



Robotics 1

Kinematic control

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



SAPIENZA
UNIVERSITÀ DI ROMA



Robot motion control

- need to “actually” realize a desired robot motion task ...
 - **regulation** of pose/configuration (constant reference)
 - **trajectory** following/**tracking** (time-varying reference)
- ... despite the presence of
 - external disturbances and/or unmodeled dynamic effects
 - initial errors (or arising later due to disturbances) w.r.t. desired task
 - discrete-time implementation, uncertain robot parameters, ...
- we use a **general** control scheme based on
 - **feedback** (from robot state measures, to impose **asymptotic stability**)
 - **feedforward** (nominal commands generated in the planning phase)
- the **error** driving the feedback part of the control law can be defined either in **Cartesian** or in **joint** space
 - control action **always** occurs at the joint level (where actuators drive the robot), but performance has to be evaluated at the **task** level

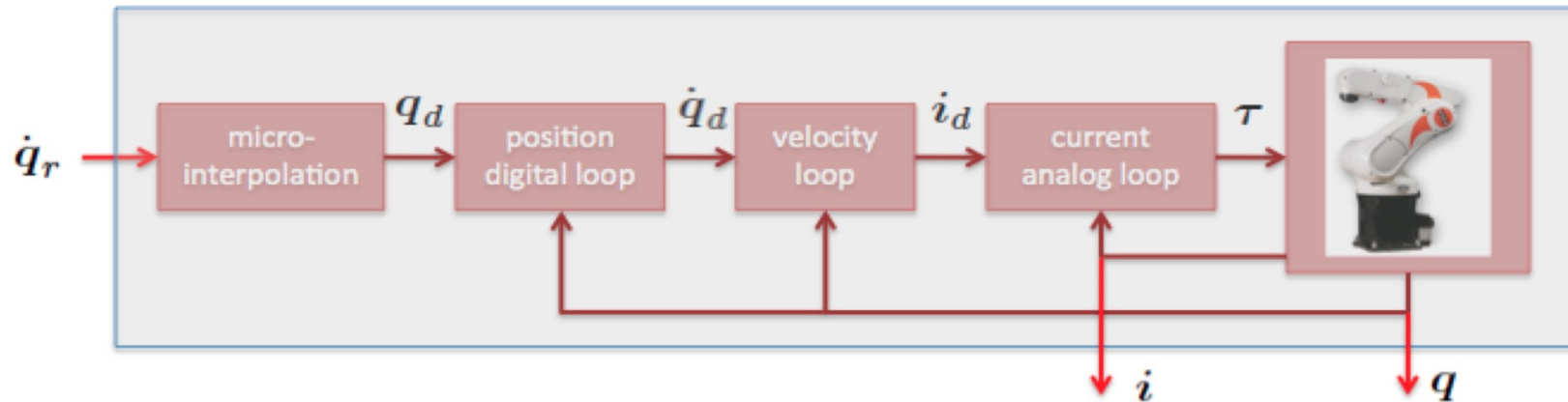


Kinematic control of robots

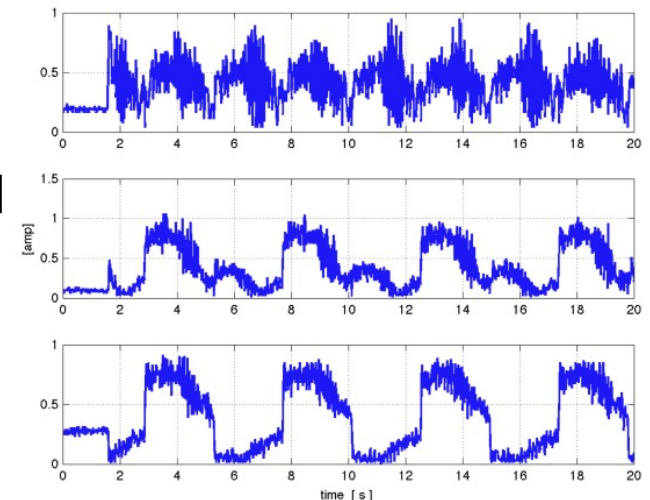
- a robot is an electro-mechanical system driven by actuating **torques** produced by the motors (with voltage/current inputs)
- it is possible, however, to consider a **kinematic command** (most often, a **velocity**) as control input to the system...
- ...thanks to the presence of **low-level feedback control** at the joints, which imposes to the robot the commanded velocities (at least, in the “ideal case”)
- these feedback loops are present in industrial robots within a **“closed” control architecture**, where users can only specify reference commands of the kinematic type
- in this way, **performance** can still be very satisfactory, provided the desired motion is **not too fast** and/or **does not require too large accelerations** (or transparent haptic feedback is not needed)

Closed control architecture

KUKA KR5 Sixx R650 robot



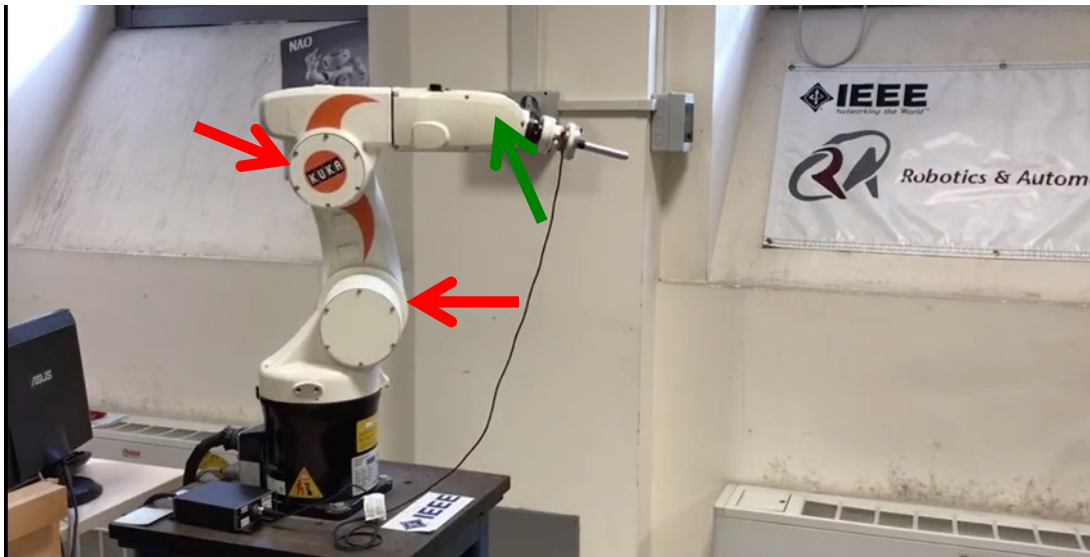
- low-level motor control laws are **not known nor accessible** by the user
- user programs based also on other exteroceptive sensors (vision, Kinect, F/T sensor) can be implemented on an **external PC via the RSI** (RobotSensorInterface), communicating with the KUKA controller **every 12 ms**
- available robot measures: **joint positions** (by encoders) and (**absolute value** of) **applied motor currents**
- the control reference is given as a **velocity** or a position **in joint space** (Cartesian commands are also accepted)



typical motor currents
on first three joints

physical Human-Robot Interaction

combining F/T sensor data and motor currents



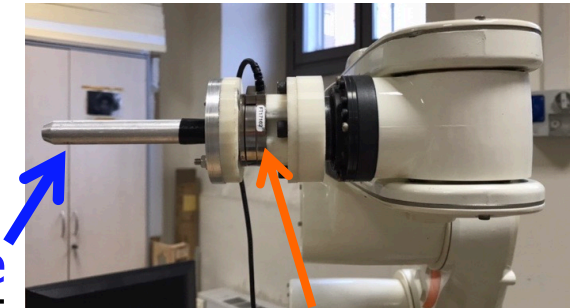
Robot in cyclic motion between four Cartesian positions

6R KUKA KR5 Sixx with
closed control architecture
and RSI interface at $T_c = 12$ ms

intentional contacts (soft)
and/or collisions (hard) may
occur anywhere along the
robot structure ...

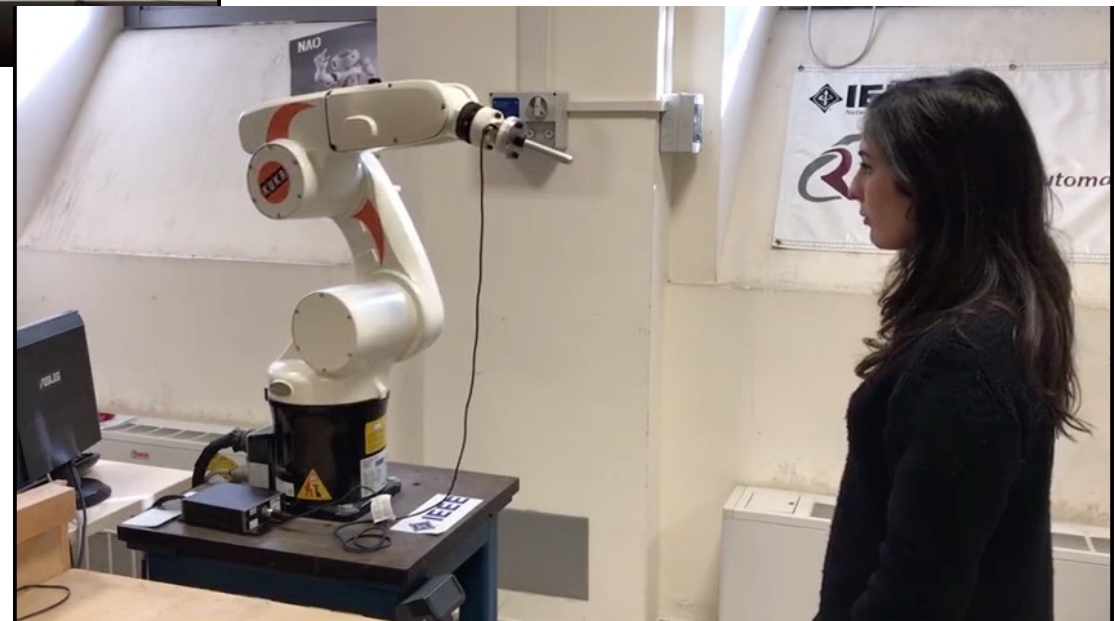
E. Mariotti, E. Magrini, A. De Luca: ICRA2019

