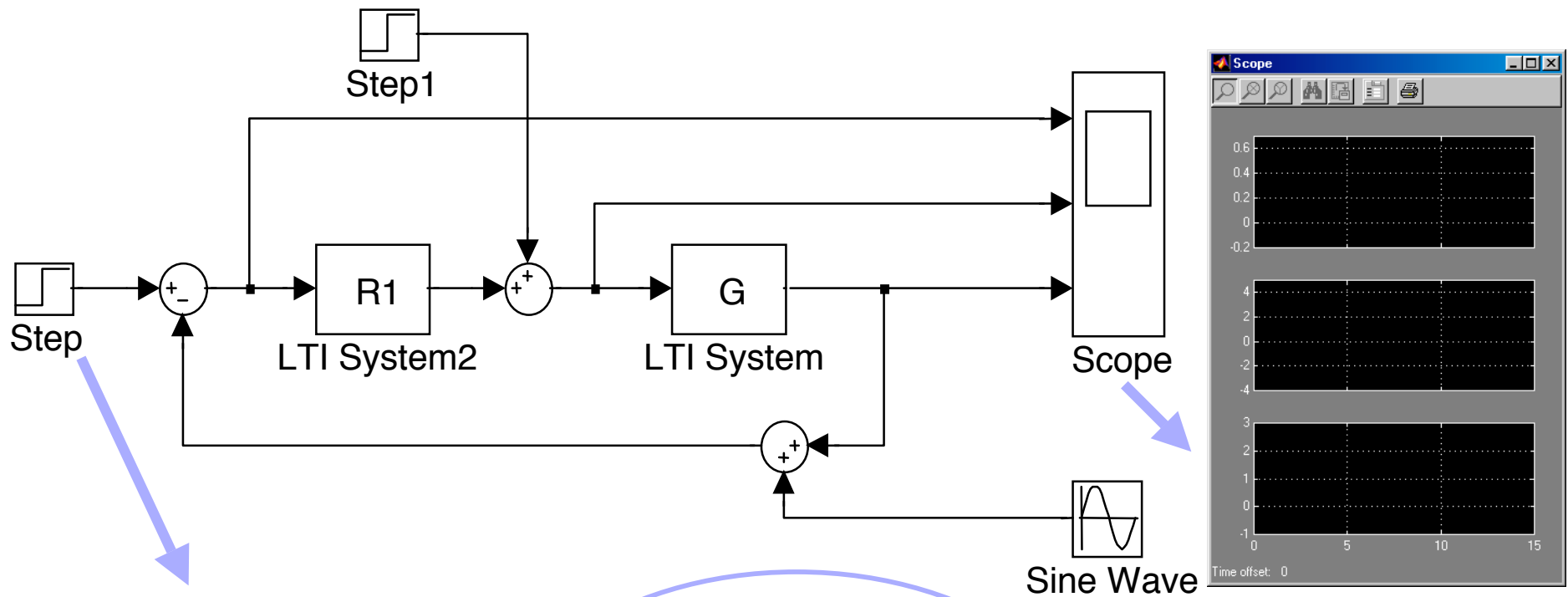
- Un segnale (scalare, vettoriale) è rappresentato da una linea che collega blocchi



# Simulink

- Possibilità di modellare in modo semplice sistemi di controllo, tenendo in considerazione:
  - generazione del riferimento (non solo gradini, rampe, ecc.)
  - disturbi presenti sul sistema
  - rumore di misura
  - nonidealità dell'impianto (es. saturazione dell'attuatore)
- Il progetto, utilizzando Matlab, si realizza in 2 passi:
  - progetto nel dominio frequenziale (es. diagrammi di Bode), con il luogo delle radici, ecc. (in Matlab)
  - verifica dei risultati in Simulink, nel dominio temporale. Possibilità di modellare effetti secondari, dinamiche non modellate, ecc.

