



Robotics 2

Introduction

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Robotics 2 – 2023-24

- **II semester** February 26 – May 29, 2024
- **master courses** Artificial Intelligence and Robotics & Control Engineering
- **credits 6** = 150h (1 ECTS = 25h of student work) **with a combination of**
 - regular lectures in the classroom, including
 - questions & answers (Q&A) on material covered in video lectures
 - exercises (on blackboard) + midterm test
 - video lectures recorded in 2019-20, available on YouTube in the **Robotics 2 playlist** of the **Video DIAG – Sapienza** channel
 - individual study (~70h)
- **schedule** Monday (8:00-10:00) - Wednesday (14:00-18:00), room **B2**
- **G-group** https://groups.google.com/a/diag.uniroma1.it/g/robotics2_2023-24



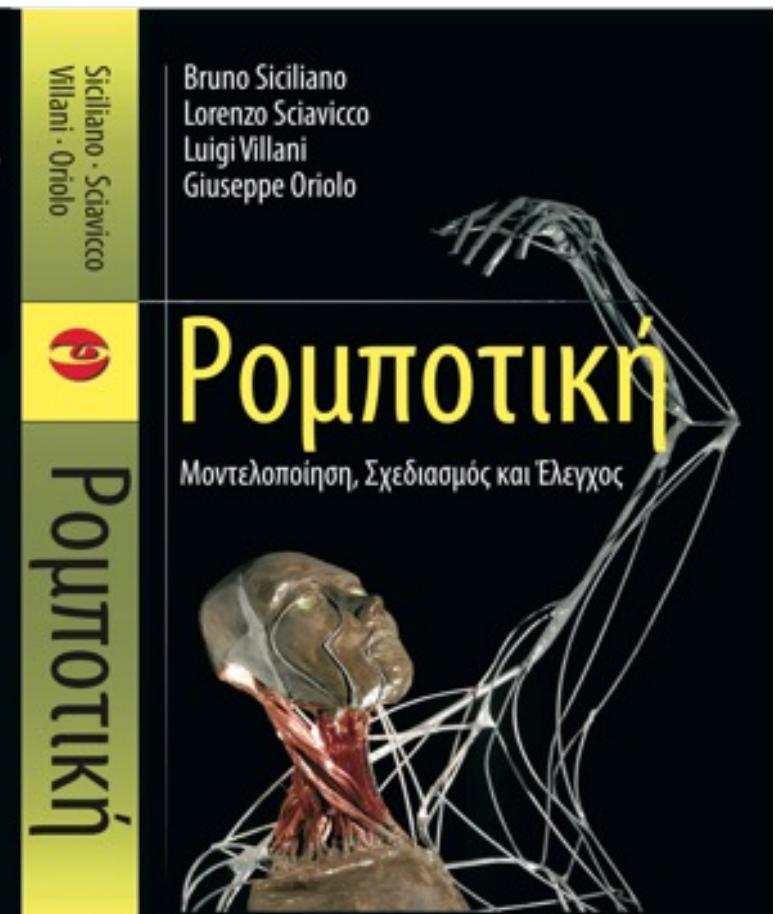
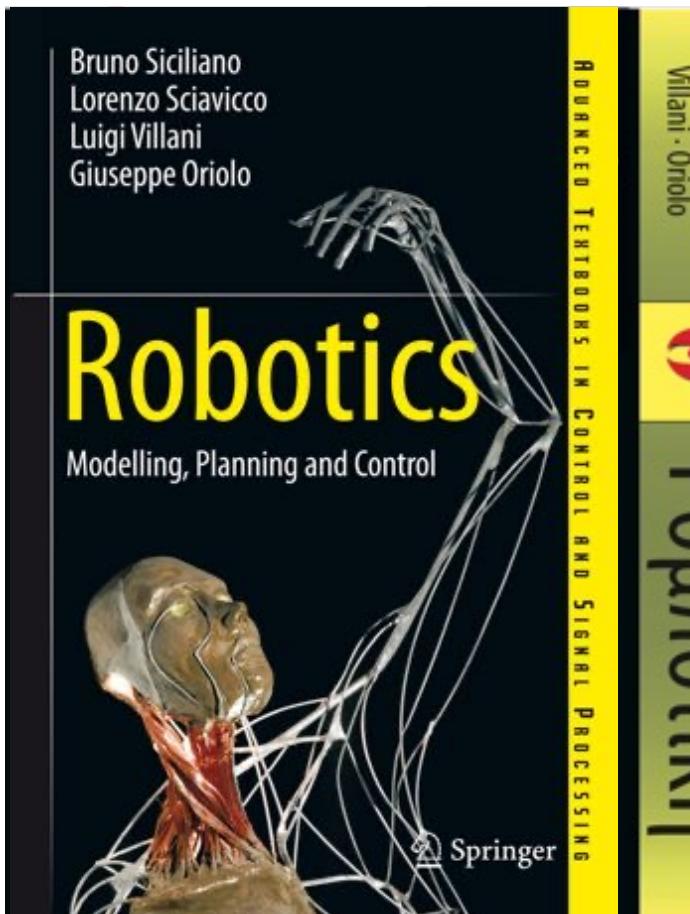
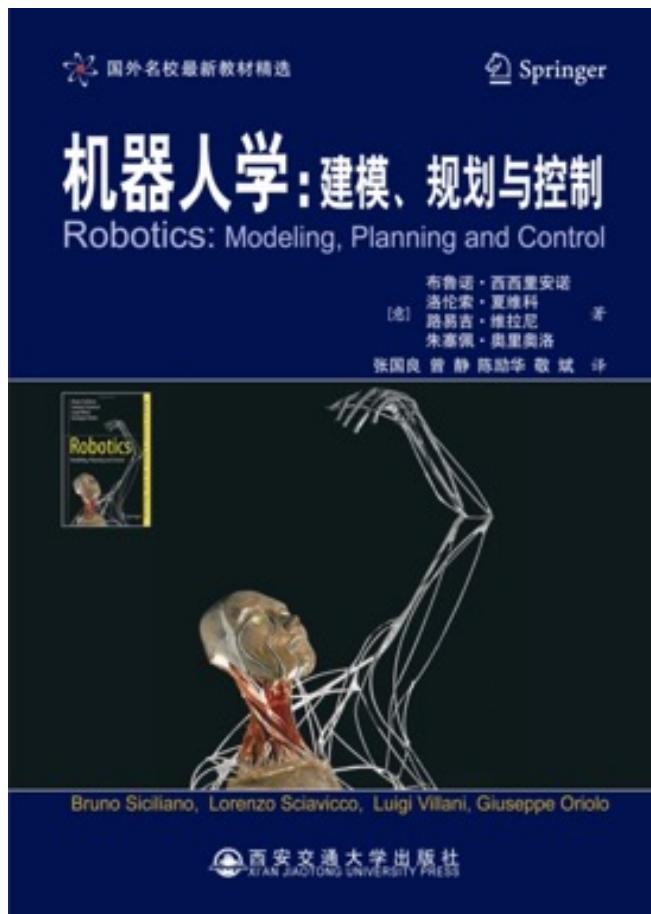
General information

- **prerequisites**
 - Robotics 1 as a prerequisite (**mandatory** for the exam)
- **aims**
 - advanced kinematics & dynamic analysis of robot manipulators
 - design of feedback control laws for **free motion** and **interaction tasks**
- **textbook**
 - B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo: *Robotics: Modelling, Planning and Control*, 3rd Edition, Springer, 2009
- **related courses**
 - Autonomous and Mobile Robotics 1st semester of year 2, 6 credits
 - Elective in Robotics whole year 2, 12 credits (four modules)
or Control Problems in Robotics 6 credits (two out of four modules)
 - Probabilistic Robotics 1st semester of year 2, 6 credits
 - Medical Robotics 2nd semester of year 2, 6 credits



An international textbook

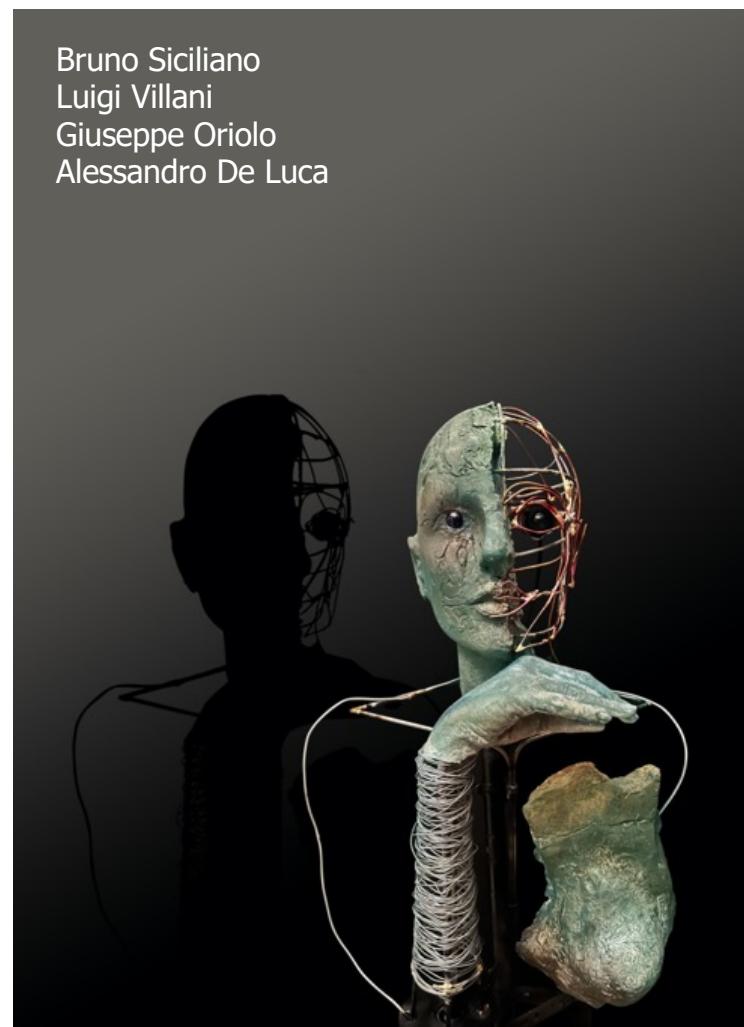
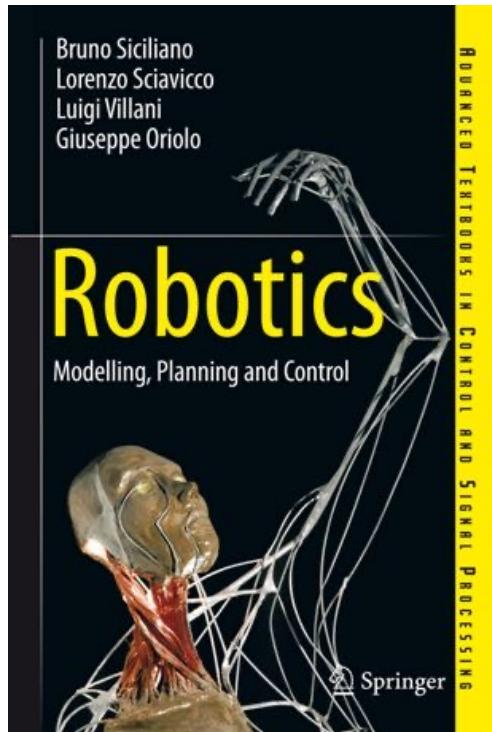
B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo: *Robotics: Modelling, Planning and Control*, 3rd Edition, Springer, 2009





... fully revised textbook coming!

B. Siciliano, L. Villani, G. Oriolo, A. De Luca: **Foundations of Robotics**, Springer, to be published **before the end of 2024**





Robotics

- **algorithms for robotics***
 - process **inputs from sensors** that provide noisy and partial data
 - build **geometric and physical models** of the robot and the world
 - **plan high- and low-level actions** at different time horizons
 - **execute these actions on actuators** with uncertainty/limited precision
- design & analysis of robot algorithms raise a unique combination of questions from many fields
 - **control theory**
 - computational geometry and topology
 - **geometrical and physical modeling**
 - reasoning under uncertainty
 - probabilistic algorithms and game theory
 - theoretical computer science

* = modified from intro to WAFR 2016



Program - 1

- advanced kinematics
 - kinematic calibration
 - kinematic redundancy and related control methods
- dynamic modeling of manipulators
 - direct and inverse dynamics
 - Euler-Lagrange formulation
 - Newton-Euler formulation
 - properties of the dynamic model
 - identification of dynamic parameters
 - inclusion of flexibility at the joints
 - inclusion of geometric constraints

all on fixed-base
robot manipulators!

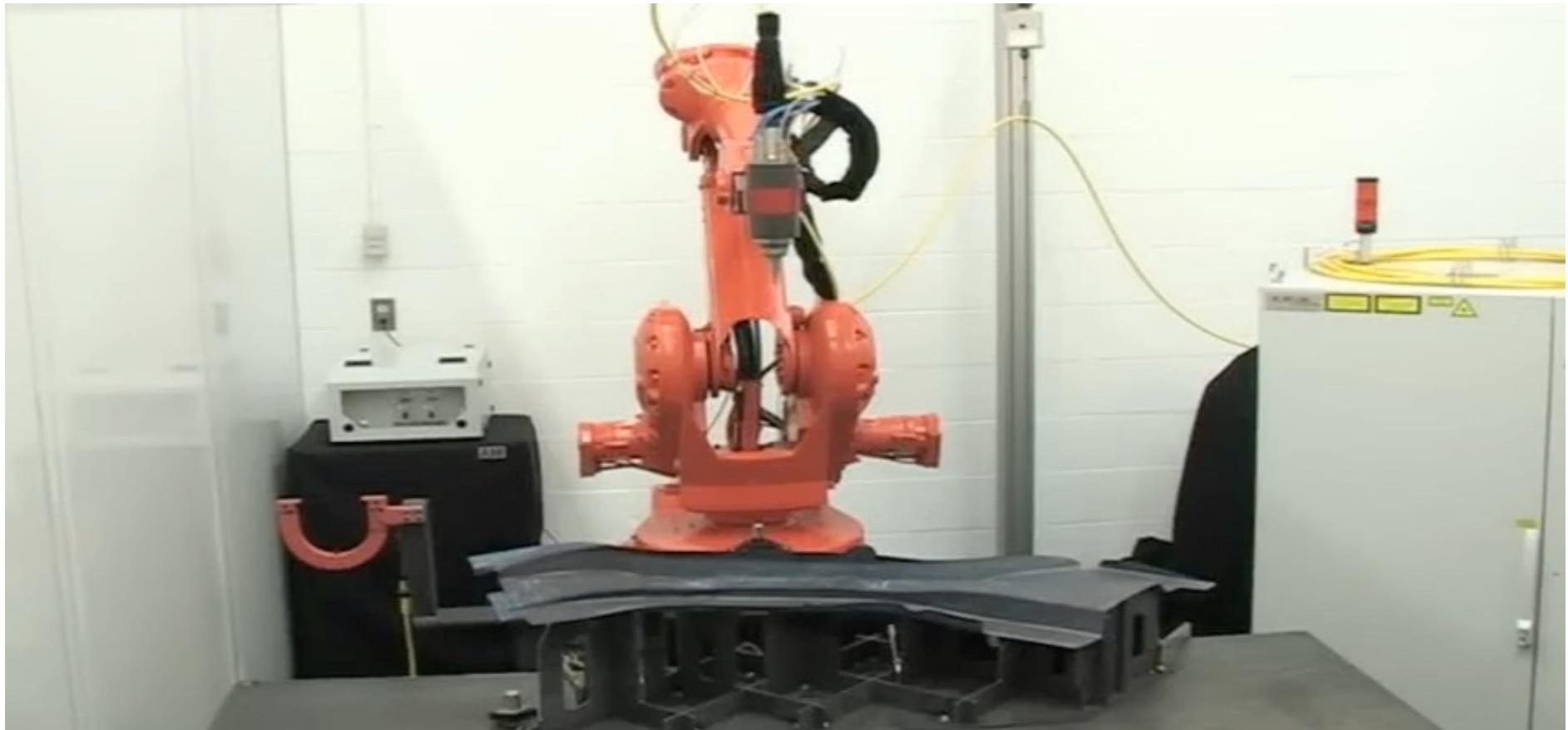
Q: are redundant robots
“special” manipulators?

Q: why/when do we need
dynamics for robot control?



Task-related redundancy

video of ABB robot in laser cutting

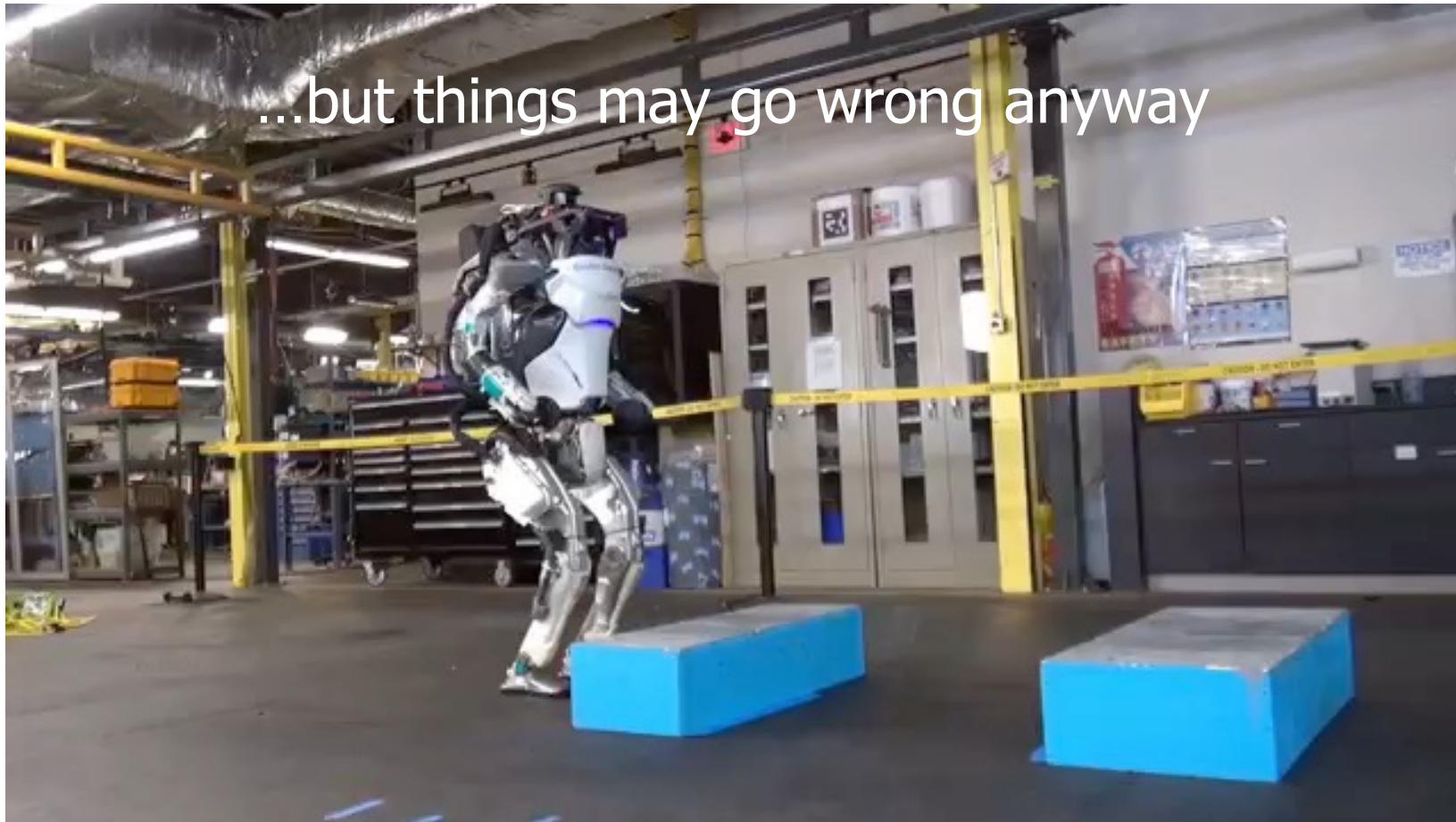


6-DOF robot for a 5-dimensional task
= 1 degree of kinematic redundancy



Robot dynamics and control

video of Atlas by Boston Dynamics, 2017



<https://youtu.be/fRj34o4hN4I>



Robot dynamics and control

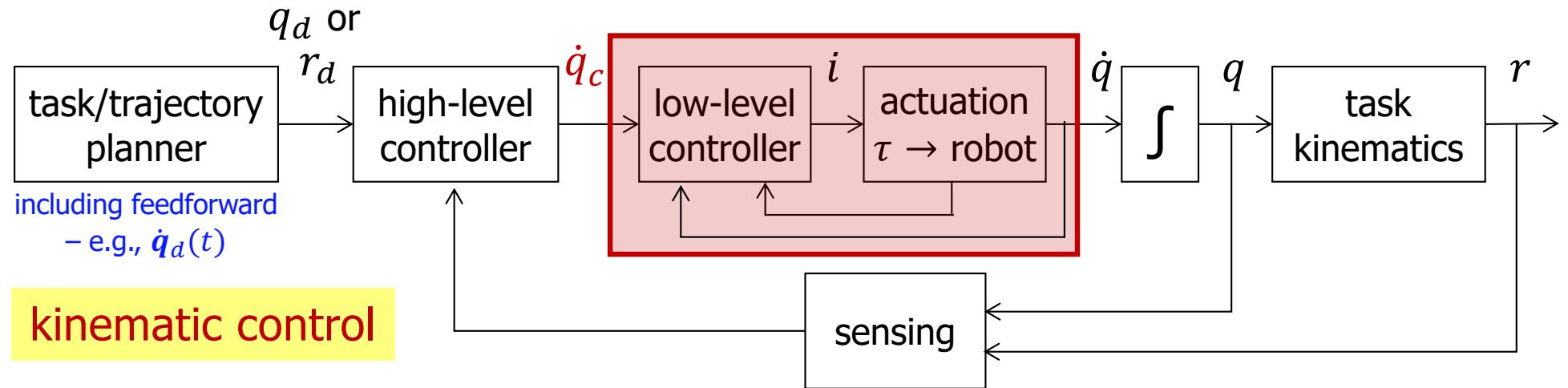
video of WAM by Barrett Technology



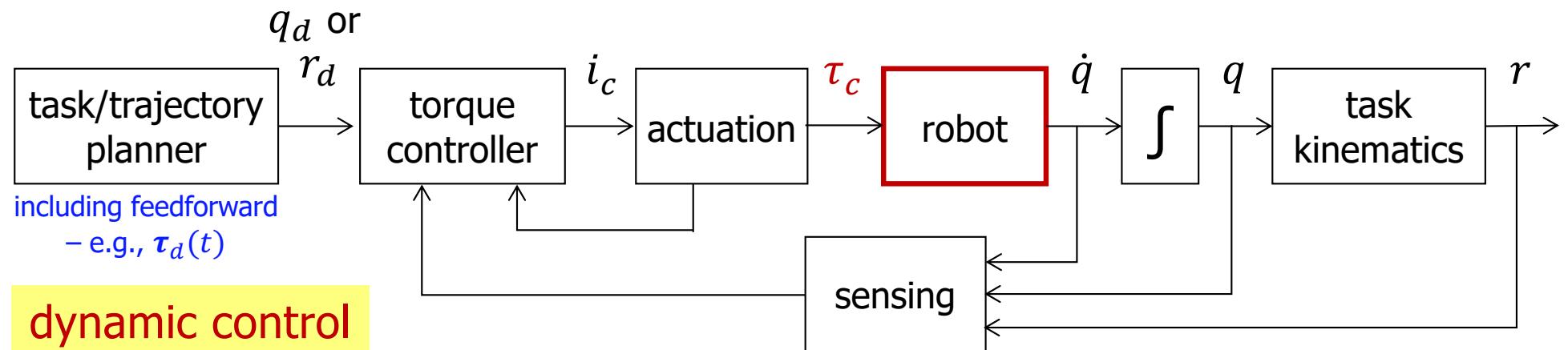
@Ishikawa Lab, Tokyo University, 2012



Position- vs. torque-controlled robots



... acceleration \ddot{q} , mass, gravity, inertia, centrifugal forces, friction?



- both modes may be present even in the same robotic system



Program - 2

■ design of feedback control laws

■ free motion tasks

■ set-point regulation

- PD with gravity cancellation or compensation
- PID or saturated PID
- iterative learning for gravity compensation
- regulation in the Cartesian/task space

■ trajectory tracking

- feedback linearization and input-output decoupling
 - in the joint space
 - in the Cartesian/task space
- passivity-based control
- adaptive (and robust) control
- on-line learning

Q: why/when is kinematic control not sufficient?

torque input commands

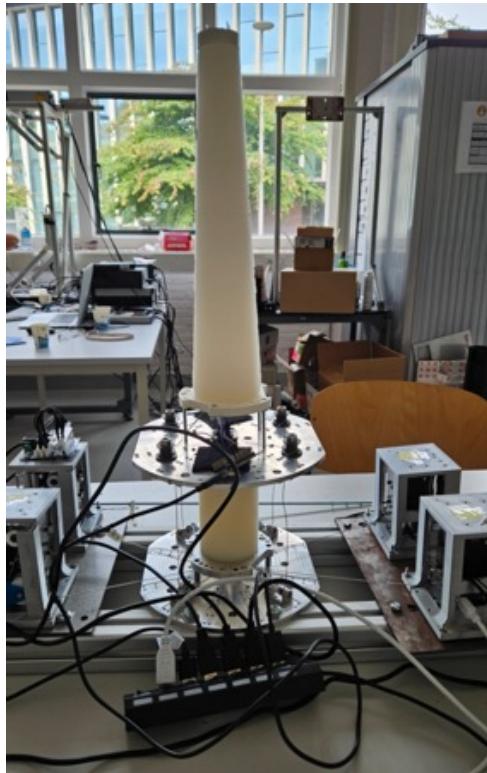


Iterative learning under gravity

continuum soft robots

- hard to model: ∞ -dimensional \Rightarrow PCC (= Piecewise Constant Curvature)
- difficult estimation of the dynamic parameters

[video](#)



two-segment prototype @TU Delft

The slide features a red circular logo with a golden eagle and the Latin motto 'STUDIVM VRBIS'. The title 'Regulation by Iterative Learning in Continuum Soft Robots' is centered in large white font. Below it, the authors' names 'Marco Montagna, Pietro Pustina, Alessandro De Luca' are listed. At the bottom, the text 'DIAG Robotics Lab' and 'Sapienza Università di Roma' is followed by the date 'October 2022'.

