



# Tecniche di controllo multivariabile

## Introduzione a Matlab e Simulink (con Control Systems Toolbox)

# Matlab e Simulink

## Matlab (Matrix Laboratory)

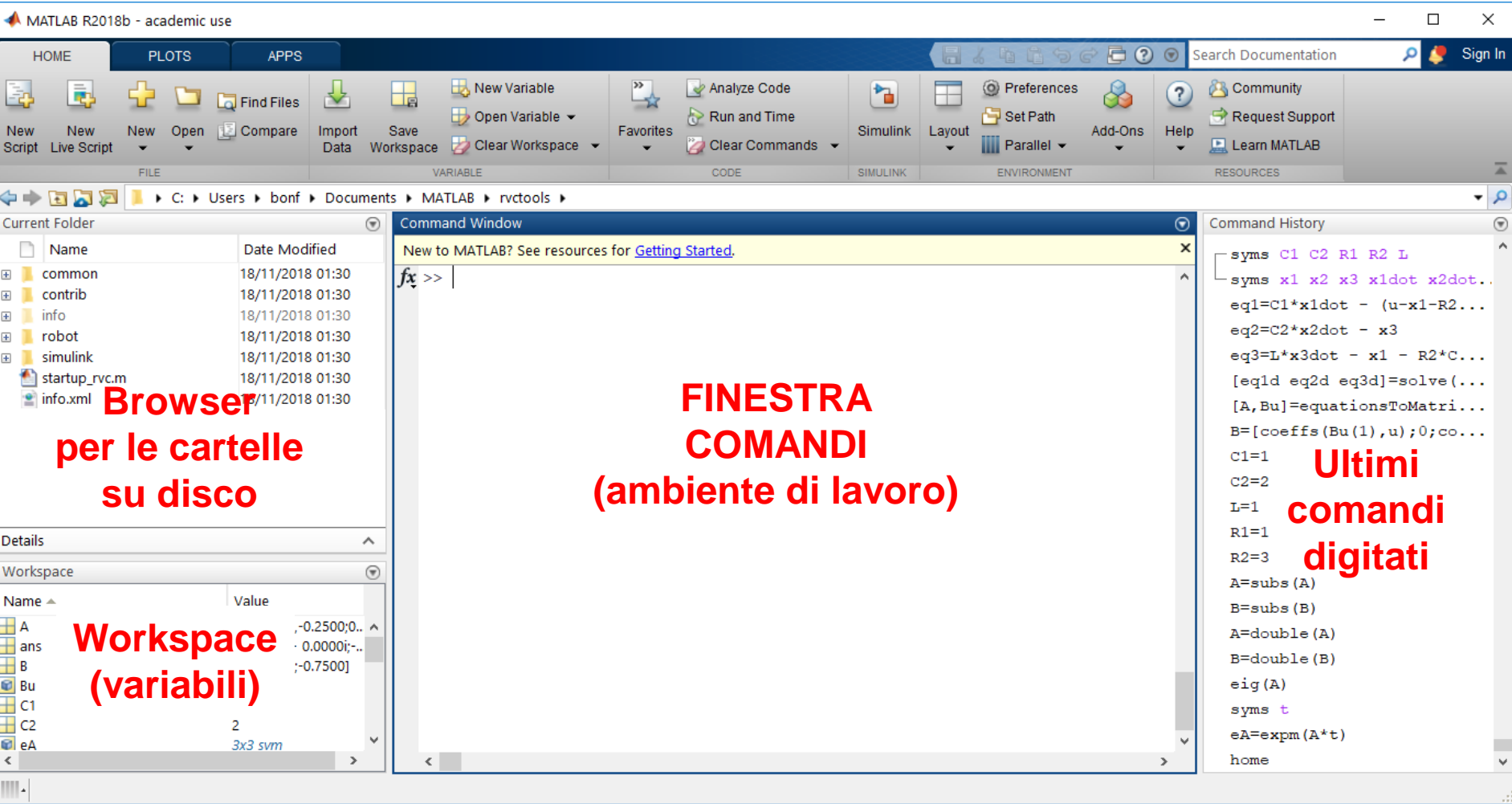
- Ambiente di sviluppo per il calcolo numerico
- Comprende l'omonimo linguaggio di programmazione con interprete dei comandi
- Include svariati toolbox (collezioni tematiche di funzioni)
- Efficace per analisi, elaborazione e visualizzazione dati

## Simulink

- Software per la modellazione e simulazione di sistemi dinamici
- Integrazione e interazione con Matlab
- Programmazione grafica basata su schemi a blocchi

Ampiamente utilizzati nel mondo della ricerca e dell'industria

# Matlab: interfaccia principale (R2018b)



**Browser  
per le cartelle  
su disco**

**FINESTRA  
COMANDI  
(ambiente di lavoro)**

**Workspace  
(variabili)**

**Ultimi  
comandi  
digitati**

# Matlab: definizione di variabili, vettori e matrici

Definire variabile scalare

```
>> x = 3
```

Definire vettore riga ( $1 \times 3$ )

```
>> x = [1 2 3]
```

Idem, ma senza echo dell'output

```
>> x = [1 2 3];
```

Definire vettore colonna ( $3 \times 1$ )

```
>> x = [1; 2; 3]
```

(oppure 

```
>> x = [1 2 3]'
```

)

Definire matrice  $3 \times 4$

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

Accedere / modificare elemento di riga 2 e colonna 1

```
>> A(2,1) = 0
```

# Matlab: operazioni su matrici

- Le "solite" operazioni matematiche:  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$
- **Es.** `>> A^3` (potenza di matrice, solo se quadrata!)
- Precedute dal punto, sono eseguite elemento per elemento anziché in senso matriciale/vettoriale
- Operazioni specifiche per matrici / vettori:
  - Trasposta: `A'`
  - Determinante: `det(A)`
  - Inversa: `inv(A)`
  - Autovalori: `eig(A)`
  - Rango: `rank(A)`
  - Polinomio caratteristico: `poly(A)`
  - Esponenziale di matrice: `expm(A)`
  - Radici di un polinomio: `roots(x)` (x vettore dei coeff.)