collaborative
forces at E-E



ATI Mini45
F/T sensor

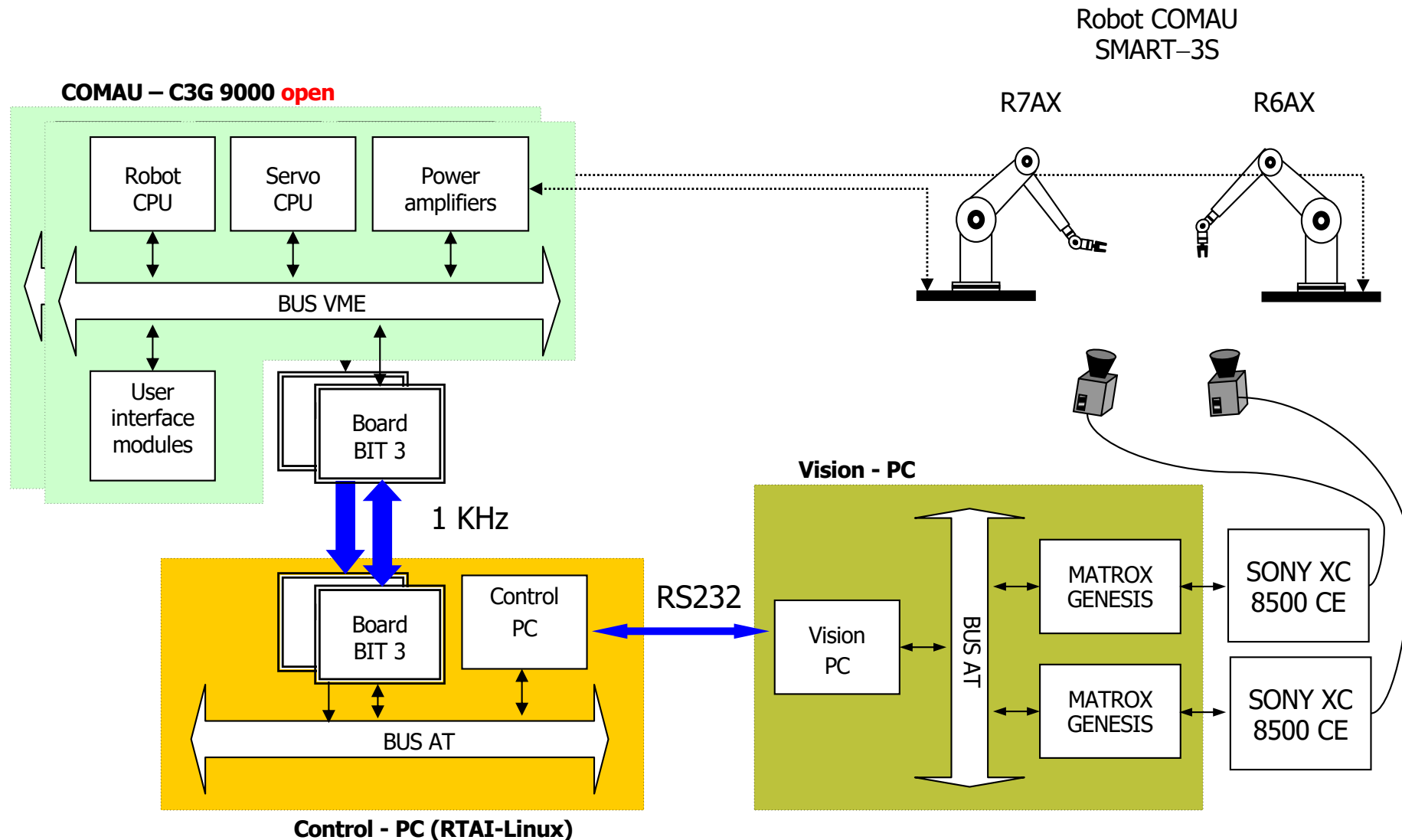
<https://youtu.be/SfZcG1Y713w>



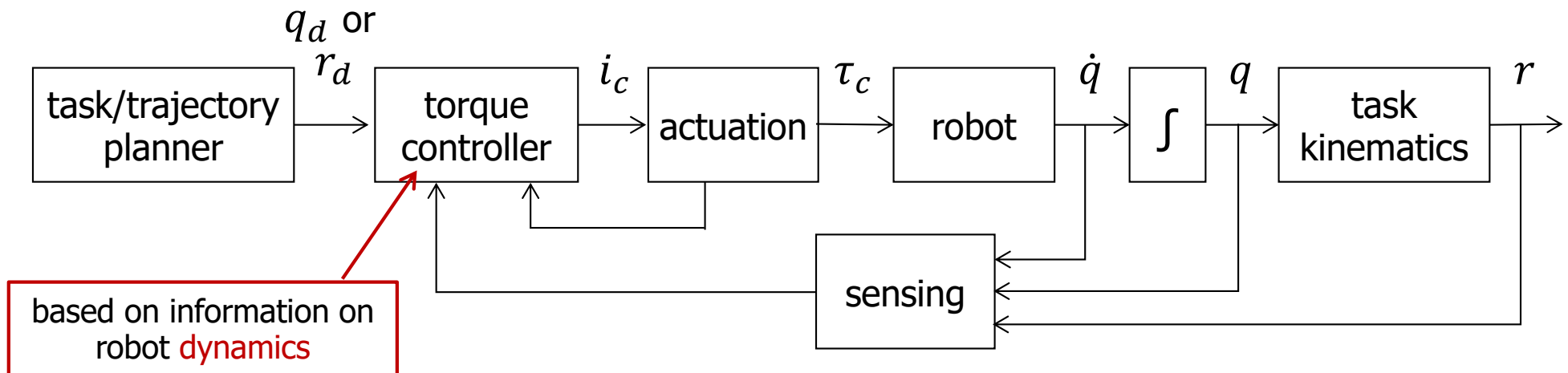
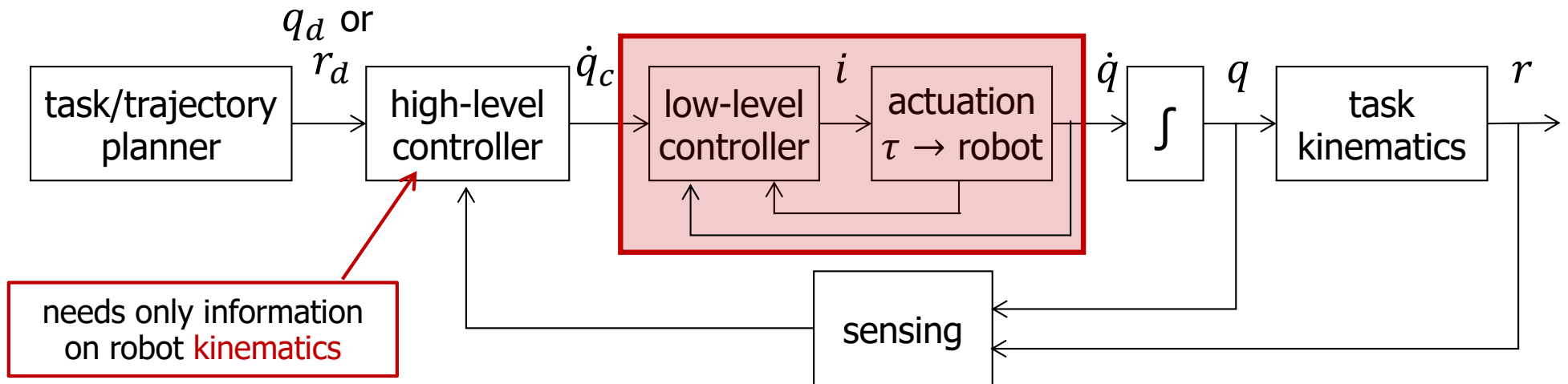


Hardware architecture

example including vision in an open controller



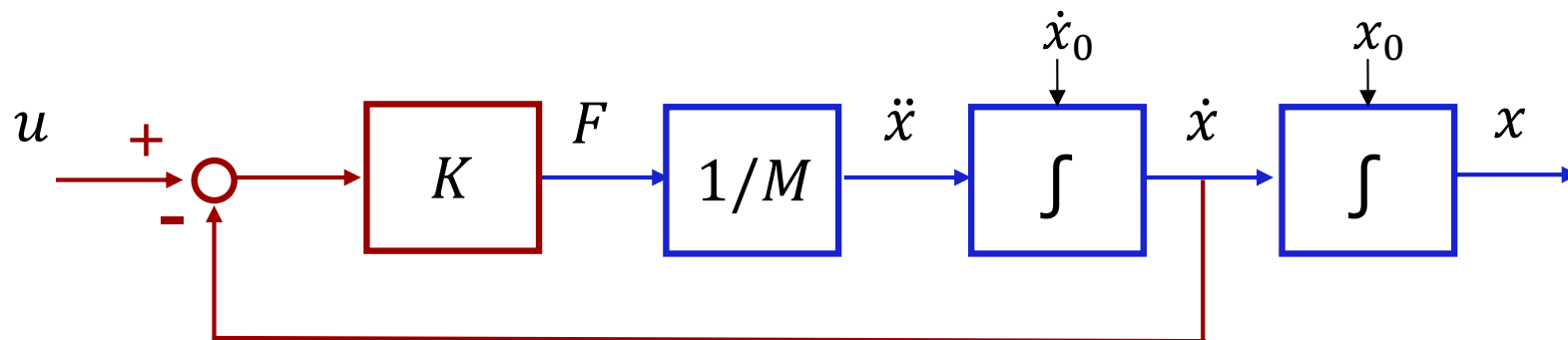
Kinematic vs. dynamic control of robots



more on this in Robotics 2 ...

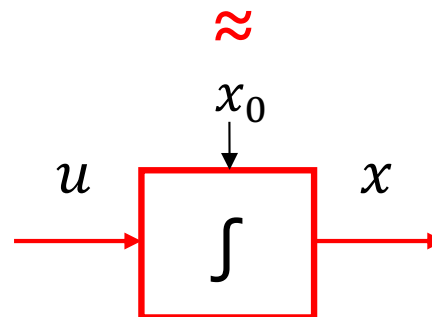
An introductory example

- a mass M in linear motion: $M\ddot{x} = F$ (dynamic model)
- low-level feedback: $F = K(u - \dot{x})$, with u = reference velocity
- equivalent scheme for $K \rightarrow \infty$: $\dot{x} \approx u$
- in practice, valid in a limited frequency “bandwidth” $\omega \leq K/M$



inner loop exact solution in continuous time
for a constant input $\bar{u}$

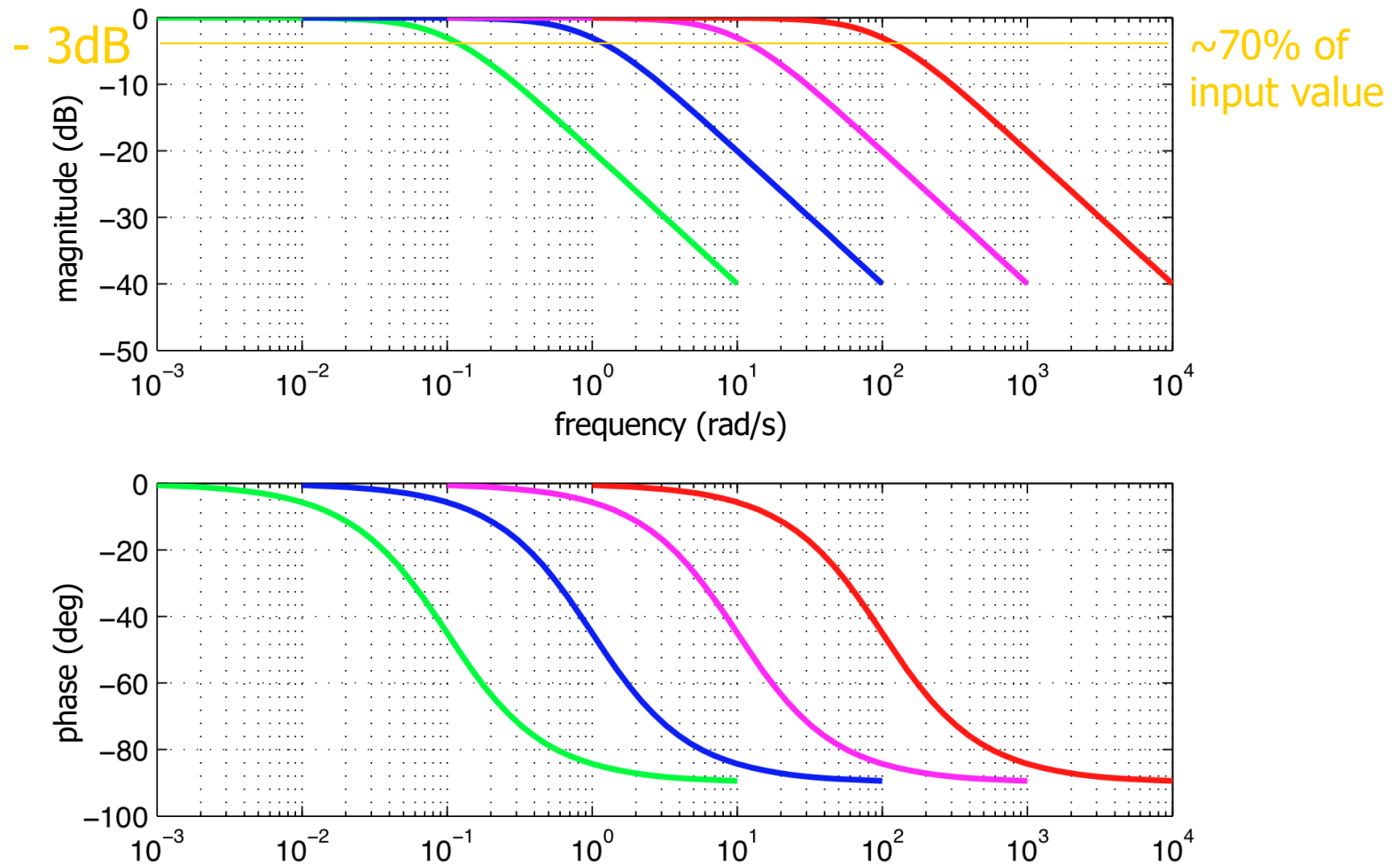
$$\dot{x}(t) = \dot{x}_0 + (\bar{u} - \dot{x}_0)[1 - \exp(-\frac{K}{M} t)]$$





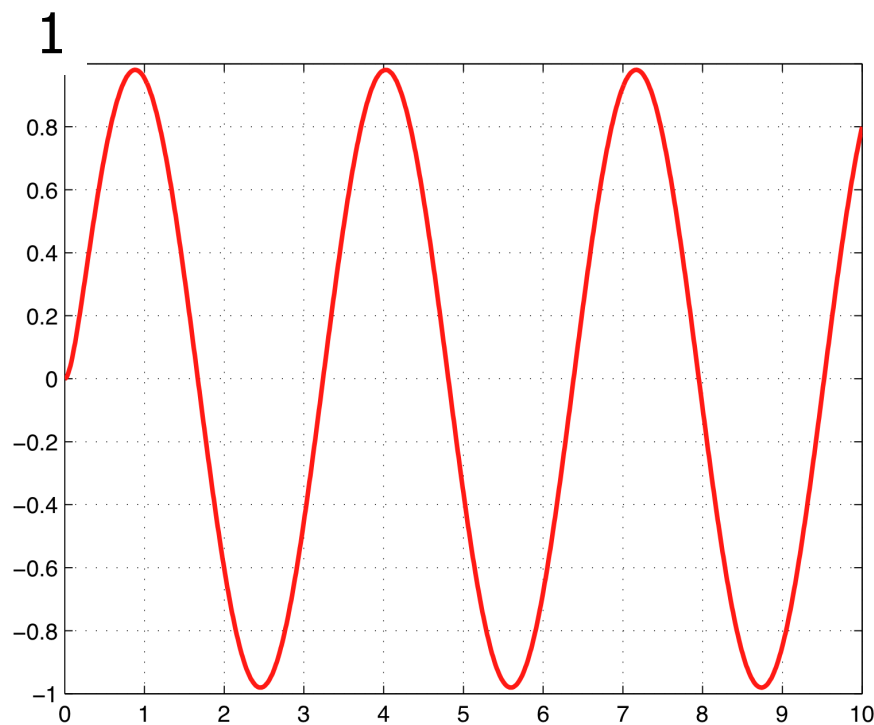
Frequency response of the closed-loop system

- Bode diagrams of $P(s) = \frac{\dot{x}(s)}{u(s)} = \frac{sx(s)}{u(s)}$ for $K/M = 0.1, 1, 10, 100$

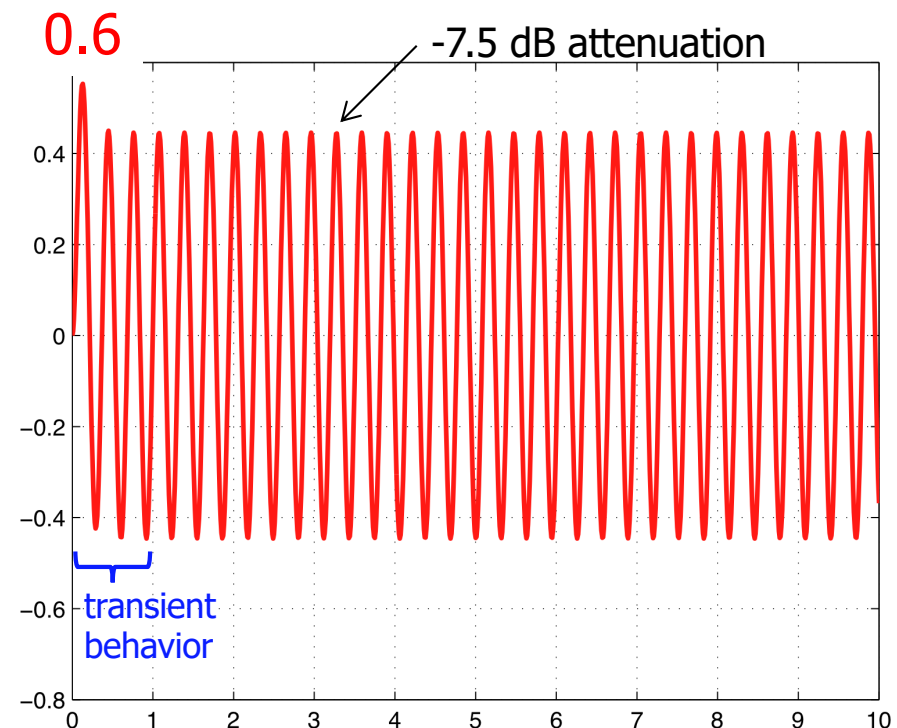


Time response

- setting $K/M = 10$ (bandwidth), we show two possible time responses to unit sinusoidal velocity reference commands at different ω