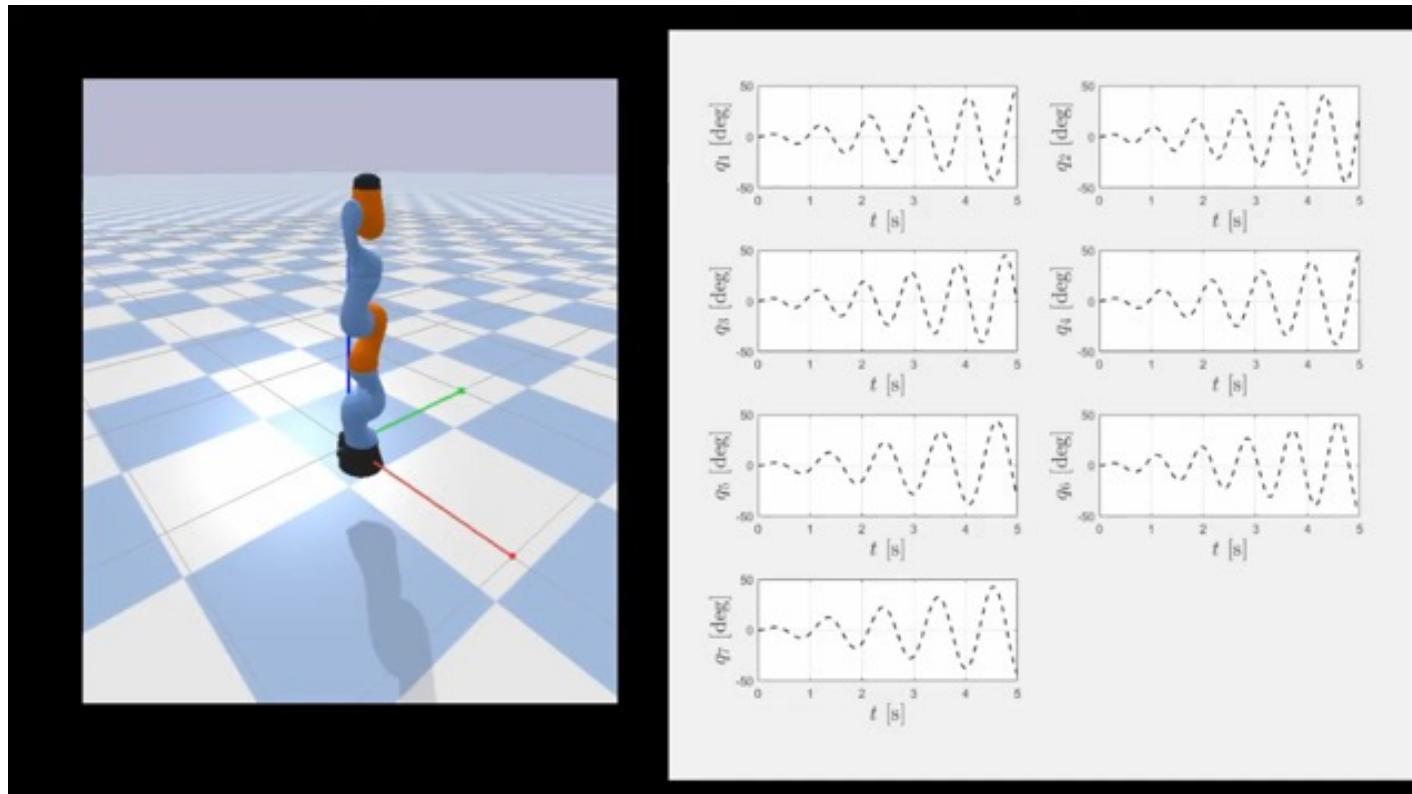
I-RIM 2022 conference



Feedback linearization and inverse dynamics

rigid multi-link robots

- use a complete dynamic model, with feedback reaction to tracking errors
- uncertainties handled by off-line identification, on-line adaptation, ...



7R KUKA LWR4+ robot

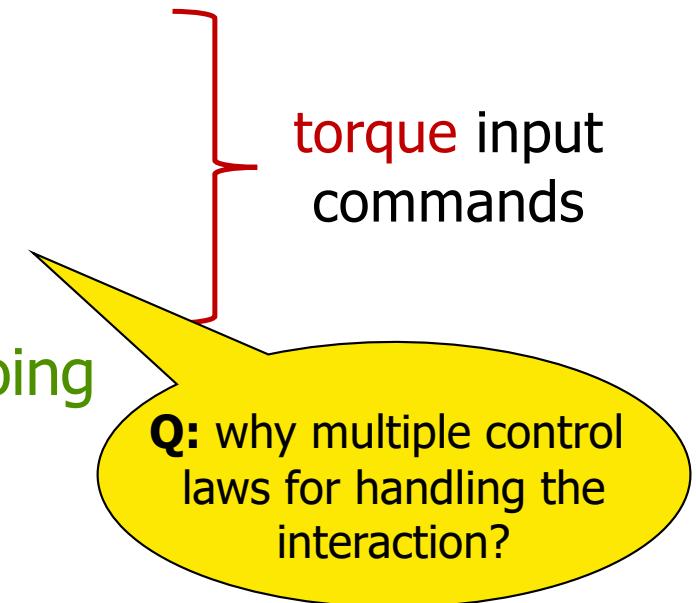
DEI UniPadova, I-RIM 2020 conference

video



Program - 3

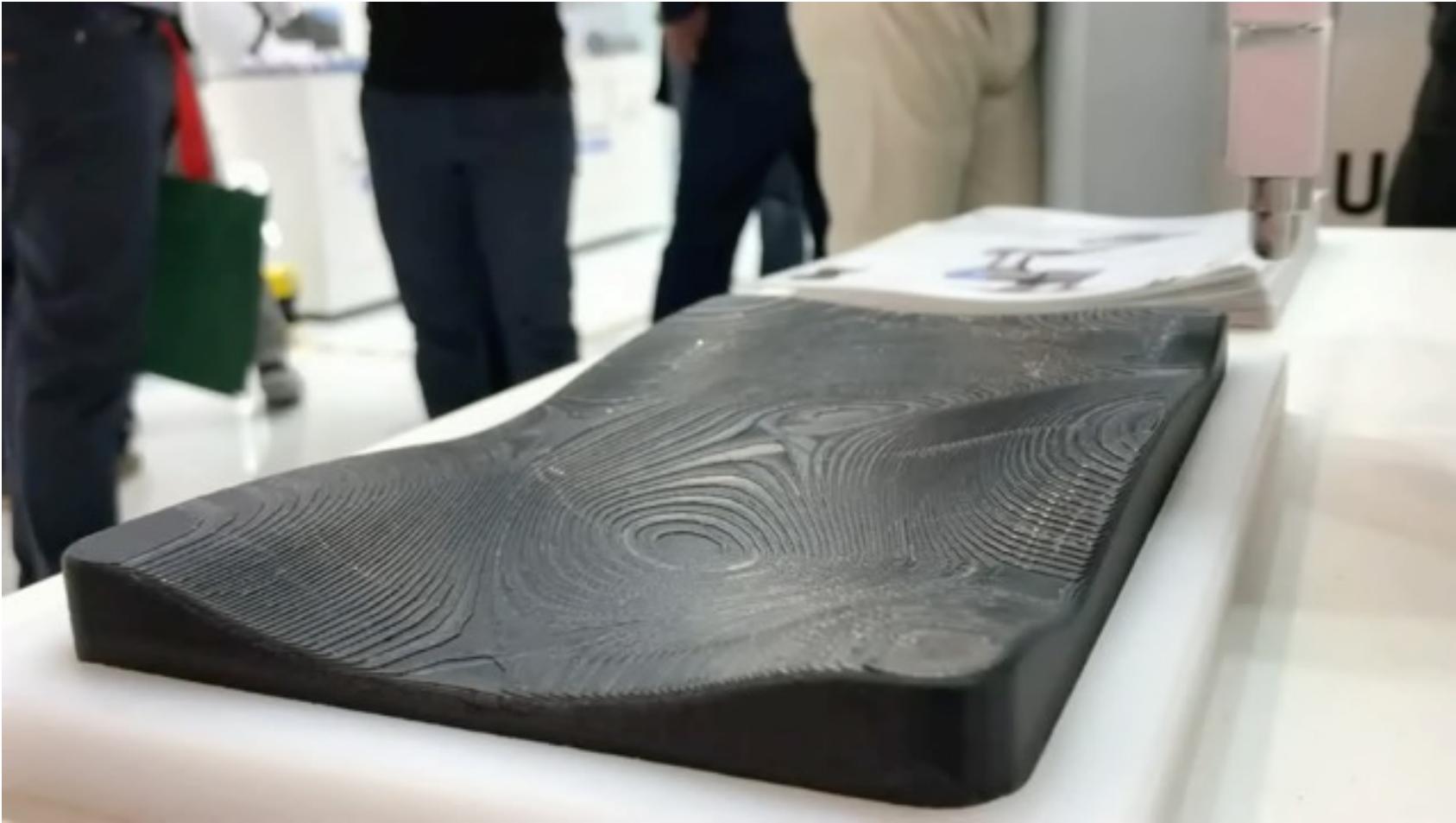
- design of feedback control laws
 - interaction tasks with the environment
 - compliance/admittance control
 - impedance control
 - hybrid force/velocity control
 - image- and position-based visual servoing
 - kinematic control treatment only
- fault diagnosis
 - detection and isolation of robot actuator faults
 - extension to a class of sensor faults
- simulation tools
 - Matlab/Simulink (including Robotics Toolbox)
 - CoppeliaSim (formerly V-REP)





Interacting with a rigid, irregular surface

[video](#)



which control law is more appropriate? what is the goal?



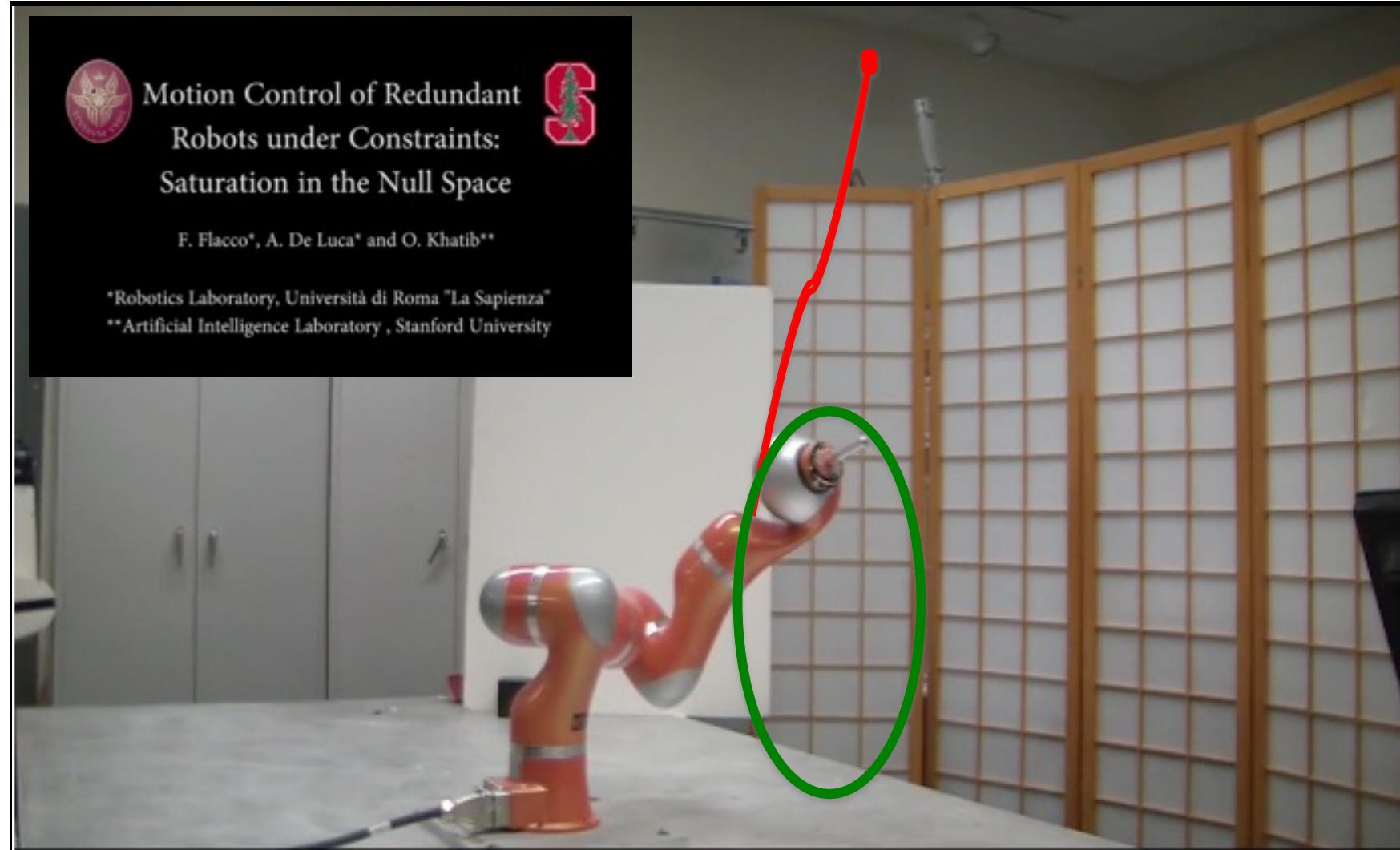
Sneak preview of videos follows ...

- kinematic redundancy and related control methods
- robot dynamic modeling and identification
- motion control in the presence of joint flexibility
- interaction with the environment: force and motion control



Kinematic/dynamic control and redundancy

SNS algorithm handles hard bounds on robot motion



KUKA LWR4+ robot

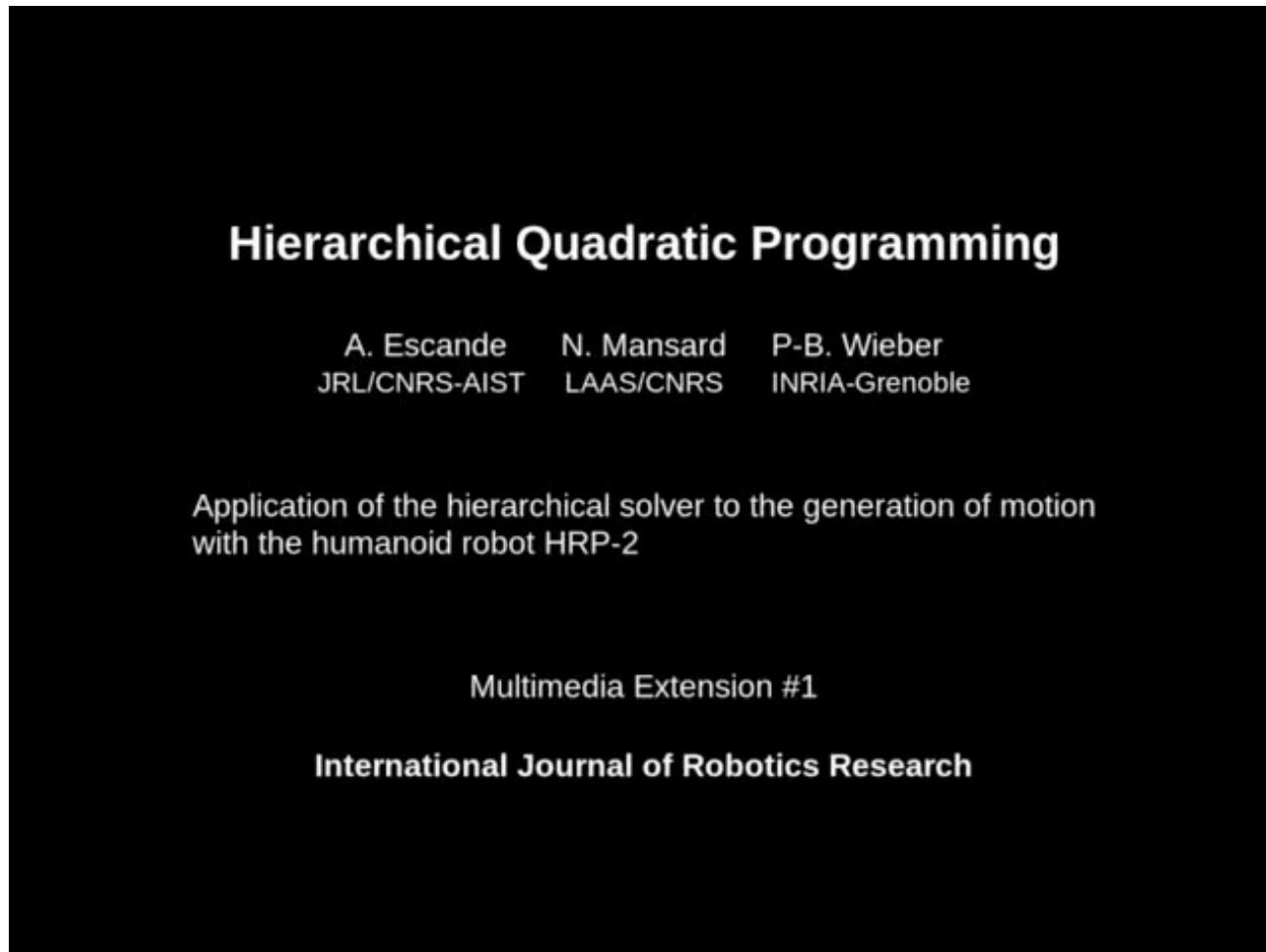
video DIAG Sapienza/Stanford, IEEE ICRA 2012



Kinematic control and redundancy

(standing) HRP-2 humanoid robot

video @LAAS/CNRS Toulouse



HQP approach for multiple equality and inequality tasks **with priorities**



Dynamic modeling and identification

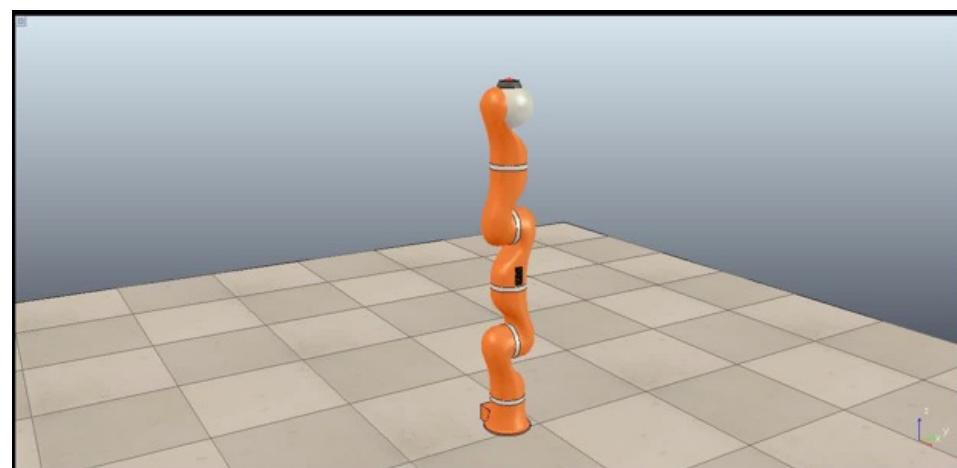


data acquisition
for identification



KUKA LWR4+ robot
with joint torque
sensing

2 videos ICRA 2014 @DIAG Robotics Lab



model validation
by torque prediction

dynamic
simulation
with V-REP



Dynamic modeling and identification

e.g., linear parametrization of gravity term in robot dynamic model

$$\pi_g = \begin{pmatrix} c_{7y}m_7 \\ c_{7x}m_7 \\ c_{6x}m_6 \\ c_{6z}m_6 + c_{7z}m_7 \\ c_{5z}m_5 - c_{6y}m_6 \\ c_{5x}m_5 \\ c_{5y}m_5 + c_{4z}m_4 + d_2(m_5 + m_6 + m_7) \\ c_{4x}m_4 \\ c_{4y}m_4 + c_{3z}m_3 \\ c_{2x}m_2 \\ c_{3x}m_3 \\ c_{2z}m_2 - c_{3y}m_3 + d_1(m_3 + m_4 + m_5 + m_6 + m_7) \end{pmatrix}$$

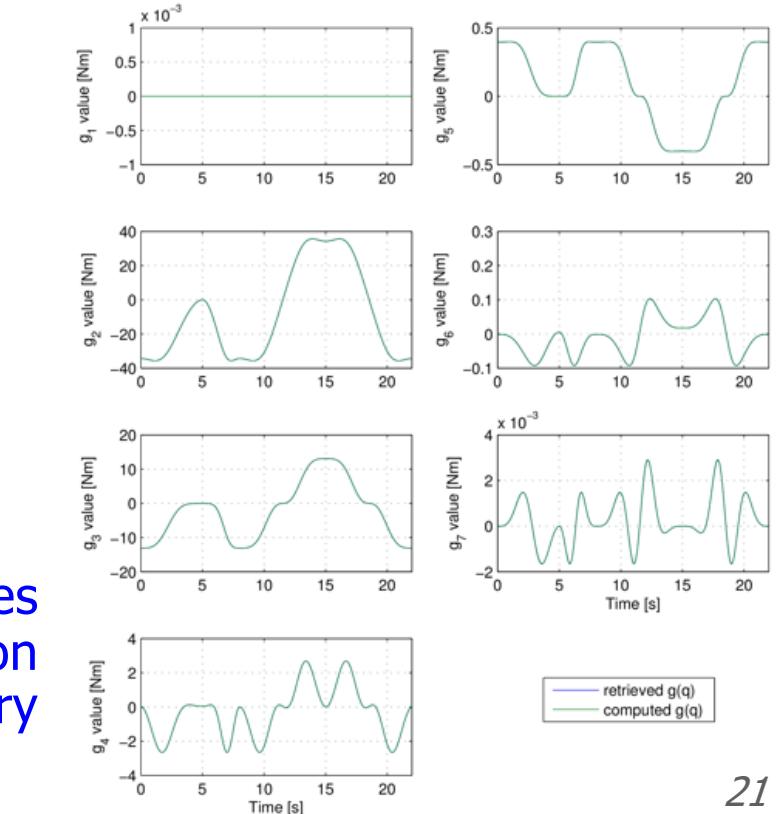
↓

$$\hat{\pi}_g = \begin{pmatrix} 9.5457 \times 10^{-4} \\ -2.9826 \times 10^{-4} \\ 8.3524 \times 10^{-4} \\ 0.0286 \\ -0.0407 \\ -6.5637 \times 10^{-4} \\ 1.334 \\ -0.0035 \\ -4.7258 \times 10^{-4} \\ 0.0014 \\ 9.4532 \times 10^{-4} \\ 3.4568 \end{pmatrix}$$

Robotics 2

symbolic expressions of gravity-related dynamic coefficients

$$\mathbf{g}(\mathbf{q}) = \mathbf{Y}_g(\mathbf{q})\boldsymbol{\pi}_g$$



numerical values identified through experiments

gravity joint torques prediction/evaluation on new validation trajectory

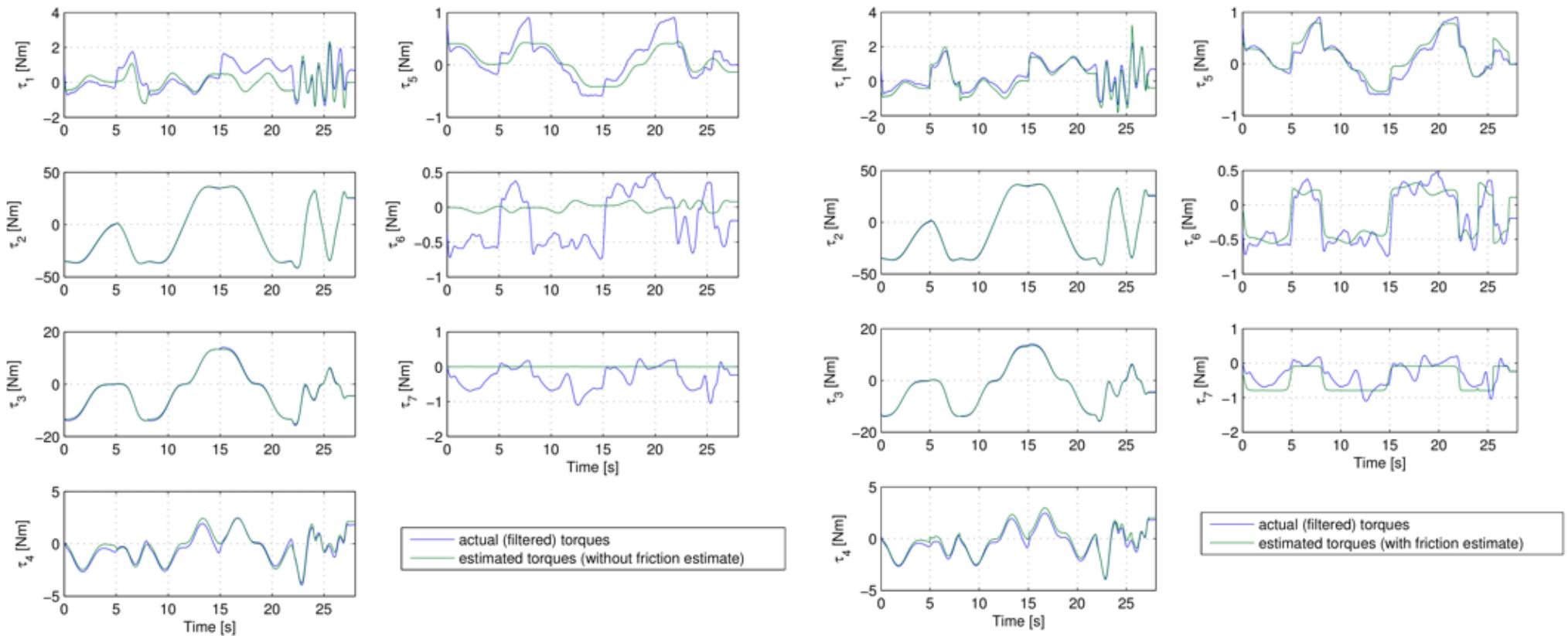


Dynamic modeling and identification

complete dynamic model estimation vs. joint torque sensor measurement

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \boldsymbol{\tau}_{friction}$$

$$\boldsymbol{\tau}_{meas}$$



without the use of a joint friction model

including an identified joint friction model



Motion and interaction control



low-damped oscillations due to flexibility
of robot transmissions at the joints
(use of Harmonic Drives)

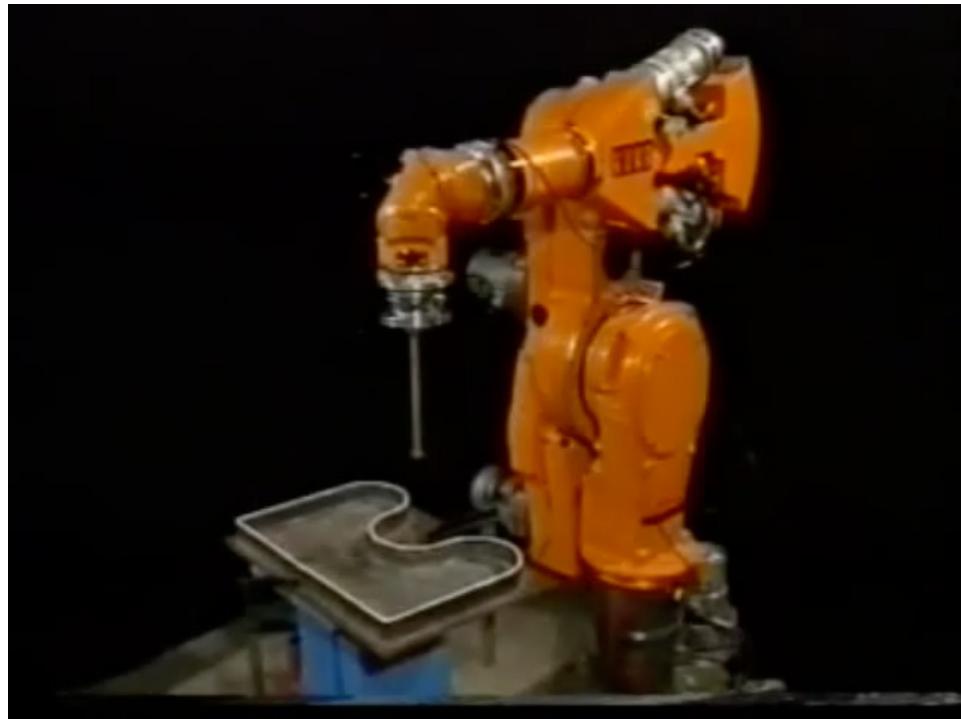


end-effector response to forces
with impedance control
(selective behavior in different directions)



Control of environment interaction

2 video clips extracted from Springer Handbook of Robotics - Multimedia



surface contour following



peg-in-hole insertion strategy

De Schutter et al @KU Leuven, Belgium (mid '90s)



Physical human-robot interaction control

video ICRA 2015 @DIAG Robotics Lab



Control of Generalized Contact Motion and Force in Physical Human-Robot Interaction

Emanuele Magrini, Fabrizio Flacco, Alessandro De Luca

Robotics Lab, DIAG
Sapienza Università di Roma

September 2014



Contacts

- **student hours** Tuesdays 12:00-13:30 (until early June 2024)
 - in presence **A-210**, left wing, floor 2, **DIAG**
 - via Zoom or G-Meet (see www.diag.uniroma1.it/deluca/Teaching.php)
 - send an email for other dates (check also “**My travel dates**”)
- **communication mode**
 - **use** the **G-group** for questions and doubts: everyone would benefit!
 - by mail (personal issues) deluca@diag.uniroma1.it
- **URL** www.diag.uniroma1.it/deluca
- **course material**
 - www.diag.uniroma1.it/deluca/rob2_en.php
 - pdf of slides, link to video lectures, vides shown in class (zipped), syllabus, written exams (most with solutions), ...
- **research video channel** www.youtube.com/user/RoboticsLabSapienza



Exams and Master Theses

- type of exam
 - midterm test **qualifies** for a final project (**OR** as part of the final exam)
 - final written exam **OR** final project + report + oral presentation
- schedule for academic year 2023-24
 - 2 sessions at the end of this semester
 - between June 3 and July 26
 - 1 session after the summer break
 - between September 2 and 24
 - 2 sessions at the end of the first semester of next year
 - January and February 2025
 - book in infostud (code 1021883) up to **one week before**, only one session is open at a time
 - **2 extra sessions only for students of previous years, part-time, etc.**
 - in March-April and October-November 2024
- theses samples at DIAG Robotics Lab www.diag.uniroma1.it/labrob



Robotics 2

Kinematic calibration

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

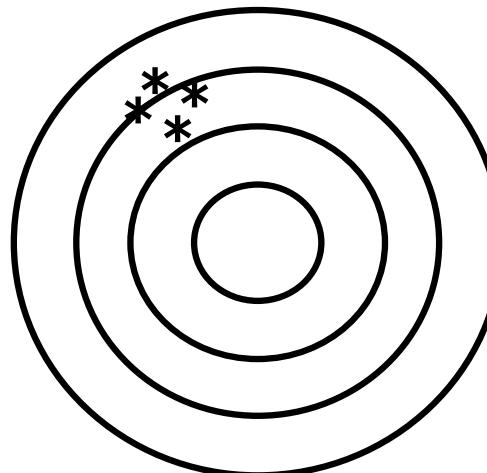
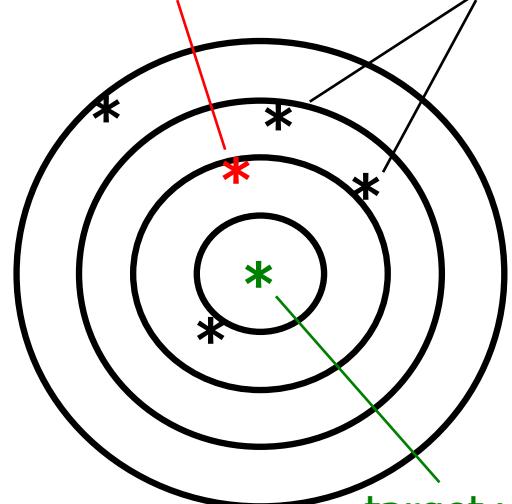




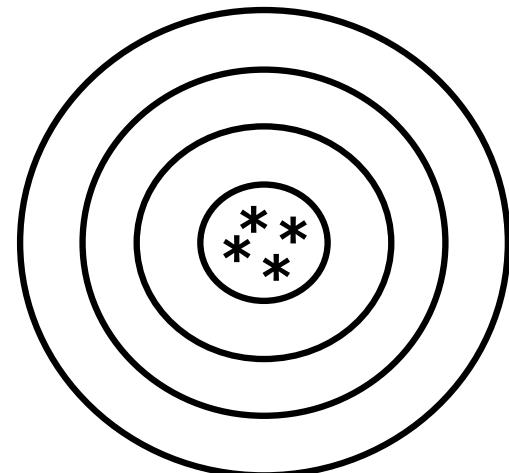
Accuracy and Repeatability

robot as a measuring device

average value measurements



from Robotics 1



low accuracy
low repeatability

low accuracy
high repeatability

high accuracy
high repeatability

better components!

calibration!