# Matlab: inizializzazione di matrici *standard*

- Comandi che forniscono matrici caratteristiche, utili per inizializzare variabili opportune:
  - Matrice  $m \times n$  con tutti elementi nulli: **zeros** ( $m, n$ )  
**NOTA:** **zeros** ( $m$ ) fornisce matrice quadrata
  - Matrice  $m \times n$  con tutti elementi unitari: **ones** ( $m, n$ )  
**NOTA:** **ones** ( $m$ ) fornisce matrice quadrata
  - Identità  $n \times n$ : **eye** ( $n$ )
  - Matrice quadrata diagonale (con elementi sulla diagonale nel vettore  $V$ ): **diag** ( $V$ )

# Matlab: il workspace

- I risultati di tutti i comandi digitati vengono memorizzati nel cosiddetto workspace della sessione
- Il workspace viene cancellato all'uscita dal Matlab!
- Il contenuto del workspace si può salvare (anche parzialmente) e ripristinare:
  - `save nomefile` (estensione di default: `.mat`)
  - `save nomefile variabile1 variabile2` (salva solo le variabili indicate)
  - `load nomefile`
  - `clear`: cancella il contenuto del workspace!!

# Matlab: script file .m

- File testuale contenente una successione di comandi modificabile tramite l'editor window
- L'esecuzione dello script (pulsante play) consiste nell'esecuzione di un comando alla volta

## Cicli

```
for i=1:10
    ...
end

while i<10
    ...
end
```

## Condizioni

```
if i<1
    ...
elseif i<20
    ...
else
    ...
end
```



# Matlab: visualizzazione dati

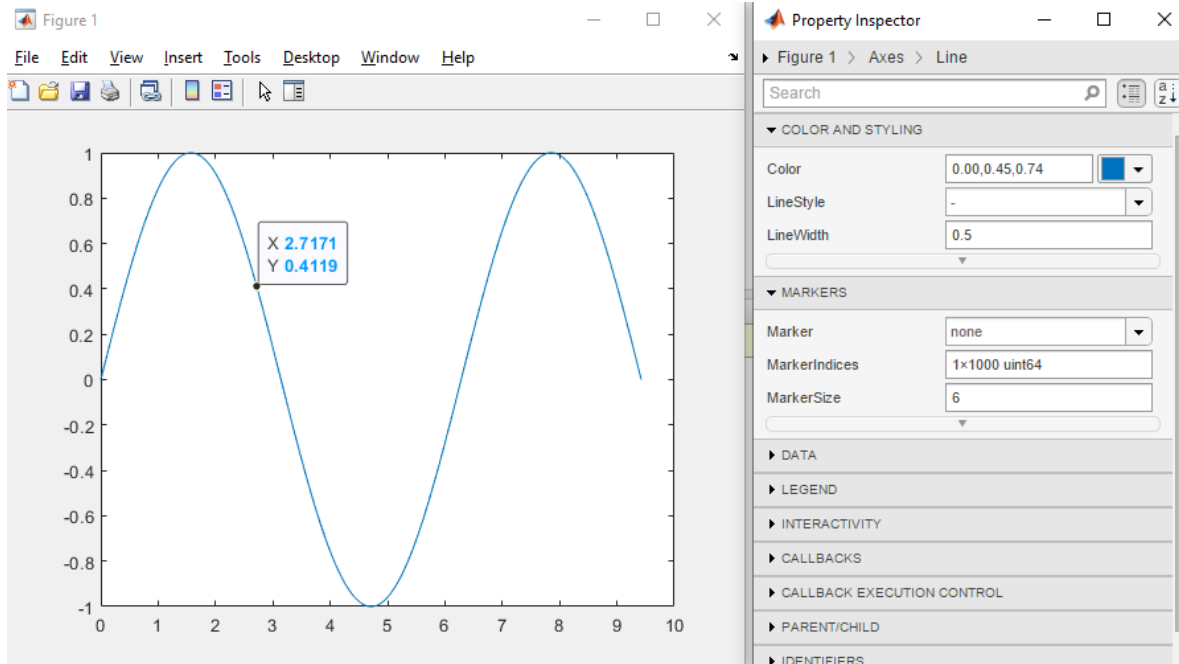
## Plot, figure e property inspector

```
>> t=0:0.001:3*pi
```

```
>> plot(t, sin(t))
```

Vettore ascisse

Vettore ordinate



# Esempio: sistema LTI tempo discreto

- Creare uno script matlab che visualizzi il moto e la risposta del seguente sistema in k passi

$$\begin{cases} x(i+1) = Ax(i) + Bu(i) \\ y(i) = Cx(i) + Du(i) \end{cases}$$

Con:

$$A = \begin{bmatrix} 0.5 & 0 \\ 3 & 0.1 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, C = [1 \ 0], D = [0 \ 1], u \text{ costante}$$

# Esempio: script

```
%% System
A = [0.5 0;3 0.1];
B = [1 1;1 1];
C = [1 0];
D = [0 1];
k = 10; % number of steps
```

```
%% input
```

```
u = zeros(2,k); % input
u(1,:) = 2*ones(1,k);
u(2,:) = 1*ones(1,k);
```

```
%% init state and output
```

```
x = zeros(2,k);
y = zeros(1,k);
```