$\omega = 2 \text{ rad/s}$



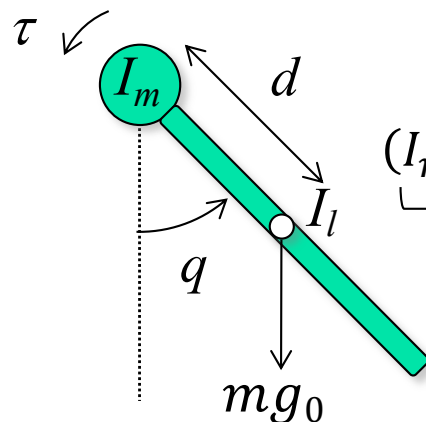
$\omega = 20 \text{ rad/s}$

realized **output** velocities

A more detailed example

including nonlinear dynamics

- single link (a thin rod) of mass m , center of mass at d from joint axis, inertia M (motor + link) at the joint, rotating in a vertical plane (the gravity torque at the joint is configuration dependent)



dynamic model

$$\underbrace{(I_m + I_l + md^2)}_M \ddot{q} + mg_0 d \sin q = \tau$$

after reduction, with
matching condition

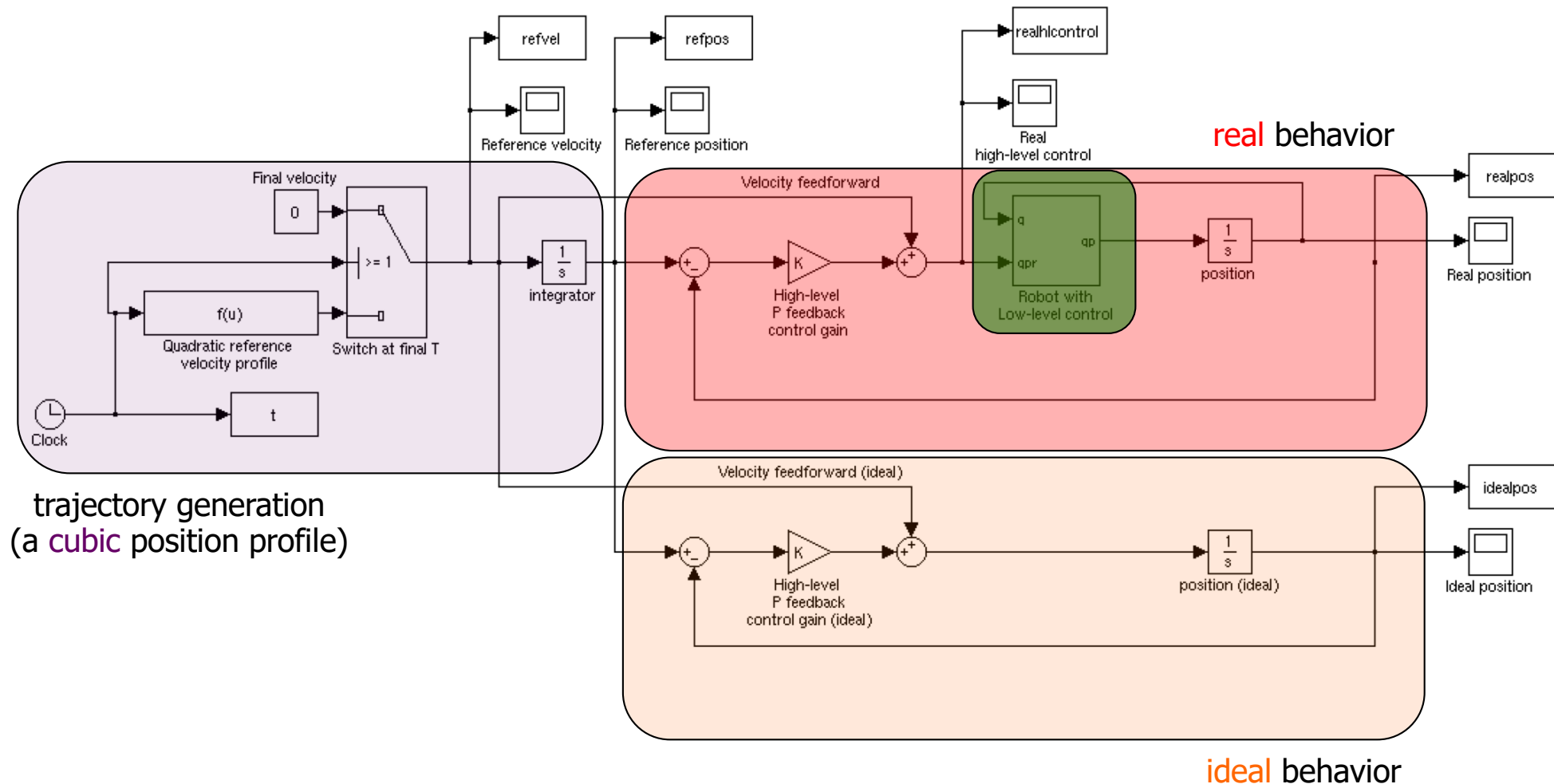
$$\begin{aligned} g_0 &= 9.81 [m/s^2] \\ m &= 10 [kg] \\ d &= l/2 = 0.2 [m] \\ I_l &= ml^2/12 = 0.1333 [kgm^2] \\ I_m &= 0.5333 [kgm^2] \\ & \quad (= I_l + md^2) \\ \Rightarrow M &= 1.0667 [kgm^2] \end{aligned}$$

- fast **low-level feedback** control loop based on a PI action on the velocity error + an approximate acceleration feedforward
- kinematic control** loop based on a **P feedback action** on the position error + feedforward of the velocity reference at the **joint level**
- evaluation of **tracking performance** for rest-to-rest motion tasks with “increasing dynamics” = higher accelerations

A more detailed example

differences between the ideal and real case

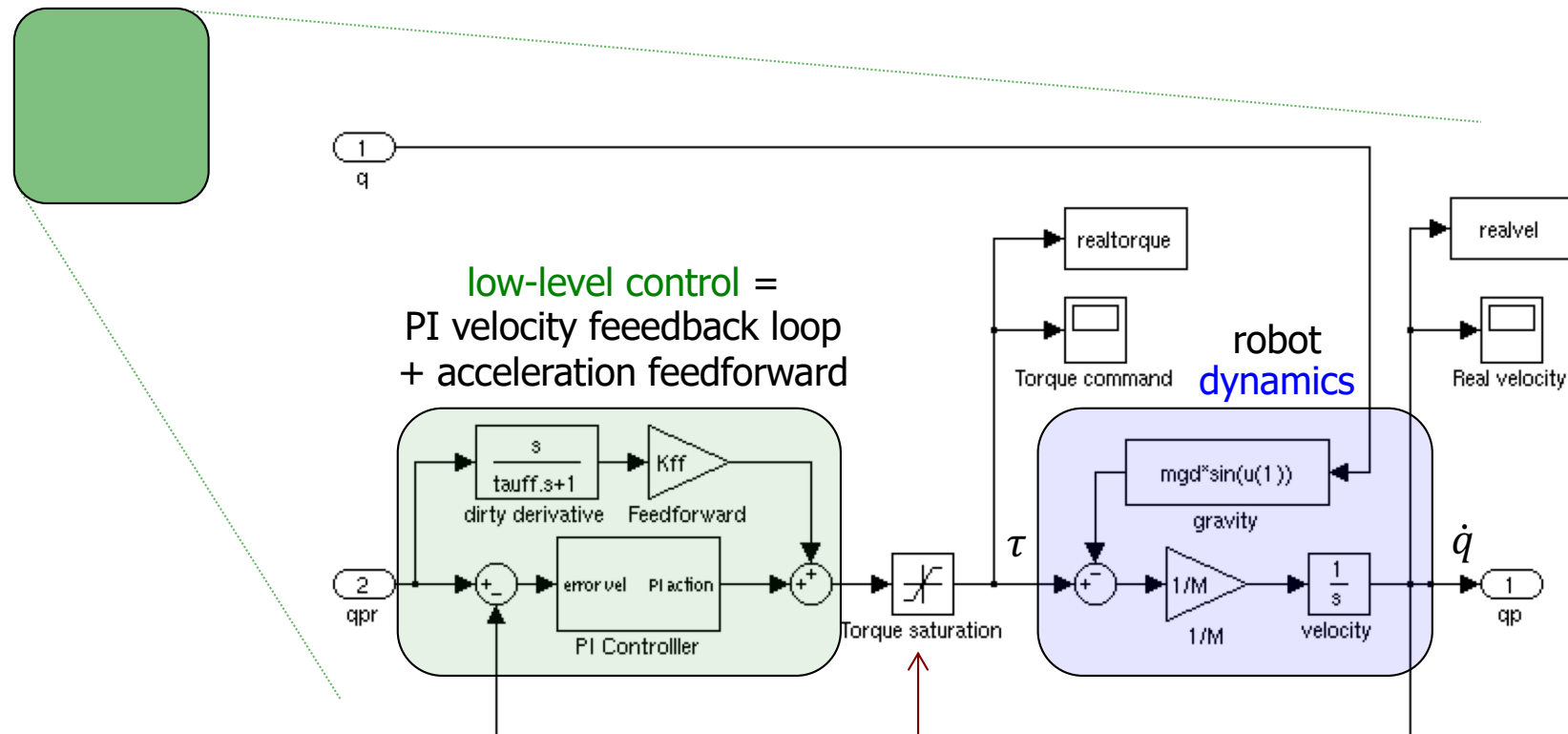
- Simulink scheme



A more detailed example

robot with low-level control

- Simulink scheme



actuator
saturation at 100 Nm

$$M\ddot{q} + mg_o d \sin q = \tau$$

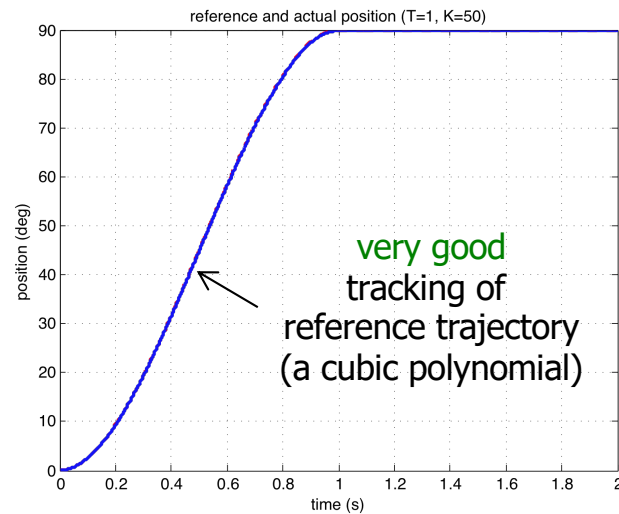
$$\ddot{q} = \frac{1}{M} (\tau - mg_o d \sin q)$$



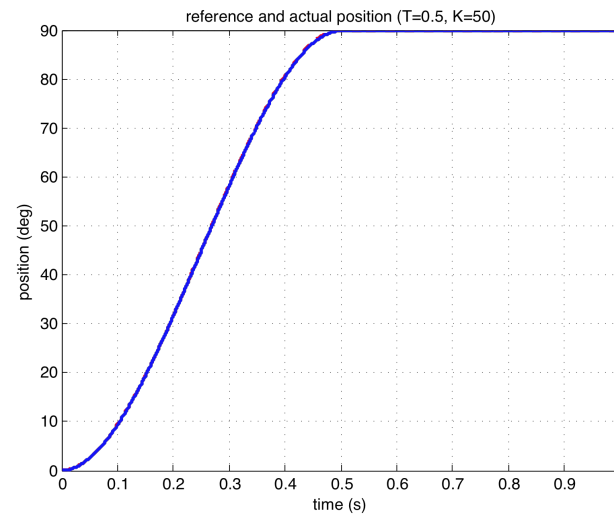
Simulation results

rest-to-rest from downward to horizontal position

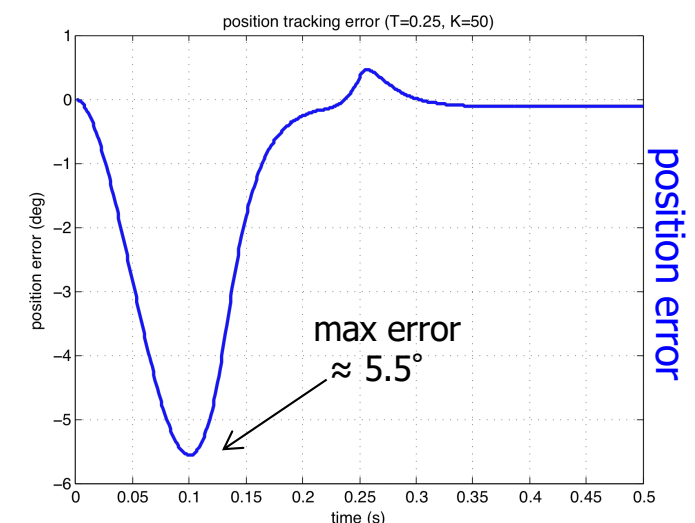
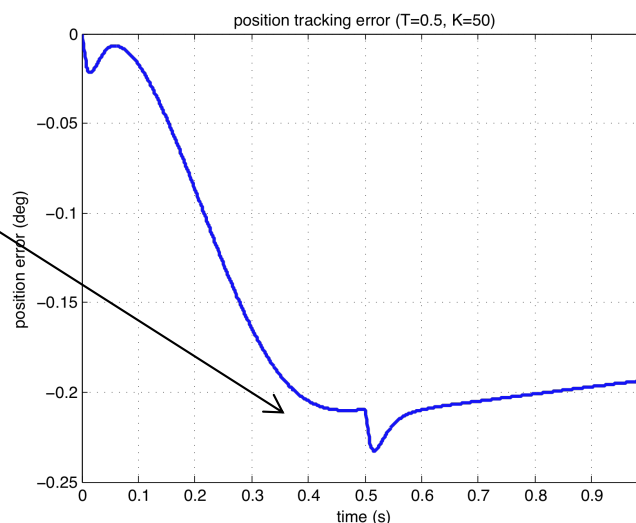
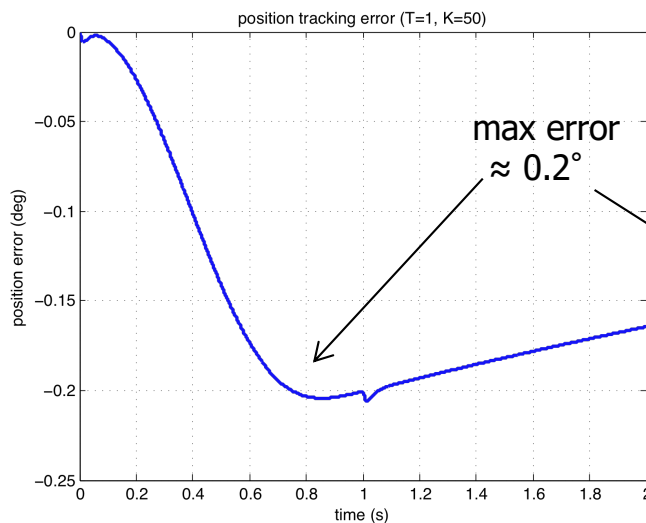
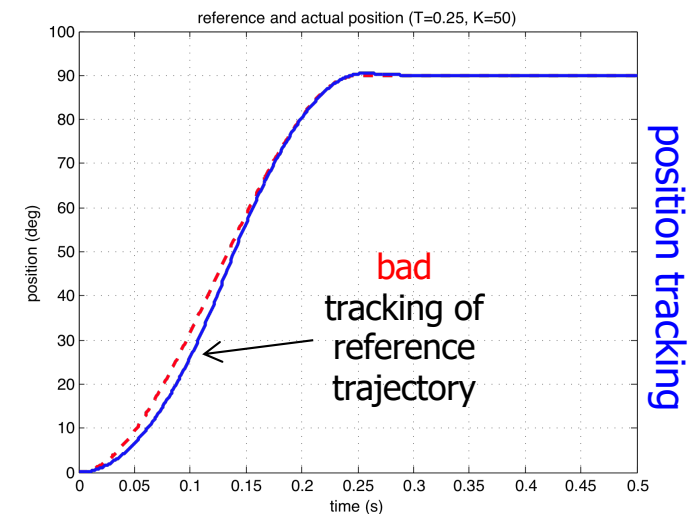
■ in $T = 1$ s