# Modello Simulink





# Libreria Simulink – elenco di alcuni blocchi

## ■ Continuous

- ☐ Integrator
  - Possibilità di definire condizioni iniziali, reset, saturazioni
- ☐ State-space
- ☐ Transfer function
  - nella forma numeratore/denominatore
- ☐ transport delay
  - Ritardo puro
- ☐ zero-pole

## ■ Functions & tables

- ☐ fcn, Matlab Fcn
  - Funzioni matematiche varie

## ■ Math

- ☐ Abs
- ☐ Gain
  - Guadagno
- ☐ Logical operator, relational operator
- ☐ Math function, trigonometric function
- ☐ sum, product
- ☐ sign



# Libreria Simulink – elenco di alcuni blocchi

## ■ Nonlinear

- ☐ dead-zone, saturation
  - Varie funzioni nonlineari
- ☐ Manual switch

## ■ Signals & Systems

- ☐ mux, demux
  - Unione di più segnali scalari in un unico segnale vettoriale (e viceversa) – v. uso con “scope” e “to workspace”
- ☐ From, Goto

## ■ Sinks

- ☐ Display
- ☐ Scope
  - Visualizzazione grafica durante la simulazione (e possibilità di salvataggio nel workspace)
- ☐ to\_workspace
  - Salvataggio delle variabili nel workspace



# Libreria Simulink – elenco di alcuni blocchi

- Sources

- ☐ Clock
- ☐ Constant
- ☐ from workspace, pulse generator, repeating sequence
  - Per la generazione di segnali temporali “complessi”
- ☐ sine
- ☐ Step
  - gradino

- Control System Toolbox

- ☐ LTI System

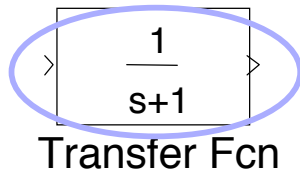
- V. Help di MATLAB/Simulink per maggiori dettagli



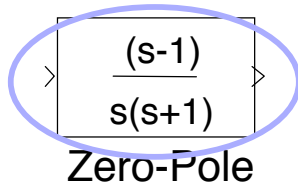
# Simulazione

- Definire in un file .m
  - il modello dell'impianto (mediante fdt)
  - altri parametri del modello (es. Ampiezza/frequenza del rumore di misura e dei disturbi, riferimenti, saturazioni, ecc.)
  - Il regolatore (mediante progetto frequenziale, ecc.)
- Creare il modello Simulink .mdl
  - Impostare i parametri del solver della simulazione
- Eseguire la simulazione ("run")
- Visualizzare i grafici
  - Direttamente in Simulink mediante Scope
  - In Matlab, importando i dati da Simulink o mediante blocchi "To Workspace", o mediante blocchi "Scope"
  - Generare i grafici mediante un file .m (per automatizzare la procedura)
- Stampare e esportare i grafici in Word/Powerpoint