Direct kinematics

- nominal set of Denavit-Hartenberg (D-H) parameters

$$\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} \quad \boldsymbol{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \quad \boldsymbol{d} = \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix}$$

for simplicity, suppose
an all-revolute joints
manipulator

- nominal direct kinematics

$$\boldsymbol{r}_{nom} = f(\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{d}, \boldsymbol{\theta})$$

θ are typically measured by encoders

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}$$

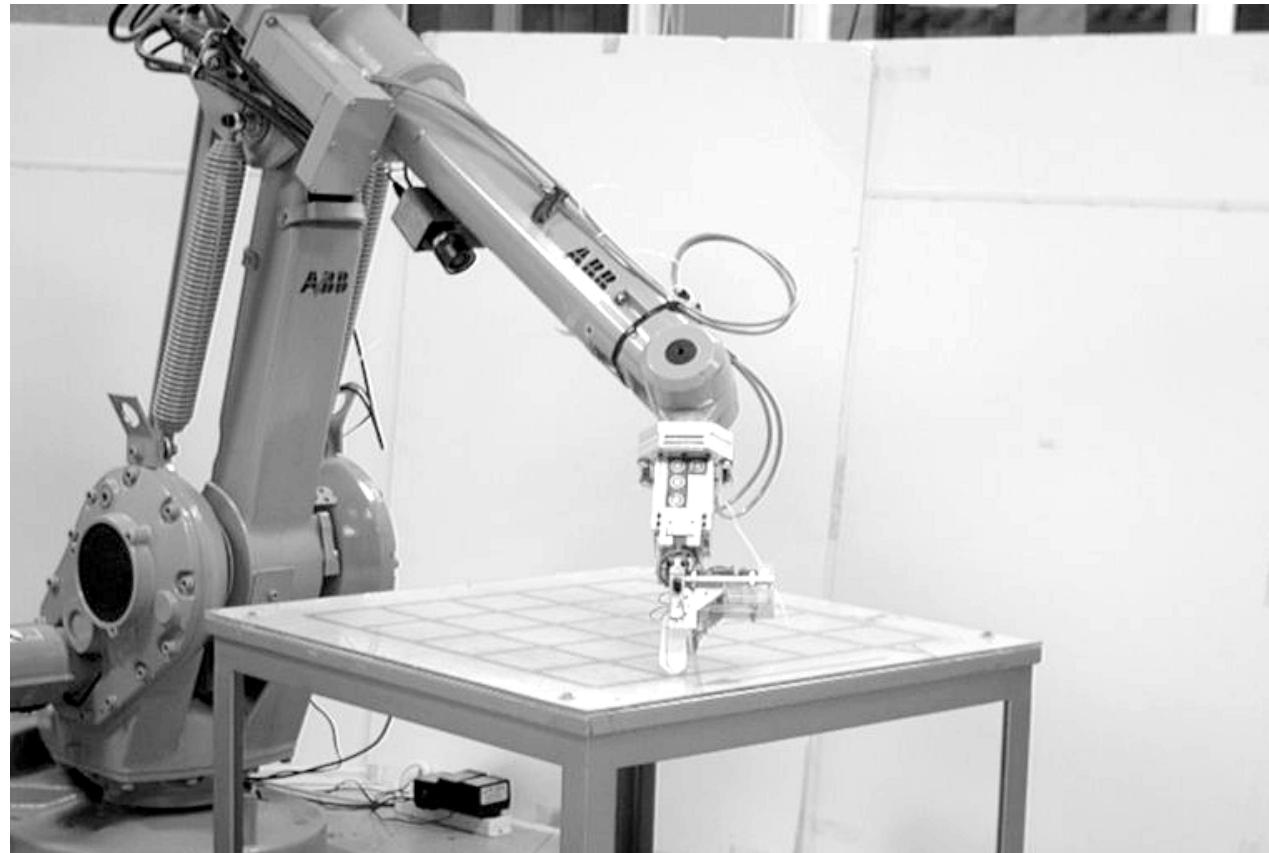


Need for calibration

- tolerances in mechanical construction and in assembly of links/joints imply small errors in actual end-effector pose ($\text{real} \neq \text{nominal}$ parameters)
- encoder mounting on motor axes may not be consistent with the “zero reference” of the robot direct kinematics (joint angle measures are constantly biased)
- errors distributed “along” the arm are amplified, due to the open chain kinematic structure of most robots
- calibration goal: recover as much as possible E-E pose errors by correcting the nominal set of D-H parameters, based on independent external (accurate!) measurements
- experiments to be done once for each robot, before starting operation... (and maybe repeated from time to time)



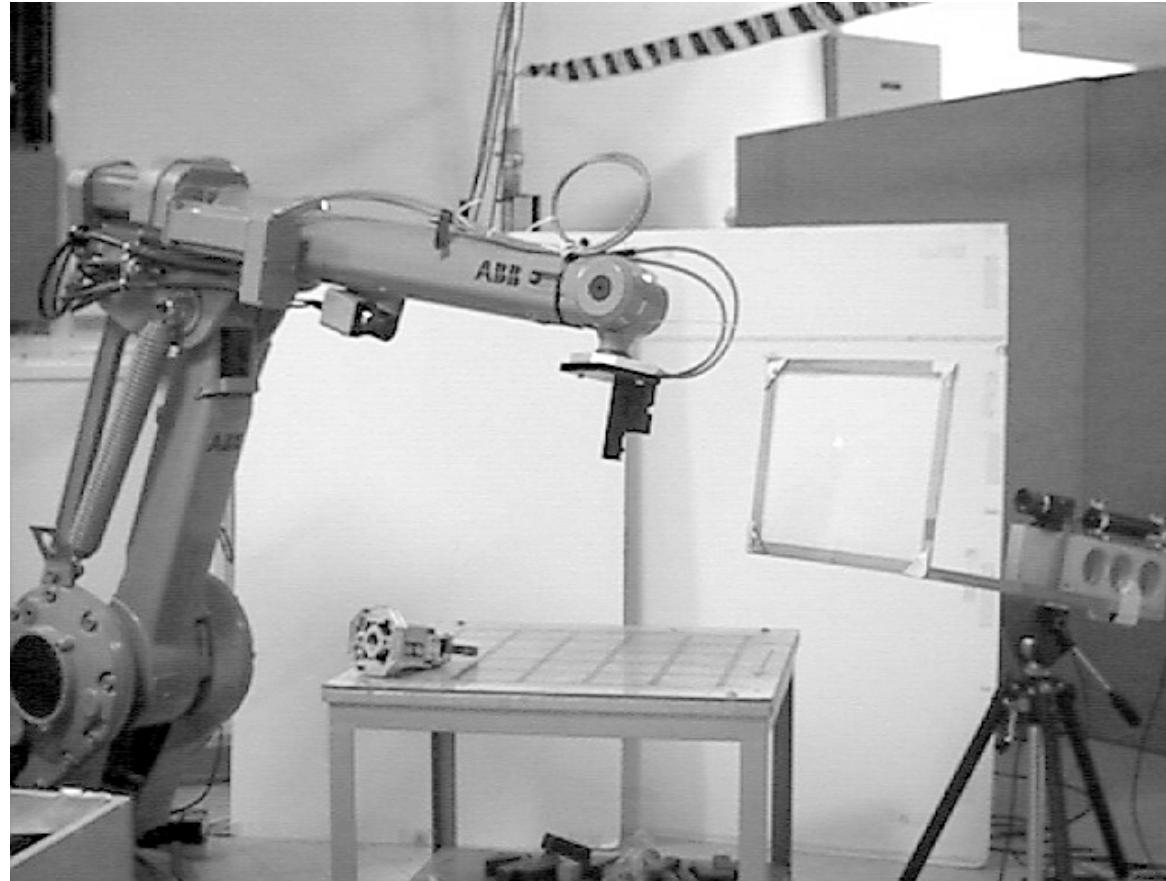
Cartesian measurement systems - 1



calibration table



Cartesian measurement systems - 2



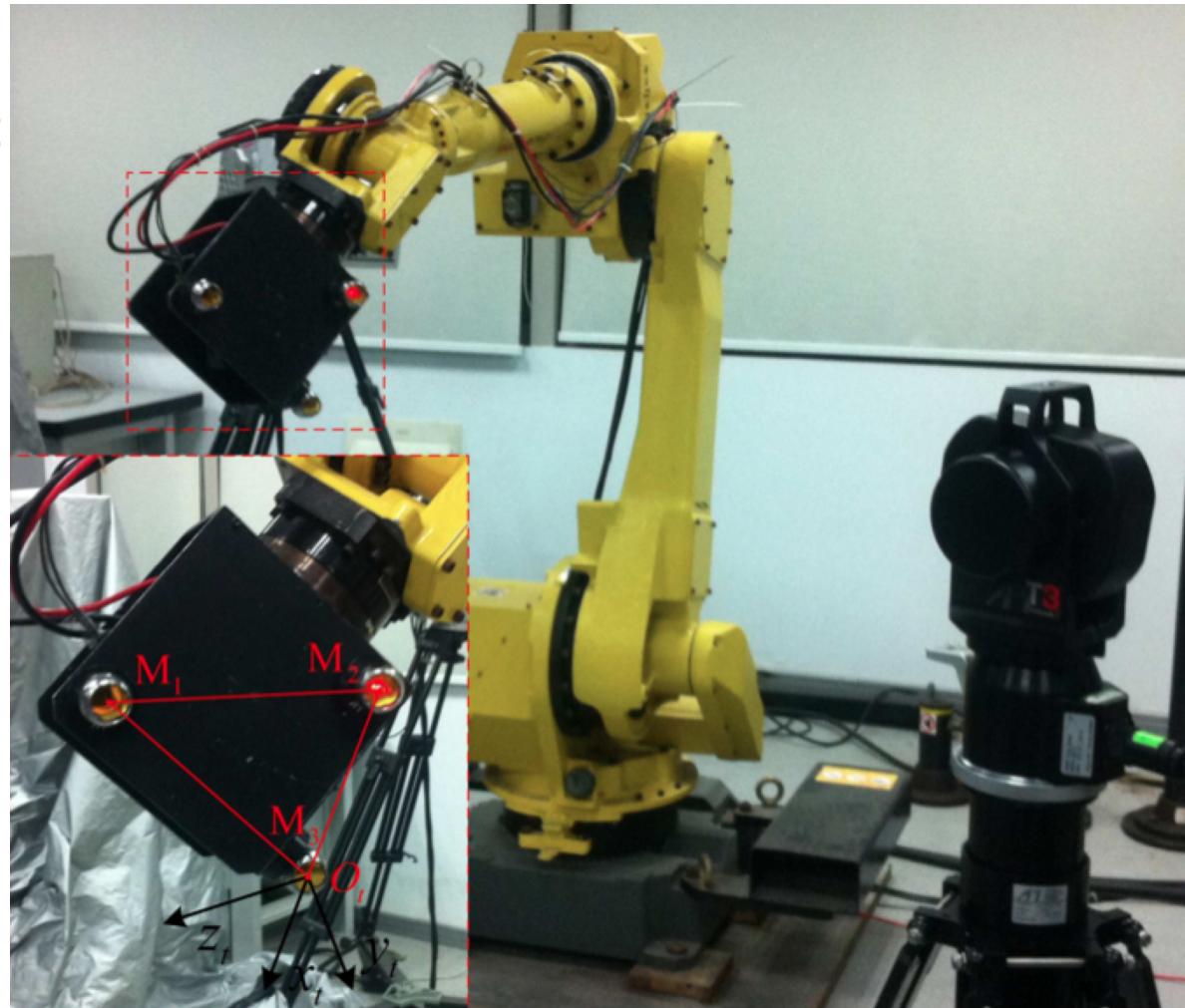
laser/camera system + triangulation



Cartesian measurement systems - 3

FANUC 6R robot
M-710iC/50

3 SMRs
(Spherically-
Mounted
Reflectors)



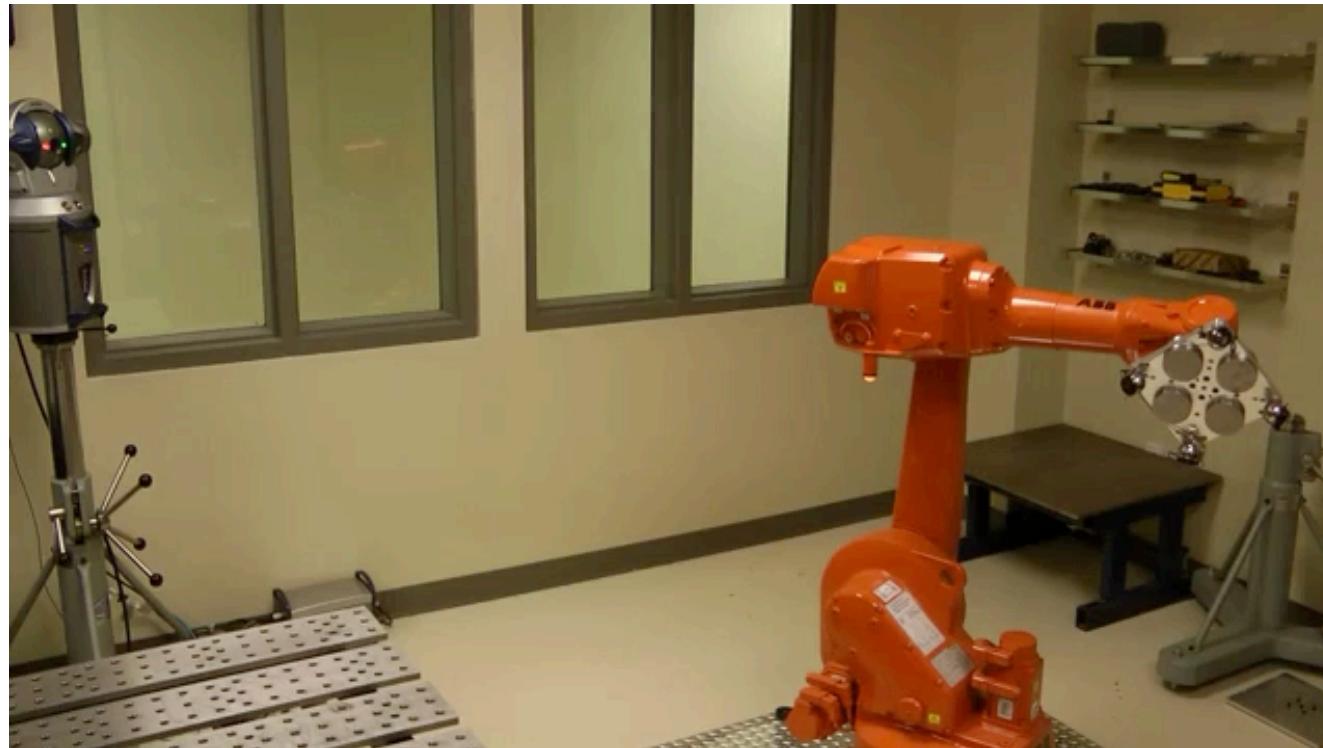
API
laser tracker III
www.apisensor.com

laser tracker + targets on end-effector



Acquiring data for calibration

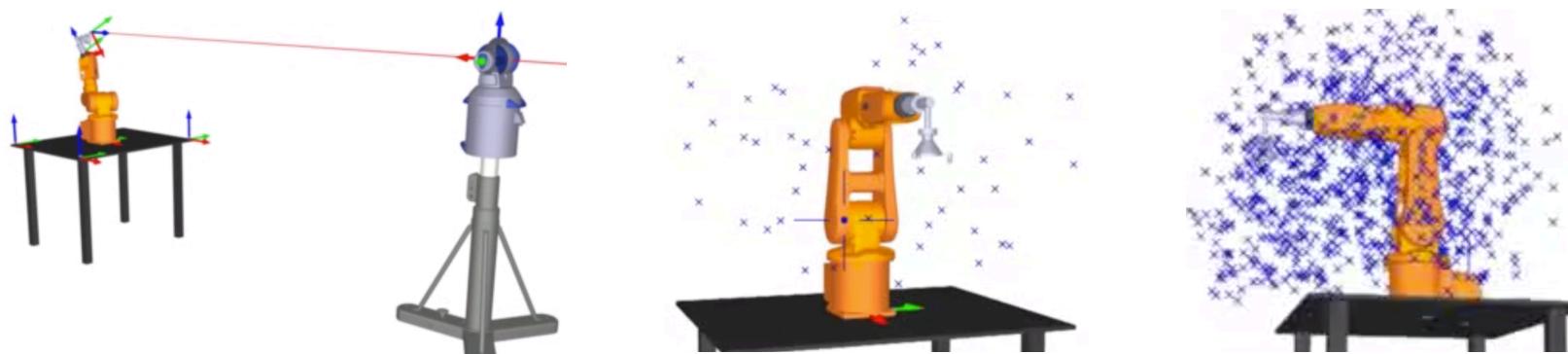
FARO ION
laser tracker



video
@CoRo Lab
ETS Montréal

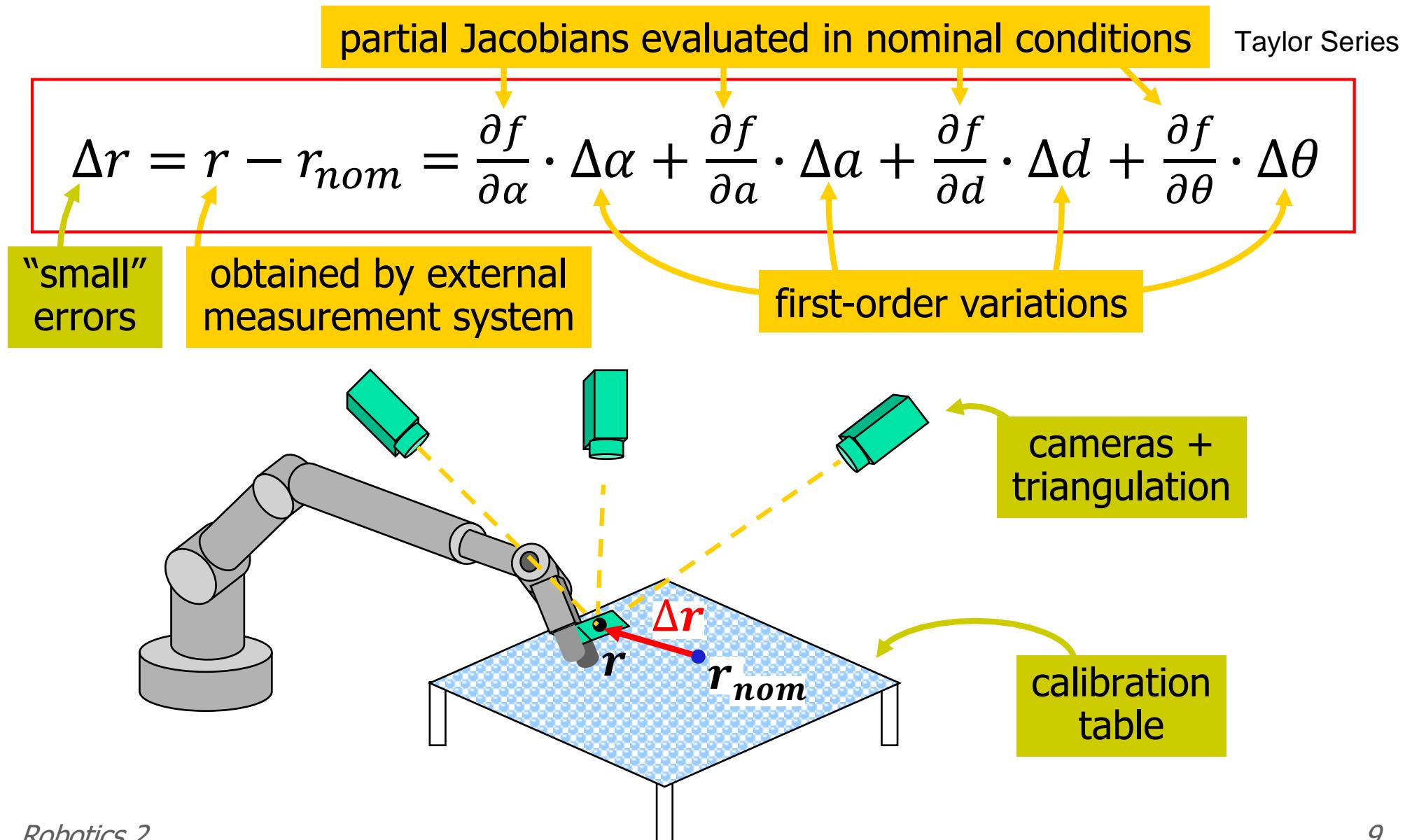
ABB
IRB 1600
robot

4 SMRs





Linearization of direct kinematics





Calibration equation

$$\Delta\varphi = \begin{pmatrix} \Delta\alpha \\ \Delta a \\ \Delta d \\ \Delta\theta \end{pmatrix} \quad \Phi = \left(\frac{\partial f}{\partial \alpha} \quad \frac{\partial f}{\partial a} \quad \frac{\partial f}{\partial d} \quad \frac{\partial f}{\partial \theta} \right)$$

Note: delta f w.r.t. to theta is just the analytic jacobian, we are extending it

$\Delta r = \Phi \cdot \Delta\varphi$

$4n \times 1$ alpha, a, d and theta are n-dimensional since we have n joints

$6 \times 4n$ 6 because we are assuming the task space has 6 dimensions i.e. f has 6 components

Doing L experiments, so L points

$$\Delta\bar{r} = \begin{pmatrix} \Delta r_1 \\ \Delta r_2 \\ \vdots \\ \Delta r_\ell \end{pmatrix} \quad \bar{\Phi} = \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_\ell \end{pmatrix}$$

$6\ell \times 1$

$6\ell \times 4n$

Φ matrix has many rows and few columns.
If you take a sufficient number of experiments
and they are independent (not aligned in one direction)
we have full column rank

**full column rank
(for sufficiently large ℓ)**

ℓ experiments ($\ell \gg n$)

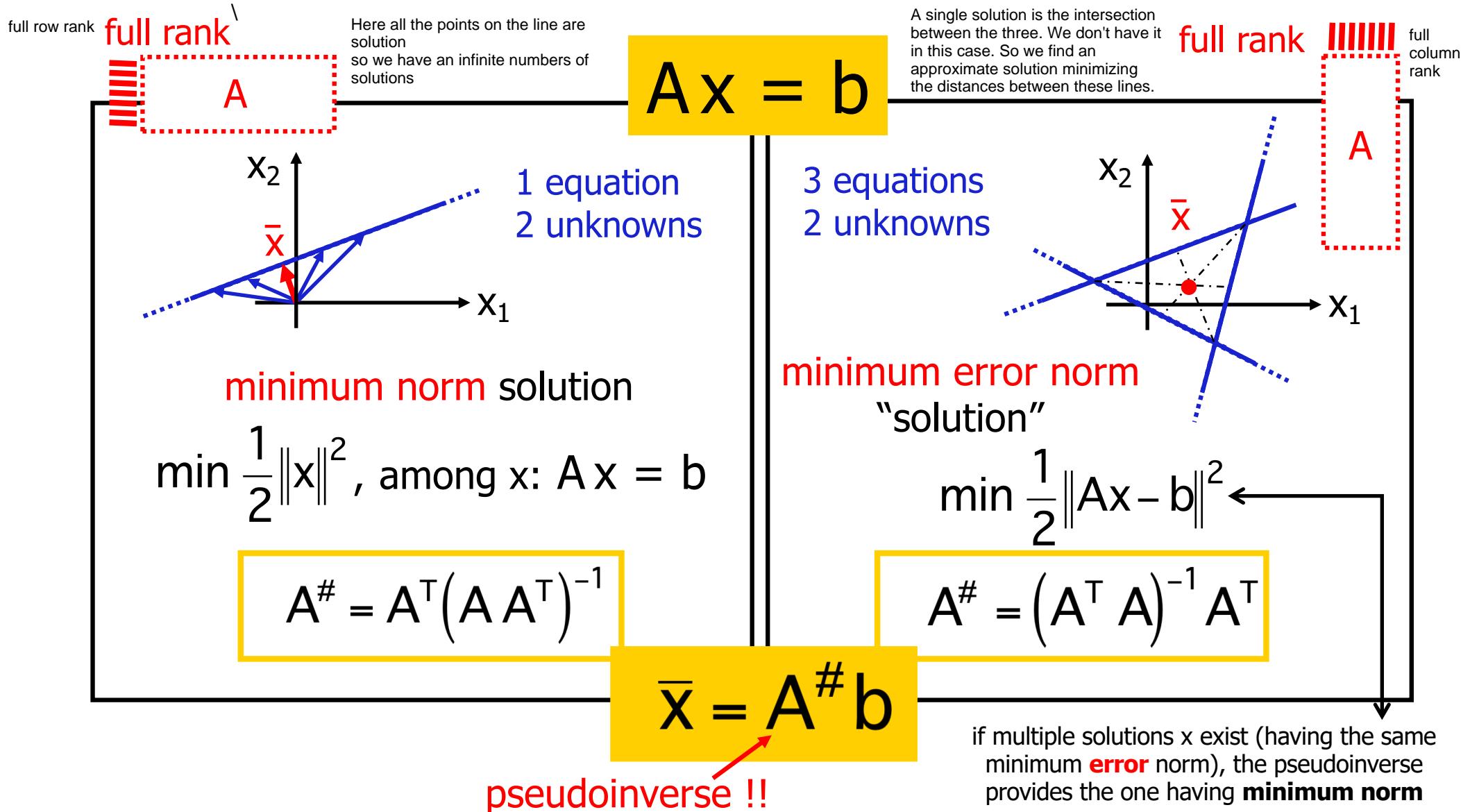
$\Delta\bar{r} = \bar{\Phi} \cdot \Delta\varphi$

measures regressor matrix evaluated at nominal parameters unknowns

Note that delta phi doesn't grow with experiments, they are the same for all the experiments -> overconstrained system! We choose a vector delta phi that reduces the error on all equations.

