

# CONTROLLO DI SISTEMI ROBOTICI

*- Controllo di Robot Antropomorfi e Robot Mobili  
in ambiente Matlab/Simulink*

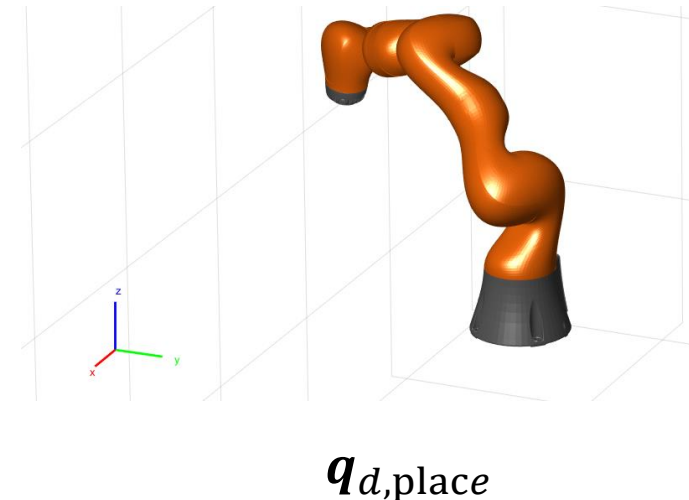
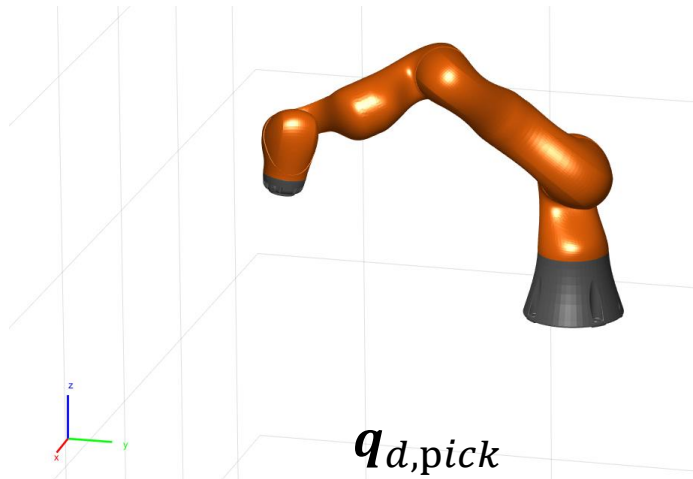
# *Toolbox utilizzati*



- ➡ Matlab Robotics System Toolbox
- ➡ Simulink 3D Animation

# Robot Antropomorfo: Pick and Place

- Risolvere un problema di pick and place (regolazione) nello spazio di giunto per il manipolatore antropomorfo Kuka LBR 14 (robot ridondante 7 DoF)
- Tale problema può essere affrontato con la tecnica di controllo in Feedback Linearization PD + compensazione di gravità
- Il robot deve essere programmato per raggiungere la configurazione di picking  $\mathbf{q}_{d,pick}$  (dove avviene la presa dell'oggetto da movimentare) e in seguito la posizione di placing  $\mathbf{q}_{d,place}$  (dove avviene il rilascio)



# Modello dinamico del manipolatore

- ➔ Il Robotics System Toolbox di Matlab include le funzionalità per definire cinematica e dinamica di un manipolatore e alcuni modelli di Robot già definiti, tra i quali il manipolatore KUKA LBR 14
- ➔ Per importare il modello è necessario lanciare i seguenti comandi:

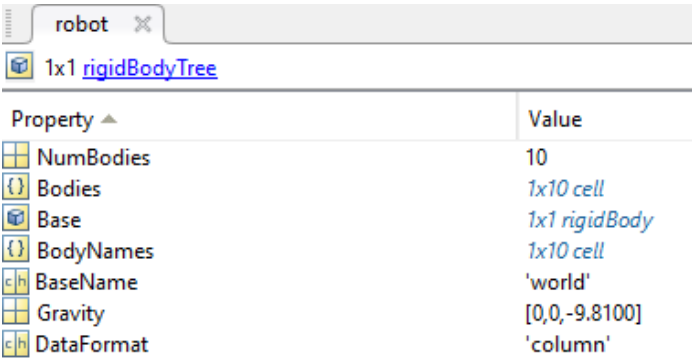
```
robot = importrobot('iiwa14.urdf');  
robot.DataFormat = 'column';  
robot.Gravity = [0;0;-9.81];
```

specifica il vettore gravità

specifica il fatto che i vettori  $q, \dot{q}$  ... sono vettori colonna

unified robot description format

- ➔ Nel Workspace di Matlab viene creata la variabile *robot*, di tipo *RigidBodyTree*

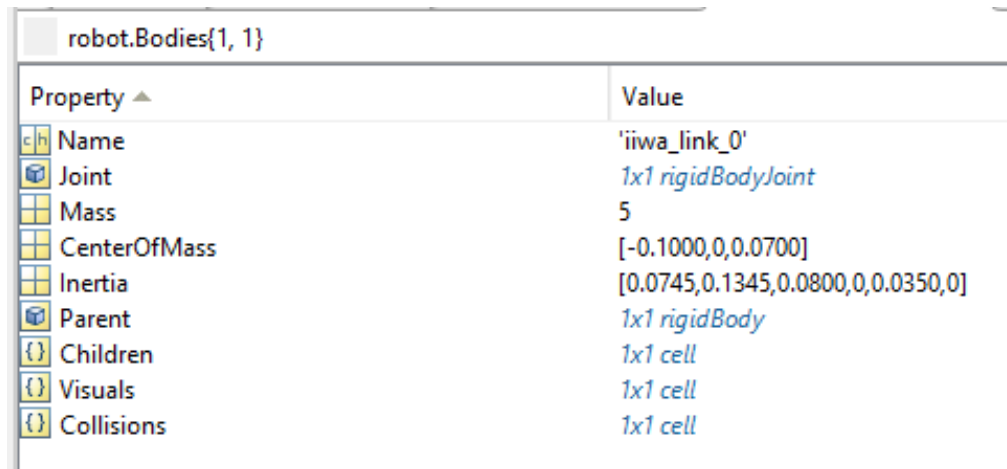


Property	Value
NumBodies	10
Bodies	1x10 cell
Base	1x1 rigidBody
BodyNames	1x10 cell
BaseName	'world'
Gravity	[0,0,-9.8100]
DataFormat	'column'