■ in $T = 0.5$ s



■ in $T = 0.25$ s

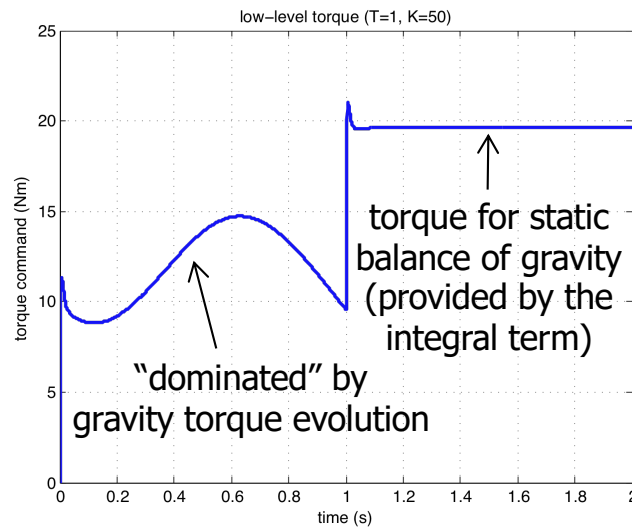
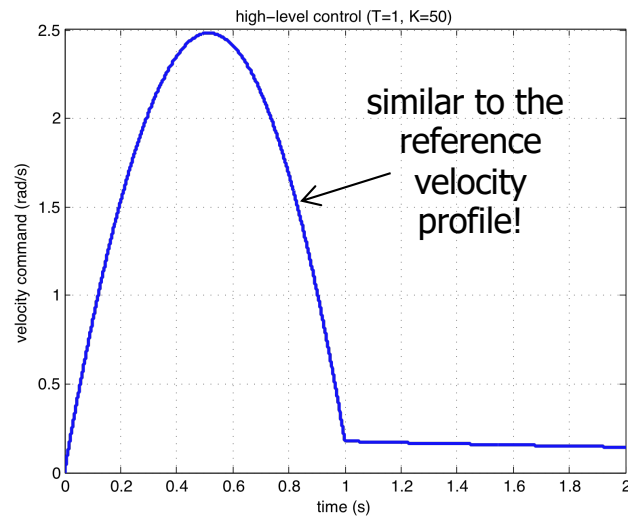




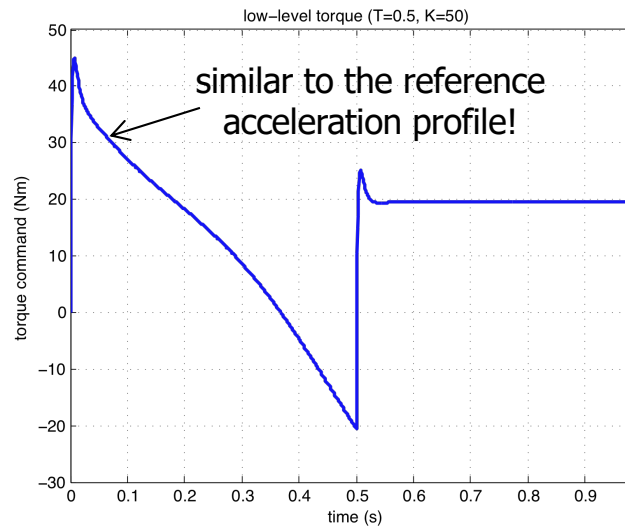
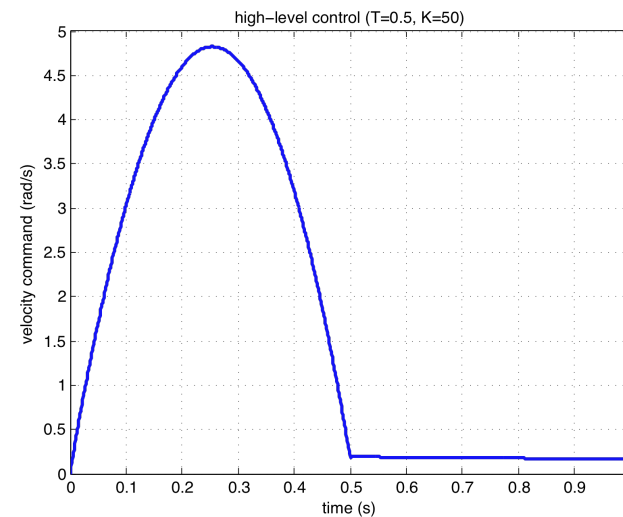
Simulation results

rest-to-rest from downward to horizontal position

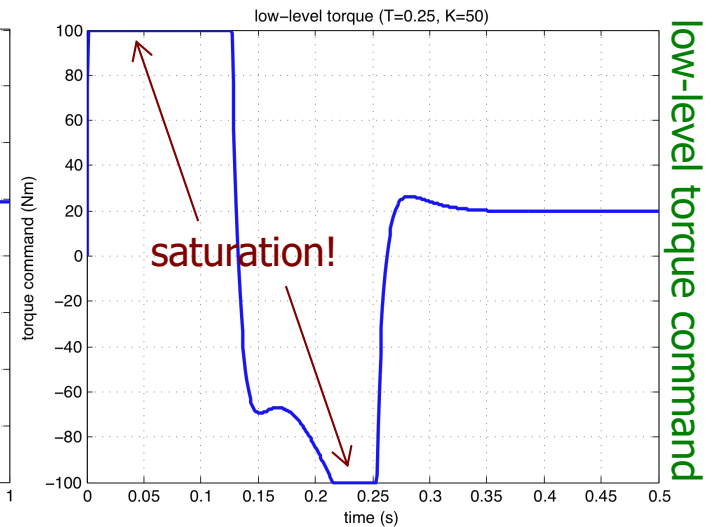
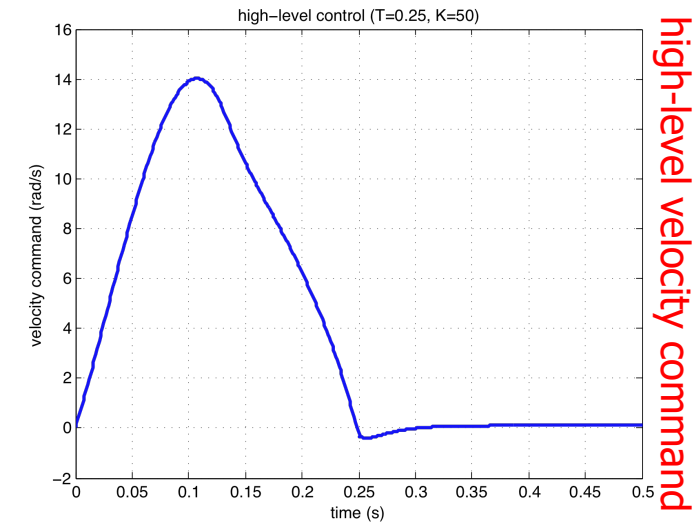
■ in $T = 1$ s



■ in $T = 0.5$ s



■ in $T = 0.25$ s

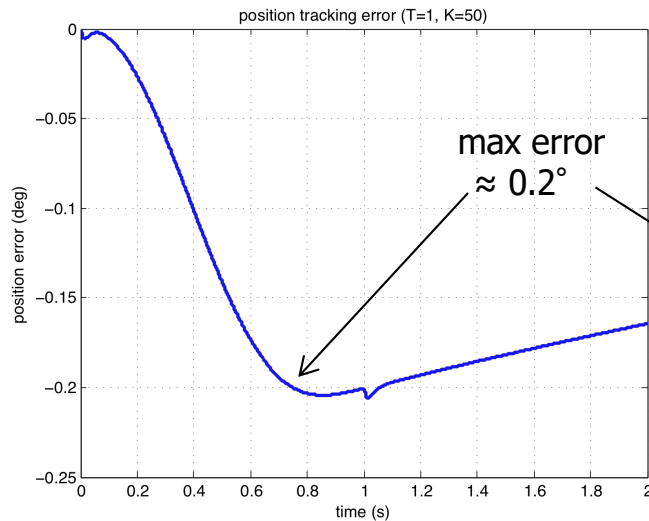




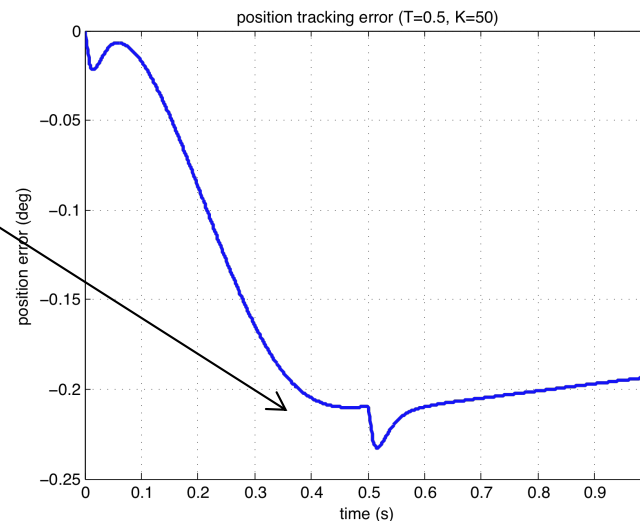
Simulation results

rest-to-rest from downward to horizontal position

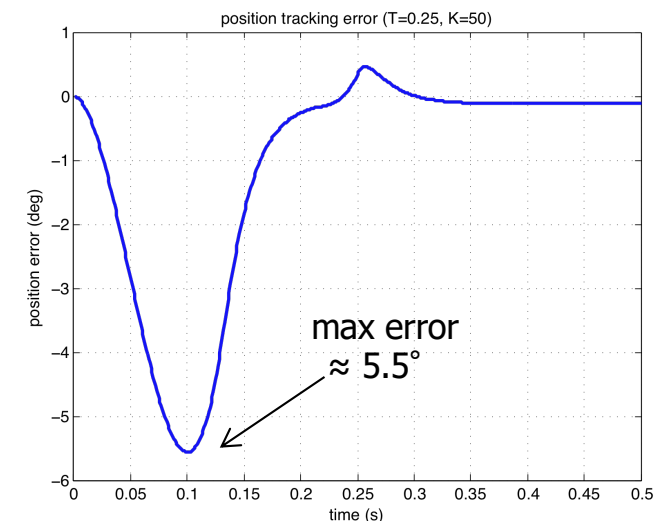
■ in $T = 1$ s



■ in $T = 0.5$ s

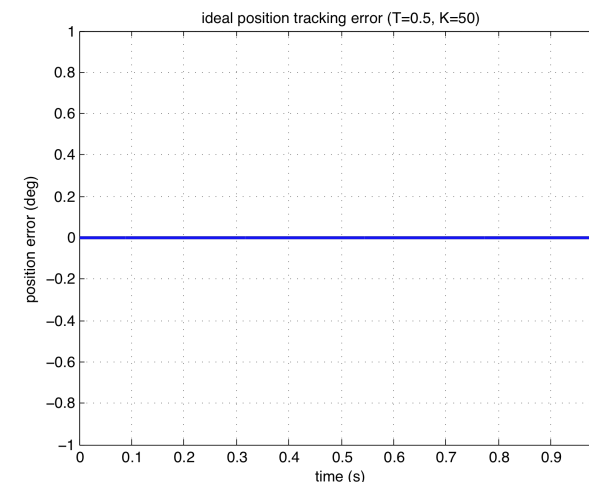


■ in $T = 0.25$ s



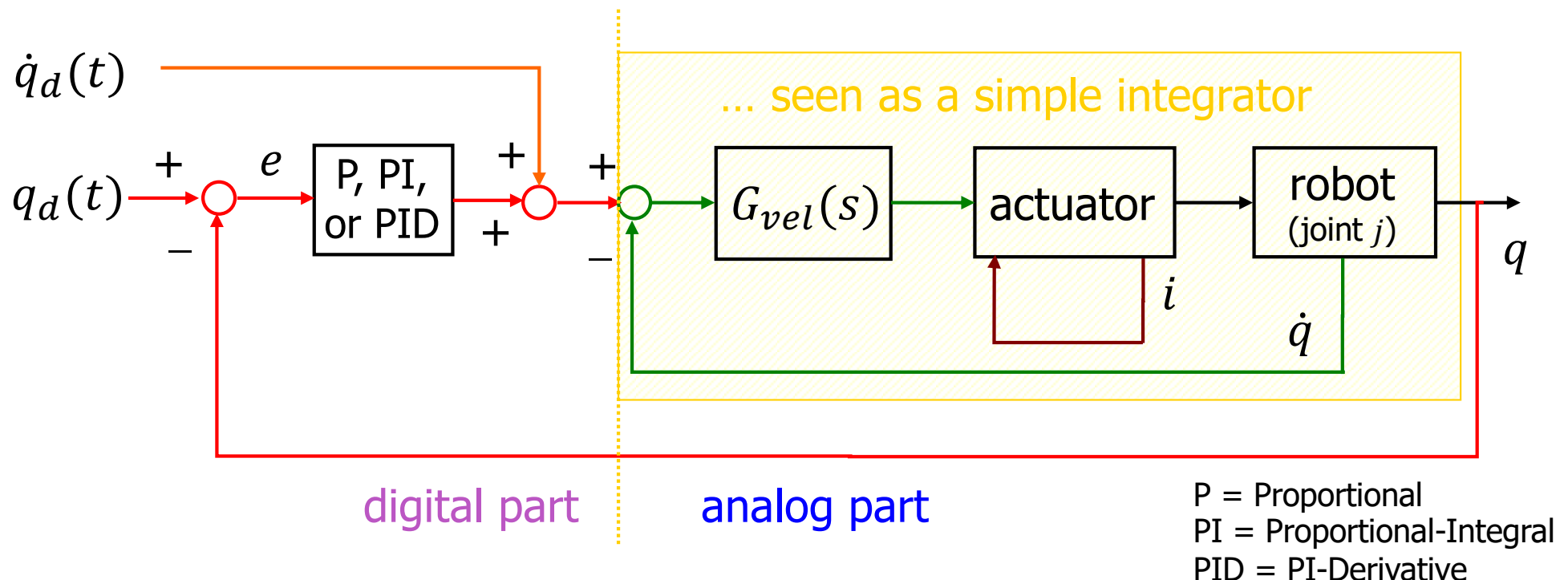
real position errors increase when reducing too much the motion time
($\Rightarrow$ acceleration is too large)

while **ideal** position errors
(based only on kinematics)
remain always the same!!
*here $\equiv 0$, thanks to the initial matching
between robot and reference trajectory*