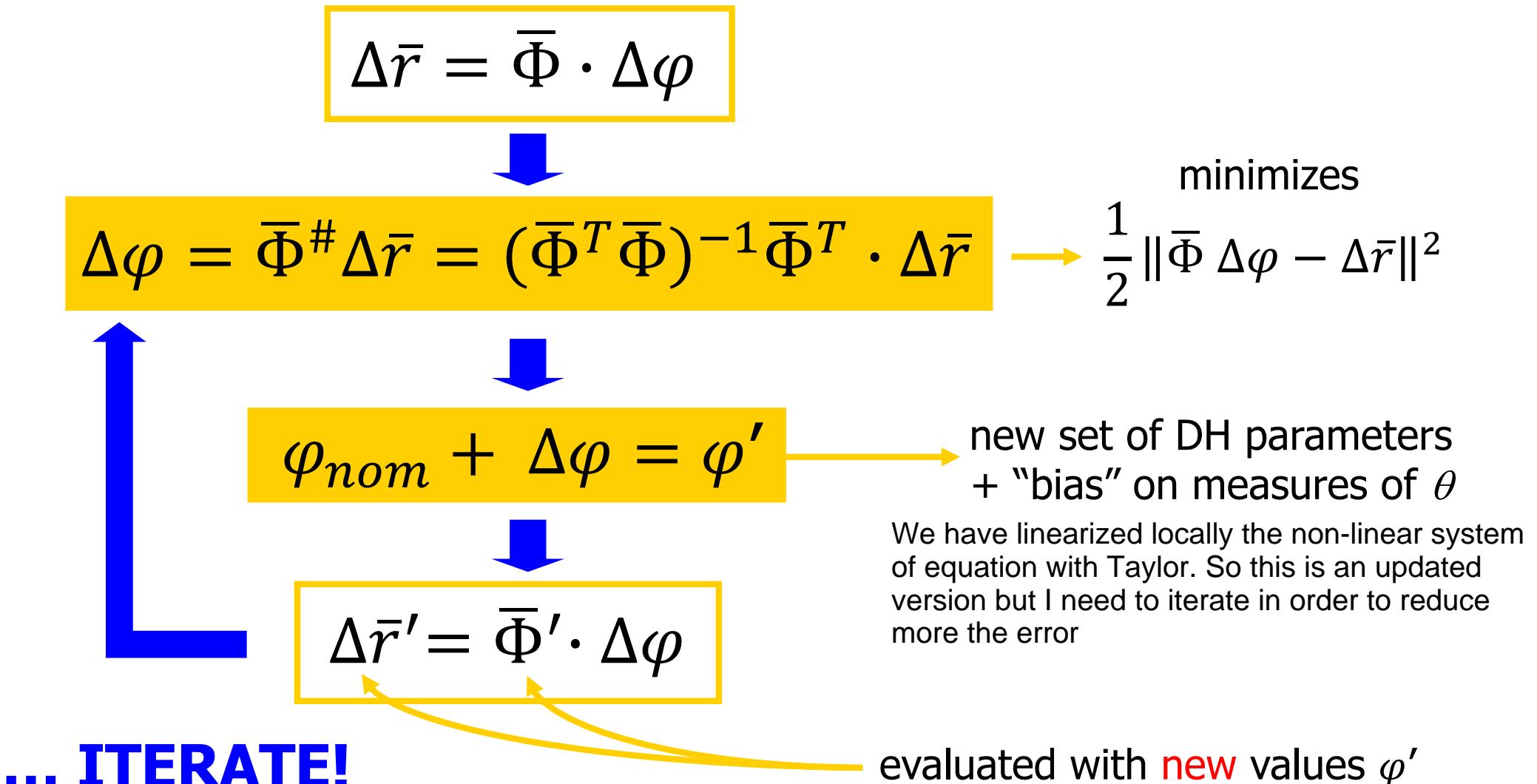


Under- and over-determined systems of linear equations





Calibration algorithm





Improvement by kinematic calibration

- ABB IRB 120 6R industrial robot
- 1000 random configurations (collision-free by simulation)
- 50 arbitrary configurations used for measurement in calibration
- 950 configurations used for validation
- Cartesian position errors

	before calibration	after calibration
Average	1.746 mm	0.193 mm
Median	1.567 mm	0.180 mm
Standard Deviation	1.043 mm	0.085 mm
Min	0.050 mm	0.010 mm
Max	4.423 mm	0.516 mm

- Improvement by **a factor $8 \div 10$**



Final comments

- an **iterative least squares** method
 - original problem is **nonlinear** in the unknowns, then linearized using first-order Taylor expansion
- it is useful to calibrate **first** and **separately** those quantities that are less accurate (typically, the encoder bias)
 - keeping the remaining ones at their nominal values
- **alternative** kinematic descriptions can be used
 - more complex than D-H parameters, but leading to a **better numerical conditioning** of the regressor matrix in calibration algorithm
 - one such description uses the POE (Product Of Exponential) formula
- more in general, **6 base parameters** should also be included
 - to locate 0-th robot frame w.r.t. world coordinate frame (of external sensor)
- accurate calibration/**estimation of real parameters** is a general problem in robotics (and beyond...)
 - for **sensors** (e.g., camera calibration)
 - for **models** (identification of dynamic parameters of a manipulator)

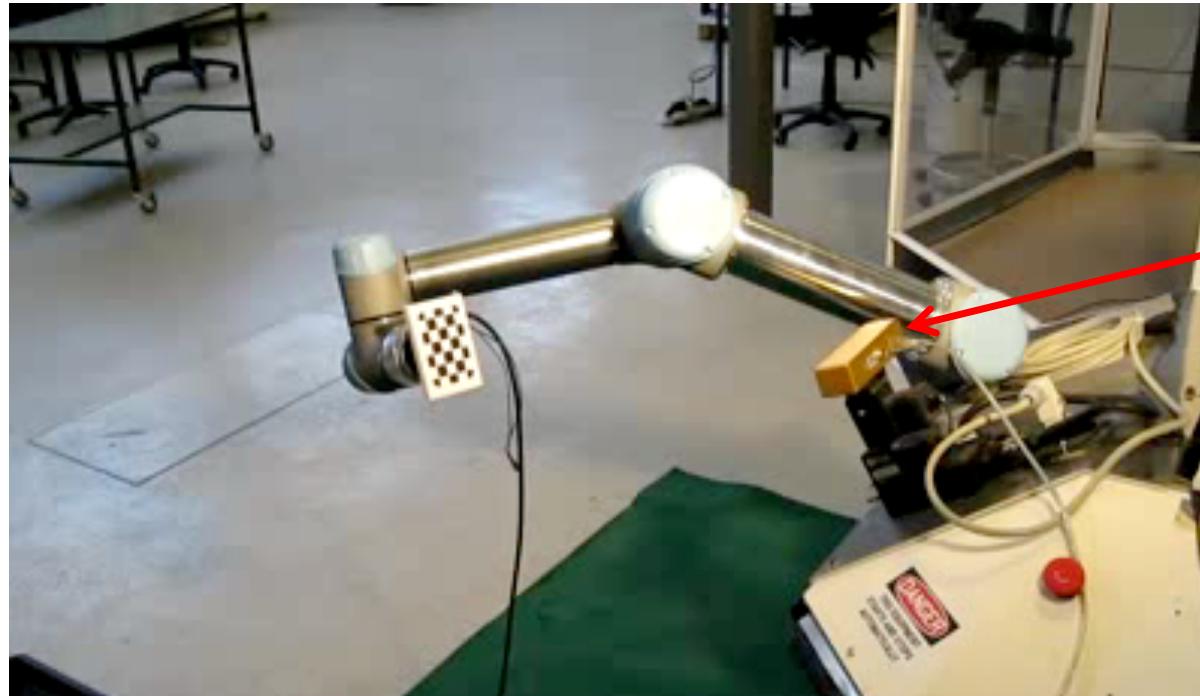
It's like normalizing values: if you have to minimize a parameters that go from 0 to 1 and another in the order of thousands first you normalize, optimize and then go back to the unnormalized space. Here is the same: theta has larger error, throwing all together, since minimizing the norm you are not doing any difference between components, we are spreading the variation on all parameters on an equal way. So we first capture the variation of the parameters which have larger effect on the positioning error. Now you just put the delta theta, keeping the other fixed, on their nominal values. You do the algorithm, getting the variation as usual, but only for delta theta. Once you have done this, the error is reduced, removed the component with larger effect, at this point you consider the full problem, allowing also the variation of alpha, a and d.



Calibration experiment

in a research environment

video



Videre Design
stereovision
camera

- automatic data acquisition for **simultaneous** calibration of
 - robot-camera transformation
 - DH parameters of the manipulator

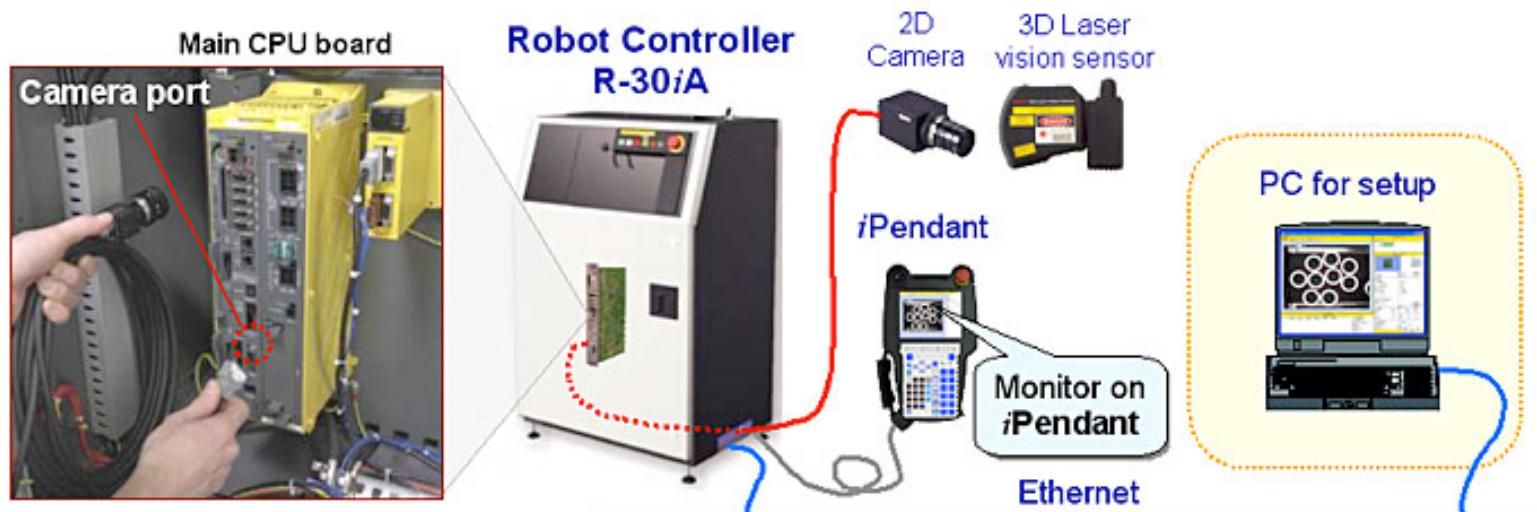


Calibration experiment in an industrial setting

FANUC
3D Laser
calibration
(with iR Vision)



video





Robotics 2

Robots with kinematic redundancy

Part 1: Fundamentals

Prof. Alessandro De Luca

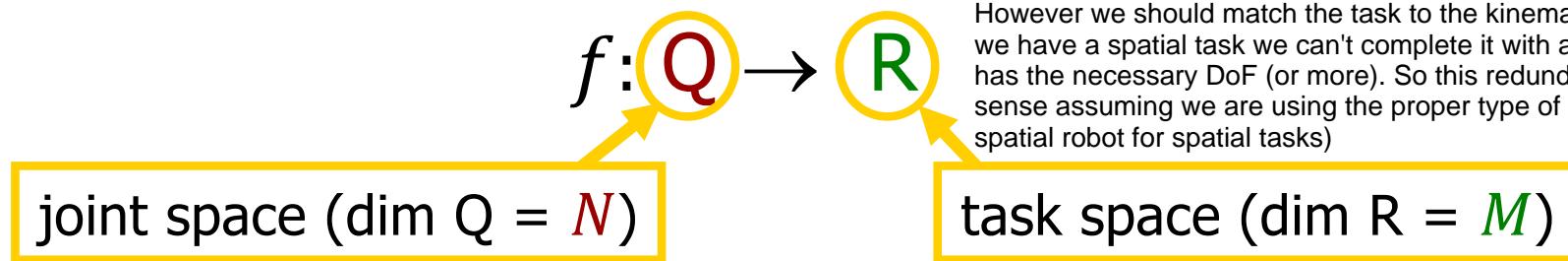
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Redundant robots

- direct kinematics of the task $r = f(q)$



- a robot is (kinematically) **redundant** for the task if $N > M$
(more degrees of freedom than strictly needed for executing the task)
- r may contain the position and/or the orientation of the end-effector or, more in general, be any parameterization of the task (even not in the Cartesian workspace)
- “redundancy” of a robot is thus a relative concept, i.e., it holds **with respect to a given task**



Some E-E tasks and their dimensions

TASKS [for the robot end-effector (E-E)]	dimension M
■ position in the plane	→ 2
■ position in 3D space	→ 3
■ orientation in the plane	→ 1
■ pointing in 3D space	two angles: rotation around vertical axis and elevation → 2
■ position and orientation in 3D space	→ 6

a planar robot with $N = 3$ joints is **redundant** for the task of **positioning its E-E in the plane ($M = 2$)**, but **NOT** for the task of **positioning AND orienting the E-E in the plane ($M = 3$)**



Typical cases of redundant robots

- 6R robot mounted on a linear track/rail
 - 7 dofs for positioning and orienting its end-effector in 3D space
- 6-dof robot used for arc welding tasks welding is a 5 dimensional task
 - task does not prescribe the final roll angle of the welding gun
- dexterous robotic hands
- multiple cooperating manipulators
- manipulator on a mobile base
- humanoid robots, team of mobile robots ...
- “kinematic” redundancy is not the only type...
 - redundancy of components (actuators, sensors)
 - redundancy in the control/supervision architecture



Uses of robot redundancy

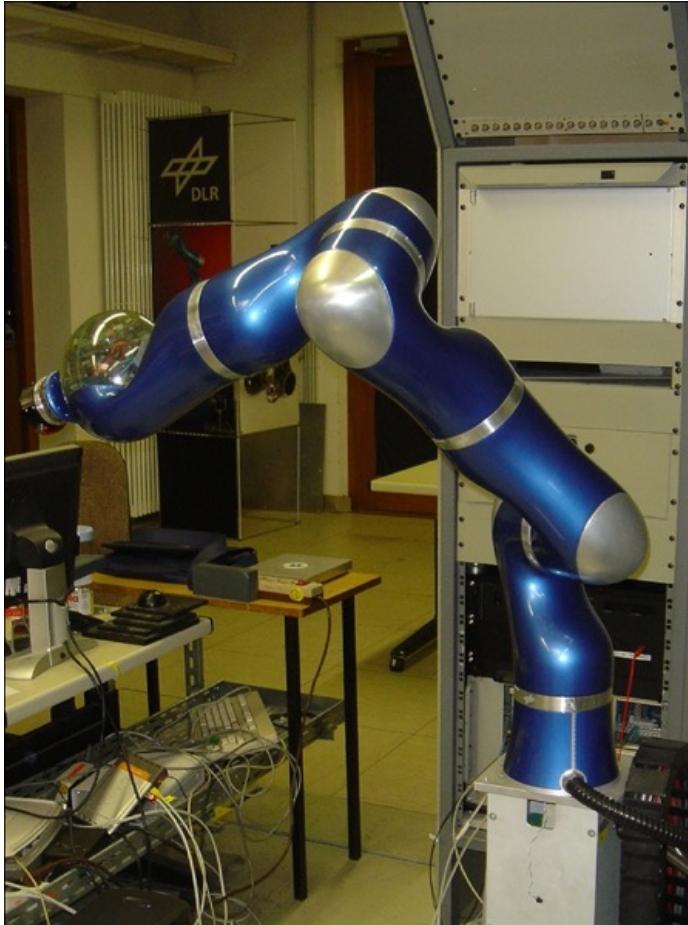
- avoid collision with obstacles (in **Cartesian** space) ...
- ... or kinematic singularities (in **joint** space)
- stay within the admissible joint ranges
- increase manipulability in specified directions
- uniformly distribute/limit joint velocities and/or accelerations
- minimize energy consumption or needed motion torques
- optimize execution time
- increase dependability with respect to faults
- ...



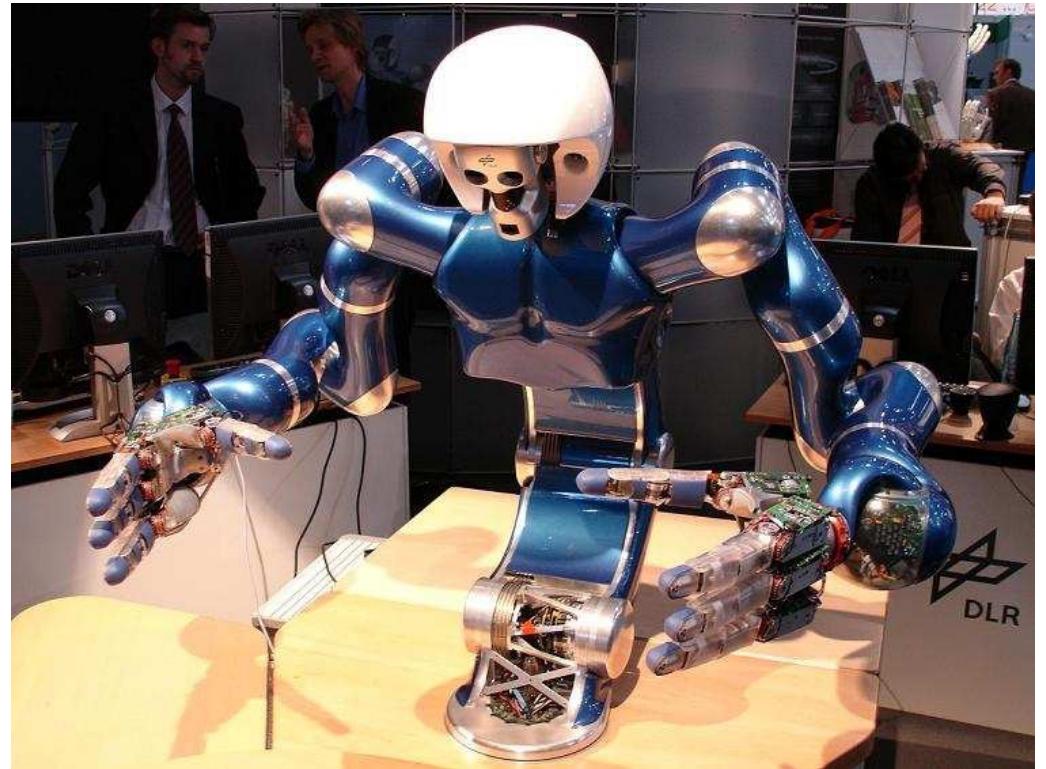
all objectives should be
quantitatively “measurable”



DLR robots: LWR-III and Justin



7R LWR-III lightweight manipulator:
elastic joints (HD), joint torque sensing,
13.5 kg weight = payload



Justin two-arm upper-body humanoid:
43R actuated =
two arms (2×7) + torso (3^*)
+ head (2) + two hands (2×12),
45 kg weight

* = one joint is dependent on the motion of the other two



Justin carrying a trailer

[video](#)



LAAS-CNRS

DLR



SIXTH FRAMEWORK PROGRAMME

motion planning for **DLR Justin robot** in the configuration space,
avoiding Cartesian obstacles and using robot redundancy



Dual-arm redundancy



video

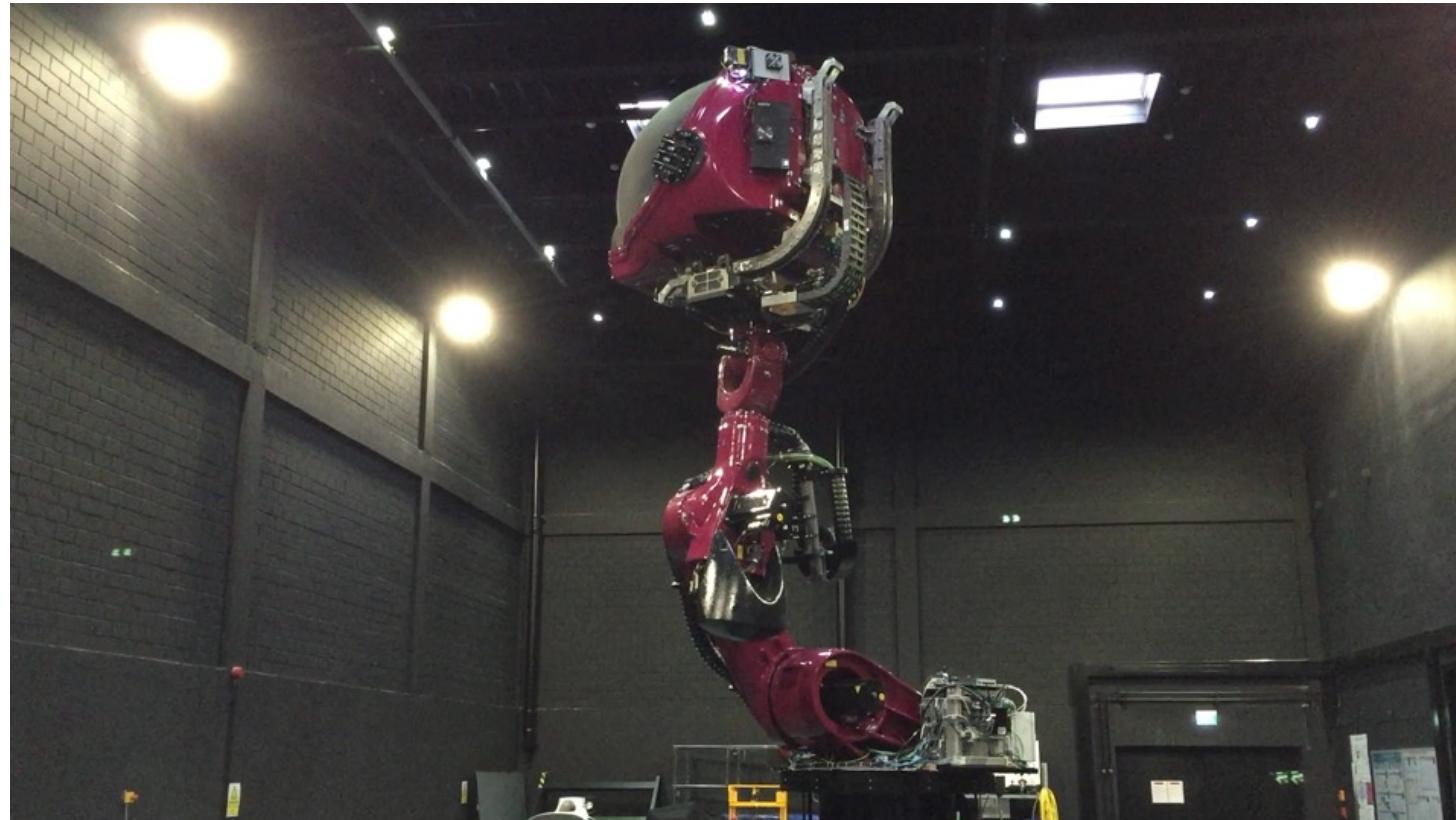
DIS, Uni Napoli

two 6R Comau robots, one mounted on a linear track (+1P)
coordinated 6D motion using the null-space of the right-side robot ($N - M = 1$)



Motion cueing from redundancy

[video](#)



Max Planck Institute for Biological Cybernetics, Tübingen

a **6R KUKA KR500** mounted on a linear track (**+1P**) with a sliding cabin (**+1R**),
used as a dynamic emulation platform for human perception ($N - M = 2$)



Self-motion

video



video



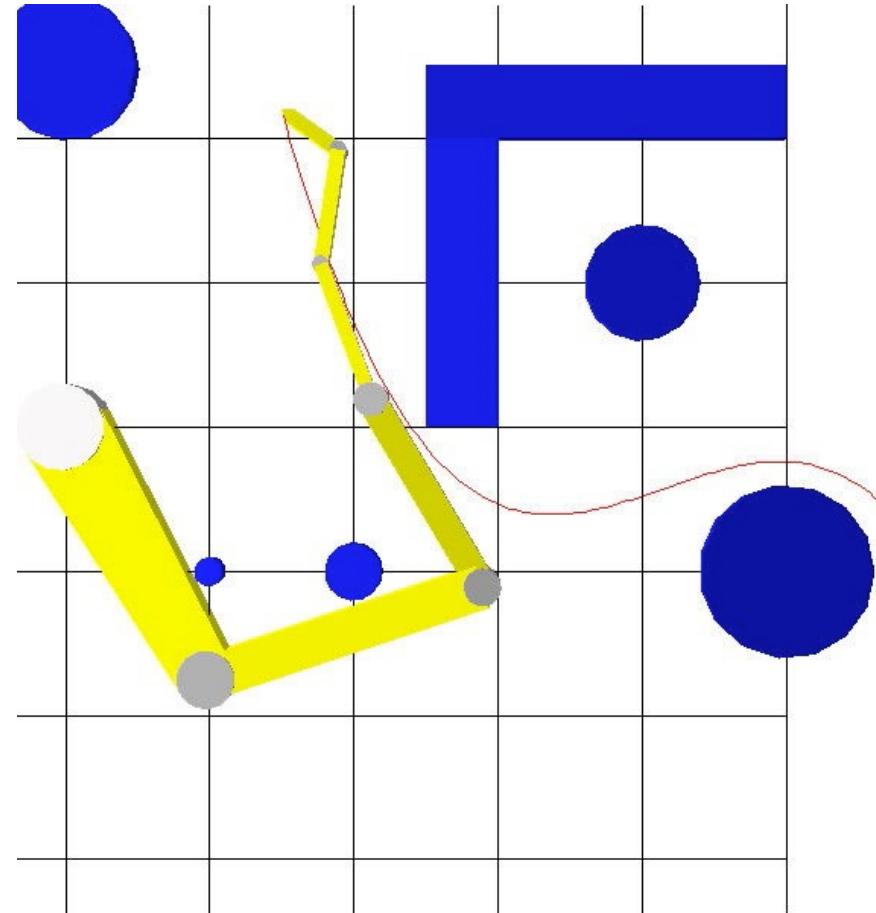
Nakamura's Lab, Uni Tokyo

8R Dexter: self-motion with
constant 6D pose of E-E ($N - M = 2$)

6R robot with spherical shoulder
in compliant tasks for the
Cartesian E-E position ($N - M = 3$)



Obstacle avoidance



video

6R planar arm moving on a given geometric path for the E-E ($N - M = 4$)



An Echord++ industrial experiment

The diagram illustrates a complex industrial robot system. A blue portal frame structure supports a grey Qirox QRC350 robotic arm. The arm has six revolute joints (A1 - A6) and three prismatic joints (x, y, z). The system is labeled as having 9 degrees of freedom and being treated as one kinematic chain. A white callout box lists results: simulation of a 6+3 DoF CLOOS robot system and Cartesian + nullspace control with all axes included.