# Modello dinamico del manipolatore

- ➔ Un RigidBodyTree è una catena cinematica di corpi rigidi (*Bodies*) a ognuno dei quali è associato un giunto, una posa, le proprietà inerziali (massa, baricentro, tensore d'inerzia) e una mesh 3D
- ➔ Tale variabile possiede dunque tutte le informazioni per calcolare il modello dinamico del manipolatore



Property	Value
Name	'iiwa_link_0'
Joint	1x1 rigidBodyJoint
Mass	5
CenterOfMass	[-0.1000,0,0.0700]
Inertia	[0.0745,0.1345,0.0800,0,0.0350,0]
Parent	1x1 rigidBody
Children	1x1 cell
Visuals	1x1 cell
Collisions	1x1 cell

# Modello dinamico del manipolatore

- ➔ Il Robotics System Toolbox espone le funzioni che permettono di calcolare le varie componenti del modello dinamico del manipolatore, nella forma semplificata (con  $D, F_a = 0$ ):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}$$

- ➔  $M = \text{massMatrix}(\text{robot}, q)$  calcola  $\mathbf{M}(\mathbf{q})$
- ➔  $Cqdot = \text{velocityProduct}(\text{robot}, q, qdot)$  calcola  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$
- ➔  $g = \text{gravityTorque}(\text{robot}, q)$  calcola  $\mathbf{g}(\mathbf{q})$

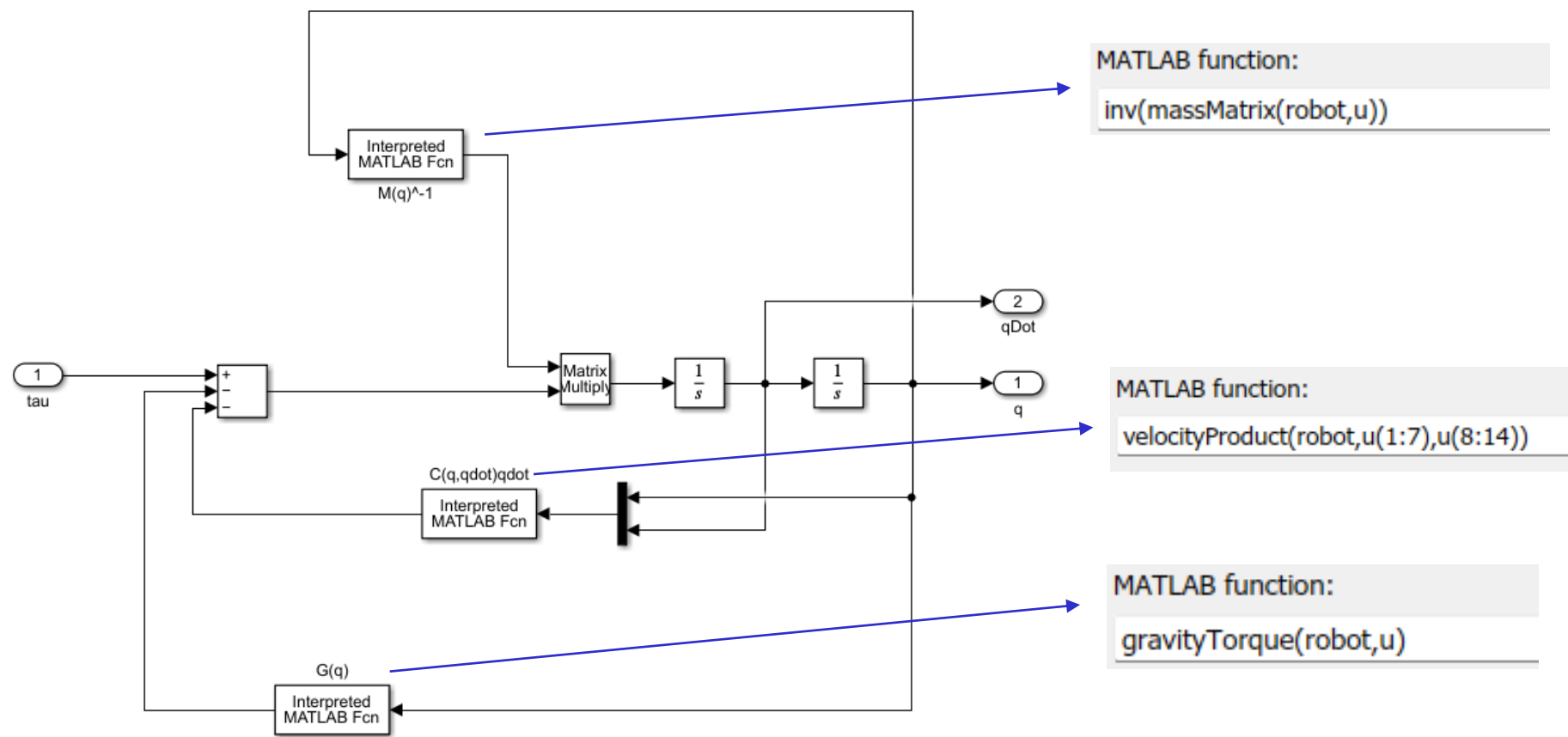
# Script di inizializzazione

```
% import modello robot
robot = importrobot('iiwa14.urdf');
robot.DataFormat = 'column';
robot.Gravity = [0;0;-9.81];

% configurazione iniziale e target
q0 = [0;0;0;0;0;0;0];
qd_pick = [-60;50;0;-60;0;60;0]*pi/180;
qd_place = [-120;50;0;-60;0;60;0]*pi/180;

% parametri controllo
Kd = 50*eye(7);
Kp = 100*eye(7);
```