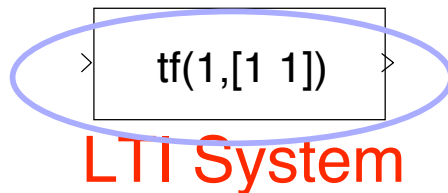
# Inserimento di una f.d.t. in Simulink



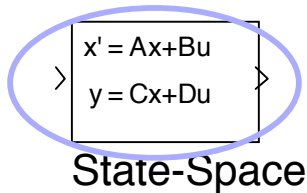
Specificando numeratore e denominatore



Specificando poli, zeri e guadagno

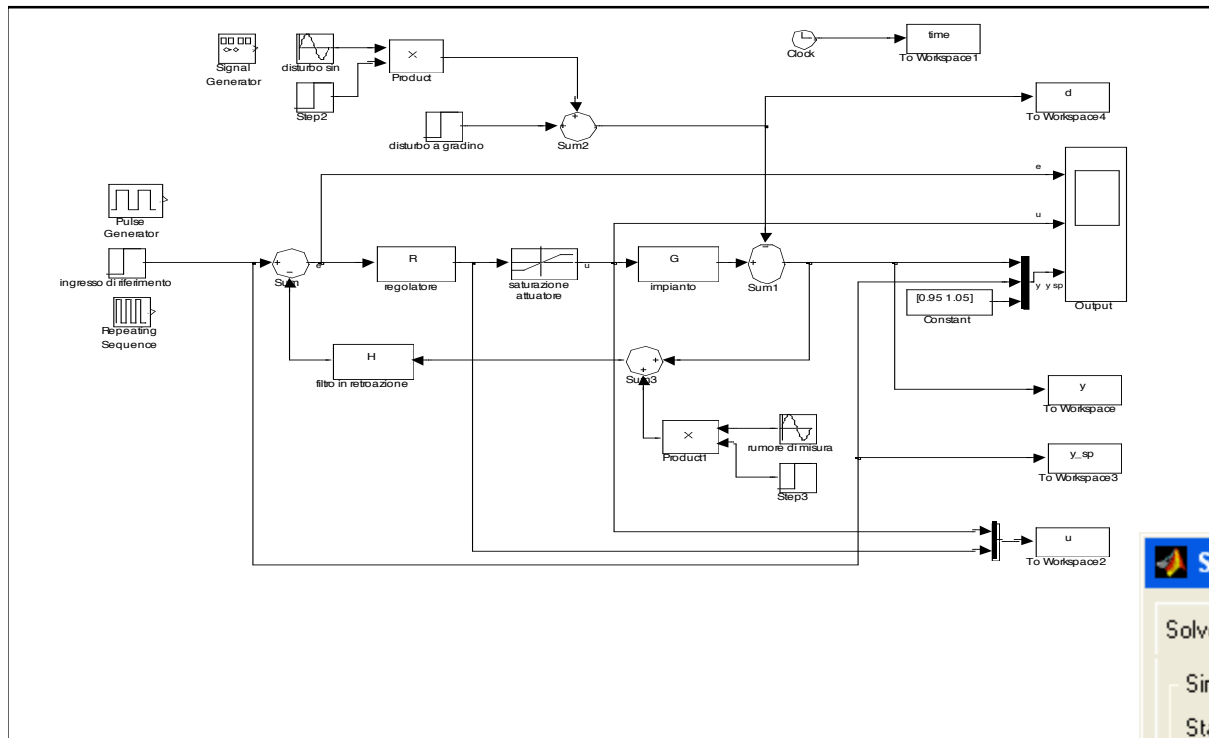


Come in CST



In forma di stato

# Come impostare i parametri della simulazione



**Simulation Parameters: esempio\_simulink**

Solver | Workspace I/O | Diagnostics | Advanced


Simulation time  
Start time: 0.0 Stop time: 30

Solver options  
Type: Variable-step ode45 (Dormand-Prince)

Max step size: 1e-2 Relative tolerance: 1e-3  
Min step size: auto Absolute tolerance: auto  
Initial step size: auto