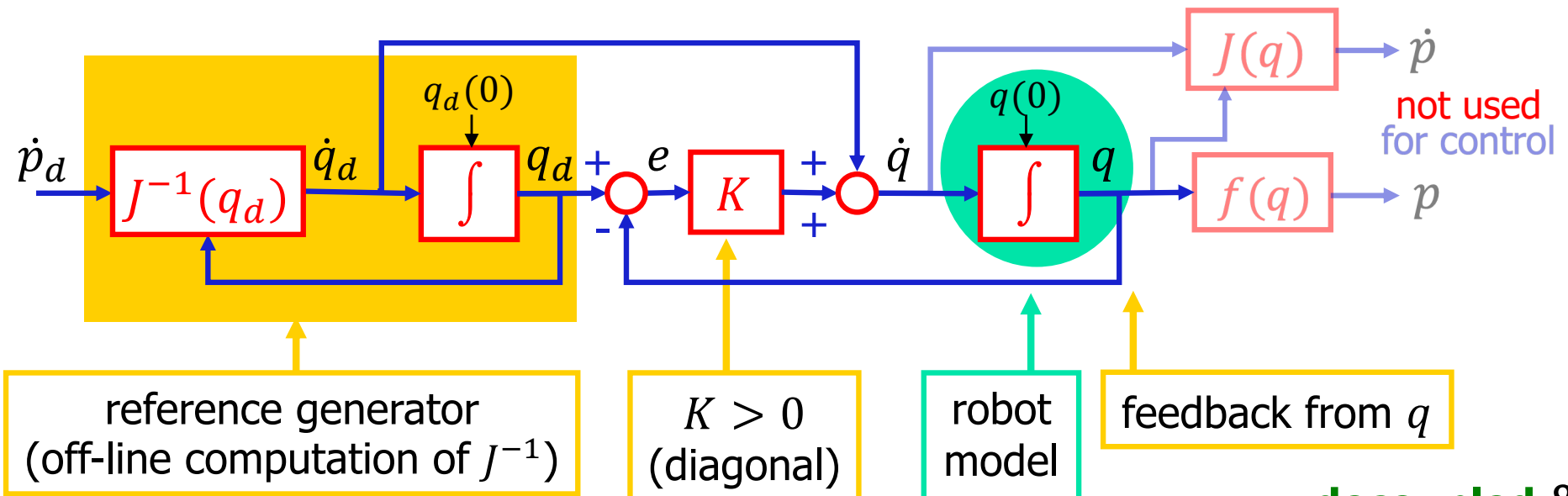


Control loops in industrial robots

- **analog** loop of large bandwidth on motor **current** ($\propto$ torque)
- **analog** loop on **velocity** ($G_{vel}(s)$, typically a PI)
- **digital feedback** loop on **position**, with **velocity feedforward**
- this scheme is local to each joint (**decentralized** control)



Kinematic control of joint motion



$$e = q_d - q \Rightarrow \dot{e} = \dot{q}_d - \dot{q} = \dot{q}_d - (\dot{q}_d + K(q_d - q)) = -Ke$$

decoupled & linear: $e_j \rightarrow 0$
 $(j = 1, \dots, n)$
exponentially,

$$e_p = p_d - p \Rightarrow \dot{e}_p = \dot{p}_d - \dot{p} = J(q_d)\dot{q}_d - J(q)(\dot{q}_d + K(q_d - q))$$

$\forall e(0)$

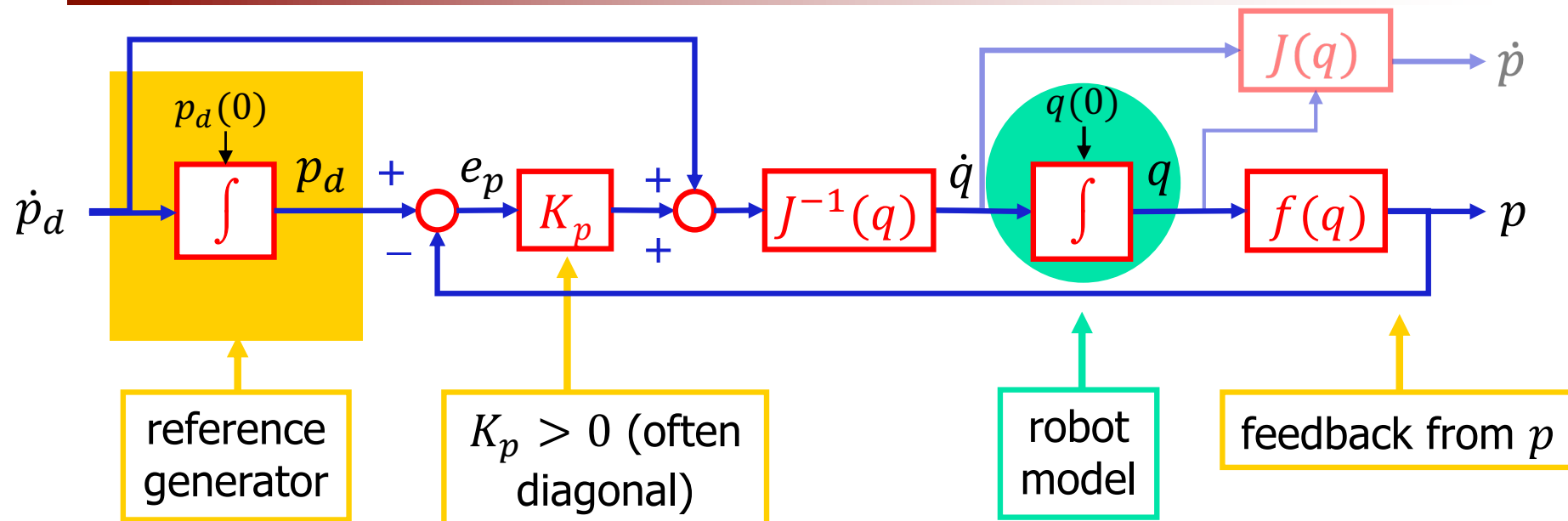
$$\begin{matrix} q \approx q_d \\ e_p \rightarrow J(q)e \end{matrix}$$

$$\Rightarrow \dot{e}_p \approx -J(q)KJ^{-1}(q)e_p$$

coupled, nonlinear
 Cartesian
 error dynamics



Kinematic control of Cartesian motion



$$e_p = p_d - p \Rightarrow \dot{e}_p = \dot{p}_d - \dot{p} = \dot{p}_d - J(q)J^{-1}(q) \left(\dot{p}_d + K_p(p_d - p) \right) = -K_p e_p$$

- **decoupled & linear:** $e_{p,i} \rightarrow 0$ ($i = 1, \dots, m$) **exponentially**, $\forall e_p(0)$
- needs on-line computation of the inverse^(*) $J^{-1}(q)$
- real-time + singularities issues

^(*) or pseudoinverse if $m < n$



Kinematic control at acceleration level

- the second-order control model is now $\ddot{q} = u$
- consider for instance the case of **Cartesian** kinematic control
- define as control law

$$u = J^{-1}(q) (\ddot{p}_d + K_d(\dot{p}_d - \dot{p}) + K_p(p_d - p) + \dot{J}(q)\dot{q})$$

with $K_p > 0, K_d > 0$ both **diagonal**

$$e_p = p_d - p \longrightarrow \dot{e}_p = \dot{p}_d - \dot{p} \longrightarrow \ddot{e}_p = \ddot{p}_d - \ddot{p}$$

second-order system:
acceleration error!

$$\begin{aligned} \longrightarrow \ddot{e}_p &= \ddot{p}_d - (J(q)\ddot{q} + \dot{J}(q)\dot{q}) = \ddot{p}_d - (J(q)u + \dot{J}(q)\dot{q}) \\ &= \ddot{p}_d - J(q)J^{-1}(q)(\ddot{p}_d + K_d(\dot{p}_d - \dot{p}) + K_p(p_d - p) + \dot{J}(q)\dot{q}) + \dot{J}(q)\dot{q} \\ &= -K_d\dot{e}_p - K_p e_p \end{aligned}$$

decoupled & **linear** 2nd-order differential equations $\dot{e}_{p,j} \rightarrow 0, e_{p,j} \rightarrow 0$ ($j = 1, \dots, m$) **exponentially**,
 $\forall e_p(0), \dot{e}_p(0)$

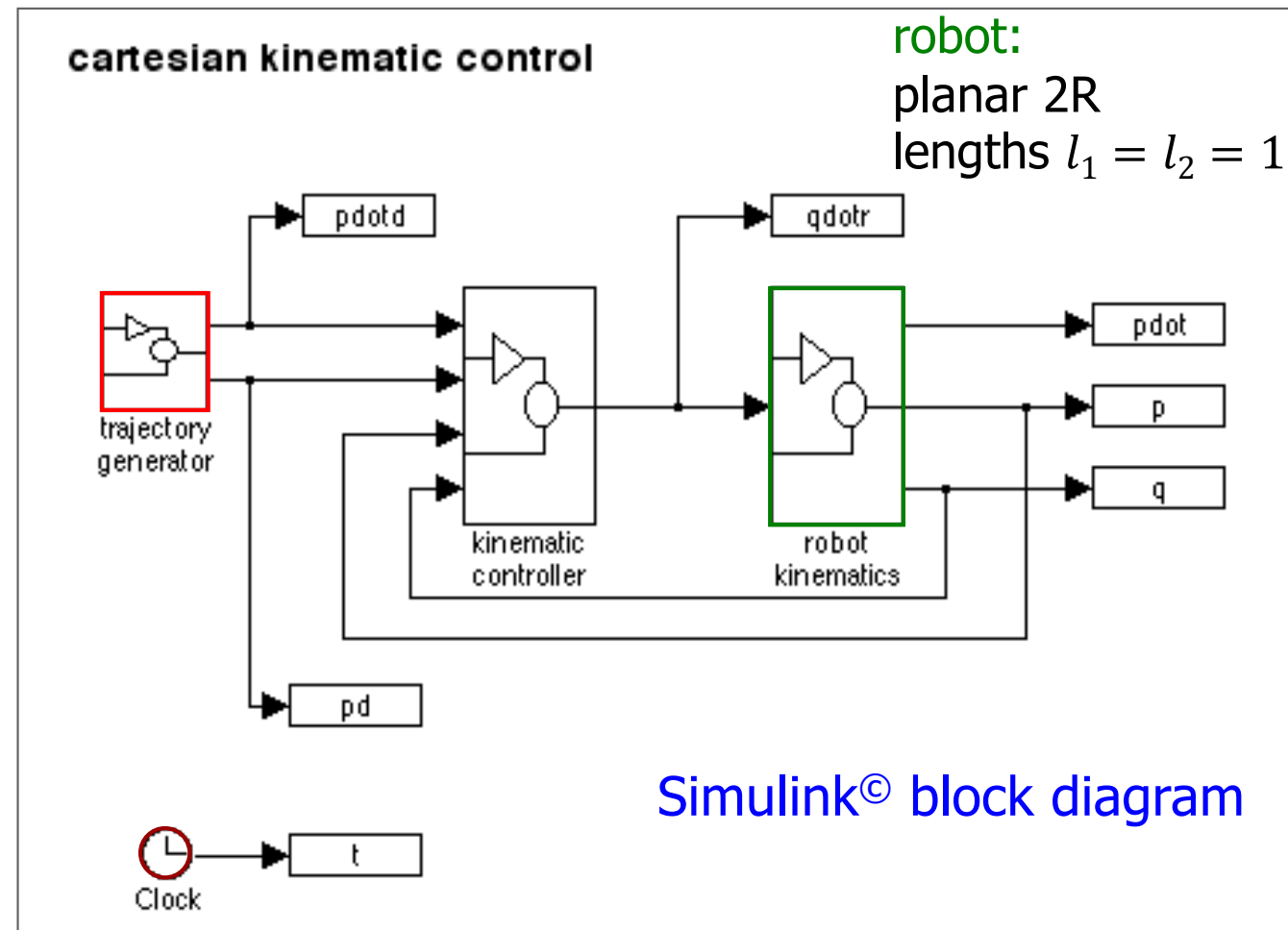


Simulation

features of kinematic control laws

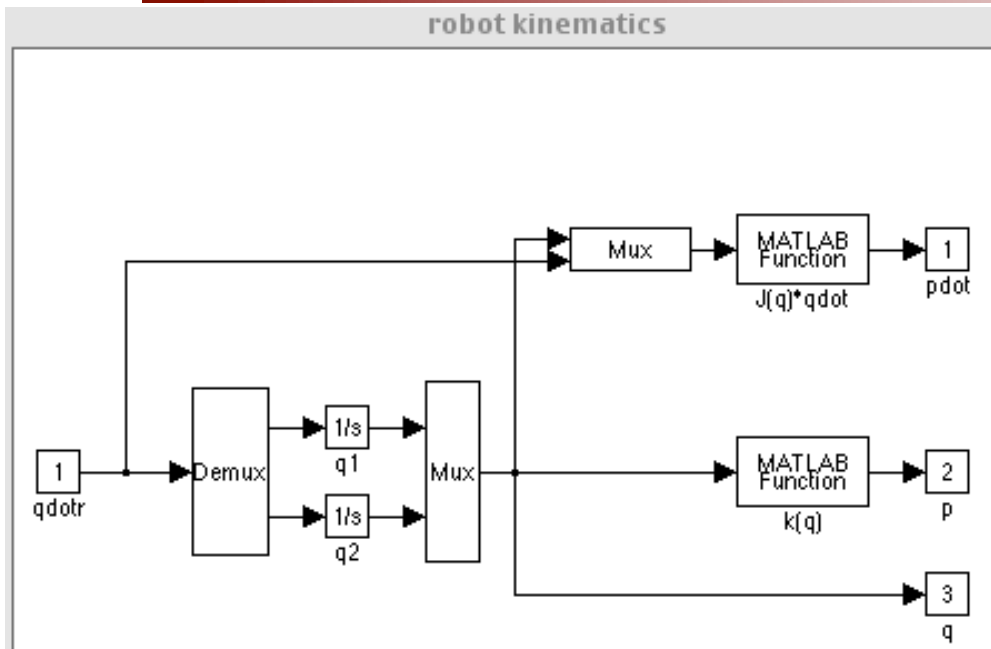
desired reference
trajectory:
two types of tasks
1. straight line
2. circular path
both with
constant speed

numerical
integration method:
fixed step
Runge-Kutta
at 1 msec





Simulink blocks



calls to Matlab functions

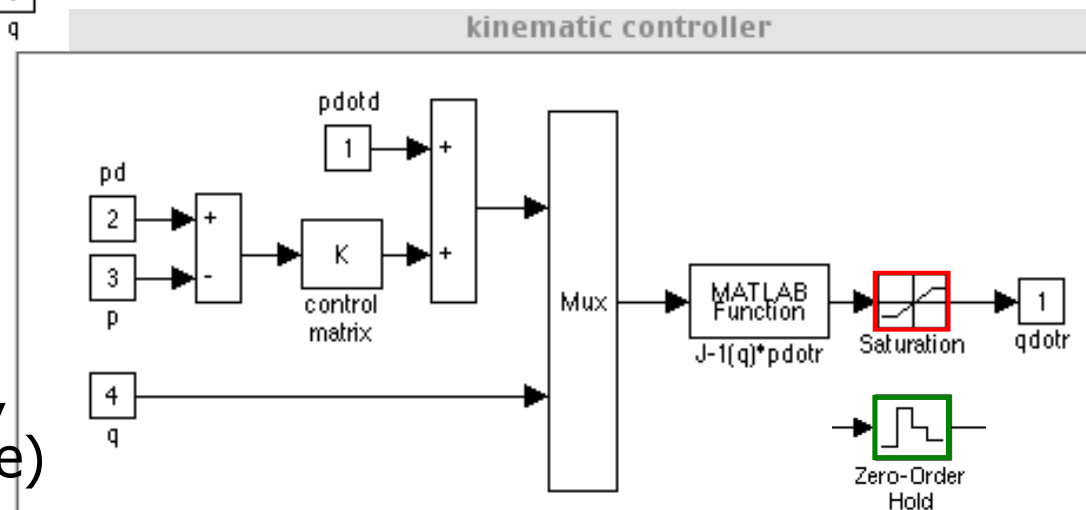
$k(q) = \text{dirkin}(\text{user})$

$J(q) = \text{jac}(\text{user})$

$J^{-1}(q) = \text{inv}(\text{jac})$ (library)

- a **saturation** (for task 1.)
or a **sample and hold** (for task 2.)
added on joint velocity commands
- **system initialization** of kinematics
data, desired trajectory, initial state,
and control parameters (in **init.m** file)

never put "numbers" inside the blocks !