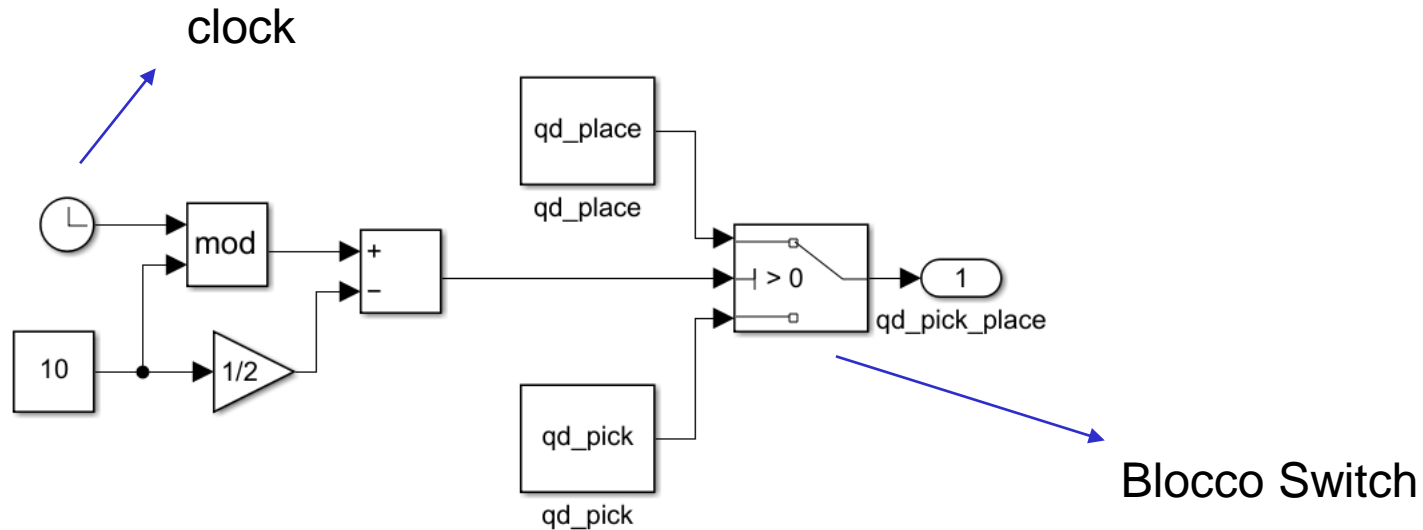
# Modello Simulink del robot



$$\ddot{q} = M^{-1}(q)(-C(q, \dot{q})\dot{q} - g(q) + \tau)$$

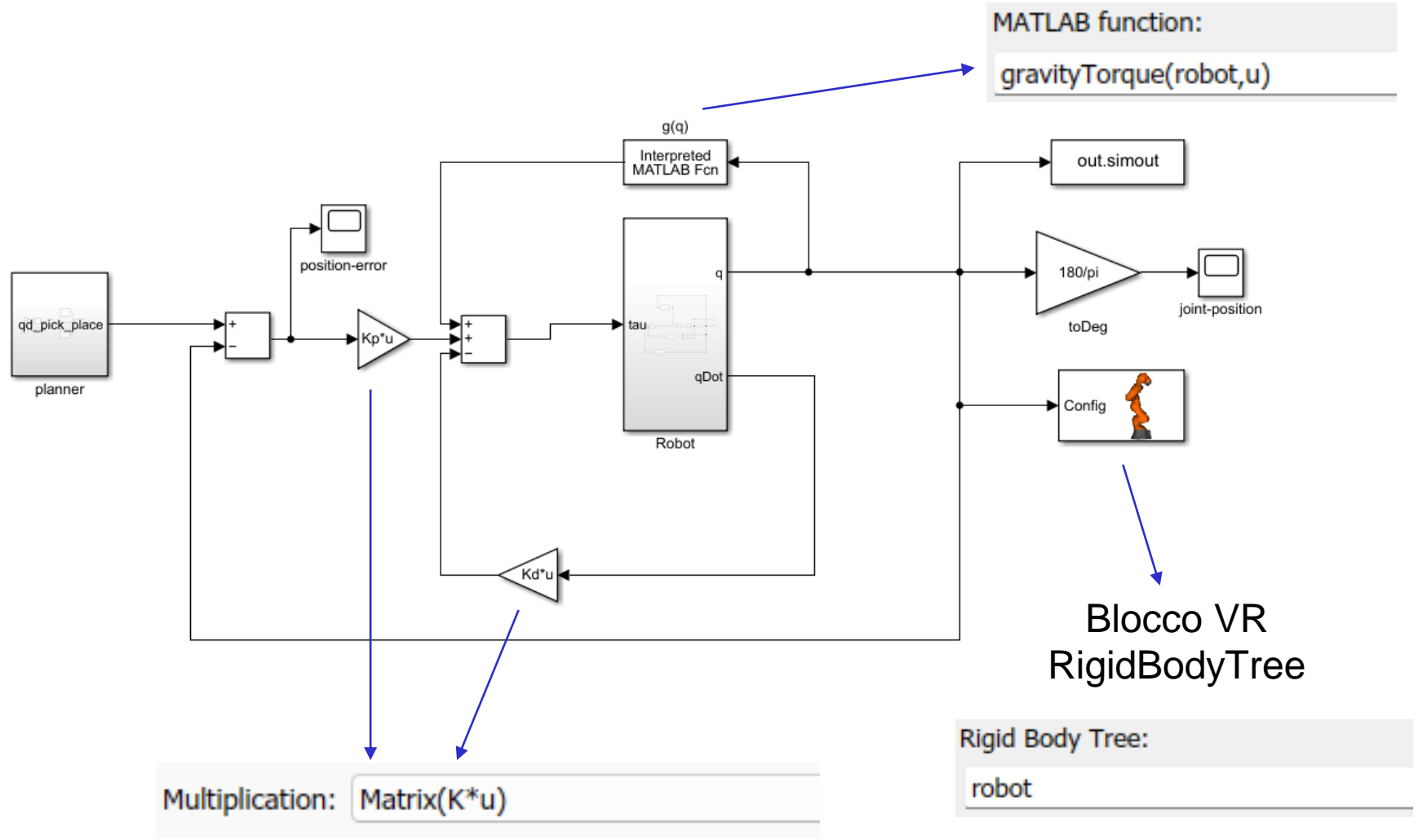


# Modello Simulink del pianificatore



- ➡ Genera un 'onda quadra' che alterna qd\_pick e qd\_place ogni 5 secondi (tempo presumibilmente sufficiente per esaurire i transistori)