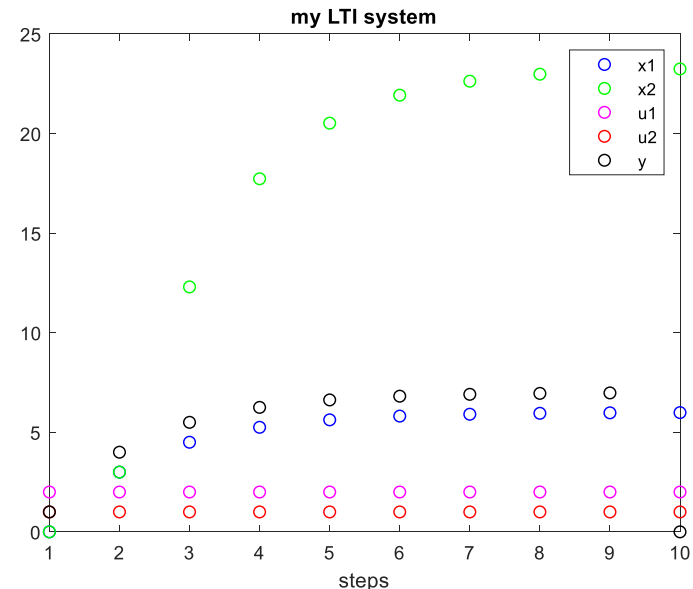
```
%% compute state and output
```

```
for i=1:k-1
    x(:,i+1) = A*x(:,i) + B*u(:,i);
    y(i) = C*x(:,i) + D*u(:,i);
```

```
end
```

```
%% plot
```

```
plot(x(1,:), 'bo')
hold on
plot(x(2,:), 'go')
plot(u(1,:), 'mo')
plot(u(2,:), 'ro')
plot(y, 'ko')
title('my LTI system')
legend('x1', 'x2', 'u1', 'u2', 'y')
xlabel('steps')
```



# Matlab: Control System Toolbox

## ► Calcolo delle matrici di raggiungibilità e osservabilità

```
>> A = [10 1 ; -1 1];
```

```
>> B = [1; 1];
```

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

```
>> P = ctrb(A,B)      →
```

**Nota:** in Matlab è detta matrice di controllabilità, equivale a  $P = [B \ A*B]$

```
P =
```

```
    1    11  
    1     0
```

```
>> Q = obsv(A,C)      →
```

**Nota:** equivale a  $Q = [C ; C*A]$

```
Q =
```

```
    0     1  
   -1     1
```

# Matlab: Control System Toolbox

- La funzione `sys=ss (A,B,C,D)` crea l'oggetto rappresentativo del modello nello spazio degli stati a partire dalle matrici A,B,C,D

```
>> sys = ss([-2 0;-1 -11],[1;0],[0 1],2)
```

```
sys =
```

```
A =
```

	x1	x2
x1	-2	0
x2	-1	-11

```
B =
```

	u1
x1	1
x2	0

```
C =
```

	x1	x2
y1	0	1

```
D =
```

	u1
y1	2

```
Continuous-time state-space  
model.
```

# Matlab: Control System Toolbox

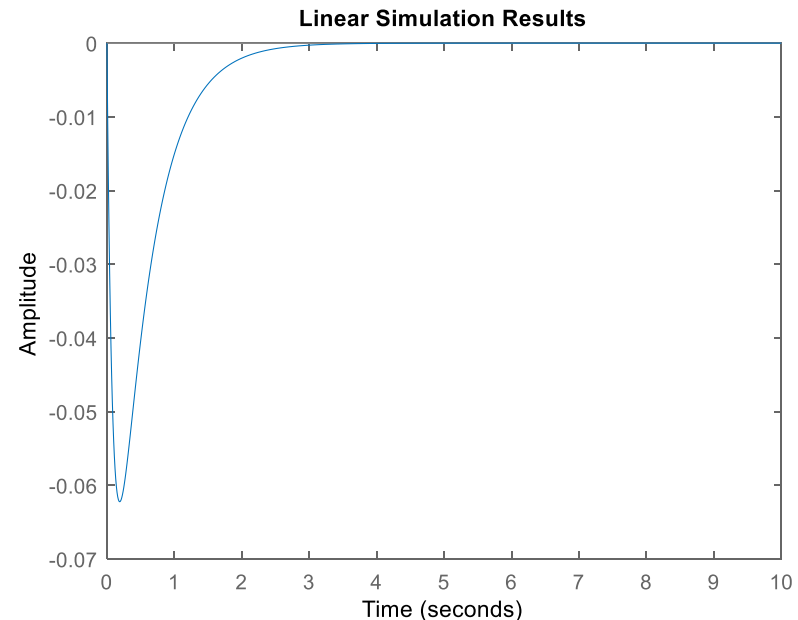
- La funzione `y=lsim(sys,u,t,x0)` simula l'andamento nel tempo del sistema a partire dalle condizioni iniziali e ne restituisce l'uscita

```
>> t=[0:0.01:10];  
>> u=zeros(size(t));  
>> x0 = [1 0];  
>> y=lsim(sys,u,t,x0);
```

**NOTA:** se chiamata così

```
>> lsim(sys,u,t,x0)
```

viene aperta la finestra di analisi grafica per sistemi LTI → → → → →



# Matlab: Control System Toolbox

- La funzione  $y = \text{step}(\text{sys}, t)$  simula l'andamento nel tempo del sistema supponendo che l'ingresso sia un gradino unitario

```
>> t=[0:0.01:10];
```