Results:

- > simulation of a 6+3 DoF CLOOS robot system
- > Cartesian + nullspace control with all axes included

Portal
- 3 prismatic joints (x,y,z)

Qirox QRC350
- 6 revolute joints (A1 - A6)

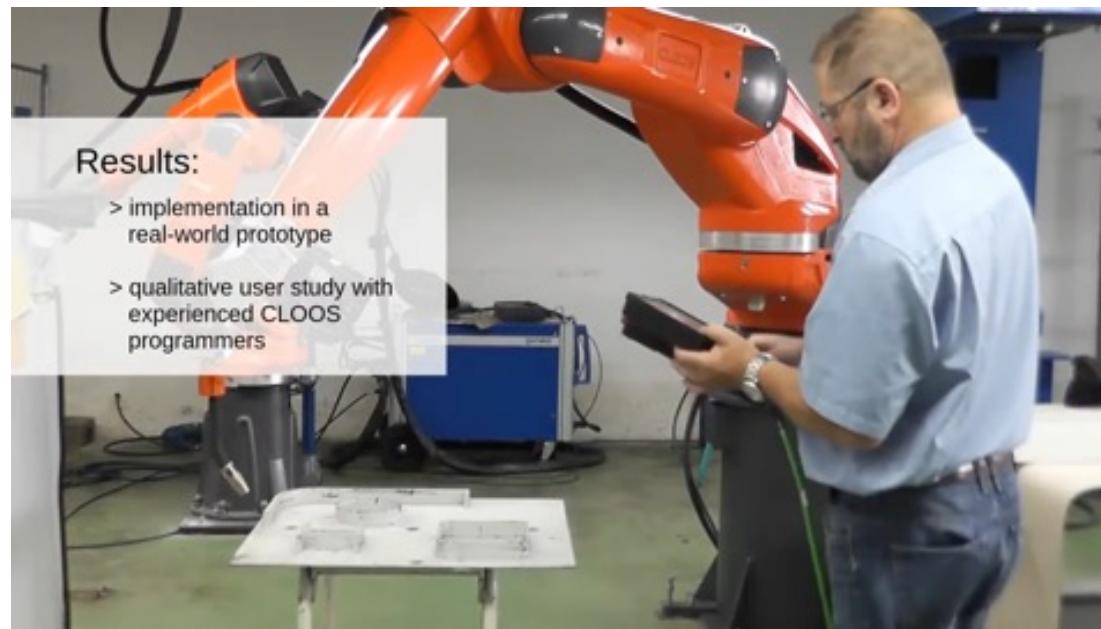
Robot system
- 9 degrees of freedom
- treated as one kinematic chain

4x

A screenshot of a 3D simulation environment. It shows a red robotic arm with a yellow cube. A coordinate system is overlaid on the cube. A text box displays sensor data: 'Selected feature: module1 (1.0.0.0)' and 'Orientation: +Z, COM: 0.00000 - 0.74387'. Another line of text shows 'quaternion: 0.30241 - 0.00000 - 0.74388 - 0.00000'.

Results:

- > easy and user-friendly demonstration of redundancy resolutions
- > automatic bootstrapping of training data
- > on-line neural-network based learning of redundancy resolutions



3 videos

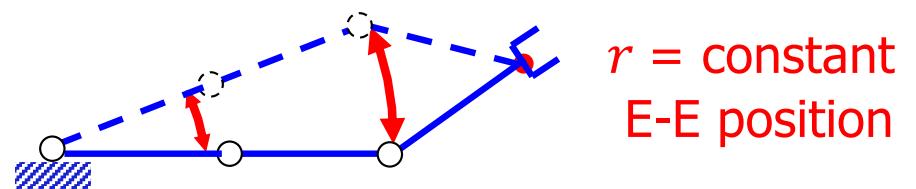


Inverse kinematics problem

- find $q(t)$ that realizes the task: $f(q(t)) = r(t)$ (at all times t)

- **infinite solutions** exist when the robot is redundant
(even for $r(t) = r = \text{constant}$)

$$N = 3 > 2 = M$$



- the robot arm may have “**internal displacements**” that are **unobservable** at the task level (e.g., not contributing to E-E motion)
 - these joint displacements can be chosen so as to **improve/optimize** in some way the behavior of the robotic system
- **self-motion**: an arm reconfiguration in the joint space that does not change/affect the value of the task variables r
- solutions are mainly sought at **differential level** (e.g., **velocity**)



Redundancy resolution

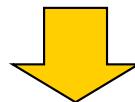
via optimization of an objective function

Local methods

given $\dot{r}(t)$ and $q(t)$, $t = kT_s$

optimization of $H(q, \dot{q})$

$\dot{q}(kT_s) \leftarrow$ ON-LINE



$$q((k+1)T_s) = q(kT_s) + T_s \dot{q}(kT_s)$$

discrete-time form

Global methods

given $r(t)$, $t \in [t_0, t_0 + T]$, $q(t_0)$

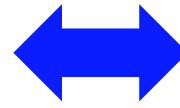
optimization of $\int_{t_0}^{t_0+T} H(q, \dot{q}) dt$

$$q(t), t \in [t_0, t_0 + T]$$

↑
OFF-LINE

Linear Quadratic (linear constraint) -> closed
form solution

relatively EASY
(a LQ problem)



quite DIFFICULT
(nonlinear TPBV problems arise)



Local resolution methods

three classes of methods for solving $\dot{r} = J(q)\dot{q}$

1 Jacobian-based methods (here, analytic Jacobian in general!)

among the infinite solutions, one is chosen, e.g., that minimizes a suitable (possibly weighted) norm For instance using pseudo-inverse

2 null-space methods

Jacobian-based method to execute the task + null space velocities which generates $r_{\text{dot}} = 0$. Although this term doesn't affect the task, it is additional velocity that can be helpful for some reason

a term is added to the previous solution so as not to affect execution of the task trajectory, i.e., belonging to the null-space $\mathcal{N}(J(q))$

3 task augmentation methods

redundancy is reduced/eliminated by adding $S \leq N - M$ further auxiliary tasks (when $S = N - M$, the problem has been “squared”)

$$r = f(q) \rightarrow \dot{r} = J(q)\dot{q}$$



1

Jacobian-based methods

we look for a solution to $\dot{r} = J(q)\dot{q}$ in the form

$$J = \boxed{\quad \quad \quad}_{N \times M}$$

$$\dot{q} = K(q)\dot{r}$$

$$K = \boxed{\quad \quad \quad}_{M \times N}$$

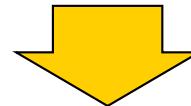
minimum requirement for K : $J(q)K(q)J(q) = J(q)$

($\Rightarrow K$ = generalized inverse of J)

r_{dot} is a linear combination of the columns of J

using $q_{\text{dot}} = K^*r_{\text{dot}}$

here we write r_{dot} as a combination of the column of J with q_{dot}



$$\forall \dot{r} \in \mathcal{R}(J(q)) \Rightarrow J(q)[K(q)\dot{r}] = J(q)K(q)J(q)\dot{q} = J(q)\dot{q} = \dot{r}$$

example:

if $J = [J_a \ J_b]$, $\det(J_a) \neq 0$, one such generalized inverse of J is $K_r = \begin{pmatrix} J_a^{-1} \\ 0 \end{pmatrix}$
(actually, this is a **stronger** right-inverse)

the zero means that we are zeroing the motion of the redundant DoF, maintaining only the motion of the joints corresponding to the non-singular minor of the jacobian



Pseudoinverse

Is a generalized inverse
with more properties

$$\dot{q} = J^\#(q)\dot{r}$$

... a very common choice: $K = J^\#$

- $J^\#$ always **exists**, and is the **unique** matrix satisfying

$$\begin{array}{ll} JJ^\#J = J & J^\#JJ^\# = J^\# \\ (JJ^\#)^T = JJ^\# & (J^\#J)^T = J^\#J \quad \text{symmetry} \end{array}$$

- if J is **full (row) rank**, $J^\# = J^T(JJ^T)^{-1}$; else, it is computed numerically using the SVD (Singular Value Decomposition) of J (**pinv** of Matlab)
- the pseudo-inverse joint velocity is the only that **minimizes the norm** $\|\dot{q}\|^2 = \dot{q}^T\dot{q}$ among all joint velocities that **minimize the task error norm** $\|\dot{r} - J(q)\dot{q}\|^2$ if we have linear and angular quantity in the same vector q , minimizing the norm makes no sense, so we need to deal with this inconsistency
- if the task is feasible ($\dot{r} \in \mathcal{R}(J(q))$), there will be **no task error**



Weighted pseudoinverse

$$\dot{q} = J_W^\#(q)\dot{r}$$

another choice: $K = J_W^\#$

- the solution \dot{q} minimizes the weighted norm

$$\|\dot{q}\|_W^2 = \dot{q}^T W \dot{q}$$

$W > 0$, symmetric
(often diagonal)

- if J is full (row) rank, $J_W^\# = W^{-1}J^T(JW^{-1}J^T)^{-1}$
- large weight $W_i \Rightarrow$ small \dot{q}_i
 - larger weights for proximity joints (carrying/moving more “mass”)
 - weights chosen proportionally to the inverse of the joint ranges $W = 1 / (q_{\max} - q_{\min})$
- it is NOT a “pseudoinverse” (4th relation does **not** hold),
but it shares similar properties



Singular Value Decomposition (SVD)

- the **SVD** routine of Matlab applied to J provides two orthonormal matrices $U_{M \times M}$ and $V_{N \times N}$, and a matrix $\Sigma_{M \times N}$ of the form

If you had a tall matrix
the sigmas are on
top and the zeros
are on the bottom

$$\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_M \end{pmatrix} \quad \begin{matrix} & & & \\ & & & \\ & & 0_{M \times (N-M)} & \\ & & & \end{matrix} \quad \begin{matrix} \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\rho > 0 \\ \sigma_{\rho+1} = \cdots = \sigma_M = 0 \\ \text{singular values of } J \end{matrix}$$

where $\rho = \text{rank}(J) \leq M$, so that their product is

so the number of non zero sigmas
is the rank of J

$$J = U \Sigma V^T$$

Proof of the first
statement is on
xournal++

- the columns of U are eigenvectors of $J J^T$ (associated to its non-negative eigenvalues σ_i^2), the columns of V are eigenvectors of $J^T J$
- the last $N - \rho$ columns of V are a basis for the **null space** of J

$$Jv_i = \sigma_i u_i \quad (i = 1, \dots, \rho)$$

$$Jv_i = 0 \quad (i = \rho + 1, \dots, N)$$



Computation of pseudoinverses

- show that the pseudoinverse of J is equal to

$$J = U\Sigma V^T \quad \Rightarrow \quad J^\# = V\Sigma^\# U^T \quad \Sigma^\# = \begin{pmatrix} \frac{1}{\sigma_1} & & & \\ & \ddots & & \\ & & \frac{1}{\sigma_\rho} & \\ \hline & & & 0_{(M-\rho) \times (M-\rho)} \\ & & & 0_{(N-M) \times M} \end{pmatrix}$$

for any rank ρ of J

- show that matrix $J_W^\#$ appears when solving the constrained linear-quadratic (LQ) optimization problem (with $W > 0$, symmetric, and assuming J of full rank)

$$\min \frac{1}{2} \|\dot{q}\|_W^2 \quad \text{s.t.} \quad J(q)\dot{q} - \dot{r} = 0$$

and that the pseudoinverse is a particular case for $W = I$

- show that a weighted pseudoinverse of J can be computed by SVD/pinv as

$$J_{aux} = JW^{-1/2} \quad J_W^\# = W^{-1/2} \text{pinv}(J_{aux})$$

applies **equally** to
square and non-square
matrices

Singularity robustness Damped Least Squares (DLS)



you want to have it
closer to a singularity
but you don't want mu when not
closer to a singularity since it
introduces an error

unconstrained
minimization
of a **suitable**
objective function

$$\min_{\dot{q}} H(\dot{q}) = \frac{\mu^2}{2} \|\dot{q}\|^2 + \frac{1}{2} \|\dot{r} - J\dot{q}\|^2$$

compromise between
large joint velocity
and task accuracy

the result is always a positive definite matrix and therefore invertible

SOLUTION $\dot{q} = J_{DLS}(q)\dot{r} = J^T(JJ^T + \mu^2 I_M)^{-1}\dot{r}$

- induces a **robust behavior** when crossing **singularities**, but in its basic version gives always a **task error** $\dot{e} = \mu^2(JJ^T + \mu^2 I_M)^{-1}\dot{r}$ (as for $N = M$)

- J_{DLS} is **not** a generalized inverse K

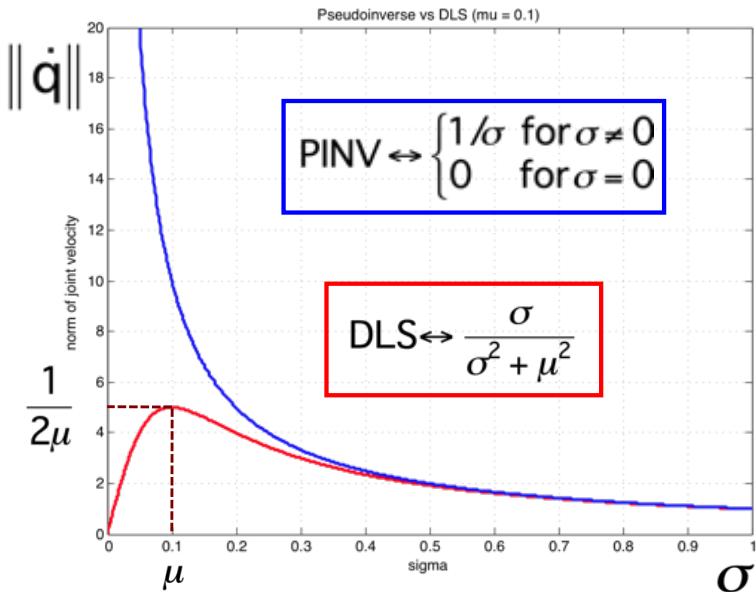
- using SVD: $J = U \Sigma V^T \Rightarrow J_{DLS} = V \Sigma_{DLS} U^T$, $\Sigma_{DLS} = \begin{pmatrix} \text{diag} \left\{ \frac{\sigma_i}{\sigma_i^2 + \mu^2} \right\} & \\ \rho \times \rho & 0_{(M-\rho) \times (M-\rho)} \end{pmatrix}$

- choice of a **variable damping factor** $\mu^2(q) \geq 0$, function of the minimum singular value $\sigma_\rho(q) > 0$ of $J \cong$ a measure of distance from a singularity (if $\rho = M$) or of further loss of rank (when $\rho < M$) when the singularity has gone you turn mu off putting it to zero, so to not have error

- numerical filtering:** introduces damping **only/mostly** in non-feasible directions for the task (see Maciejewski and Klein, *J of Rob Syst*, 1988)

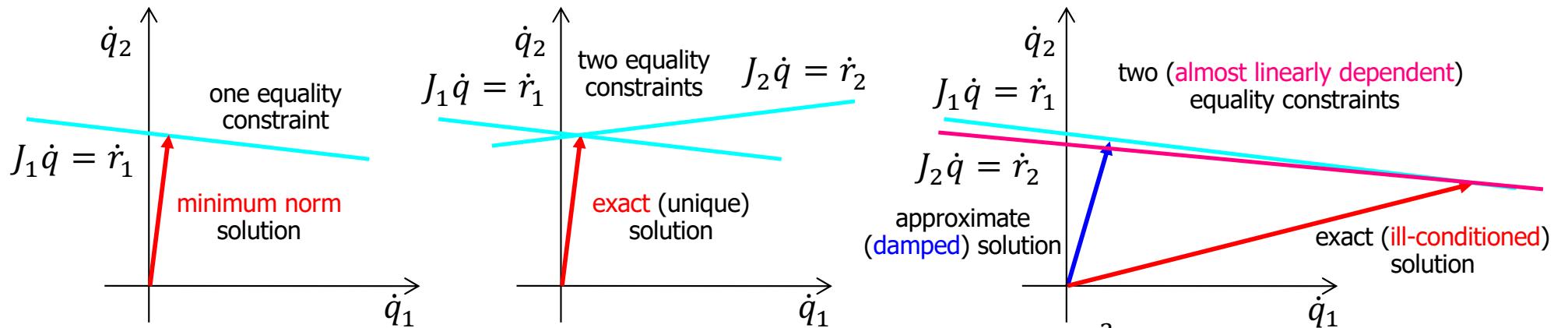


Behavior of DLS solution



- a. comparison of joint velocity norm with **PINV** (pseudoinverse) or **DLS** solutions
- in a task direction along a vector u of U , when the associated singular value $\sigma \rightarrow 0$
 - PINV** goes to infinity (and then is 0 at $\sigma = 0$)
 - DLS** peaks a value of $1/2\mu$ at $\sigma = \mu$ (and then smoothly goes to 0...)

- b. graphical interpretation of “damping” effect (here $M = N = 2$, for simplicity)

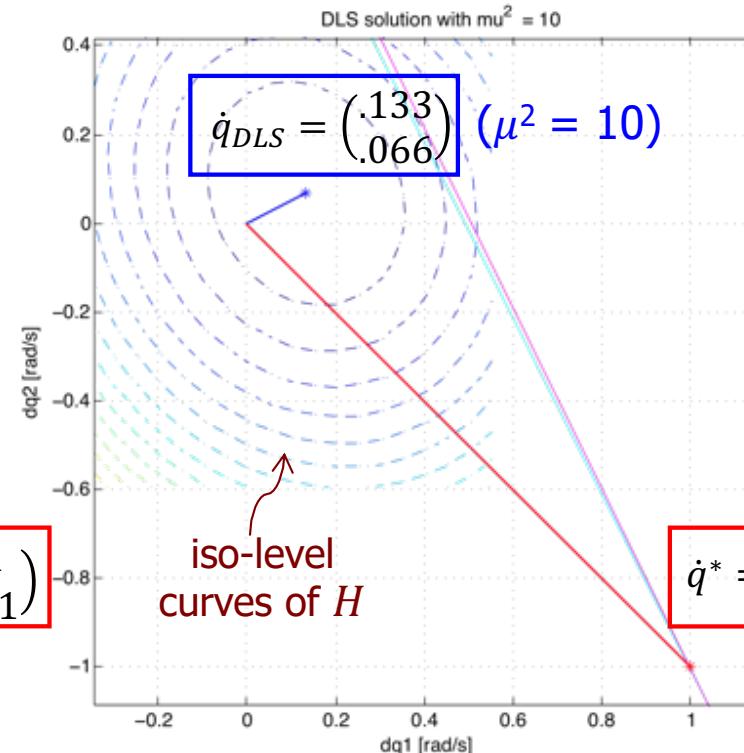
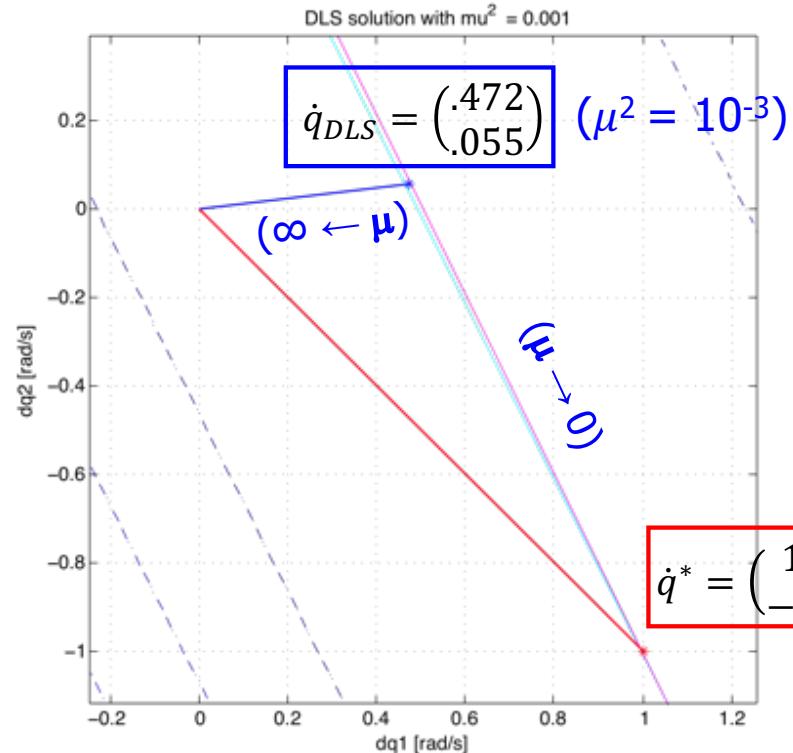


$$H(\dot{q}) = \frac{\mu^2}{2} \|\dot{q}\|^2 + \frac{1}{2} \|\dot{r} - J\dot{q}\|^2$$



Numerical example of DLS solution

planar 2R arm, unit links, close to (stretched) singular configuration $q_1 = 45^\circ$, $q_2 = 1.5^\circ$



$$\dot{r} = \begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

$\in \mathcal{R}(J)$ even
@singularity!



exact
solution
($\mu=0$)

$$H = \frac{\mu^2}{2} \|\dot{q}\|^2 + \frac{1}{2} \|\dot{r} - J\dot{q}\|^2$$

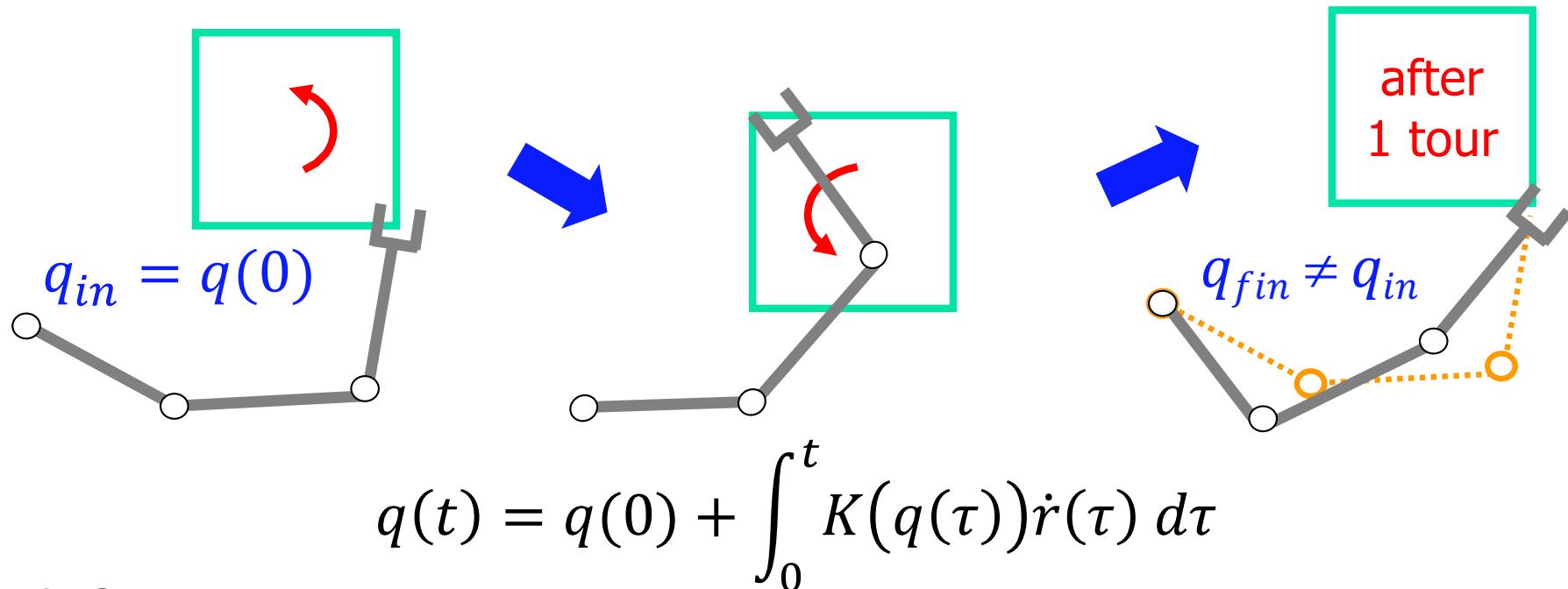
μ^2	0	10^{-4}	10^{-3}	10^{-2}	10
$\ \dot{q}\ $	$\sqrt{2}$.8954	.4755	.4467	.1490
$\ \dot{e}\ $	0	$6.6 \cdot 10^{-3}$	$1.4 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$.6668
H_{min}	0	$7.7 \cdot 10^{-5}$	$2.2 \cdot 10^{-4}$	$1.2 \cdot 10^{-3}$	$3.4 \cdot 10^{-1}$



Limits of Jacobian-based methods

- no guarantee that **singularities** are globally avoided during task execution
 - despite joint velocities are kept to a minimum, this is only a local property and “avalanche” phenomena may occur
- typically lead to **non-repeatable** motion in the joint space
 - cyclic motions in **task space** do not map to cyclic motions in **joint space**

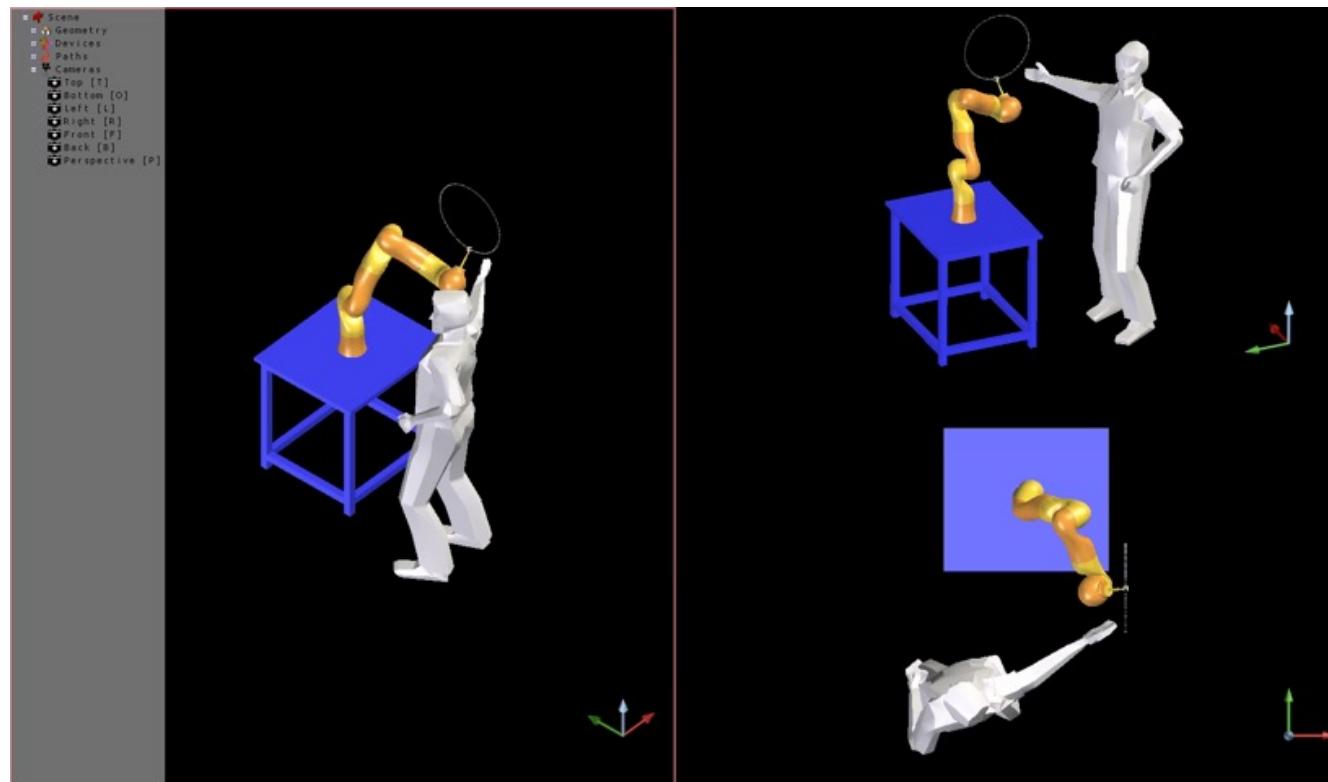
Infinite solutions since the robot is 1 degree redundant. If you have a constrained workspace, changing configuration you may collide with an object





Drift with Jacobian pseudoinverse

- a 7R KUKA LWR4 robot moves in the vicinity of a **human** operator
- we command a cyclic Cartesian path (only in position, $M = 3$), to be repeated **several** times using the **pseudoinverse** solution
- (**unexpected**) **collision** of a link occurs during the **third cycle** ...



video



2

Null-space methods

general solution of $J\dot{q} = \dot{r}$

$$\dot{q} = J^\# \dot{r} + (I - J^\# J) \dot{q}_0$$

a particular solution
(here, the pseudoinverse)
in $\mathcal{R}(J^T)$

“orthogonal” projection
of \dot{q}_0 in $\mathcal{N}(J)$

all solutions of the associated
homogeneous equation $J\dot{q} = 0$
(self-motions)

- symmetric
- idempotent: $[I - J^\# J]^2 = [I - J^\# J]$
- $[I - J^\# J]^\# = [I - J^\# J]$
- $J^\# \dot{r}$ is orthogonal to $[I - J^\# J] \dot{q}_0$

properties of
projector $[I - J^\# J]$

even more in general...

$$\dot{q} = K_1 \dot{r} + (I - K_2 J) \dot{q}_0$$

... but with less nice properties!