# Modello Simulink del sistema controllato



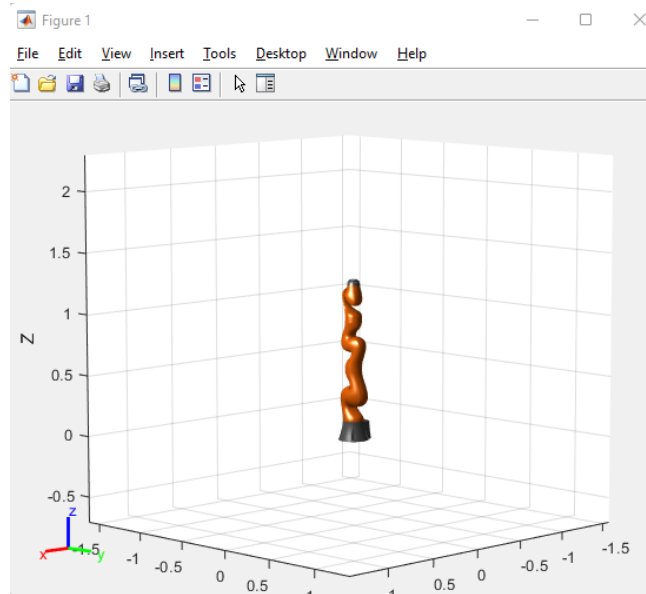
# Script di visualizzazione risultati (post-simulazione)

```
% visualizzazione 3D configurazioni assunte dal robot
n = length(out.tout);

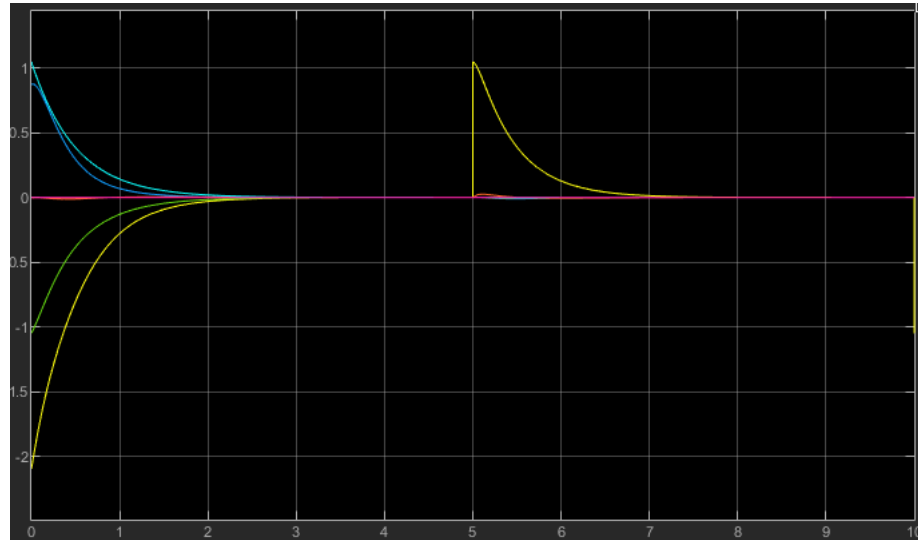
for i=1:n

    configNow = out.simout(i,:)';
    show(robot,configNow,'PreservePlot',false,'Frames','off');
    drawnow;

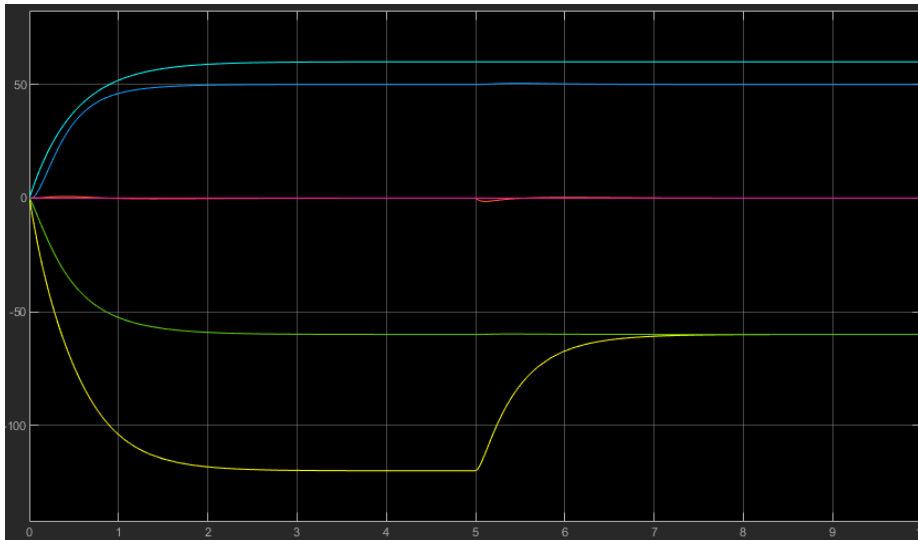
end
```