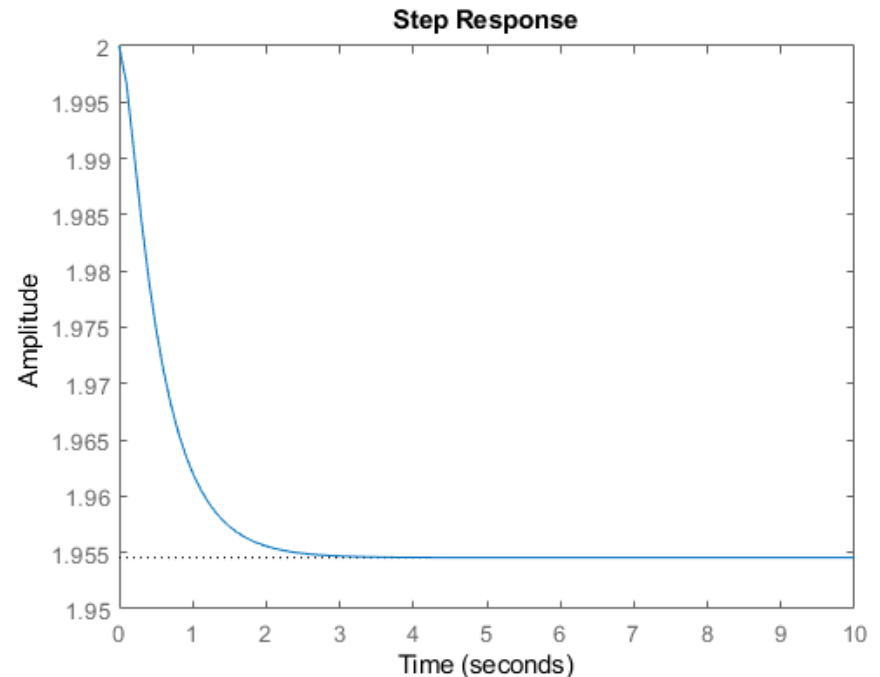
```
>> y=step(sys, t);
```

**NOTA:** se chiamata così

```
>> step(sys, t)
```

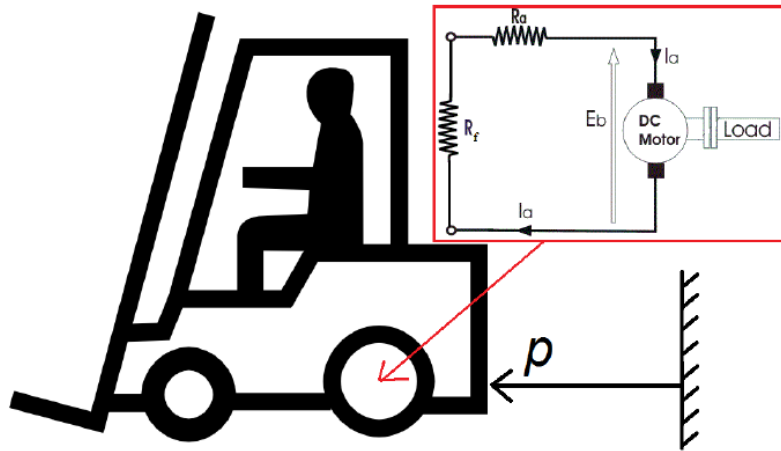
viene aperta la finestra di analisi

grafica per sistemi LTI → → → → →



# Esercizio

- ➡ Modello di un carrello elevatore a trazione elettrica in modalità di frenatura



$$m\ddot{p} + b\dot{p} + \frac{k_m^2}{R_a + R_f}\dot{p} = 0$$

$$x_1 = p; x_2 = \dot{p};$$

$$m = 1000; b = 100; R_a = 10; R_f = 90; k_m = 300;$$

Si determini lo spazio percorso e la velocità raggiunta in 12 secondi dal veicolo (i.e.  $x(t)$  con  $t=12$ ) in modalità di frenata, considerando una velocità iniziale di 5m/s, vale a dire:

$$x(0) = [0 \quad 5]^T$$



# Soluzione esercizio

```
%% parameters
```

```
m = 1000;
```

```
b = 100;
```

```
Ra = 10;
```

```
Rf = 90;
```

```
Km = 300;
```

```
%% system
```

```
A = [0, 1;
```

```
0, -(Km^2 + Ra*b +  
Rf*b) / (m*(Ra + Rf))];
```

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

```
% velocity as output
```

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

```
D = 0;
```

```
sys = ss(A,B,C,D);
```

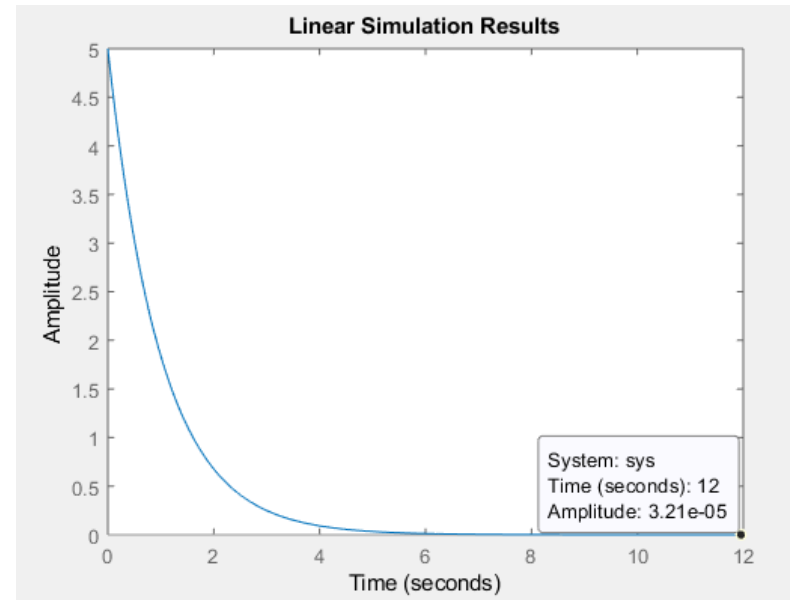
```
t = linspace(0,12,300);
```

```
u = zeros(size(t)); % no input
```

```
x0 = [0; 5];
```

```
%% simulate LTI system
```

```
lsim(sys,u,t,x0)
```