Matlab functions

dirkin.m

```
function [p] = dirkin(q)

global l1 l2

px=l1*cos(q(1))+l2*cos(q(1)+q(2));
py=l1*sin(q(1))+l2*sin(q(1)+q(2));
```

jac.m

```
function [J] = jac(q)

global l1 l2

J(1,1)=-l1*sin(q(1))-l2*sin(q(1)+q(2))
J(1,2)=-l2*sin(q(1)+q(2));
J(2,1)=l1*cos(q(1))+l2*cos(q(1)+q(2));
J(2,2)=l2*cos(q(1)+q(2));
```

init.m

```
% controllo cartesiano di un robot 2R
% initialization

clear all; close all
global l1 l2

% lunghezze bracci robot 2R

l1=1; l2=1;

% velocità cartesiana desiderata (costante)

vxd=0; vyd=0.5;

% tempo totale

T=2;

% configurazione desiderata iniziale

q1d0=-45*pi/180; q2d0=135*pi/180;

pd0=dirkin([q1d0 q2d0]');
pxd0=pd0(1); pyd0=pd0(2);

% configurazione attuale del robot

q10=-45*pi/180; q20=90*pi/180;

p0=dirkin([q10 q20]');

% matrice dei guadagni cartesiani

K=[20 20]; K=diag(K);

% saturazioni di velocità ai giunti (input in deg/sec, convertito in rad/sec)

vmax1=120*pi/180; vmax2=90*pi/180;
```

init.m
script
(for task 1.)



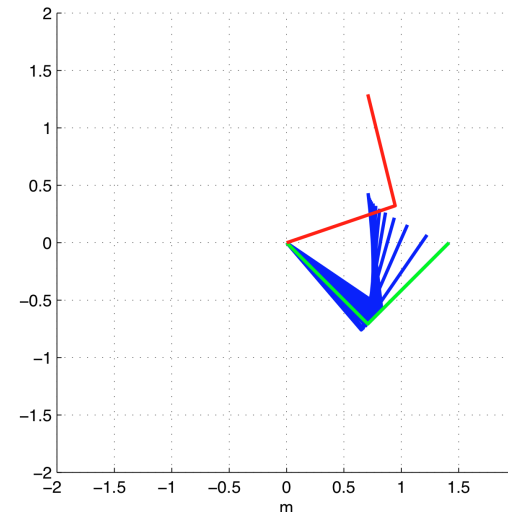
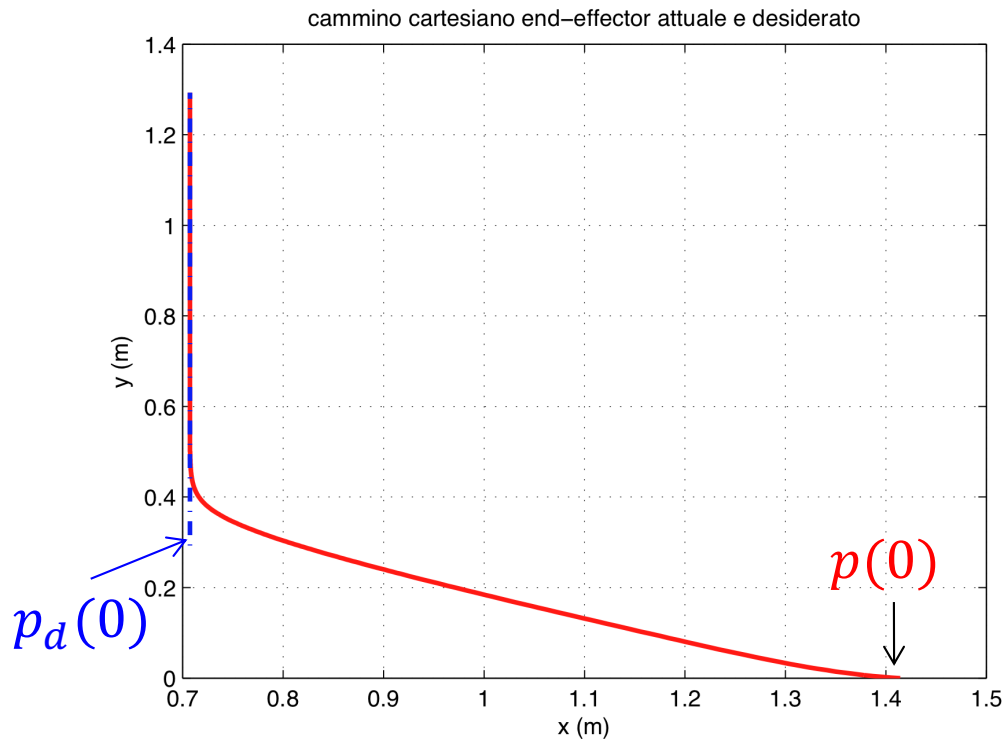
Simulation data for task 1

- **straight line** path with constant velocity
 - $x_d(0) = 0.7 \text{ m}$, $y_d(0) = 0.3 \text{ m}$; $v_{d,y} = 0.5 \text{ m/s}$, for $T = 2 \text{ s}$
- **large initial error** on end-effector position
 - $q(0) = (-45^\circ, 90^\circ) \Rightarrow e_p(0) = (-0.7, 0.3) \text{ m}$
- Cartesian control gains
 - $K_p = \text{diag}\{20, 20\}$
- (a) **without** joint velocity command saturation
- (b) **with** saturation $|\dot{q}_j| \leq v_{\max,j}$, $j = 1, 2$:
 - $v_{\max,1} = 120^\circ/\text{s}$, $v_{\max,2} = 90^\circ/\text{s}$



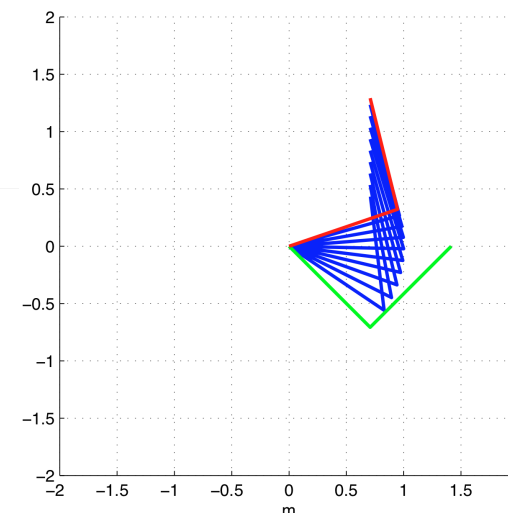
Results for task 1a

straight line: initial error, **no** saturation



initial
transient
phase
(about 0.2 s)

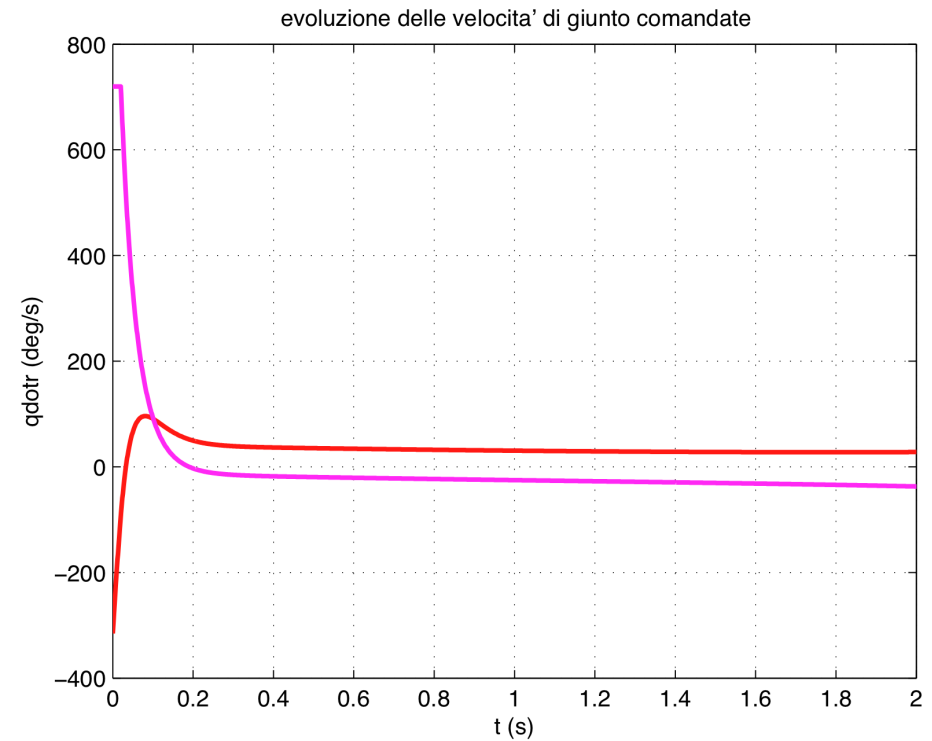
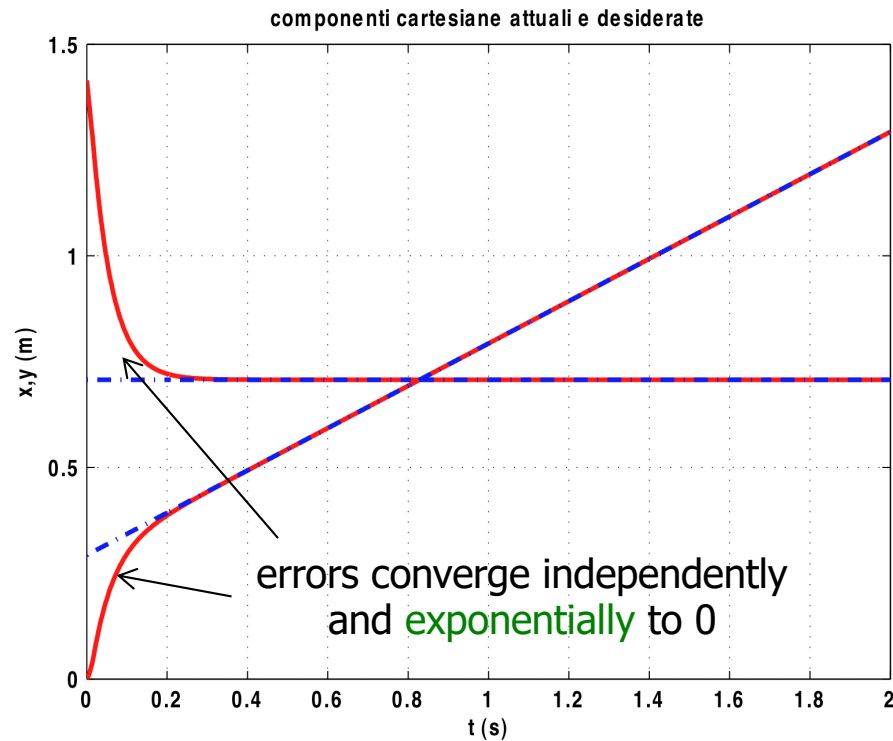
stroboscopic view of motion
(**start** and **end** configurations)



trajectory
following
phase
(about 1.8 s)

Results for task 1a

straight line: initial error, **no** saturation



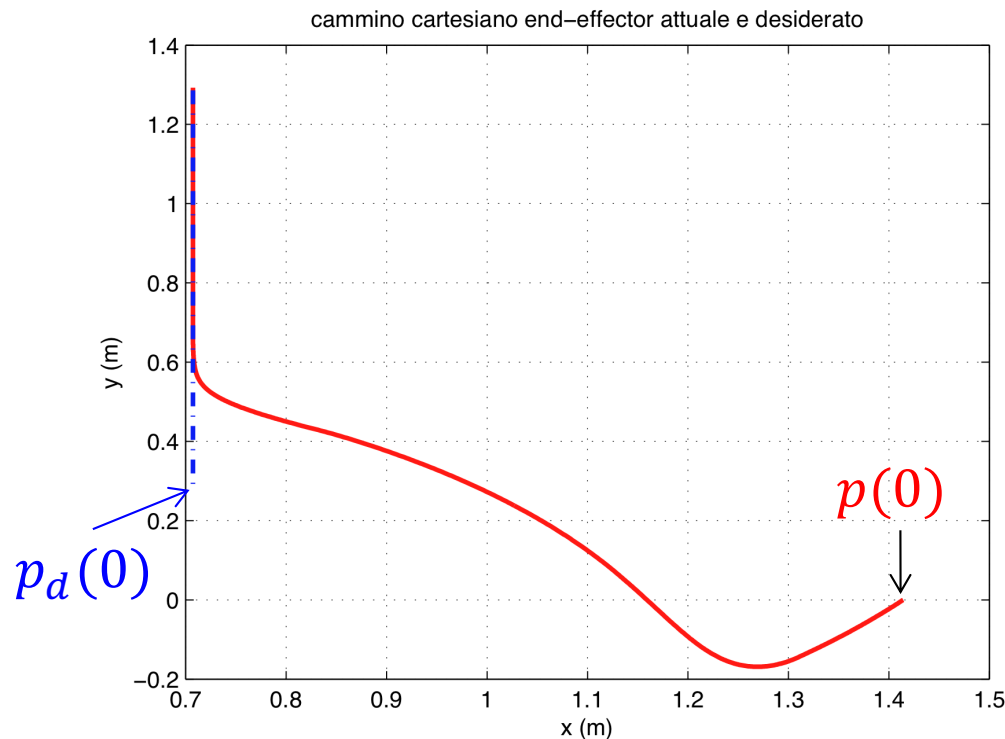
p_x, p_y actual and desired

control inputs $\dot{q}_{r1}, \dot{q}_{r2}$

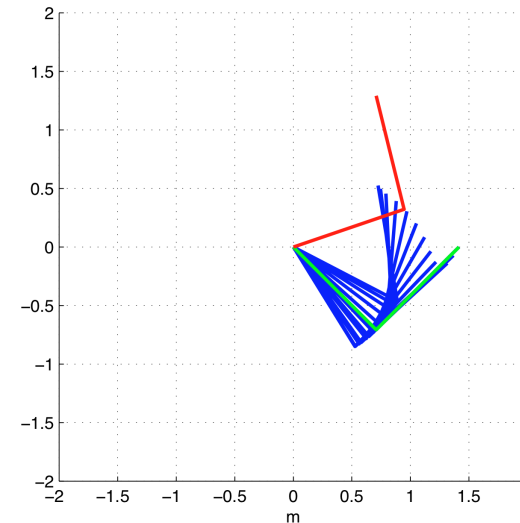


Results for task 1b

straight line: initial error, **with** saturation

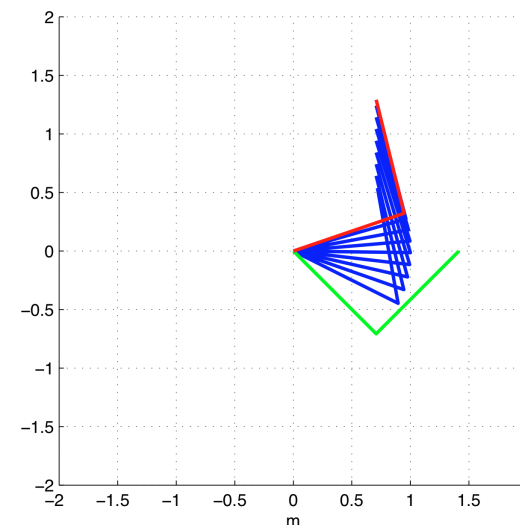


path executed by the
robot end-effector
(**actual** and **desired**)



initial
transient
phase
(about 0.5 s)

stroboscopic view of motion
(**start** and **end** configurations)