# Risultati



errore di posizione  
(rad)

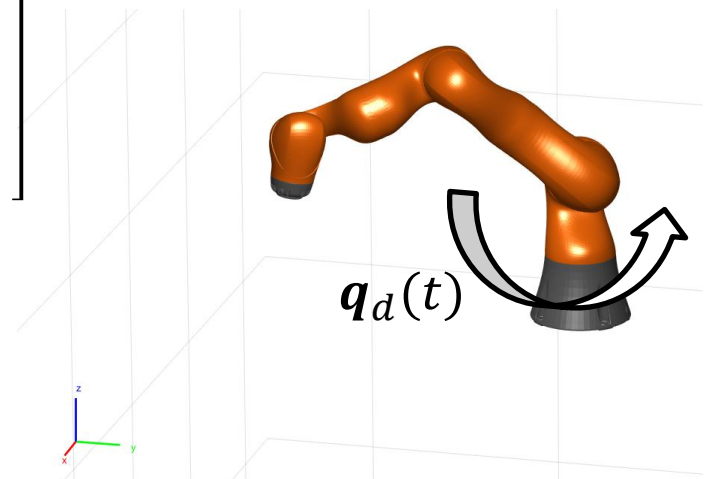


posizione giunti  
(deg)

# Robot Antropomorfo: Inseguimento di traiettoria

- Risolvere un problema di inseguimento di una traiettoria desiderata nello spazio di giunto per il manipolatore antropomorfo Kuka LBR 14 (robot ridondante 7 DoF)
- Tale problema può essere affrontato con la tecnica di controllo in Feedback Linearization denominata 'controllo a dinamica inversa'
- In particolare si desidera inseguire la seguente traiettoria:

$$\mathbf{q}_d(t) = \begin{bmatrix} \sin(t) \\ 50 \frac{\pi}{180} \\ 0 \\ -60 \frac{\pi}{180} \\ 0 \\ 60 \frac{\pi}{180} \\ 0 \end{bmatrix} \quad \dot{\mathbf{q}}_d(t) = \begin{bmatrix} \cos(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \ddot{\mathbf{q}}_d(t) = \begin{bmatrix} -\sin(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



# Robot Antropomorfo: Inseguimento di traiettoria

- L'azione di controllo cancella le non-linearità tramite feedback dello stato:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\mathbf{v} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \Rightarrow \ddot{\mathbf{q}} = \mathbf{v}$$

- La dinamica dell'errore di tracking viene resa asintoticamente stabile imponendo:

$$\mathbf{v} = \ddot{\mathbf{q}}_d - \mathbf{K}_d(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d) - \mathbf{K}_p(\mathbf{q} - \mathbf{q}_d)$$