# Soluzione alternativa esercizio

➡ Risolvere l'esercizio in forma numerica

```
>> x0 = [0;5];
```

```
>> tf = 12;
```

```
x_12 = expm(A*tf)*x0
```

```
x_12 =
```

```
5.0000
```

```
0.0000
```

**NOTA:** il risultato è arrotondato, di default Matlab mostra solo 4 cifre dopo il punto decimale. Per mostrare più cifre:

```
>> format long
```

```
>> x_12
```

```
x_12 =
```

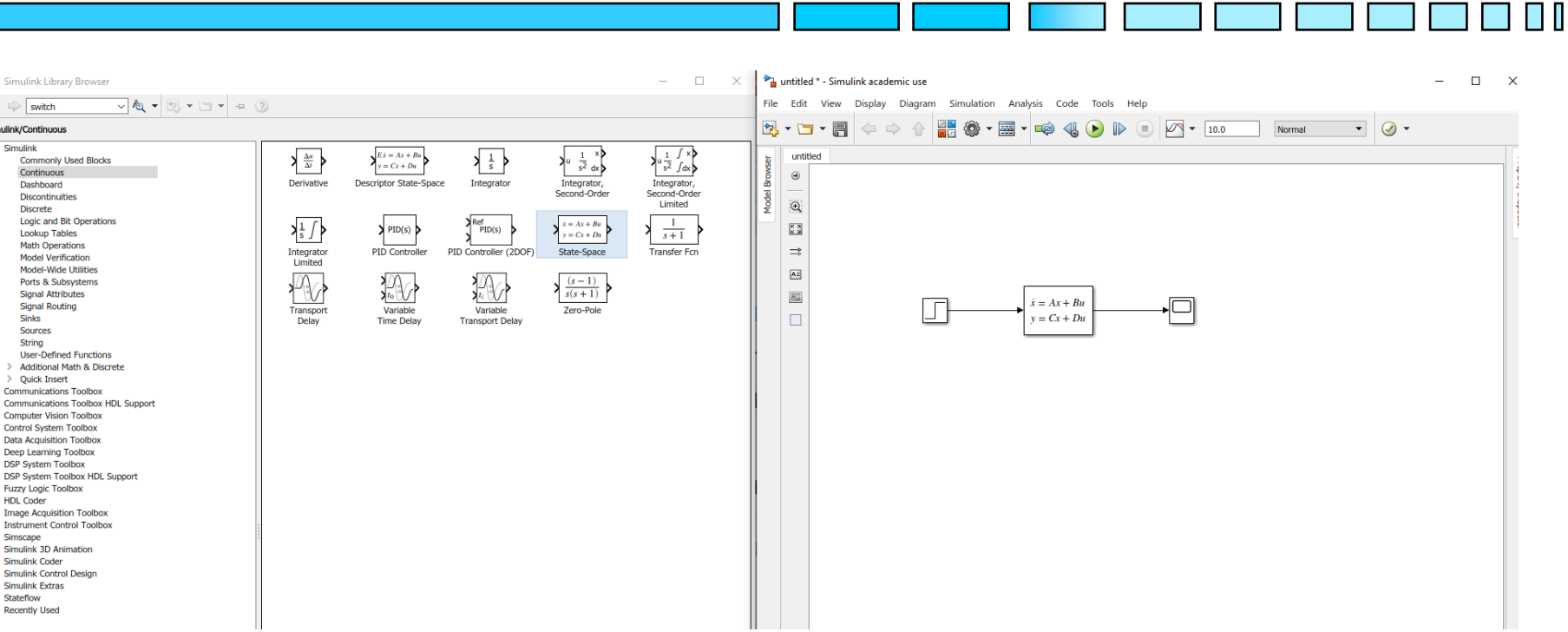
```
4.999969278938233
```

```
0.000030721061767
```

La simulazione di un sistema dinamico con Simulink consiste in due fasi fondamentali:

- **Modellistica:** Creazione di un modello grafico del sistema da simulare, inserendo e connettendo blocchi rappresentativi di sistemi multivariabile tempo continuo, tempo discreto, lineari e non lineari
- **Simulazione** del comportamento del sistema durante una sua evoluzione temporale su un arco di tempo definito. Simulink in questa fase utilizza le informazioni contenute nel modello grafico per generare le equazioni dinamiche ad esso associate e risolverle numericamente

# Simulink: interfaccia principale



Library browser



Model editor

# Simulink: blocchi principali

## Continuous:

$\frac{\Delta u}{\Delta t}$  Derivative  
 $\begin{cases} E\dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$  Descriptor State-Space  
 $\frac{1}{s}$  Integrator  
 $u \frac{1}{s^2} x$  Integrator, Second-Order  
 $u \frac{1}{s^2} \int dx$  Integrator, Second-Order Limited  
 $\frac{1}{s} \int$  Integrator Limited  
 PID(s) PID Controller  
 $\text{Ref PID}(s)$  PID Controller (2DOF)  
 $\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$  State-Space  
 $\frac{1}{s+1}$  Transfer Fcn  
 Transport Delay  
 Variable Time Delay  
 Variable Transport Delay  
 $\frac{(s-1)}{s(s+1)}$  Zero-Pole