Output options  
Refine output Refine factor: 1

OK Cancel Help Apply

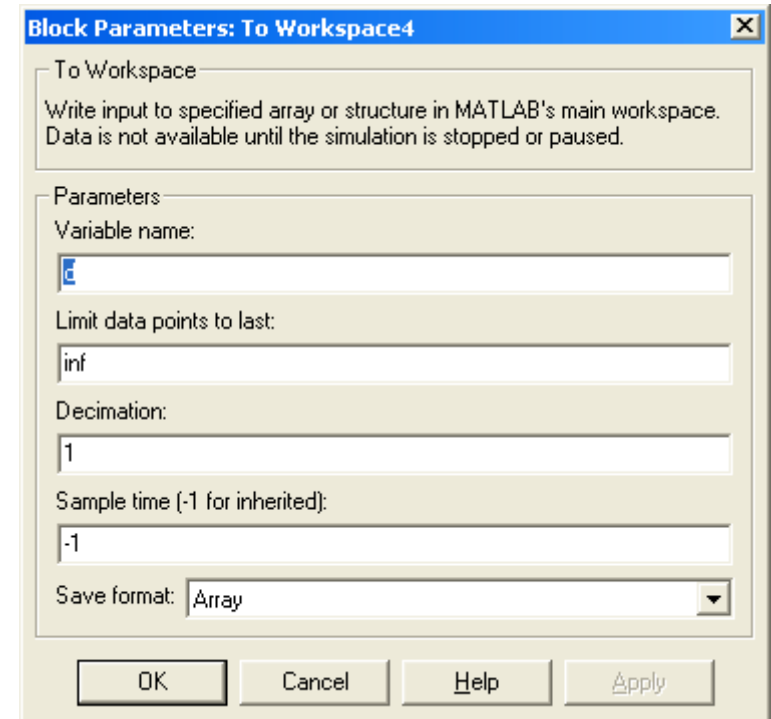


## Come impostare i parametri della simulazione (vedi Simulation, Simulation parameters...)

- Istante iniziale e finale della simulazione
- Tipo della tecnica di integrazione numerica
  - (es. variable step, ODE45)
- Max step size (massimo intervallo fra un istante di calcolo della soluzione dell'ODE e il successivo)
  - DEVE essere scelto:
    - Minore della costante di tempo più veloce del sistema
    - Minore del periodo del segnale periodico più veloce che agisce sul sistema (es. 1/10 più piccolo)
    - In modalità AUTO, si hanno simulazioni poco precise
- Min step size (minimo intervallo fra un istante di calcolo della soluzione dell'ODE e il successivo)
  - Può essere utilizzato per ridurre il tempo richiesto per la simulazione (a discapito della precisione)
- Relative/absolute tolerance
  - Accuratezza del risultato ottenuto con la simulazione
  - Impostare "auto" o a valori sufficientemente piccoli (es. Relative tolerance 1E-4)

# Esportazione di dati da Simulink a Matlab

- Con il blocco “to Workspace”
  - Scegliere il nome della variabile
  - Formato di salvataggio (save format):
    - Array – variabile in forma vettoriale
    - Struct with time:
      - Definita “Y” la variabile in cui vengono salvati i dati, si accede ai valori salvati con:
        - Y.time  
– tempo
        - Y.signals.values  
- variabili salvate
  - Utilizzare i comandi standard di Matlab per generare i grafici
    - Es. Plot(Y.time, Y.signals.values)



# Esportazione di dati da Simulink a Matlab

## ■ Con il blocco “Scope”

### □ Properties

#### ■ Impostare il numero di assi

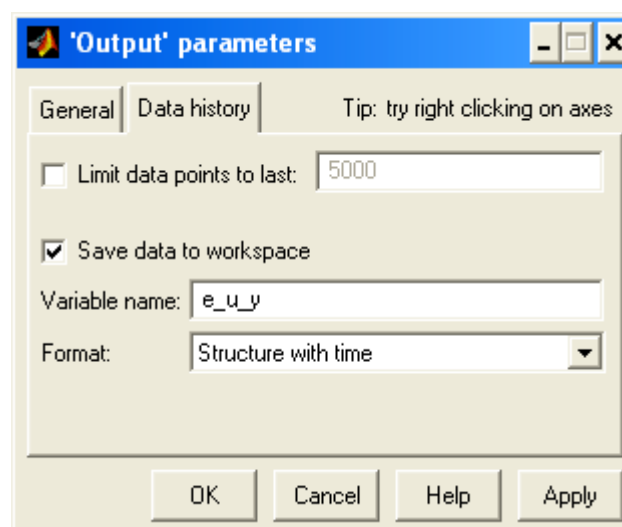
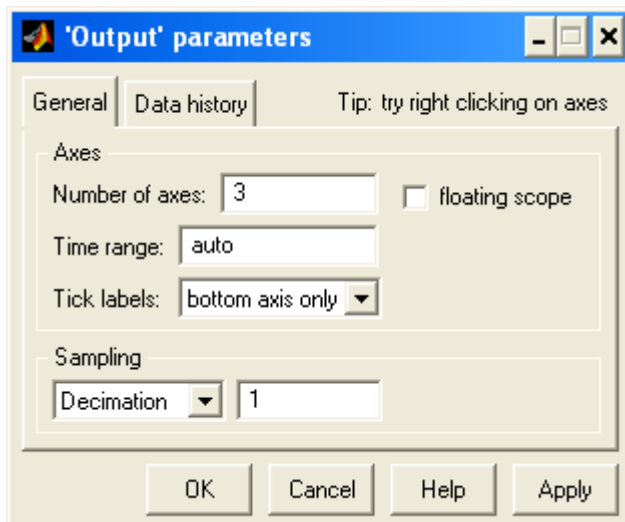
#### ■ Data History