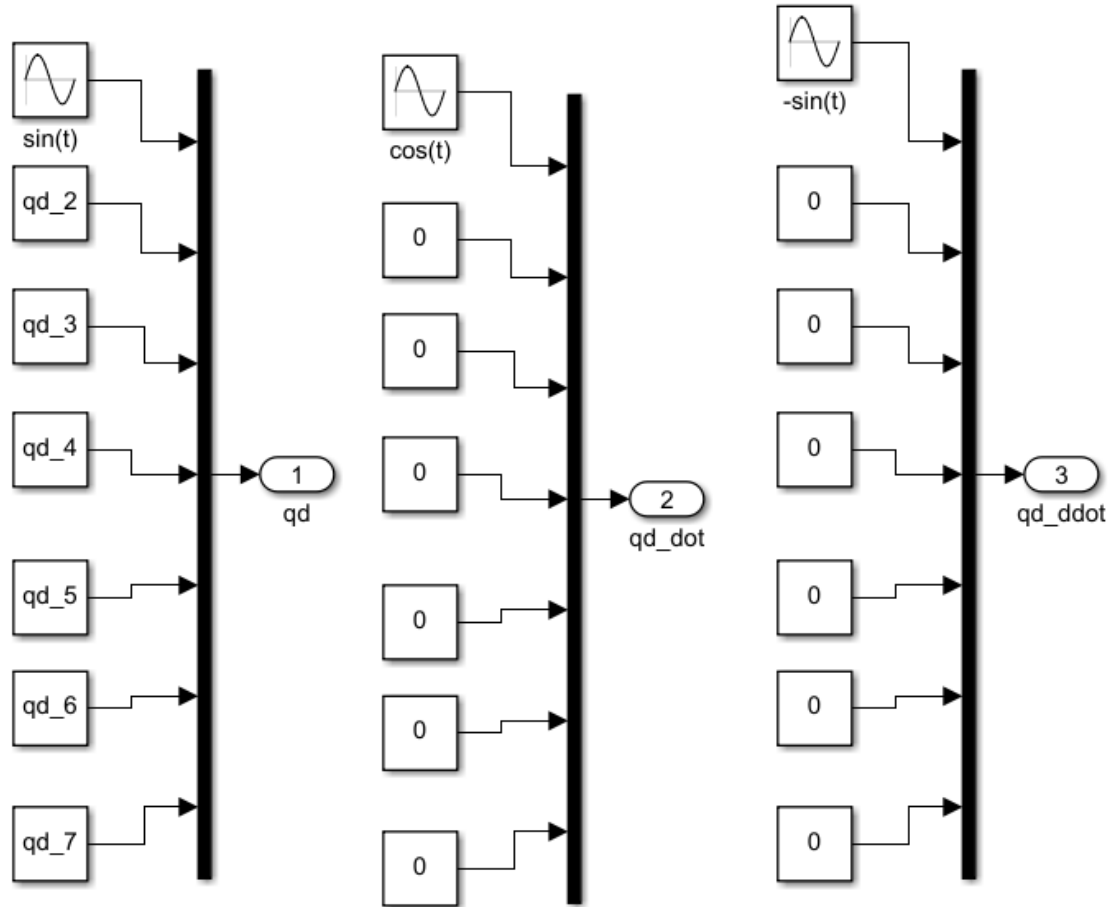
# Script di inizializzazione

```
% import robot
robot = importrobot('iiwa14.urdf');
robot.DataFormat = 'column';
robot.Gravity = [0;0;-9.81];

% configurazione iniziale e target
(componenti 2->7)
q0 = [0;0;0;0;0;0;0];
qd_2 = 50*pi/180;
qd_3 = 0;
qd_4 = -60*pi/180;
qd_5 = 0;
qd_6 = 60*pi/180;
qd_7 = 0;

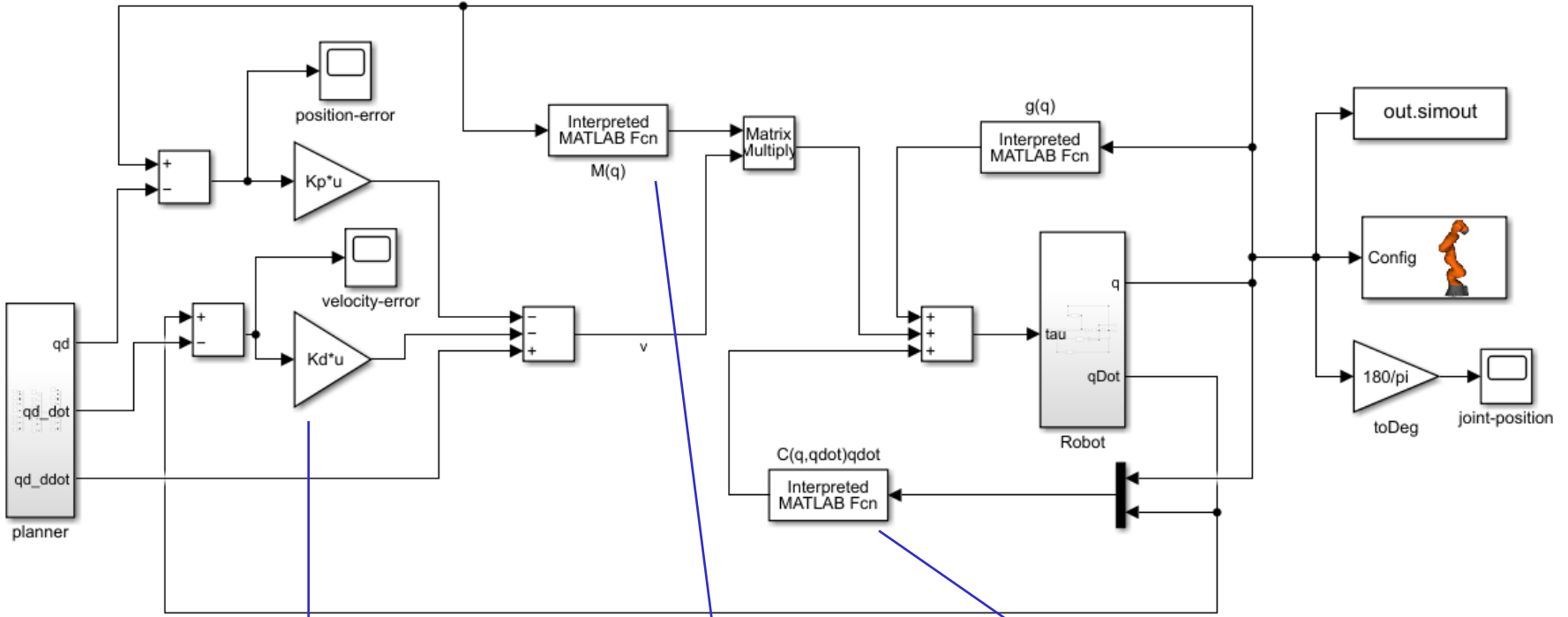
% controllo
Kd = 10*eye(7);
Kp = 10*eye(7);
```