## Discrete:

$z^{-2}$  Delay  
 $\frac{z-1}{z}$  Difference  
 $\frac{K(z-1)}{Ts z}$  Discrete Derivative  
 $\frac{1}{1+0.5z^{-1}}$  Discrete Filter  
 $\frac{0.5+0.5z^{-1}}{1}$  Discrete FIR Filter  
 PID(z) Discrete PID Controller  
 $\text{Ref PID}(z)$  Discrete PID Controller (2DOF)  
 $\begin{cases} x_{n+1} = Ax_n + Bu_n \\ y_n = Cx_n + Du_n \end{cases}$  Discrete State-Space  
 $\frac{K Ts}{z-1}$  Discrete-Time Integrator  
 $\frac{1}{z+0.5}$  Discrete Transfer Fcn  
 $\frac{(z-1)}{z(z-0.5)}$  Discrete Zero-Pole  
 $u \frac{1}{z^{-2}}$  Enabled Delay  
 First-Order Hold  
 Memory  
 $\frac{u}{s} z^{-1}$  Resettable Delay  
 Tapped Delay  
 $\frac{0.05z}{z-0.95}$  Transfer Fcn First Order  
 $\frac{z-0.75}{z-0.95}$  Transfer Fcn Lead or Lag  
 $\frac{z-0.75}{z}$  Transfer Fcn Real Zero  
 $\frac{1}{z}$  Unit Delay  
 $u \frac{1}{d} z^{-d}$  Variable Integer Delay  
 Zero-Order Hold

# Simulink: blocchi principali

## ➡ Math:

A grid of Simulink Math blocks. Red boxes highlight the following blocks:

- Abs
- Add
- Gain
- Product
- Sum
- Trigonometric Function

Other visible blocks include: Algebraic Constraint, Assignment, Bias, Complex to Magnitude-Angle, Complex to Real-Imag, Divide, Dot Product, Find Nonzero Elements, Magnitude-Angle to Complex, Math Function, Matrix Concatenate, MinMax, MinMax Running Resettable, Permute Dimensions, Polynomial, Product of Elements, Real-Imag to Complex, Reciprocal Sqrt, Reshape, Rounding Function, Sign, Signed Sqrt, Sine Wave Function, Slider Gain, Sqrt, Squeeze, Subtract, Sum of Elements, Unary Minus, and Vector Concatenate.

## ➡ Sinks:

A row of Simulink Sink blocks. Red boxes highlight the following blocks:

- Out1
- Scope
- To Workspace

Other visible blocks include: Display, Floating Scope, Out Bus Element, Stop Simulation, Terminator, To File, and XY Graph.

# Simulink: blocchi principali

## Sources:

Grid of Simulink source blocks:

- Band-Limited White Noise
- Chirp Signal
- Clock
- Constant**
- Counter Free-Running
- Counter Limited
- Digital Clock
- Enumerated Constant
- From File
- From Spreadsheet
- From Workspace**
- Ground
- In Bus Element
- In1**
- Pulse Generator
- Ramp**
- Repeating Sequence
- Repeating Sequence Interpolated
- Repeating Sequence Stair
- Signal Builder
- Scenario Signal Editor
- Signal Generator
- Sine Wave**
- Step**
- Uniform Random Number
- Waveform Generator

## Signal routing:

Grid of Simulink signal routing blocks:

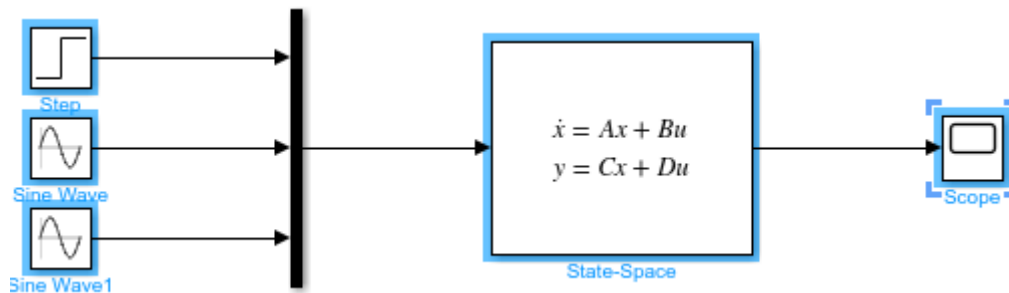
- Bus Element In
- Bus Element Out
- Bus Assignment
- Bus Creator
- Bus Selector
- Data Store Memory
- Data Store Read
- Data Store Write
- Demux
- Environment Controller
- From
- Goto
- Goto Tag Visibility
- Index Vector
- Manual Switch**
- Manual Variant Sink
- Manual Variant Source
- Merge
- Multiport Switch
- Mux**
- Parameter Writer
- Selector
- State Reader
- State Writer
- Switch
- Variant Sink
- Variant Source
- Vector Concatenate

# Esempio: Sistema LTI multivariabile

## ► Inizializzazione parametri:

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

## ► Simulink model:

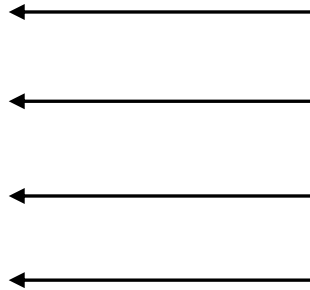




# Esempio: Sistema LTI multivariabile

## ➡ Configurazione blocco state-space

Variabili  
caricate dal  
workspace di  
Matlab



Block Parameters: State-Space

State Space

State-space model:  
 $dx/dt = Ax + Bu$   
 $y = Cx + Du$

Parameters

A:

B:

C:

D:

Initial conditions:

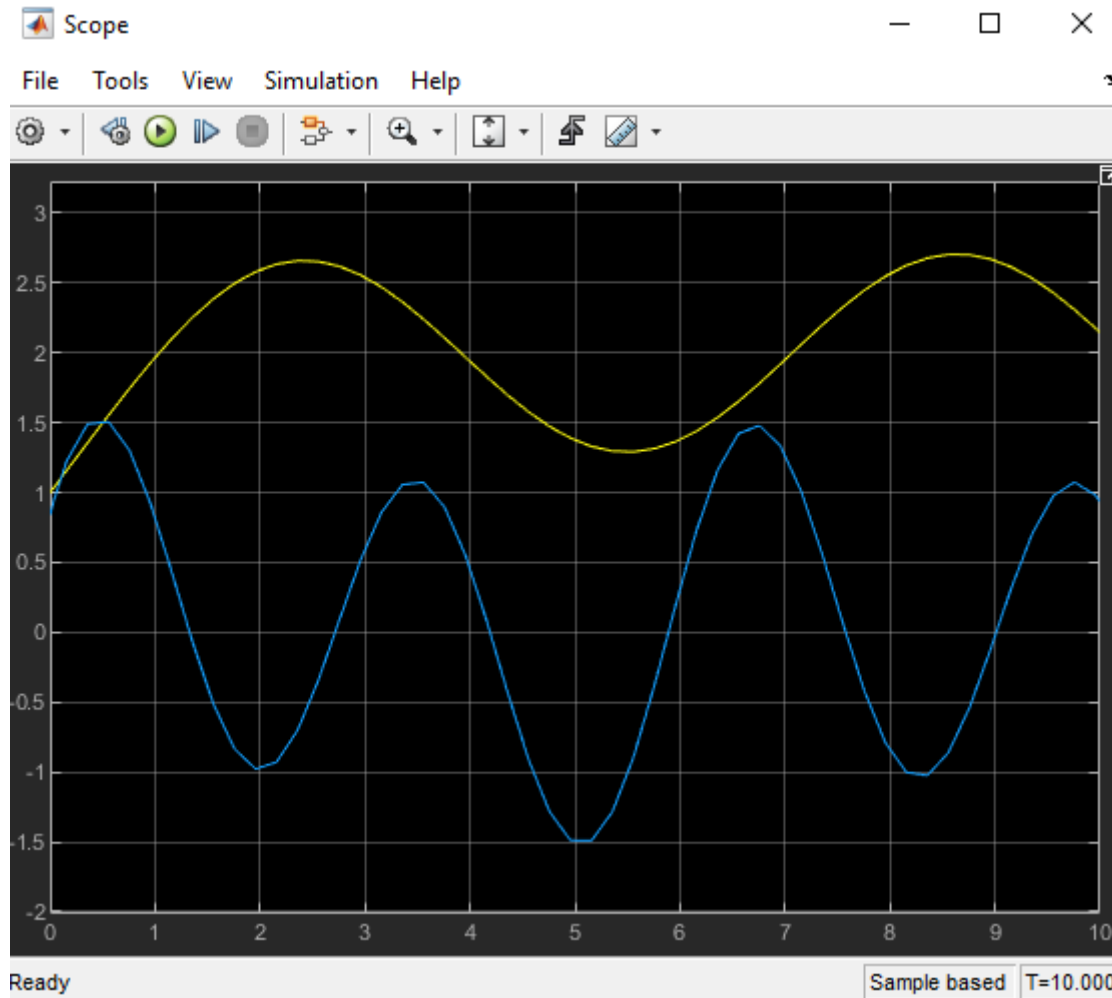
Absolute tolerance:

State Name: (e.g., 'position')

? OK Cancel Help Apply

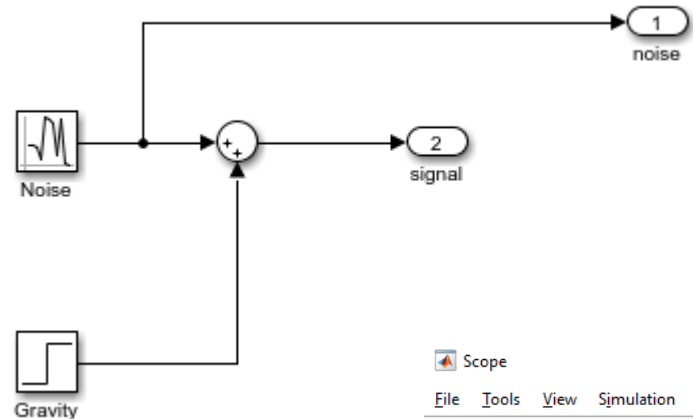
# Esempio: Sistema LTI multivariabile

➡ Simulazione (ingressi parametrizzati a piacere)