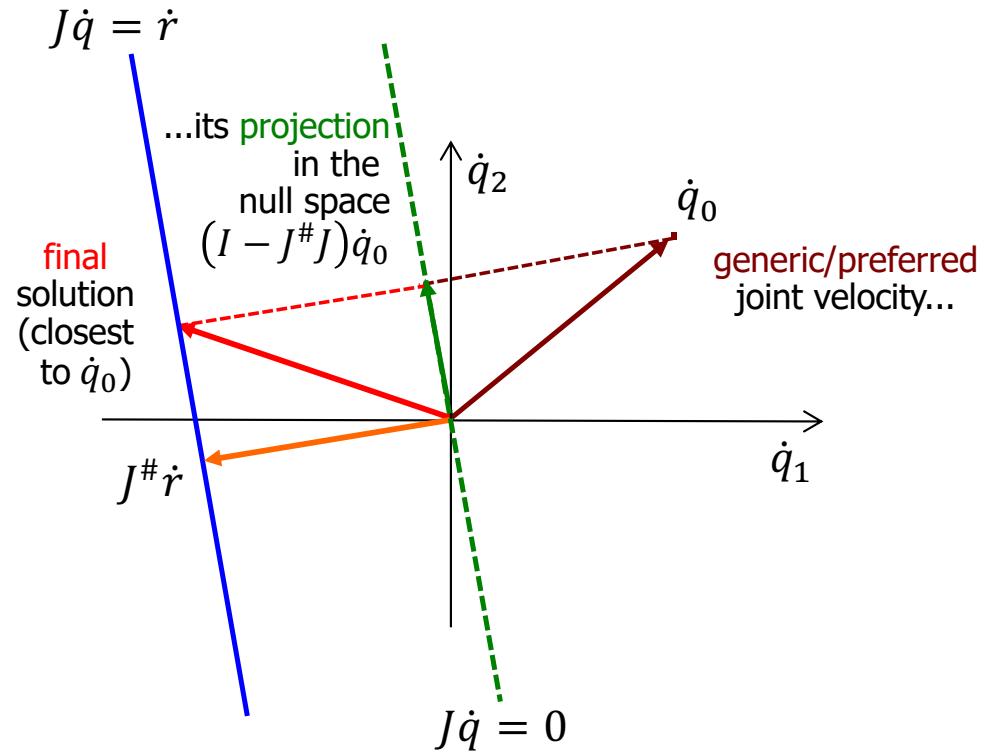
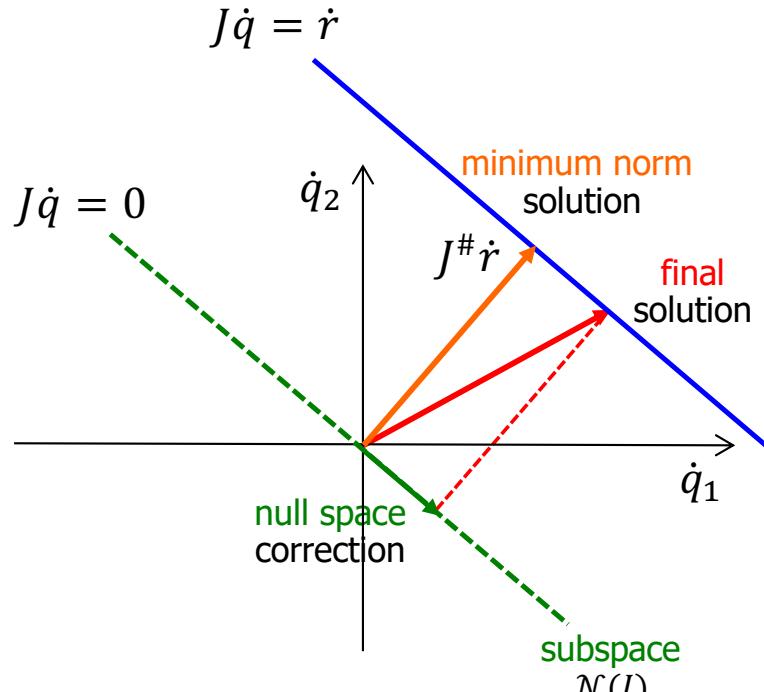
K_1, K_2 generalized
inverses of J
($J K_i J = J$)

how do we choose \dot{q}_0 ?



Geometric view on Jacobian null space

in the space of velocity commands



a correction is added to the original pseudoinverse (minimum norm) solution

- i) which is in the **null space** of the Jacobian
- ii) and possibly satisfies **additional criteria** or objectives



Linear-Quadratic Optimization

generalities

$$\begin{aligned} \min_x H(x) &= \frac{1}{2} (x - x_0)^T W (x - x_0) \\ \text{s.t. } Jx &= y \end{aligned}$$

M × N

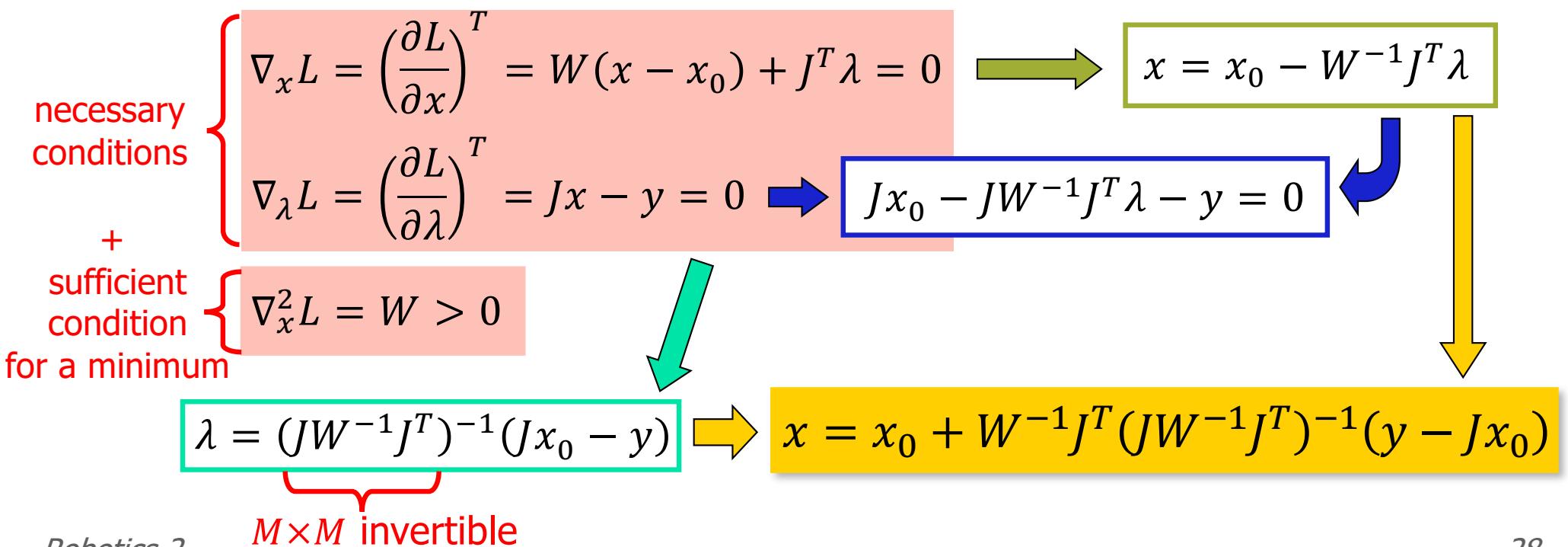
$$x \in \mathbb{R}^N$$

$$W > 0 \text{ (symmetric)}$$

$$y \in \mathbb{R}^M$$

$$\text{rank}(J) = \rho(J) = M$$

$$L(x, \lambda) = H(x) + \lambda^T (Jx - y) \leftarrow \text{Lagrangian (with multipliers } \lambda\text{)}$$





Linear-Quadratic Optimization

application to robot redundancy resolution

PROBLEM

$$\begin{aligned} \min_{\dot{q}} H(\dot{q}) &= \frac{1}{2} (\dot{q} - \dot{q}_0)^T W (\dot{q} - \dot{q}_0) \\ \text{s.t. } J\dot{q} &= \dot{r} \end{aligned}$$

\dot{q}_0 is a
“privileged”
joint velocity

SOLUTION

$$\dot{q} = \dot{q}_0 + W^{-1} J^T (J W^{-1} J^T)^{-1} (\dot{r} - J \dot{q}_0)$$

$$J_W^\#$$

$$\dot{q} = J_W^\# \dot{r} + (I - J_W^\# J) \dot{q}_0$$

minimum weighted norm
solution (for $\dot{q}_0 = 0$)

“projection” matrix in
the null-space $\mathcal{N}(J)$

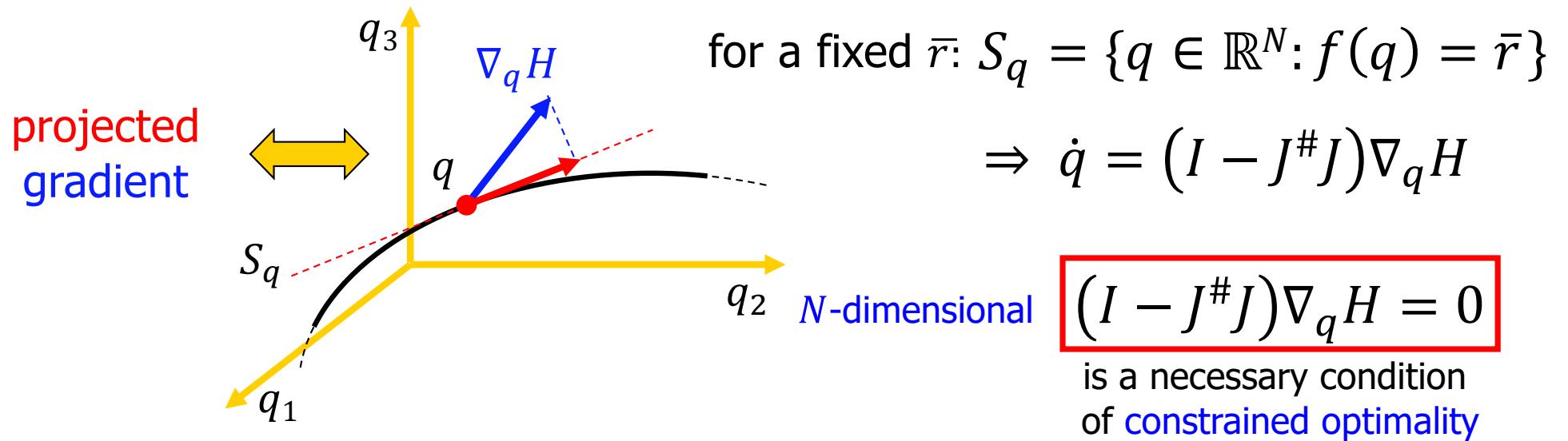


Projected Gradient (PG)

$$\dot{q} = J^\# \dot{r} + (I - J^\# J) \dot{q}_0$$

the choice $\dot{q}_0 = \nabla_q H(q)$ → differentiable objective function
 realizes one step of a constrained optimization algorithm

while executing the time-varying task $r(t)$
 the robot tries to increase the value of $H(q)$





Typical objective functions $H(q)$

- **manipulability** (maximize the “distance” from singularities)

$$H_{\text{man}}(q) = \sqrt{\det[J(q)J^T(q)]}$$

- **joint range** (minimize the “distance” from the mid points of the joint ranges)

$$\begin{aligned} q_i &\in [q_{m,i}, q_{M,i}] \\ \bar{q}_i &= \frac{q_{M,i} + q_{m,i}}{2} \end{aligned}$$

$$H_{\text{range}}(q) = \frac{1}{2N} \sum_{i=1}^N \left(\frac{q_i - \bar{q}_i}{q_{M,i} - q_{m,i}} \right)^2$$

\downarrow
 $\dot{q}_0 = -\nabla_q H(q)$

- **obstacle avoidance** (maximize the minimum distance to Cartesian obstacles)

also known as
“clearance”

$$H_{\text{obs}}(q) = \min_{\substack{a \in \text{robot} \\ b \in \text{obstacles}}} \|a(q) - b\|^2$$

potential difficulties due
to non-differentiability
(this is a max-min problem)

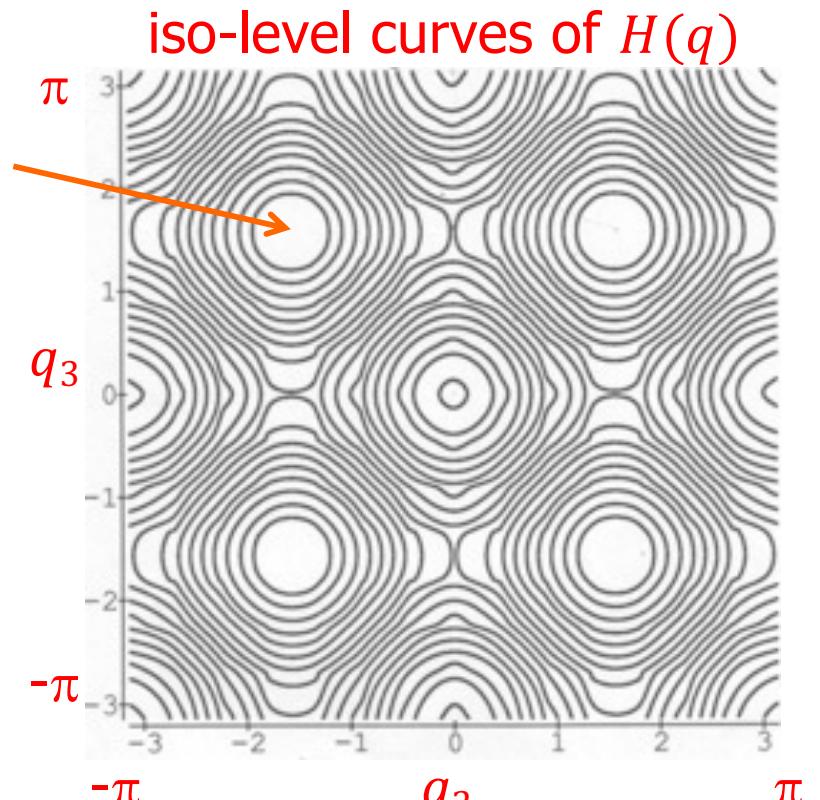
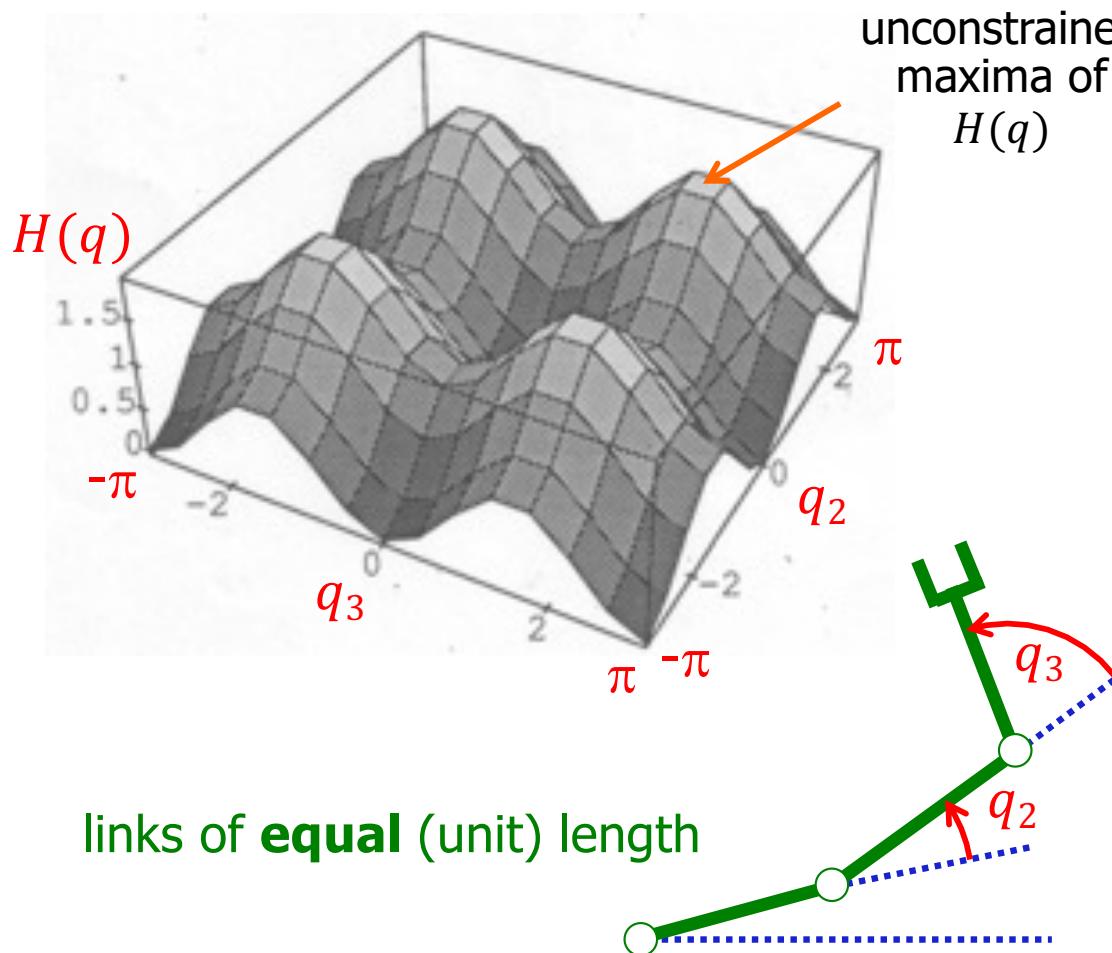


Singularities of planar 3R arm

the robot is redundant
for a positioning task
in the plane ($M = 2$)

$$H(q) = \sin^2 q_2 + \sin^2 q_3$$

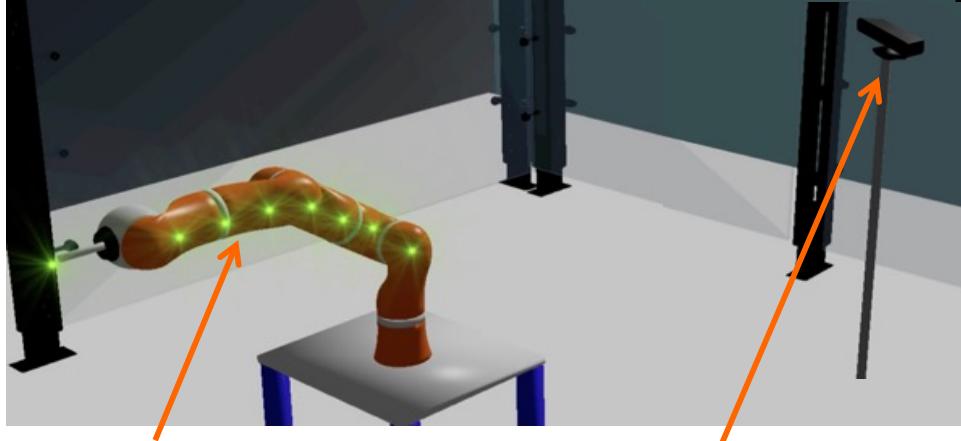
this H is **not** H_{man}
but has the **same** minima



independent from q_1 !



Minimum distance computation in human-robot interaction

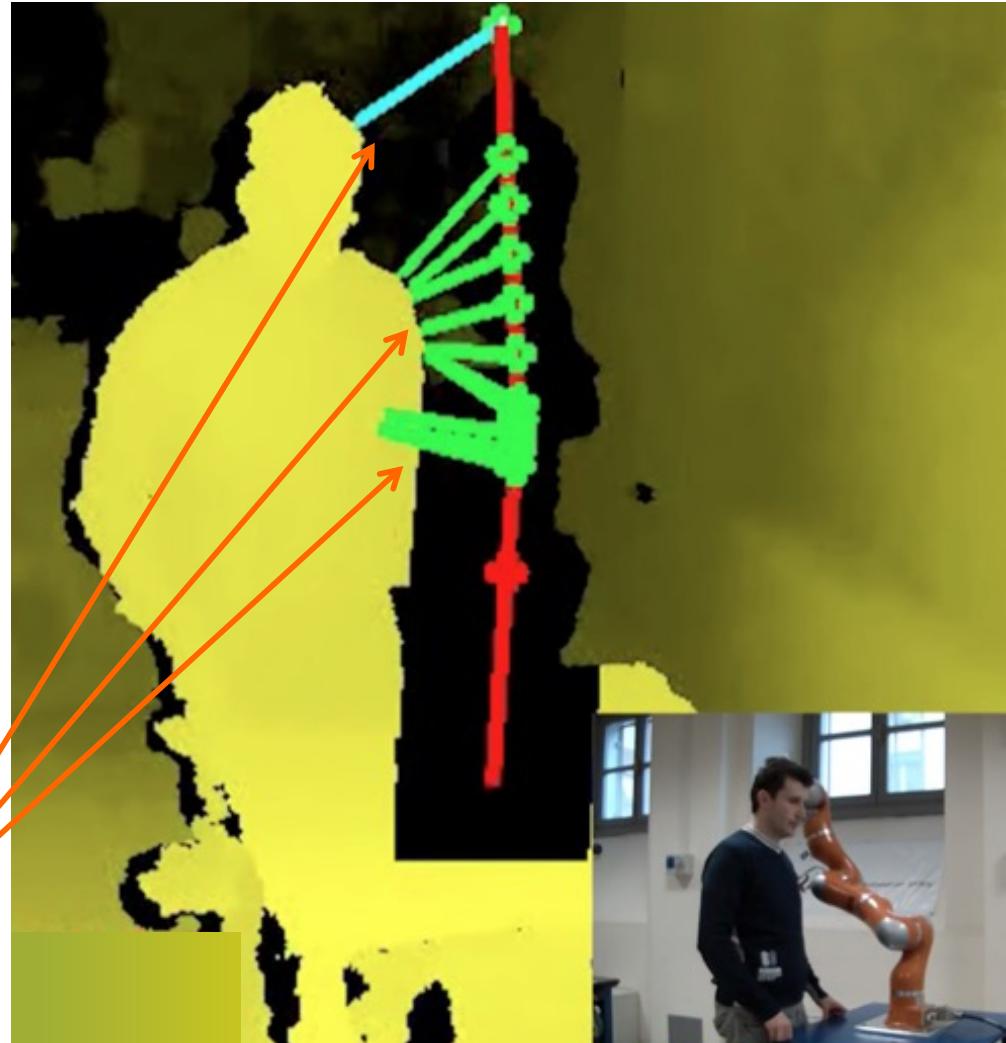


LWR4 robot with
a finite number of
control points $a(q)$
(8, including the E-E)

a Kinect sensor monitors
the workspace giving the
3D position of points b
on obstacles that are
fixed or moving
(like humans)

distances in 3D (and then the clearance)
are computed in this case as

$$\min_{\substack{a \in \{\text{control points}\} \\ b \in \text{human body}}} \|a(q) - b\|^2$$





Comments on null-space methods

- the projection matrix $(I - J^{\#}J)$ has dimension $N \times N$, but only rank $N - M$ (if J is full rank M), with some **waste of information**
- actual (efficient) evaluation of the solution

$$\dot{q} = J^{\#}\dot{r} + (I - J^{\#}J)\dot{q}_0 = \dot{q}_0 + J^{\#}(\dot{r} - J\dot{q}_0)$$

but the pseudoinverse is needed anyway, and this is **computationally intensive** (SVD in the general case)

- in principle, the additional complexity of a redundancy resolution method should depend only on the **redundancy degree $N - M$**
- a constrained optimization method is available, which is known to be more efficient than the projected gradient (PG) —at least when the **Jacobian has full rank ...**



Decomposition of joint space

- if $\rho(J(q)) = M$, there exists a **decomposition** of the set of joints (possibly, after a reordering)

$$q = \begin{pmatrix} q_a \\ q_b \end{pmatrix} \quad \begin{matrix} \text{\scriptsize } M \\ \text{\scriptsize } N - M \end{matrix}$$

such that $J_a(q) = \overbrace{\frac{\partial f}{\partial q_a}}^{M \times M}$ is nonsingular

- from the **implicit function theorem**, there exists an inverse function g

$$f(q_a, q_b) = r \quad \rightarrow \quad q_a = g(r, q_b)$$

$$\text{with } \frac{\partial g}{\partial q_b} = - \left(\frac{\partial f}{\partial q_a} \right)^{-1} \frac{\partial f}{\partial q_b} = -J_a^{-1}(q)J_b(q)$$

- the **$N - M$ variables q_b** can be selected **independently** (e.g., they are used for optimizing an objective function $H(q)$, “reduced” via the use of g to a **function of q_b only**)
- $q_a = g(r, q_b)$ is then chosen so as to correctly **execute the task**



Reduced Gradient (RG)

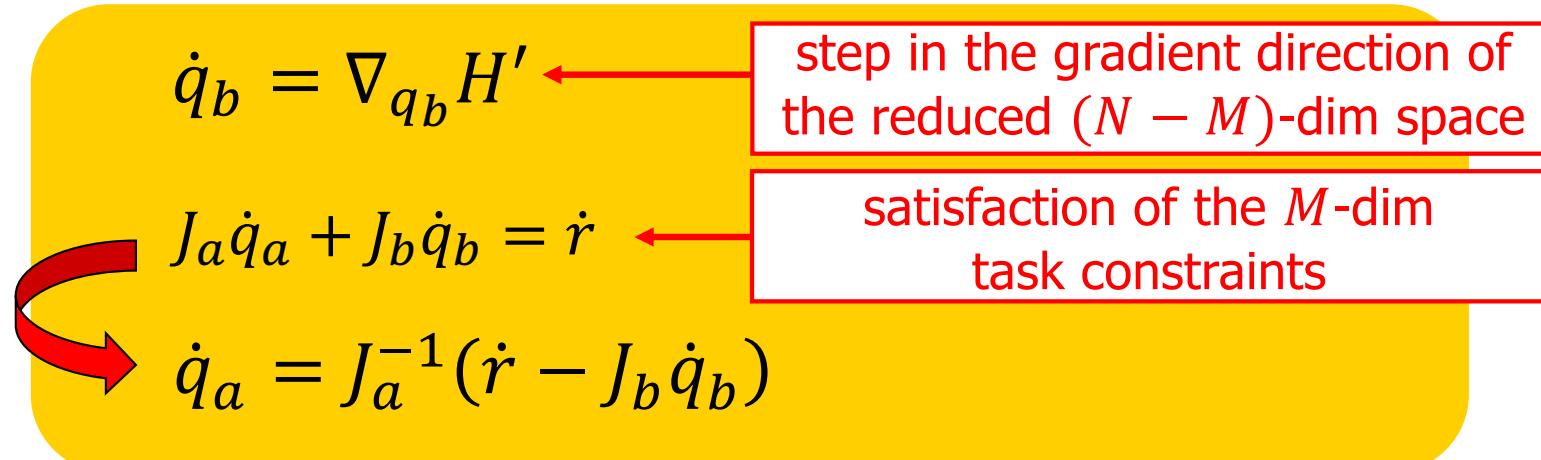
- $H(q) = H(q_a, q_b) = H(g(r, q_b), q_b) = H'(q_b)$, with r at **current** value
- the **Reduced Gradient** (w.r.t. q_b only, but still keeping the effects of this choice into account) is

$$\nabla_{q_b} H' = [- (J_a^{-1} J_b)^T \quad I_{N-M}] \nabla_q H$$

$(\neq \nabla_{q_b} H \text{ only}!!)$

$\boxed{\nabla_{q_b} H' = 0}$
is a “compact”
(i.e., $N - M$ dimensional)
necessary condition
of **constrained optimality**