# Pianificatore traiettoria





# Schema di controllo



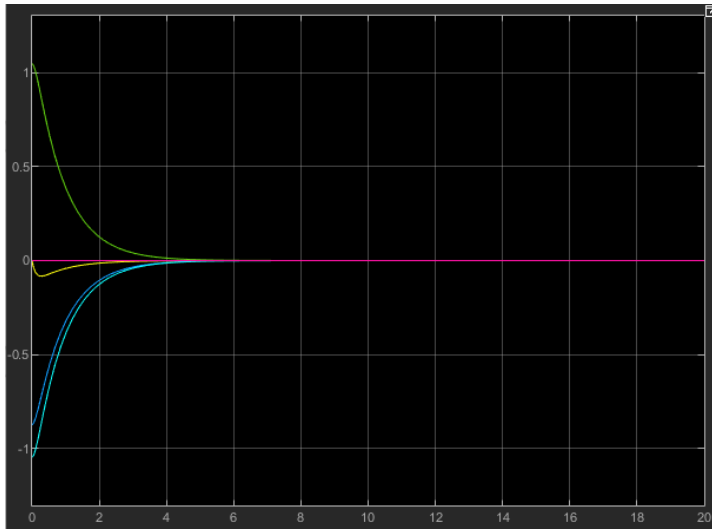
Multiplication: `Matrix(K*u)`

MATLAB function:  
`massMatrix(robot,u)`

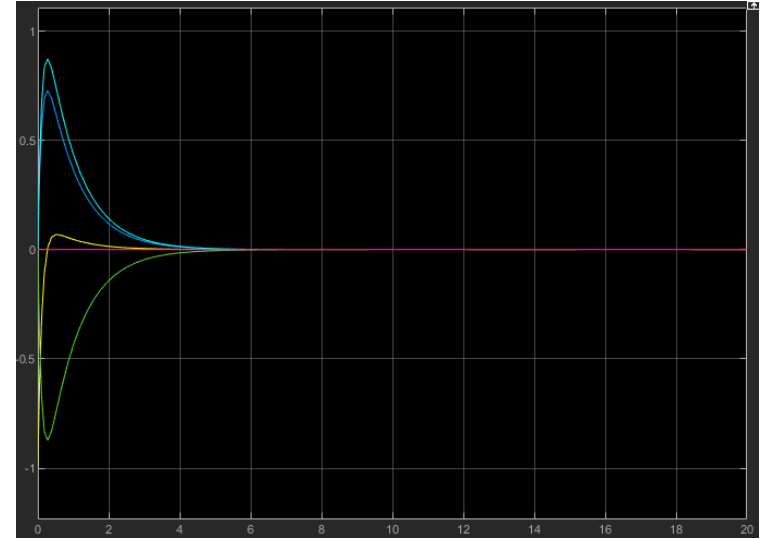
MATLAB function:  
`velocityProduct(robot,u(1:7),u(8:14))`

# Risultati

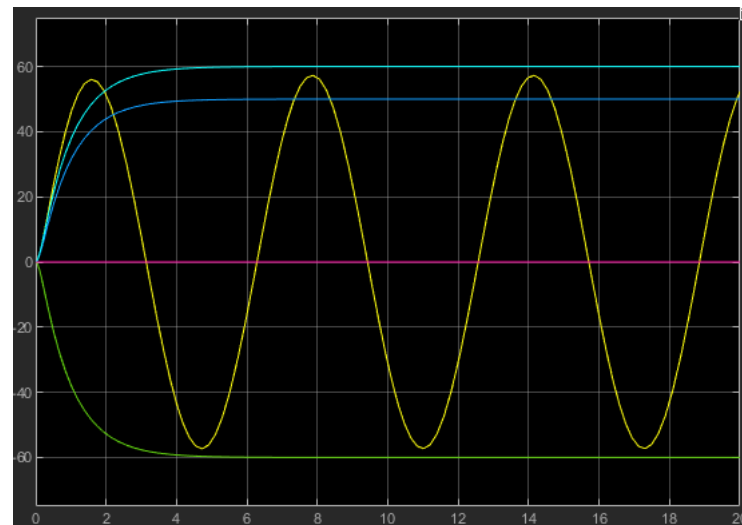
➔ errore di posizione (rad)



➔ errore di velocità (rad)



➔ posizione giunti (deg)



# Robot Mobile, inseguimento di traiettoria

- Si consideri un Robot Mobile con modello uniciclo a guida differenziale, con i seguenti parametri:
  - Raggio delle ruote  $r = 0.05 \text{ m}$
  - Distanza tra le ruote  $d = 0.2 \text{ m}$
  - Distanza del punto di riferimento dal centro asse  $b = 0.1 \text{ m}$

- Condizioni iniziali:  $\mathbf{q}_0(t) = \begin{bmatrix} -2 \\ -2 \\ 0 \end{bmatrix} \text{m}$

- Si risolva un problema tracking per la seguente traiettoria desiderata:

$$\mathbf{q}_d(t) = \begin{bmatrix} x_B(t) \\ y_B(t) \end{bmatrix} = \begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix}$$

# Script di inizializzazione

```
%% robot params
```

```
r = 0.05;
```

```
d = 0.2;
```

```
b = 0.1;
```

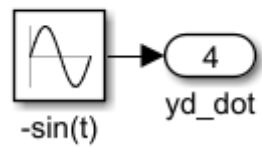
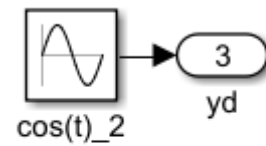
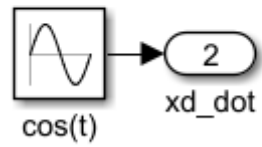
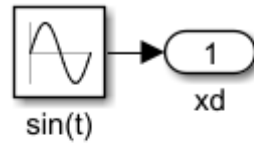
```
q0 = [-2;-2;0];
```

```
%% control params
```