### □ NON selezionare “Limit data point to last”

- Altrimenti viene salvato solo un numero limitato di dati della simulazione

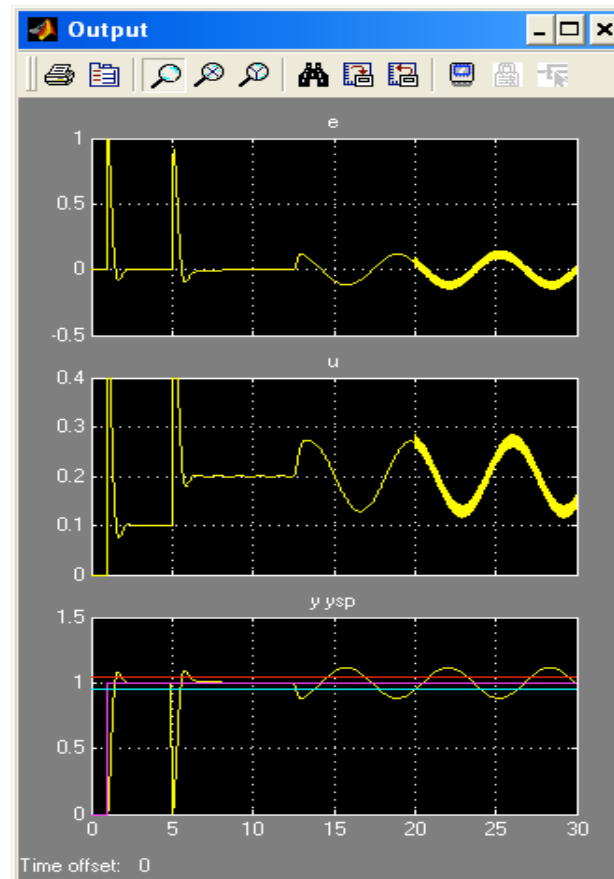
### □ Save Data to workspace

- salvataggio delle variabili, come con “to workspace”. Scegliere “Struct with time” e il nome della variabile (es. Y)
- Se si utilizzano più assi, i dati visualizzati nell’asse k-esimo sono contenuti in Y.signals(k).values



## Note sull'esportazione di dati da Simulink a Matlab

- In Matlab, utilizzare i comandi per la rappresentazione delle variabili `Y.signals(k).values` (es. `plot`)
- Oppure utilizzare il comando `"simplot(Y)"` per generare in Matlab la stessa figura ottenuta nello Scope
  - L'aspetto della figura (es. i colori) è modificabile (v. Edit Figure)



# Esportare i grafici

## ■ Esportazione in Word/Powerpoint

- In Matlab, “Edit, Copy Figure” per copiare la figura negli appunti di Windows
- In Matlab, “Edit, Copy Options” per modificare le impostazioni
  - Scegliere “Metafile” come “clipboard format”
- In Word/Powerpoint, incollare con “Modifica, Incolla speciale” e selezionare “Enhanced Metafile”