- algorithm





Comparison between PG and RG

- Projected Gradient (PG)

$$\dot{q} = J^\# \dot{r} + (I - J^\# J) \nabla_q H$$

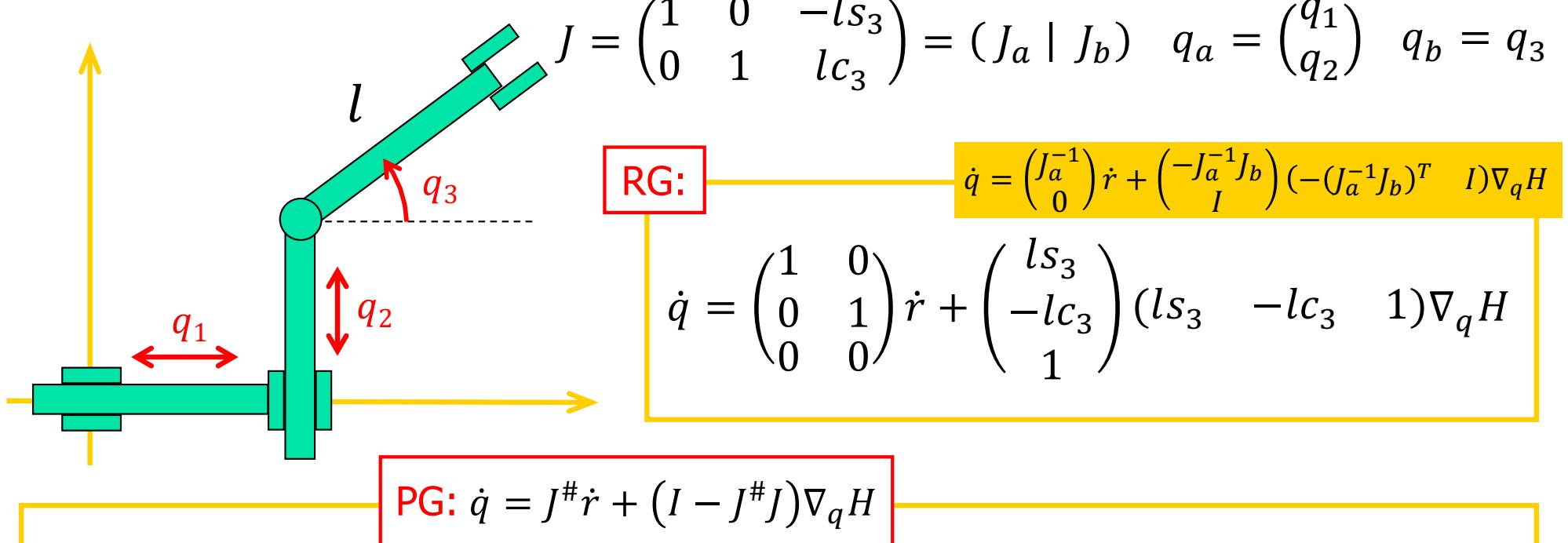
- Reduced Gradient (RG)

$$\dot{q} = \begin{pmatrix} \dot{q}_a \\ \dot{q}_b \end{pmatrix} = \begin{pmatrix} J_a^{-1} \\ 0 \end{pmatrix} \dot{r} + \begin{pmatrix} -J_a^{-1} J_b \\ I \end{pmatrix} (-J_a^{-1} J_b)^T \quad I \nabla_q H$$

- RG is **analytically simpler** and **numerically faster** than PG, but requires the search for a non-singular minor (J_a) of the robot Jacobian
- if $r = \text{cost}$ & $N - M = 1 \Rightarrow$ same (unique) direction for \dot{q} , but RG has automatically a **larger** optimization step size
- else \Rightarrow RG and PG methods provide always **different evolutions**



Analytic comparison PPR robot



$$J^\# = \frac{1}{1+l^2} \begin{pmatrix} 1+l^2c_3^2 & l^2s_3c_3 \\ l^2s_3c_3 & 1+l^2s_3^2 \\ -ls_3 & lc_3 \end{pmatrix} \quad I - J^\# J = \frac{1}{1+l^2} \begin{pmatrix} l^2s_3^2 & l^2s_3c_3 & ls_3 \\ l^2s_3c_3 & l^2c_3^2 & -lc_3 \\ ls_3 & -lc_3 & 1 \end{pmatrix}$$

always $< 1!!$



Joint range limits

$$q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \theta = T\theta$$

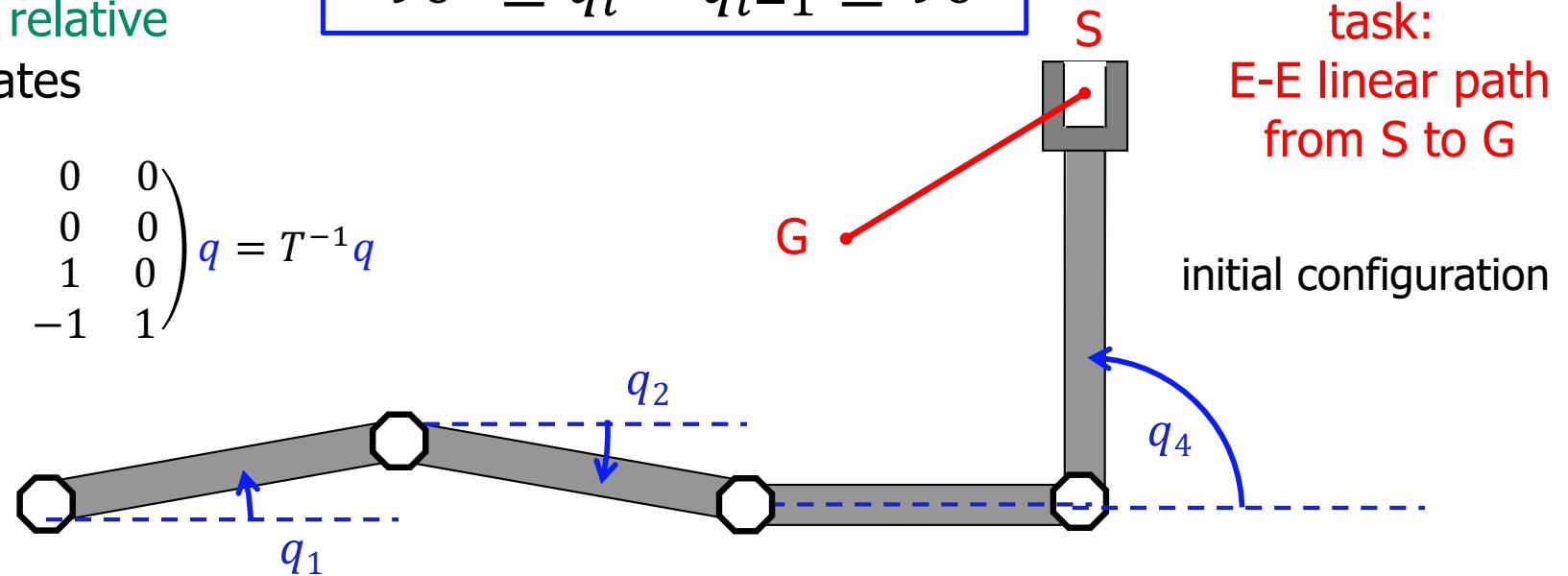
absolute \Leftrightarrow relative
coordinates

$$\theta = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} q = T^{-1}q$$

$$-90^\circ \leq \theta_i \leq 90^\circ$$

\Updownarrow

$$-90^\circ \leq q_i - q_{i-1} \leq 90^\circ$$

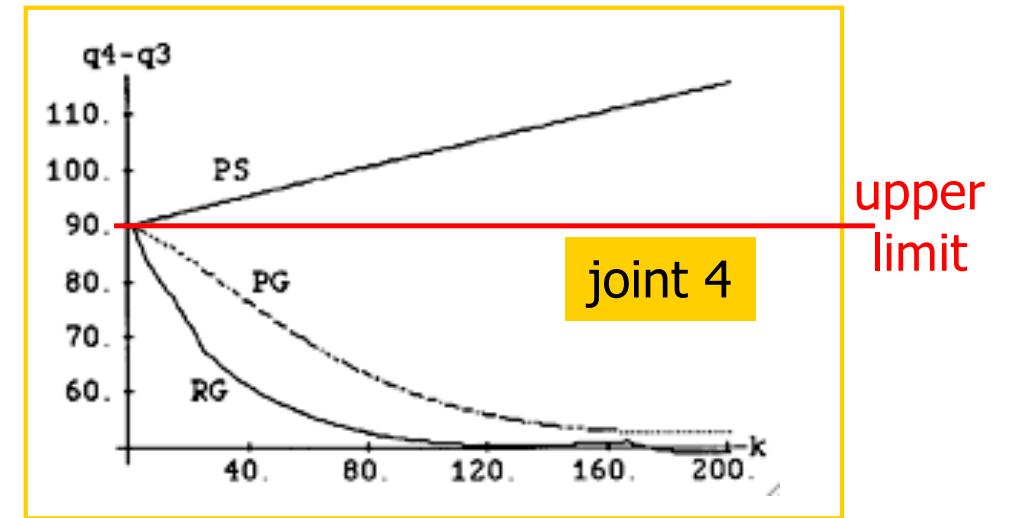
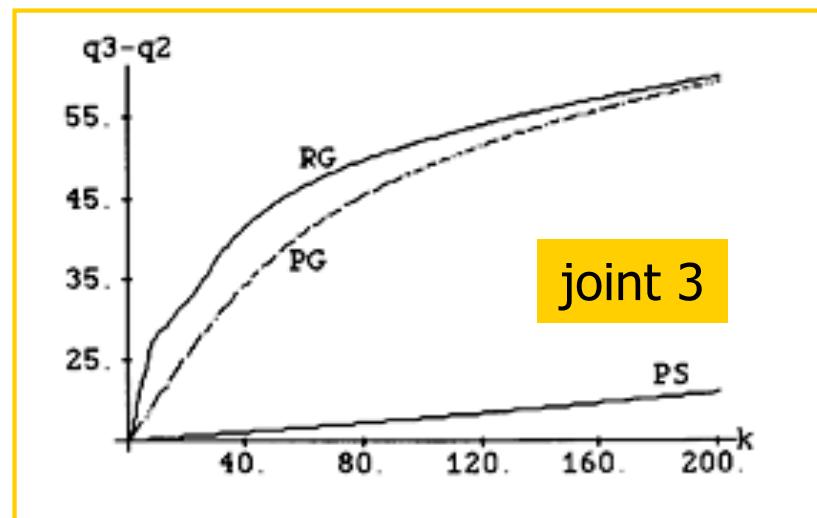
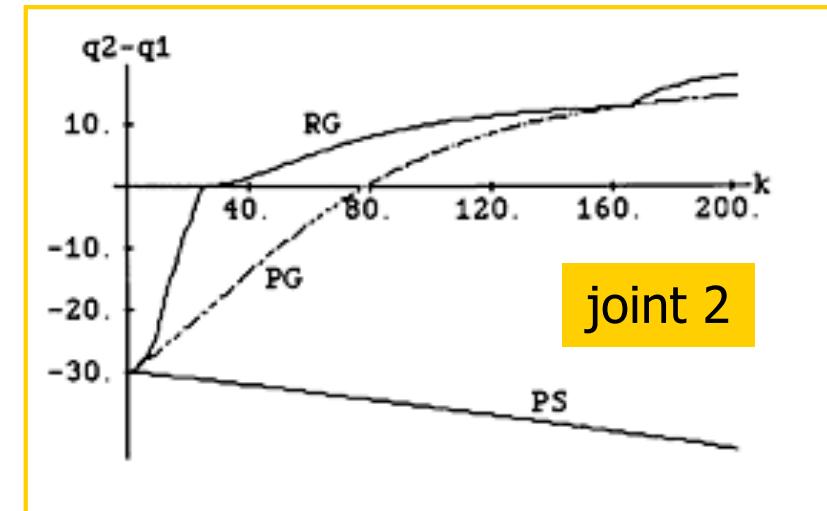
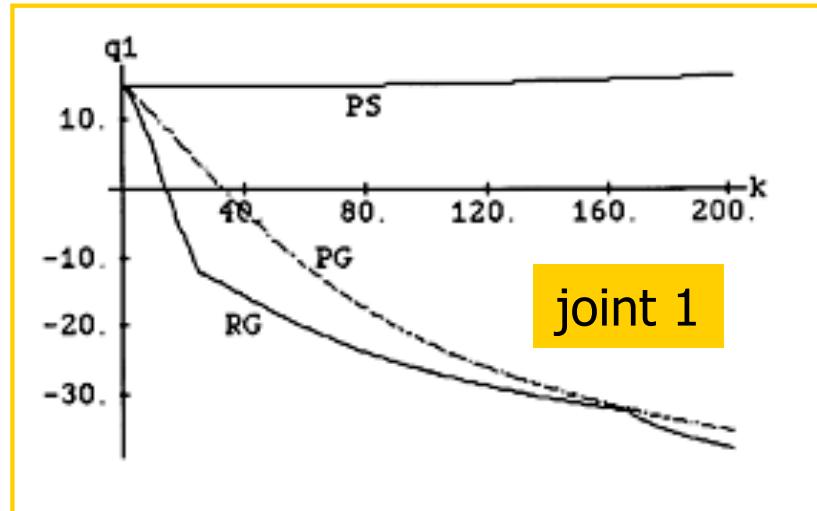


numerical comparison among pseudoinverse (PS),
projected gradient (PG), and reduced gradient (RG) methods



Numerical results

minimizing distance from mid joint range

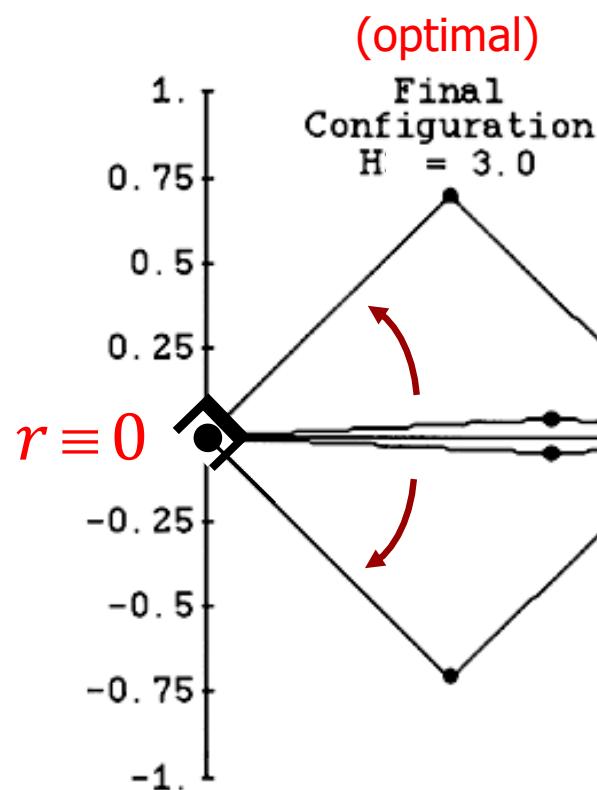


steps of numerical simulation



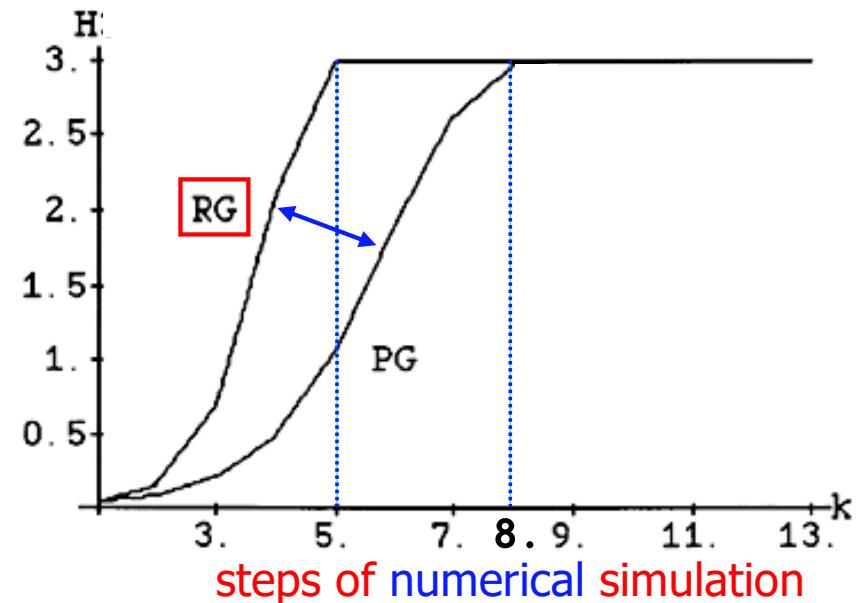
Numerical results

self-motion for escaping singularities



$$\max H(q) = \sum_{i=1}^3 \sin^2(q_{i+1} - q_i)$$

this function is **NOT**
the manipulability index,
but has the same minima (= 0)



RG is faster than PG
(keeping the same accuracy on r)



3 Task augmentation methods

- an **auxiliary task** is added (task augmentation)

$$S \updownarrow f_y(q) = y \quad S \leq N - M$$

corresponding to some desirable feature for the solution

$$r_A = \begin{pmatrix} r \\ y \end{pmatrix} = \begin{pmatrix} f(q) \\ f_y(q) \end{pmatrix} \rightarrow \dot{r}_A = \begin{pmatrix} J(q) \\ J_y(q) \end{pmatrix} \dot{q} = J_A(q) \dot{q}$$

$\boxed{J_A}$ $M + S$
 $\underbrace{}_N$

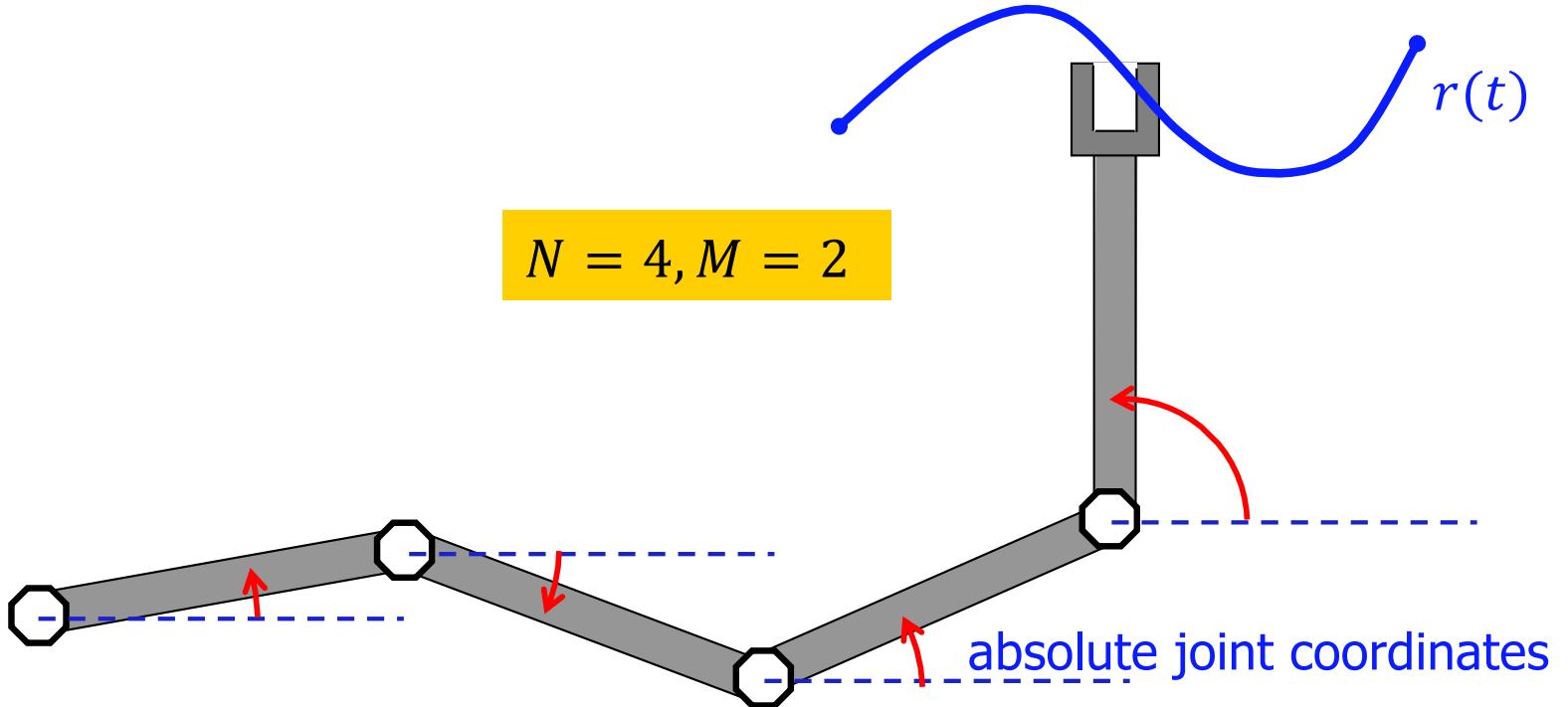
- a **solution** is chosen still in the form of a generalized inverse

$$\dot{q} = K_A(q) \dot{r}_A$$

or by adding a term in the null space of the **augmented Jacobian** matrix J_A



Augmented task example



$$f_y(q) = q_4 = \pi/2 \quad (S = 1)$$

last link is to be held vertical...



Augmenting the task ...

- **advantage:** better shaping of the inverse kinematic solution
- **disadvantage:** algorithmic singularities are introduced when

$$\rho(J) = M \quad \rho(J_y) = S \quad \text{but} \quad \rho(J_A) < M + S$$

to avoid this, it should be always

$$\mathcal{R}(J^T) \cap \mathcal{R}(J_y^T) = \emptyset$$

difficult to be obtained globally!

rows of J AND rows of J_y
are all together linearly independent



Extended Jacobian ($S = N - M$)

- square J_A : in the absence of **algorithmic** singularities, we can choose

$$\dot{q} = J_A^{-1}(q)\dot{r}_A$$

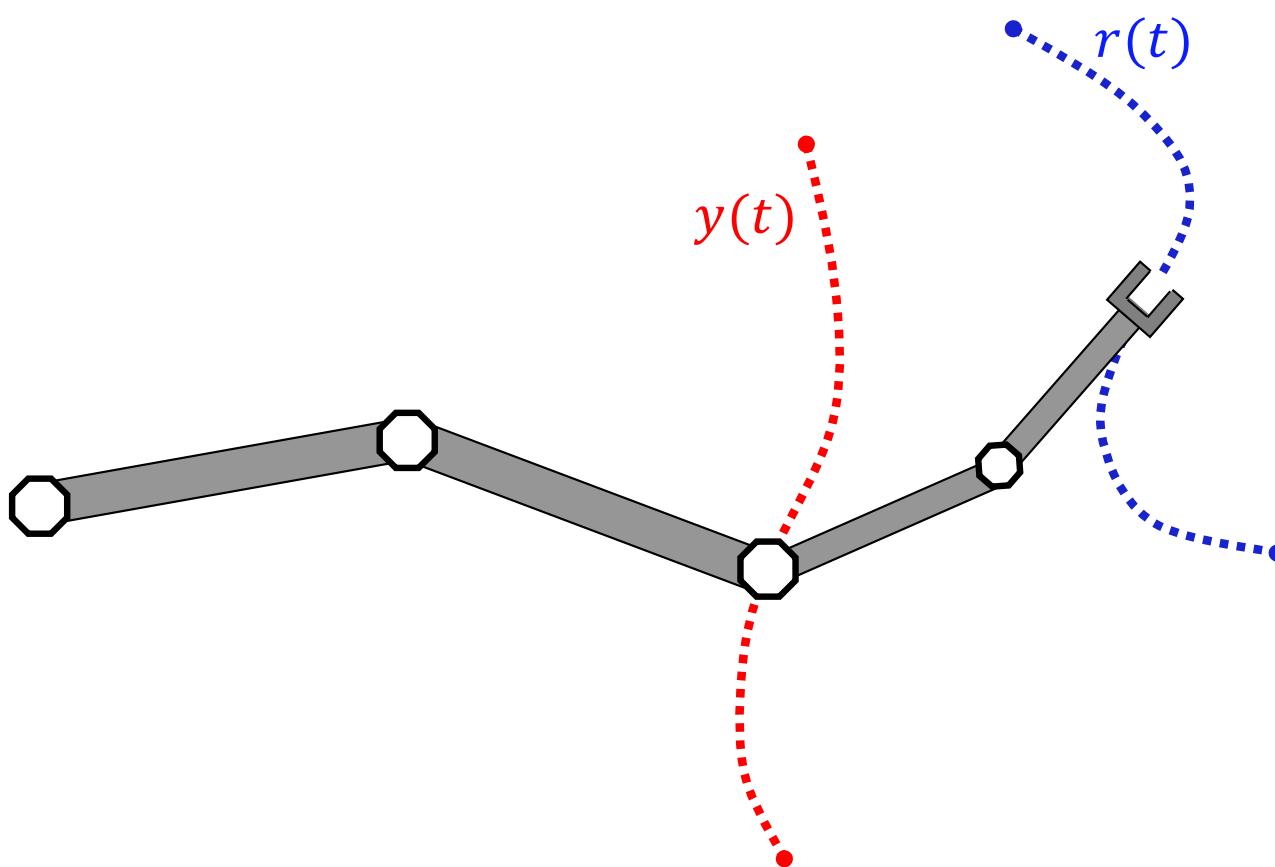
- the scheme is then **repeatable**
 - provided no singularities are encountered during a complete task cycle*
- when the $N - M$ conditions $f_y(q) = 0$ correspond to necessary (and sufficient) conditions for constrained optimality of a given objective function $H(q)$ (see RG method, slide #36), this scheme guarantees that constrained **optimality** is locally **preserved** during task execution
- in the vicinity of algorithmic singularities, for the simultaneous execution of the **original task** and the **auxiliary task(s)**, one can use the DLS method; however, **both** tasks will be affected by **errors**

* there exists an unexpected phenomenon in some 3R manipulators having “generic” kinematics: the robot may sometimes perform a pose change after a full cycle, even if no singularity has been encountered during motion (see J. Burdick, *Mech. Mach. Theory*, 30(1), 1995)



Extended Jacobian example

MACRO-MICRO manipulator



$$N = 4, M = 2$$

$$\dot{r} = J(q_1, \dots, q_4)\dot{q}$$

$$\dot{y} = J_y(q_1, q_2)\dot{q}$$



$$J_A = \left(\begin{array}{c|c} * & * \\ * & 0 \end{array} \right) \quad 4 \times 4$$



Task Priority

if the original (primary) task $\dot{r}_1 = J_1(q)\dot{q}$ has **higher priority** than the auxiliary (secondary) task $\dot{r}_2 = J_2(q)\dot{q}$

- we **first** address the task with highest priority

$$\dot{q} = J_1^\# \dot{r}_1 + (I - J_1^\# J_1) v_1$$

- and **then** choose v_1 so as to satisfy, if possible, also the secondary (lower priority) task

$$\dot{r}_2 = J_2 \dot{q} = J_2 J_1^\# \dot{r}_1 + J_2 (I - J_1^\# J_1) v_1 = J_2 J_1^\# \dot{r}_1 + J_2 P_1 v_1$$

the general solution for v_1 takes the usual form

this second term is an extra term

$$v_1 = (J_2 P_1)^\# (\dot{r}_2 - J_2 J_1^\# \dot{r}_1) + (I - (J_2 P_1)^\# (J_2 P_1)) v_2$$

v_2 is available for the execution of further tasks of lower (ordered) priorities



Task Priority (continue)