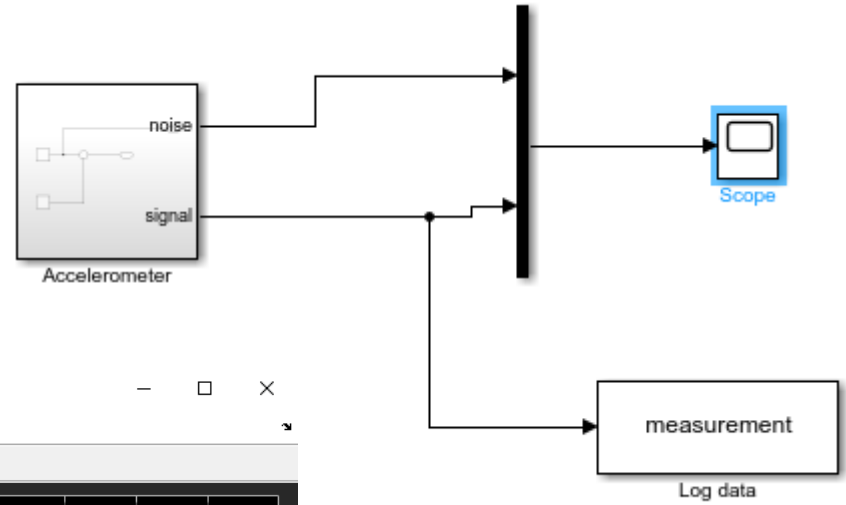


# Esempio: l'accelerometro

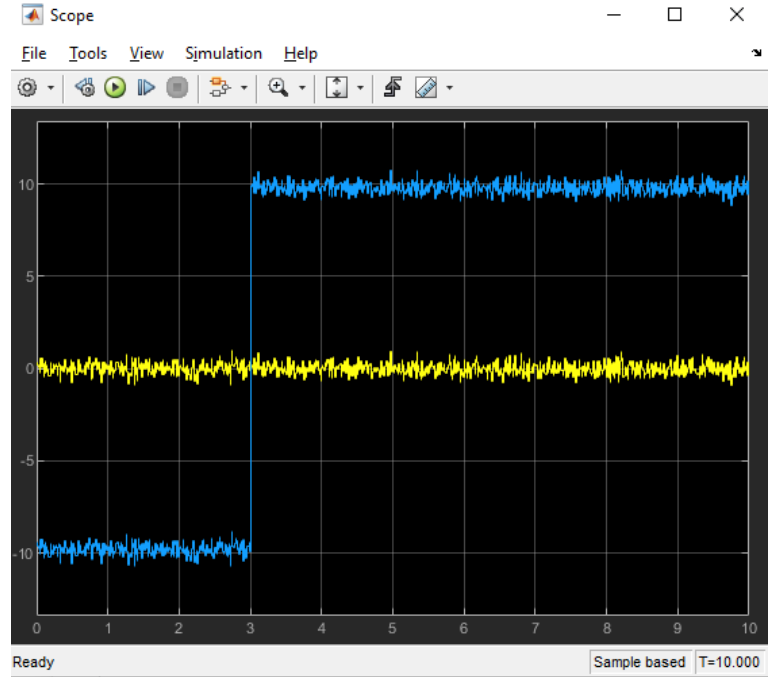
➡ Subsystem:



➡ System:



➡ Scope:



# Esercizio: retroazione stato - ingresso

► Dato il sistema:

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

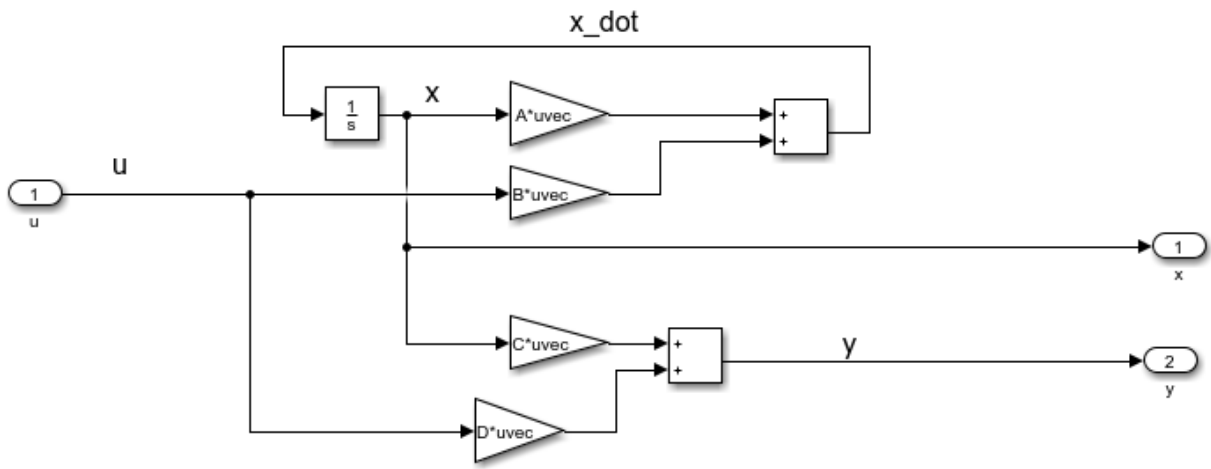
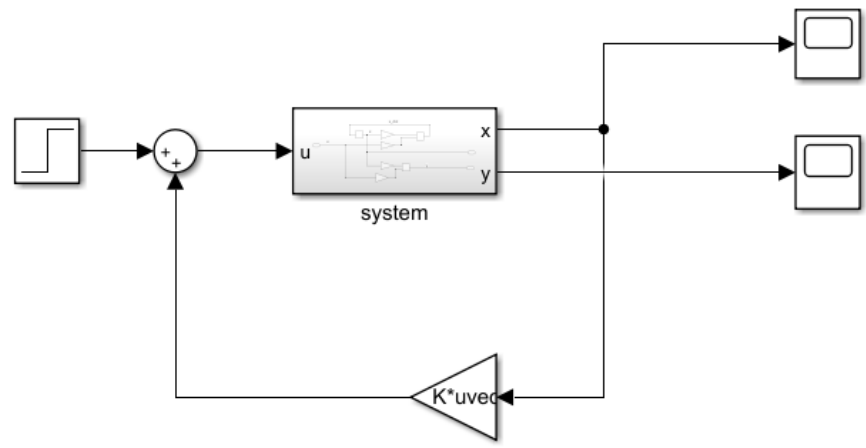
Con:

$$A = \begin{bmatrix} -2 & -20 & -2 \\ 0 & 0 & 1 \\ 1 & -5 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 20 \\ 0 \\ 0 \end{bmatrix} \quad C = [0 \quad 2 \quad 0] \quad D = 0$$

- Costruire il modello Simulink del sistema controllato tramite opportuna retroazione stato-ingresso che stabilizzi il sistema
- Simulare e visualizzare l'andamento di stato e uscita a fronte di un ingresso a gradino

# Esercizio: retroazione stato - ingresso

## ► Soluzione



# Esercizio: retroazione stato - ingresso

► Script di inizializzazione:

```
A = [-2 -20 -2;0 0 1;1 -5 -1];
```

```
B = [20;0;0];
```

```
C = [0 2 0];
```

```
D = 0;
```

```
H = -place(A, B, [-1; -2; -3]);
```

Autovalori desiderati  
(**attenzione:** immettere  
sempre un vettore di  
autovalori desiderati con  
valori differenti tra loro)

Comando che risolve il problema  
dell'assegnamento degli  
autovalori di  $A - B^*H$



# INTRODUZIONE A MATLAB e SIMULINK

**FINE**