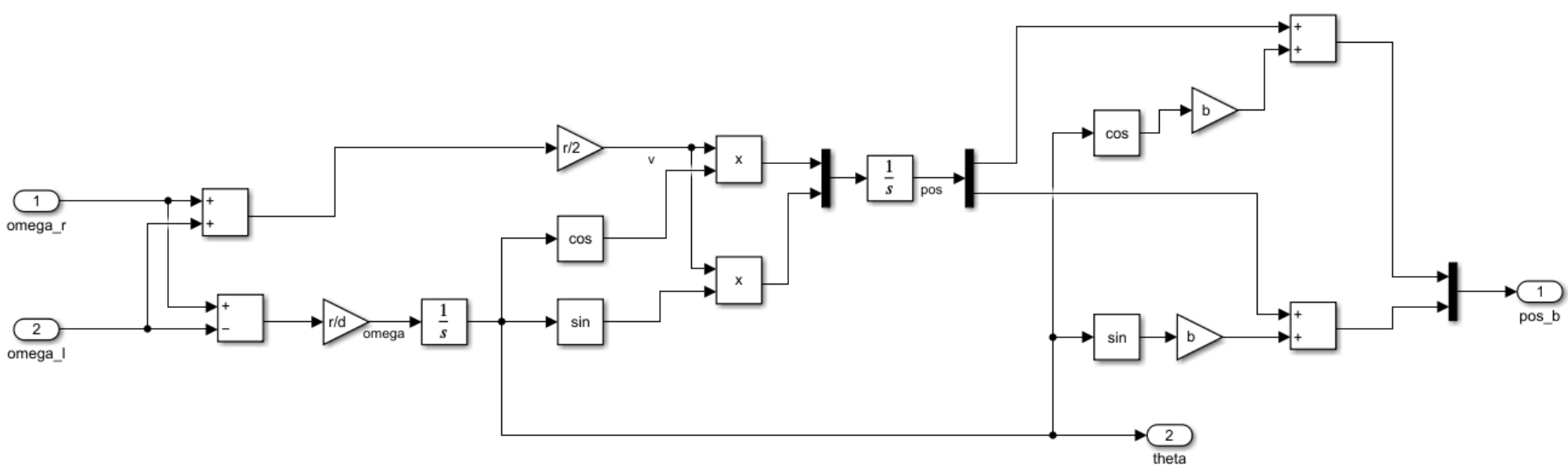
```
k1 = 0.5;
```

```
k2 = 0.5;
```

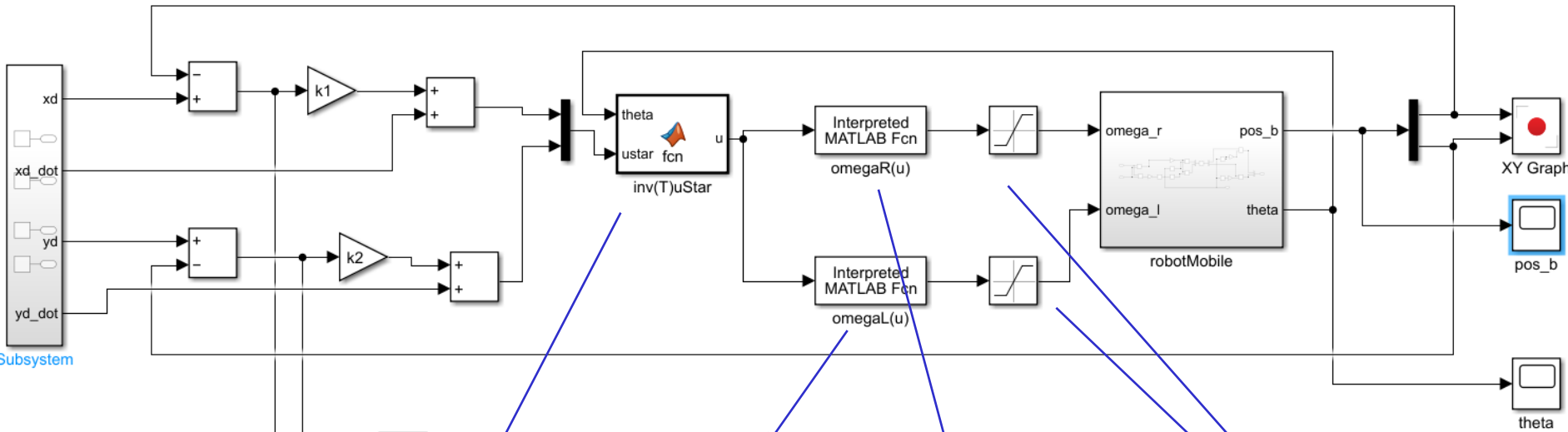
# Pianificatore



# Modello del Robot Mobile



# Controllo differenziale driving



MATLAB function:  
 $(2*u(1)-d*u(2))/(2*r)$

MATLAB function:  
 $(2*u(1)+d*u(2))/(2*r)$

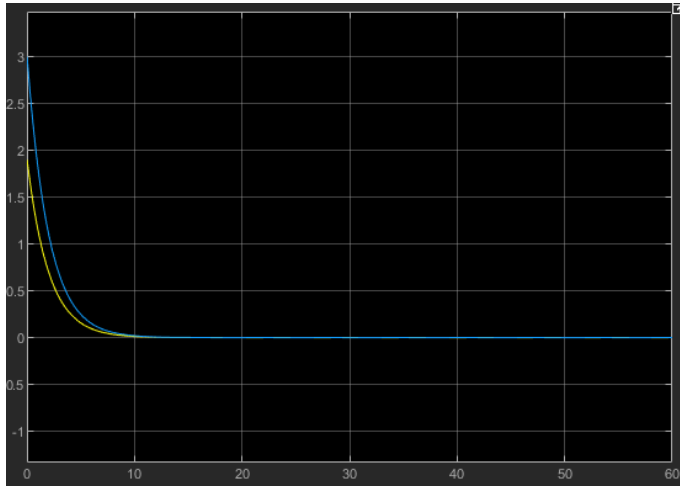
Upper limit:  
 50  
 Lower limit:  
 -50

Limiti attuatori

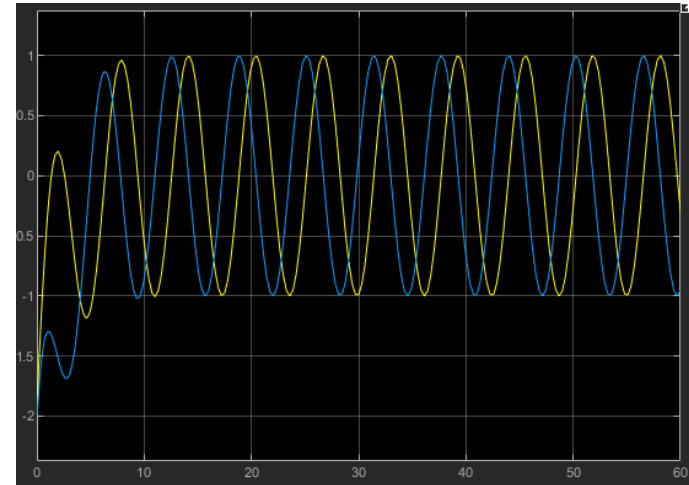
```
function u = fcn(theta,ustar,b)
u = [cos(theta) sin(theta);-1/b*sin(theta) 1/b*cos(theta)]*ustar;
```

# Risultati

➡ errore di posizione



➡ Traiettoria (nel tempo)



➡ Traiettoria (nello spazio)