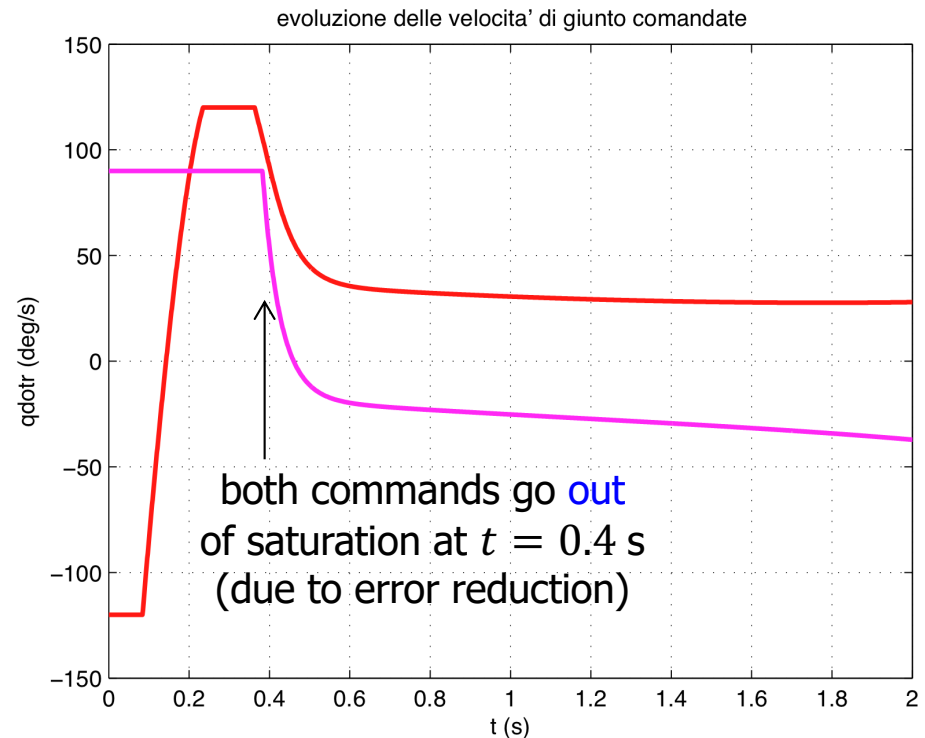
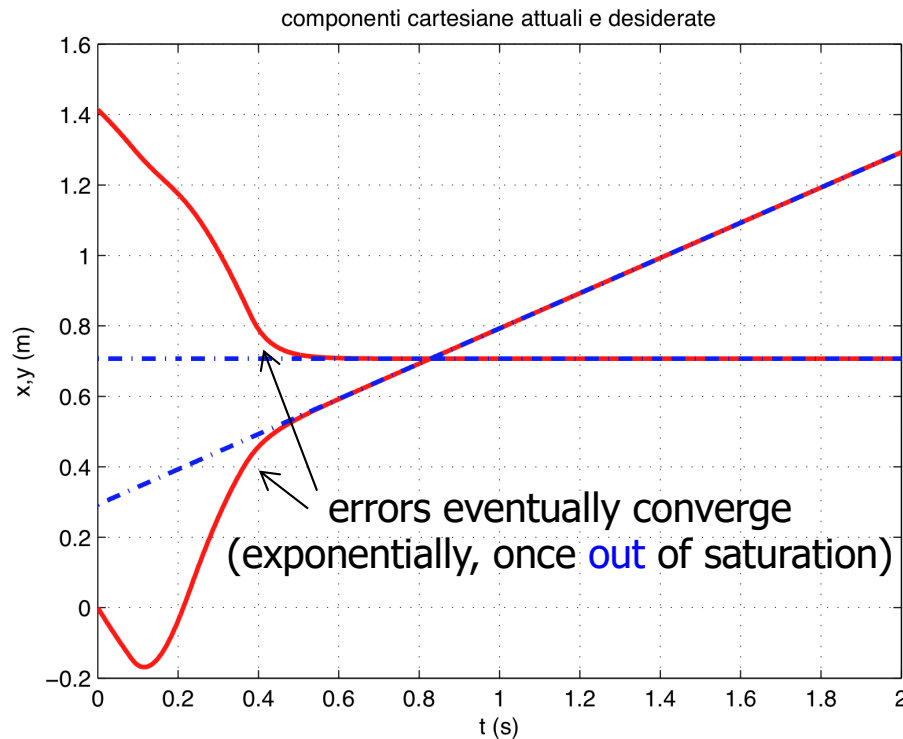


trajectory
following
phase
(about 1.5 s)



Results for task 1b

straight line: initial error, **with** saturation



p_x, p_y **actual** and **desired**

control inputs $\dot{q}_{r1}, \dot{q}_{r2}$
(saturated at $\pm v_{\max,1}, \pm v_{\max,2}$)



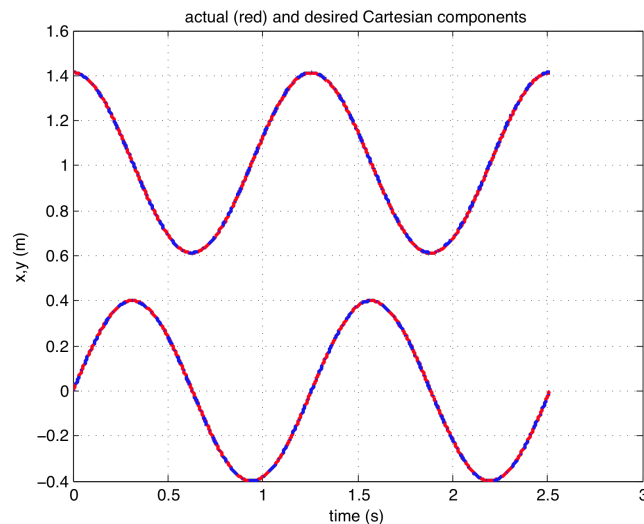
Simulation data for task 2

- **circular** path with constant velocity
 - centered at $(1.014, 0)$ with radius $R = 0.4$ m;
 - $v = 2$ m/s, performing **two** rounds $\Rightarrow T \approx 2.5$ s
- **zero initial error** on Cartesian position ("match")
 - $q(0) = (-45^\circ, 90^\circ) \Rightarrow e_p(0) = 0$
- (a) ideal **continuous** case (**1 kHz**), even **without** feedback
- (b) **with** sample and hold (ZOH) of $T_{\text{hold}} = 0.02$ s (joint velocity command updated at **50 Hz**), but **without** feedback
- (c) as before, but **with** Cartesian feedback using the gains
 - $K_p = \text{diag}\{25, 25\}$

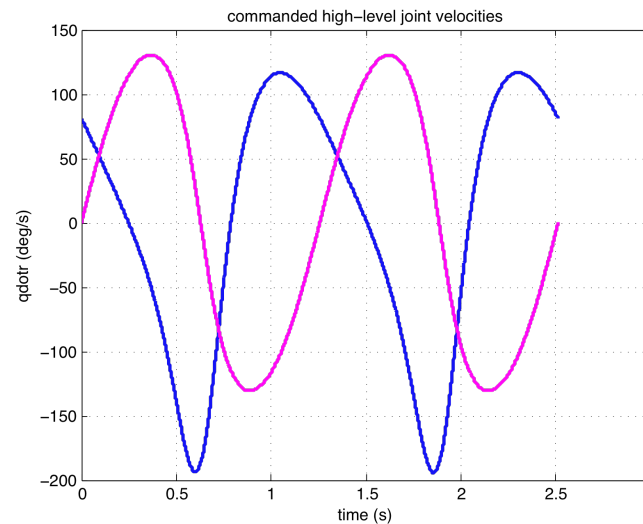


Results for task 2a

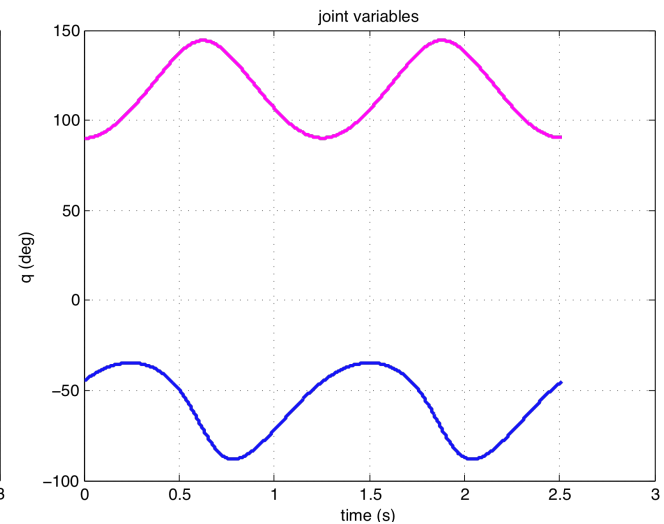
circular path: no initial error, **continuous** control (ideal case)



p_x, p_y **actual** and **desired**

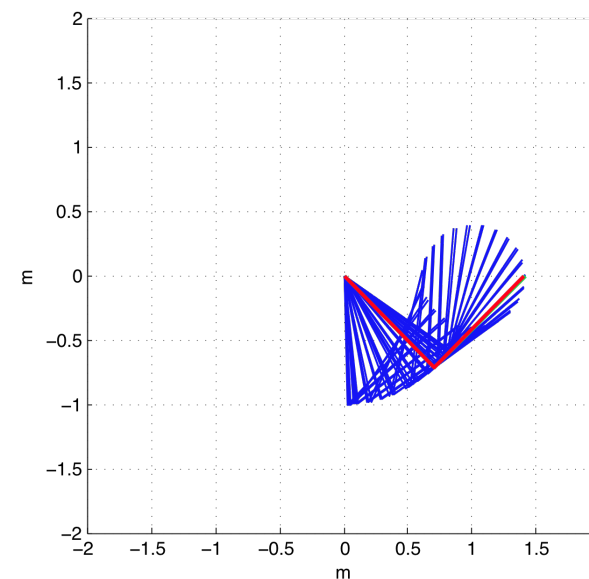
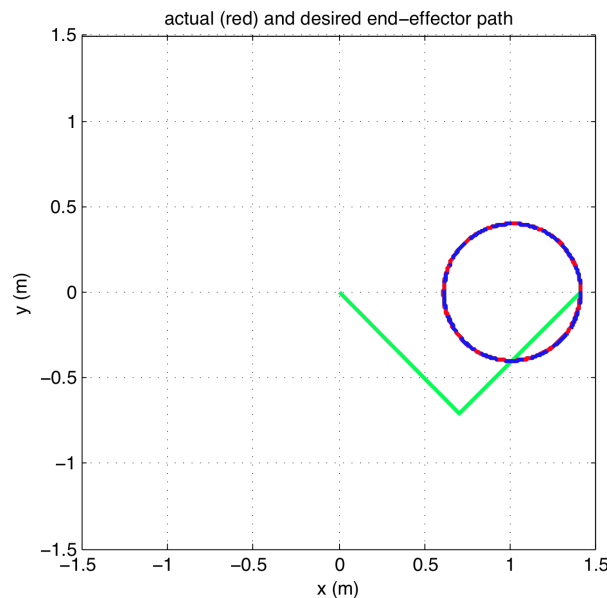


control inputs $\dot{q}_{r1}, \dot{q}_{r2}$



joint variables q_1, q_2

zero tracking
error is kept
at all times

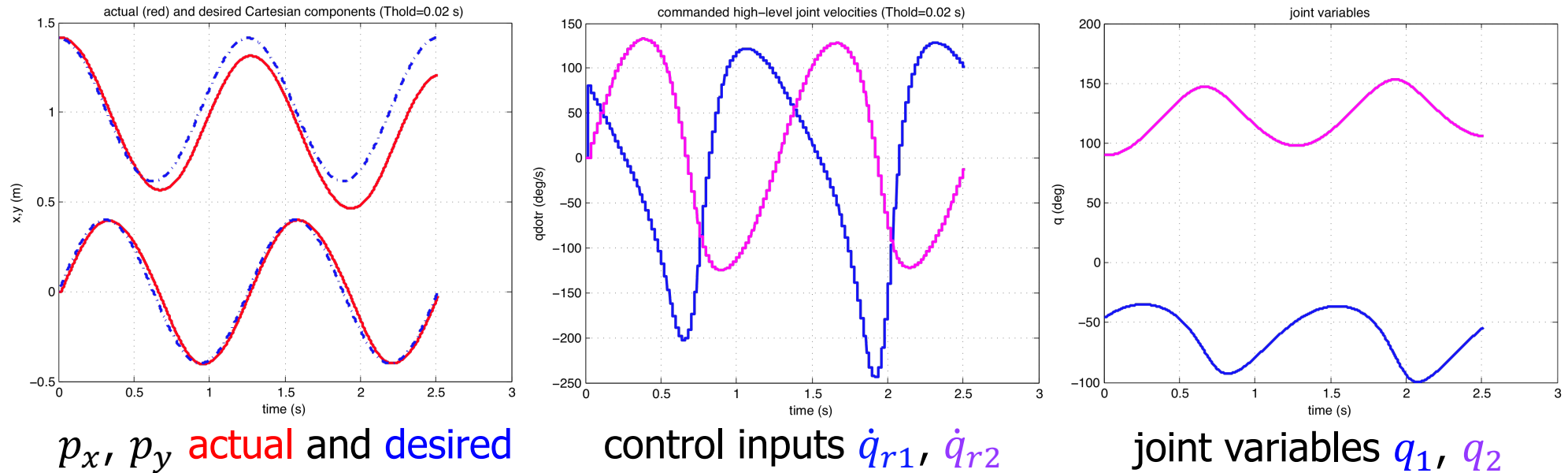


final configuration
(after two rounds)
coincides with
initial configuration

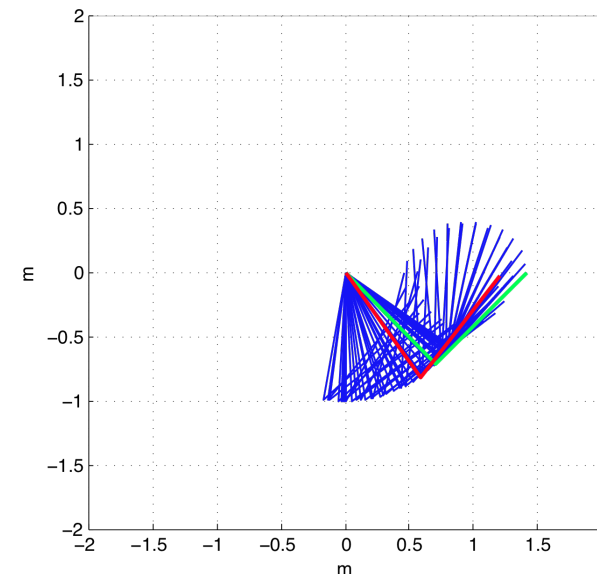
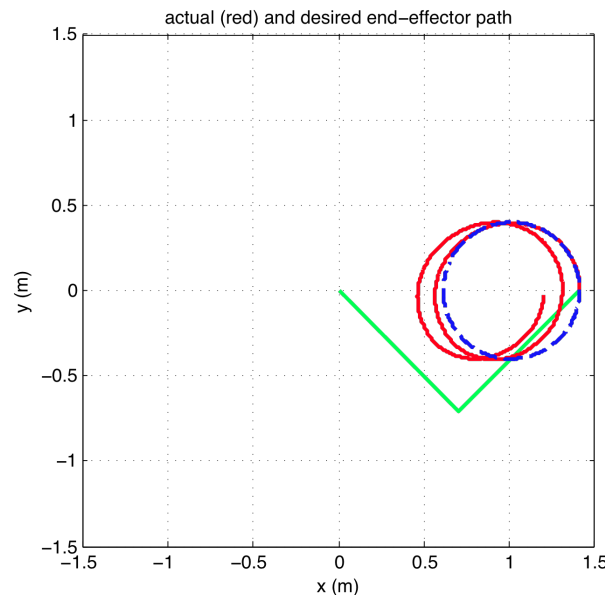


Results for task 2b

circular path: no initial error, **ZOH** at 50 Hz, **no** feedback



a **drift** occurs along the path due to the "linearization error" along the path tangent

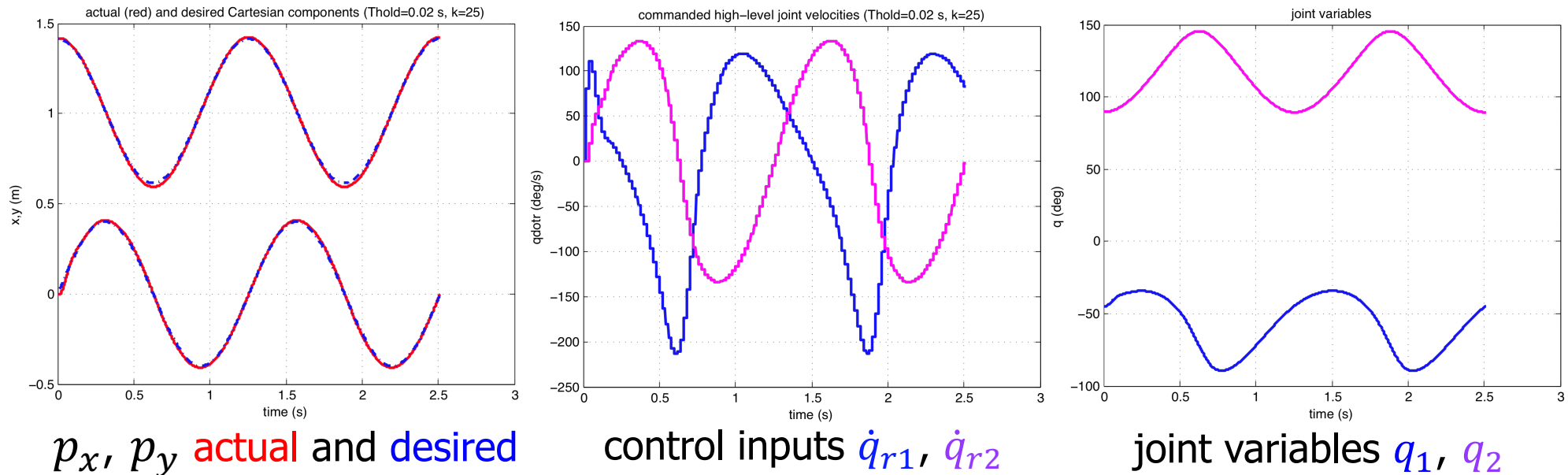


final configuration (after **two** rounds) differs from **initial** configuration

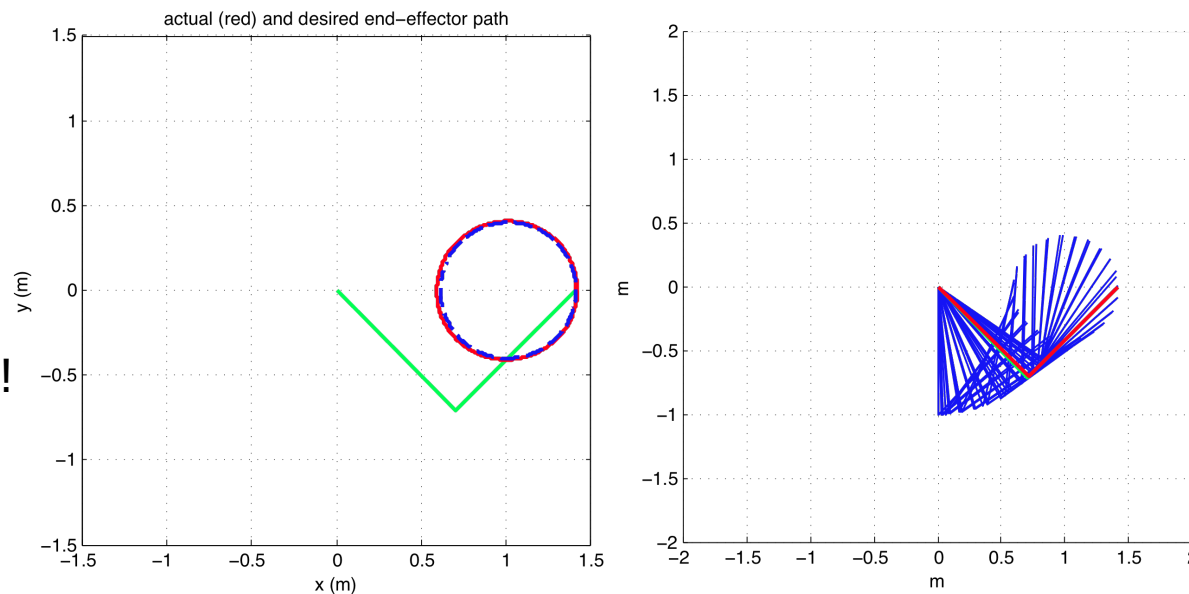


Results for task 2c

circular path: no initial error, **ZOH** at 50 Hz, **with** feedback



(almost) same performance of the continuous case is recovered!!

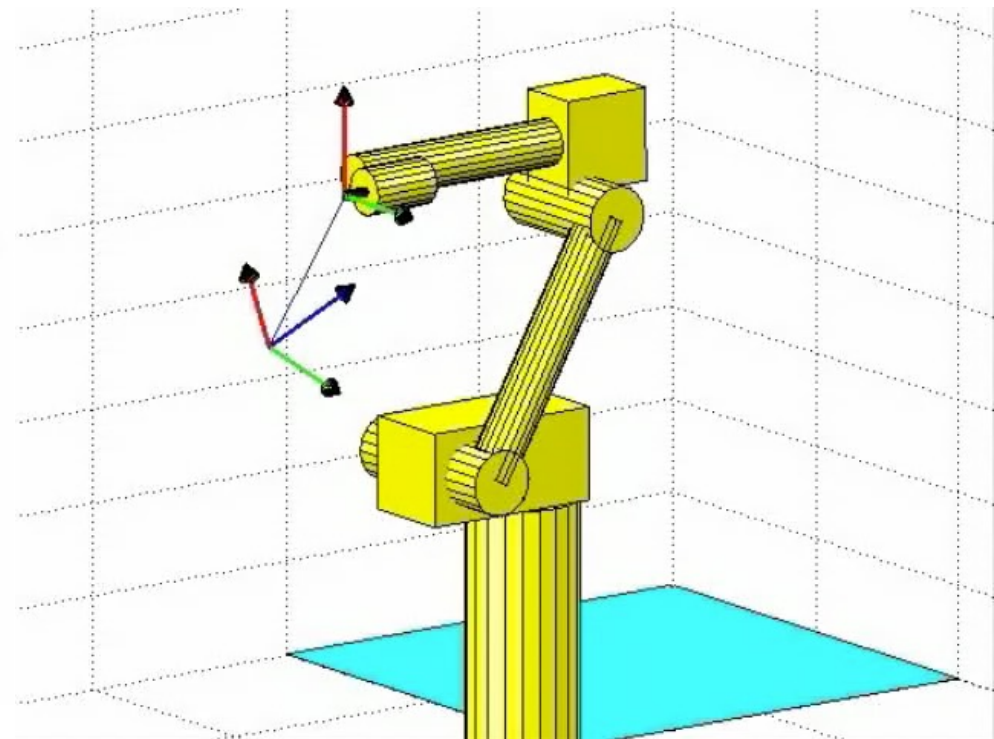
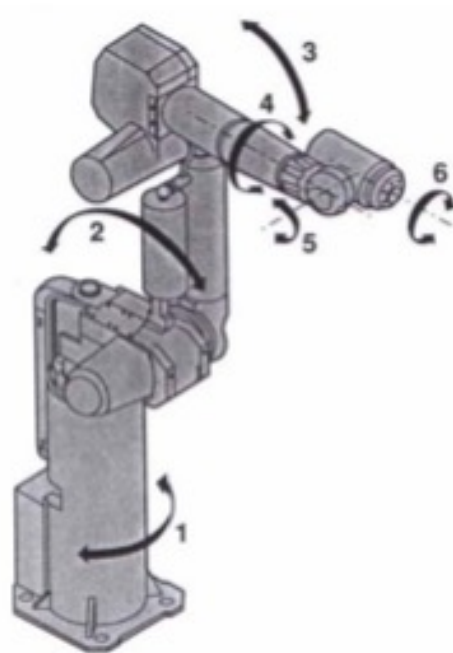


larger **P** gains will eventually lead to **unstable** behavior (stability limits for **discrete-time** implementations of a continuous control law)

3D simulation

why is final convergence so slow here?

video



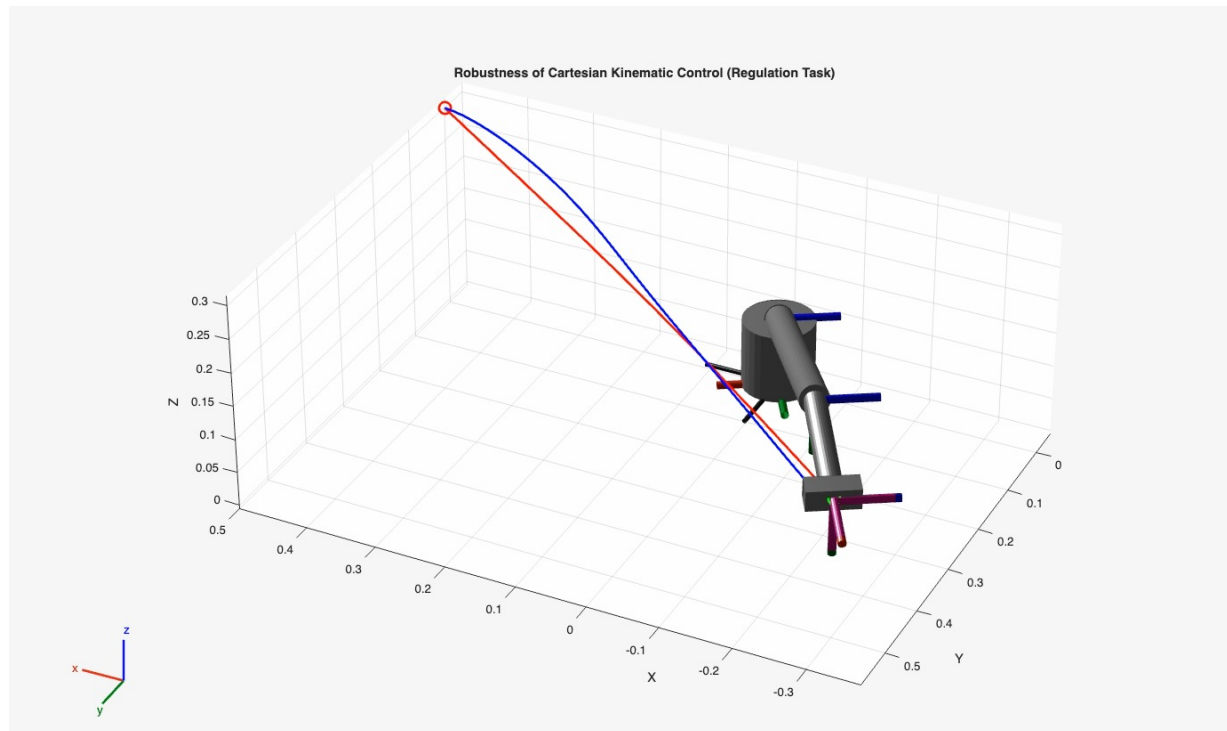
kinematic control of Cartesian motion of Fanuc 6R (Arc Mate S-5) robot
simulation and visualization in Matlab



Robust Cartesian kinematic control

- **regulation** task to constant p_d with **uncertain** Jacobian, using an external measure of Cartesian position: $\dot{q} = \hat{J}^{-1}(q)K_p(p_d - p) = \hat{J}^{-1}(q)K_p e_p$
- 3R elbow-type robot with link lengths: $l_1 = 0.1, l_2 = 0.4, l_3 = 0.2$
- **estimated** lengths: $\hat{l}_1 = l_1, \hat{l}_2 = 0.45 (+12.5\%), \hat{l}_3 = 0.35 (+75\%)$
- control gains: $K_p = k_p \cdot I_{3 \times 3}, k_p = 30; e_p(0) = p_d - p(0) = (-0.8, 0.5, -0.1)$

— nominal motion
— actual motion





Robust Cartesian kinematic control

$$V = \frac{1}{2} \|e_p\|^2 \geq 0 \quad \dot{e}_p = -\dot{p} = -J(q)\dot{q} = -J(q)\hat{J}^{-1}(q)K_p e_p = -k_p A(q)e_p$$

$$\begin{aligned} \dot{V} &= e_p^T \dot{e}_p = -k_p e_p^T A(q) e_p = -k_p e_p^T \frac{A(q) + A^T(q)}{2} e_p \\ &= -k_p e_p^T A_s(q) e_p \leq -k_p \lambda_{\min}(A_s(q)) \|e_p\|^2 \leq 0 \end{aligned}$$

a **sufficient** condition for convergence is $\lambda_{\min}(A_s) > 0$ over the whole path