- substituting the expression of v_1 in \dot{q}

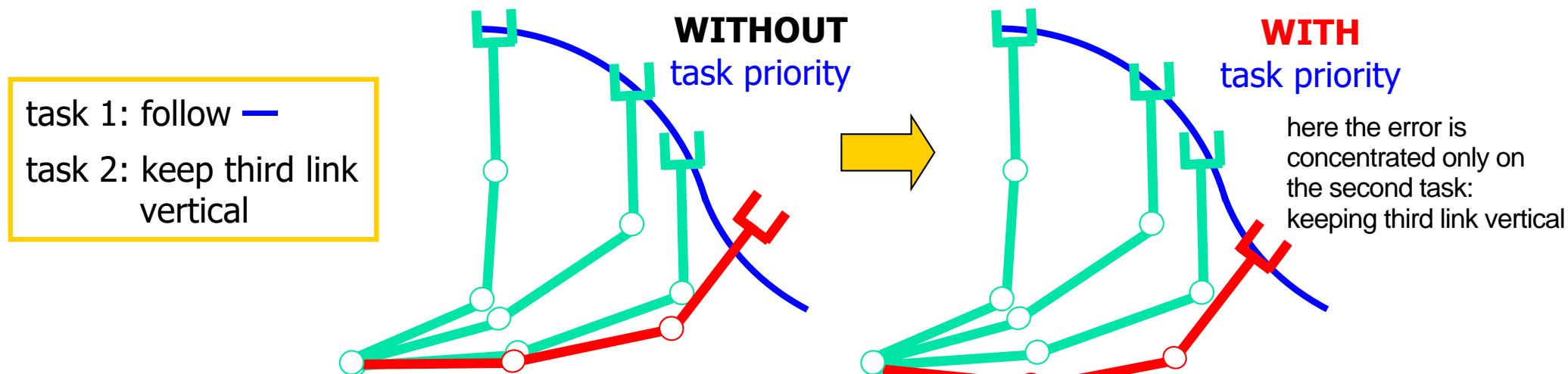
$$\dot{q} = J_1^\# \dot{r}_1 + P_1 (J_2 P_1)^\# (\dot{r}_2 - J_2 J_1^\# \dot{r}_1) + P_1 (I - (J_2 P_1)^\# (J_2 P_1)) v_2$$

$P(BP)^\# = (BP)^\#$
 when matrix P is
 idempotent and symmetric

$= (J_2 P_1)^\#$

possibly $= 0$

- main advantage: highest priority task is ideally no longer affected by algorithmic singularities (error is restricted only to secondary task)





Robotics 2

Robots with kinematic redundancy

Part 2: Extensions

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





A general task priority formulation

- consider a **large** number p of tasks to be executed **at best** and **with strict priorities** by a robotic system having **many** dofs
- everything should run efficiently in real time, with possible **addition**, **deletion**, **swap**, or **reordering** of tasks
- a **recursive** formulation that reduces computations is convenient

$$\dot{q} \in \mathbb{R}^n \quad \dot{r}_k \in \mathbb{R}^{m_k} \quad \dot{r}_k = J_k(q)\dot{q} \quad k = 1, \dots, p$$

k-th task

$$P_k(q) = I - J_k^\#(q)J_k(q)$$

projector in the null-space of k -th task

$i < j \Rightarrow$ task i has higher priority than task j

$$\sum_{k=1}^p m_k = m (\leq n)$$

even larger!

$$\dot{r}_{A,k} = \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \vdots \\ \dot{r}_k \end{pmatrix} \quad J_{A,k} = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_k \end{pmatrix}$$

stack of first k tasks

augmented Jacobian of first k tasks

$$P_{A,k} = I - J_{A,k}^\# J_{A,k}$$

projector in the null-space of the augmented Jacobian of the first k tasks

$$J_i P_{A,k} = O \quad \forall i \leq k$$

$\iff J_{A,k} P_{A,k} = O$



Recursive solution with priorities - 1

- start with the first task and **reformulate** the problem so as to provide **always** a “solution”, at least in terms of **minimum error norm**

for $k = 1$

$$\begin{cases} \dot{\mathbf{q}}_1 = \arg \min_{\dot{\mathbf{q}} \in \mathbb{R}^n} \frac{1}{2} \|\dot{\mathbf{q}}\|^2 \\ \text{s.t. } J_1 \dot{\mathbf{q}} = \dot{\mathbf{r}}_1 \end{cases} \xrightarrow{\quad} \begin{cases} \dot{\mathbf{q}}_1 = \arg \min_{\dot{\mathbf{q}} \in \mathcal{S}_1} \frac{1}{2} \|\dot{\mathbf{q}}\|^2 \\ \mathcal{S}_1 = \left\{ \arg \min_{\dot{\mathbf{q}} \in \mathbb{R}^n} \frac{1}{2} \|J_1 \dot{\mathbf{q}} - \dot{\mathbf{r}}_1\|^2 \right\} \end{cases}$$

$$\xrightarrow{\quad} \dot{\mathbf{q}}_1 = J_1^\# \dot{\mathbf{r}}_1 \quad \xrightarrow{\quad} \mathcal{S}_1 = \{\dot{\mathbf{q}}_1 + P_1 \mathbf{v}_1, \mathbf{v}_1 \in \mathbb{R}^n\}$$

for $k = 2$

$$\begin{cases} \dot{\mathbf{q}}_2 = \arg \min_{\dot{\mathbf{q}} \in \mathcal{S}_2} \frac{1}{2} \|\dot{\mathbf{q}}\|^2 \\ \mathcal{S}_2 = \left\{ \arg \min_{\dot{\mathbf{q}} \in \mathcal{S}_1} \frac{1}{2} \|J_2 \dot{\mathbf{q}} - \dot{\mathbf{r}}_2\|^2 \right\} \end{cases} \xrightarrow{\quad} \begin{aligned} \dot{\mathbf{q}}_2 &= \dot{\mathbf{q}}_1 + (J_2 P_1)^\# (\dot{\mathbf{r}}_2 - J_2 \dot{\mathbf{q}}_1) \\ \mathcal{S}_2 &= \{\dot{\mathbf{q}}_2 + P_{A,2} \mathbf{v}_2, \mathbf{v}_2 \in \mathbb{R}^n\} \end{aligned}$$



Recursive solution with priorities - 2

generalizing to step k

\dot{q}_{k-1}
prioritized solution
up to task $k-1$

$$\mathcal{S}_{k-1} = \{\dot{q}_{k-1} + P_{A,k-1} v_{k-1}, v_{k-1} \in \mathbb{R}^n\}$$

set of all solutions up to task $k-1$

LQ problem
for k -th task

$$\left\{ \begin{array}{l} \dot{q}_k = \arg \min_{\dot{q} \in \mathcal{S}_k} \frac{1}{2} \|\dot{q}\|^2 \\ \mathcal{S}_k = \left\{ \arg \min_{\dot{q} \in \mathcal{S}_{k-1}} \frac{1}{2} \|J_k \dot{q} - \dot{r}_k\|^2 \right\} \end{array} \right.$$

recursive formula
(Siciliano, Slotine:
ICAR 1991)

$$\dot{q}_k = \dot{q}_{k-1} + (J_k P_{A,k-1})^\# (\dot{r}_k - J_k \dot{q}_{k-1})$$

prioritized
solution
up to task k

correction needed when
the solution up to task $k-1$
is not satisfying also task k

over the steps, the search set
is progressively reduced

$$\Leftrightarrow \mathbb{R}^n = \mathcal{S}_0 \supseteq \mathcal{S}_1 \supseteq \dots \supseteq \mathcal{S}_{p-1} \supseteq \mathcal{S}_p$$

initialization

$$\begin{aligned} \dot{q}_0 &= 0 \\ P_{A,0} &= I \end{aligned}$$



Recursive solution with priorities

properties and implementation

- the solution considering the first k tasks with their priority

$$\dot{\mathbf{q}}_k = \dot{\mathbf{q}}_{k-1} + (\mathbf{J}_k \mathbf{P}_{A,k-1})^\# (\dot{\mathbf{r}}_k - \mathbf{J}_k \dot{\mathbf{q}}_{k-1})$$

satisfies also ("does not perturb") the previous $k - 1$ tasks

$$\mathbf{J}_{A,k-1} \dot{\mathbf{q}}_k = \mathbf{J}_{A,k-1} \dot{\mathbf{q}}_{k-1}$$

since

$$\mathbf{J}_{A,k-1} \underbrace{(\mathbf{J}_k \mathbf{P}_{A,k-1})^\#}_{=} = \mathbf{J}_{A,k-1} \underbrace{\mathbf{P}_{A,k-1} (\mathbf{J}_k \mathbf{P}_{A,k-1})^\#}_{=} = \mathbf{O}$$

(Maciejewski, Klein: IJRR 1985): check the four defining properties of a pseudoinverse

- recursive expression also for the null-space projector

$$\boxed{\mathbf{P}_{A,k} = \mathbf{P}_{A,k-1} - (\mathbf{J}_k \mathbf{P}_{A,k-1})^\# \mathbf{J}_k \mathbf{P}_{A,k-1}} \quad \mathbf{P}_{A,0} = \mathbf{I}$$

(Baerlocher, Boulic: IROS 1998): for the proof, see Appendix A

- when the k -th task is (close to be) incompatible with the previous ones (**algorithmic singularity**), use "DLS" instead of "#" in k -th solution...



A list of extensions

- up to now, only “basic” redundancy resolution schemes
 - defined at **first-order** differential level (velocity)
 - it is possible to work in **acceleration**
 - useful for obtaining **smoother** motion
 - allows including the consideration of **dynamics**
 - seen within a **planning**, not a **control** perspective what if we have a perturbation at the initial point, so we don't start on top of the task?
 - take into account and recover errors in task execution by using **kinematic control** schemes
 - applied to robot manipulators with **fixed base**
 - extend to **wheeled mobile manipulators**
 - tasks specified only by **equality constraints**
 - add also **linear inequalities** in a complete QP formulation
 - very common also for **humanoid robots** in multiple tasks
 - consider **hard limits** in joint/command space



Resolution at acceleration level

$$r = f(q) \rightarrow \dot{r} = J(q)\dot{q} \rightarrow \ddot{r} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$$

- rewritten in the form

$$J(q)\ddot{q} = \ddot{r} - \dot{J}(q)\dot{q} \triangleq \ddot{x}$$

to be chosen given
 (at time t) known q, \dot{q}
 (at time t)

the problem is formally equivalent to the previous one,
with **acceleration** in place of velocity commands

- for instance, in the null-space method

$$\ddot{q} = \underbrace{J^\#(q)\ddot{x}}_{\text{solution with minimum acceleration norm } \|\ddot{q}\|^2} + \underbrace{(I - J^\#(q)J(q))\ddot{q}_0}_{\text{solution with minimum norm } \|\ddot{q} - \ddot{q}_0\|^2}$$

“preferred”
 acceleration
 to **damp/stabilize**
 joint motion
 in the null space
 $(K_D > 0)$

$q_{ddot} = -K_d q_{dot}$
 positive velocity \rightarrow decelerate
 negative velocity \rightarrow accelerate
 so it brings to $q_{dot} = 0$, damping velocity in a decoupled way (joint by joint)
 So it's perfect to stop the robot when the task are finished



Dynamic redundancy resolution

- dynamic model of a robot manipulator (more later!)

$$M(q)\ddot{q} + n(q, \dot{q}) = \tau \quad J(q)\ddot{q} = \ddot{x} (= \ddot{r} - j(q)\dot{q})$$

↑ ↑ ↑
 $N \times N$ symmetric
inertia matrix,
positive **definite** for all q input torque vector
(provided by the motors) M -dimensional
acceleration task
Coriolis/centrifugal vector $c(q, \dot{q})$
+ gravity vector $g(q)$

- we can formulate and solve interesting dynamic problems in the general framework of LQ optimization^(o)
 - closed-form expressions can be obtained by the solution formula^(o) (assuming a full rank Jacobian J)

(o) in block *Kinematic redundancy - Part 1*, slide #28



Dynamic redundancy resolution

as Linear-Quadratic optimization problems

- typical **dynamic** objectives to be **locally minimized** at (q, \dot{q})

torque norm

$$H_1(\ddot{q}) = \frac{1}{2} \|\tau\|^2 = \frac{1}{2} \ddot{q}^T M^2(q) \ddot{q} + n^T(q, \dot{q}) M(q) \ddot{q} + \frac{1}{2} n^T(q, \dot{q}) n(q, \dot{q})$$

(squared inverse inertia weighted) torque norm

$$\begin{aligned} H_2(\ddot{q}) &= \frac{1}{2} \|\tau\|_{M^{-2}}^2 = \frac{1}{2} \tau^T M^{-2}(q) \tau \\ &= \frac{1}{2} \ddot{q}^T \ddot{q} + n^T(q, \dot{q}) M^{-1}(q) \ddot{q} + \frac{1}{2} n^T(q, \dot{q}) M^{-2}(q) n(q, \dot{q}) \end{aligned}$$

(inverse inertia weighted) torque norm

$$\begin{aligned} H_3(\ddot{q}) &= \frac{1}{2} \|\tau\|_{M^{-1}}^2 = \frac{1}{2} \tau^T M^{-1}(q) \tau \\ &= \frac{1}{2} \ddot{q}^T M(q) \ddot{q} + n^T(q, \dot{q}) \ddot{q} + \frac{1}{2} n^T(q, \dot{q}) M^{-1}(q) n(q, \dot{q}) \end{aligned}$$



Closed-form solutions

minimum torque norm solution

$$\frac{1}{2} \|\tau\|^2 \rightarrow \tau_1 = (J(q)M^{-1}(q))^{\#}(\ddot{r} - \dot{J}(q)\dot{q} + J(q)M^{-1}(q)n(q, \dot{q}))$$

- good for **short** trajectories (in fact, it is still only a “local” solution!)
- for **longer** trajectories it leads to torque “oscillation/explosion” (**whipping** effect)

minimum (squared inverse inertia weighted) torque norm solution

$$\frac{1}{2} \|\tau\|_{M^{-2}}^2 \rightarrow \tau_2 = M(q)J^{\#}(q)(\ddot{r} - \dot{J}(q)\dot{q} + J(q)M^{-1}(q)n(q, \dot{q}))$$

- good performance in general, to be preferred

minimum (inverse inertia weighted) torque norm solution

$$\frac{1}{2} \|\tau\|_{M^{-1}}^2 \rightarrow \tau_3 = J^T(q)(J(q)M^{-1}(q)J^T(q))^{-1}(\ddot{r} - \dot{J}(q)\dot{q} + J(q)M^{-1}(q)n(q, \dot{q}))$$

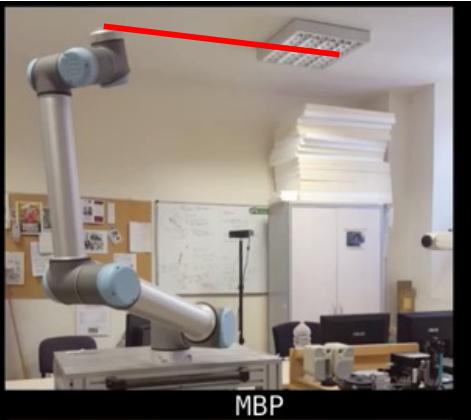
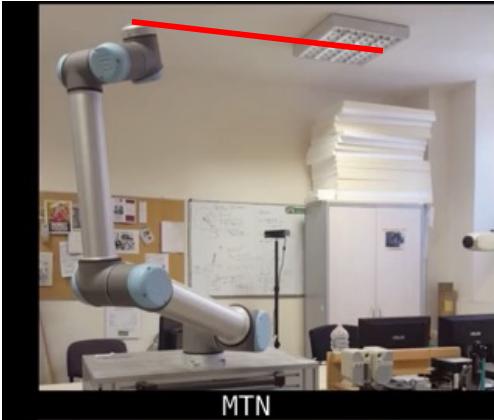
- a solution with a **leading $J^T(q)$** term: what is its nice physical interpretation?

May we add also a term τ_0 in a (dynamic) null space? Easy to do in the LQ framework!



Stabilizing the minimum torque solution

Universal
Robots
UR-10
(6-dof)



video

$$\min \frac{1}{2} \|\tau\|^2 = \text{MTN}$$

versus

video

KUKA
LRW 4
(7-dof,
last joint
not used)

Stable Torque Optimization for Redundant Robots using a Short Preview

K. Al Khudir, G. Halvorsen, L. Lanari, A. De Luca

Robotics Lab, DIAG
Sapienza Università di Roma

September 2018

- MBP = minimizing torque also at a short preview instant
- MTND = damping joint velocity in the null space
- MBPD = ... do both

IEEE Robotics and Automation Lett. 2019



Kinematic control

- given a desired M -dimensional task $r_d(t)$, in order to recover a task error $e = r_d - r$ due to initial mismatch or due to
 - disturbances
 - inherent linearization error in using the Jacobian (first-order motion)
 - discrete-time implementation

we need to “close” a **feedback loop on task execution**, by replacing (with diagonal matrix gains $K > 0$ or $K_P, K_D > 0$)

$$\dot{r} \xrightarrow{\quad\textcolor{red}{\longrightarrow}\quad} \dot{r}_d + K(r_d - r) \qquad \text{in velocity-based...}$$

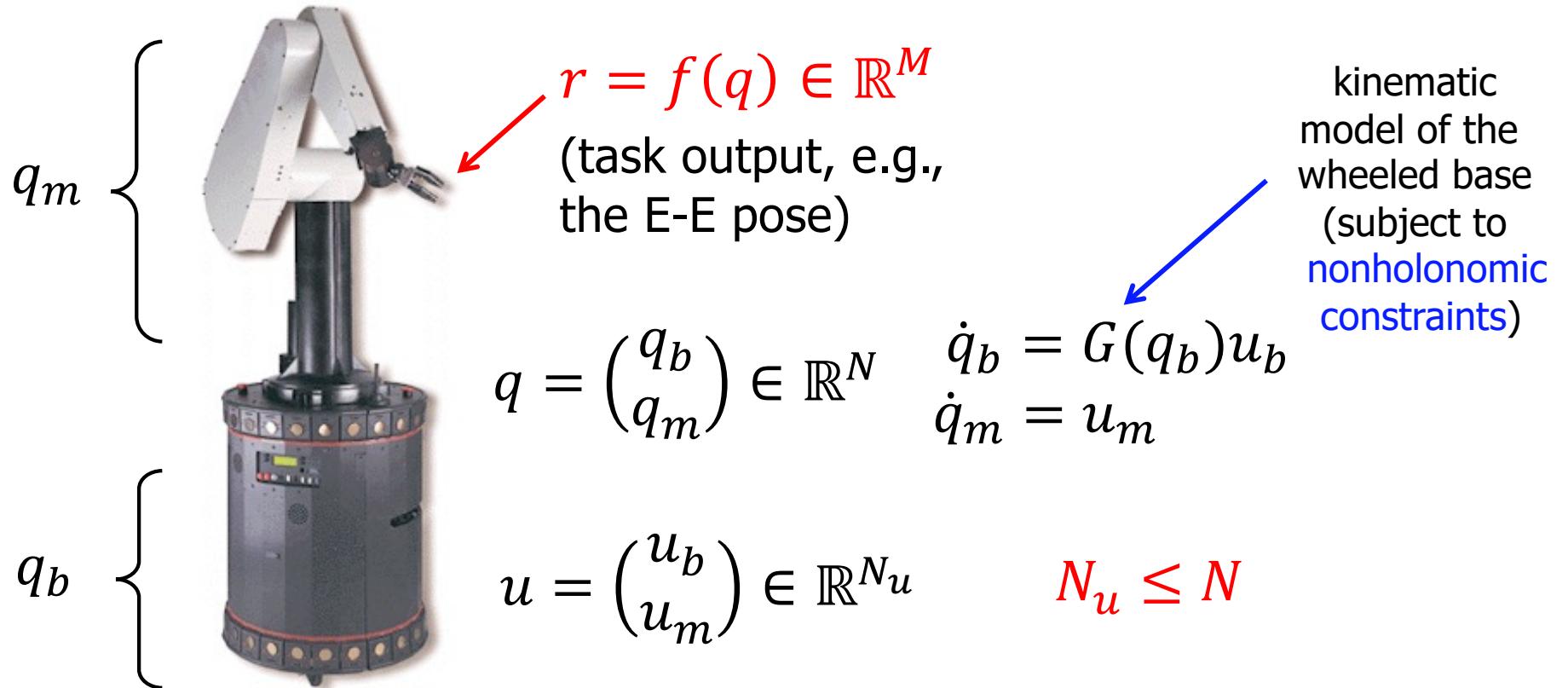
$$\ddot{r} \xrightarrow{\quad\textcolor{red}{\longrightarrow}\quad} \ddot{r}_d + K_D(\dot{r}_d - \dot{r}) + K_P(r_d - r) \quad \dots \text{in acceleration-based methods}$$

where $r = f(q)$, $\dot{r} = J(q)\dot{q}$



Mobile manipulators

- coordinates: q_b of the base and q_m of the manipulator
- differential map: **from** available commands u_b on the mobile base and u_m on the manipulator **to** task output velocity





Mobile manipulator Jacobian

$$r = f(q) = f(q_b, q_m)$$

$$\dot{r} = \frac{\partial f(q)}{\partial q_b} \dot{q}_b + \frac{\partial f(q)}{\partial q_m} \dot{q}_m = J_b(q)\dot{q}_b + J_m(q)\dot{q}_m$$

$$= J_b(q)G(q_b)u_b + J_m(q)u_m = (J_b(q)G(q_b) \quad J_m(q)) \begin{pmatrix} u_b \\ u_m \end{pmatrix}$$

$$= J_{NMM}(q)u$$

Nonholonomic Mobile Manipulator (NMM)
Jacobian ($M \times N_u$)

- ... most previous results follow by just replacing

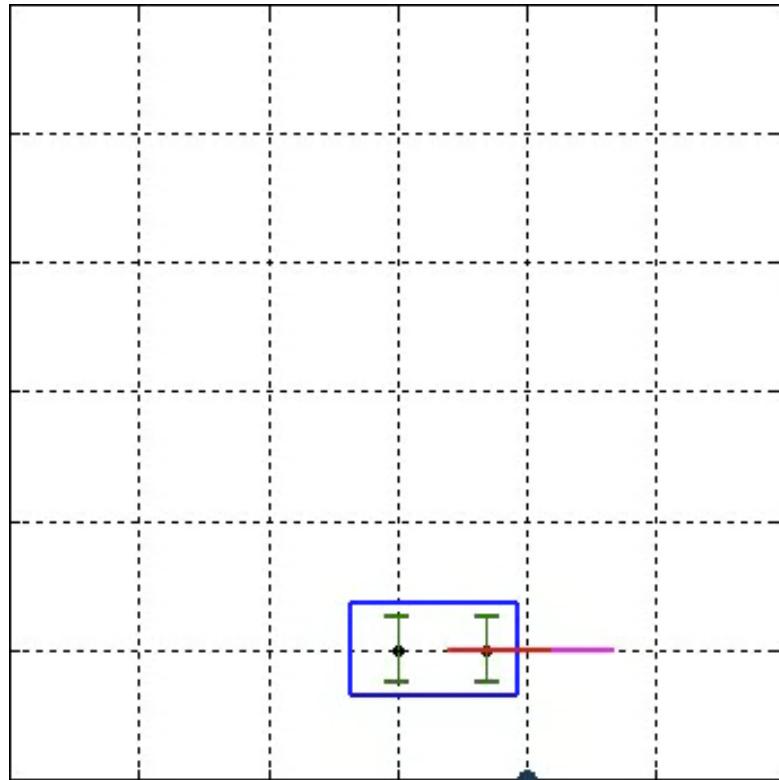
$$J \Rightarrow J_{NMM} \quad \dot{q} \Rightarrow u \quad (\text{redundancy if } N_u - M > 0)$$

↑
namely, the
available velocity commands



Mobile manipulators

video



car-like+2R planar arm

$(N = 6, N_u = 4)$:

E-E trajectory control on a line ($N_u - M = 2$)
with maximization of J_{NMM} manipulability

Automatica Fair 2008



video

wheeled Justin with centered
steering wheels

$(N = 3 + 4 \times 2, N_u = 8)$
“dancing” in controlled
but otherwise passive mode



Quadratic Programming (QP)

with equality and inequality constraints

- minimize a **quadratic** objective function (typically positive definite, like when using norms of vectors) subject to **linear** equality and inequality constraints, all expressed in terms of **joint velocity** commands

$$J\dot{q} = \dot{r} \quad C\dot{q} \leq d \quad \dot{q} \in \Omega \subseteq \mathbb{R}^n$$

within a given **convex** set

solution set, with **only equality** constraints

$$\mathcal{S}_{eq} = \arg \min_{\dot{q} \in \Omega} \frac{1}{2} \|J\dot{q} - \dot{r}\|^2$$

given $\dot{q}^* \in \mathcal{S}_{eq}$ $\Rightarrow \mathcal{S}_{eq} = \{\dot{q} \in \Omega : J\dot{q} = J\dot{q}^*\}$

solution set, with **only inequality** constraints

$$\mathcal{S}_{ineq} = \arg \min_{\dot{q} \in \Omega} \frac{1}{2} \|w\|^2$$

s.t. $C\dot{q} - d \leq w \quad w \in \mathbb{R}_+^m$
(non-negative) **slack** variables

QP complete formulation

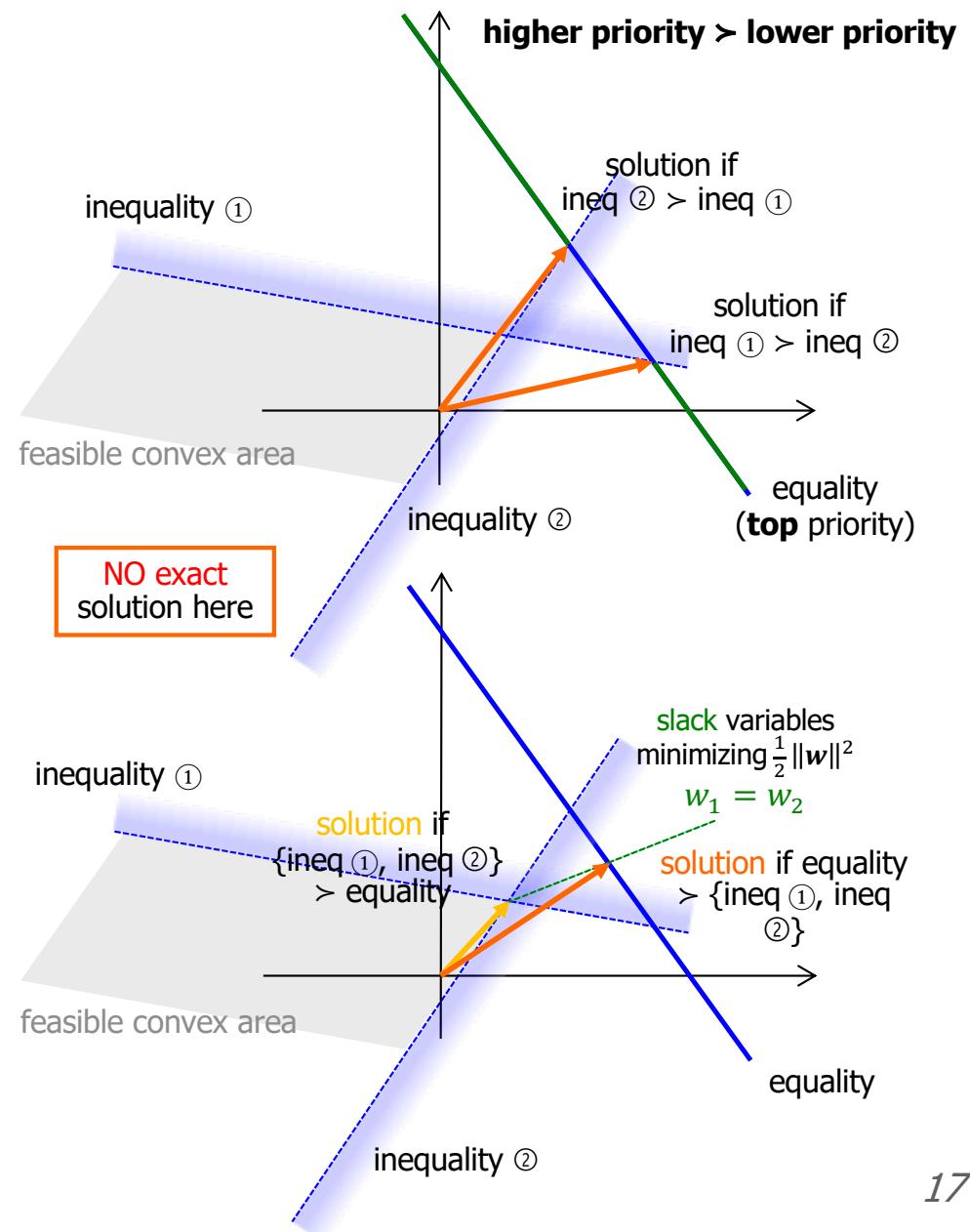
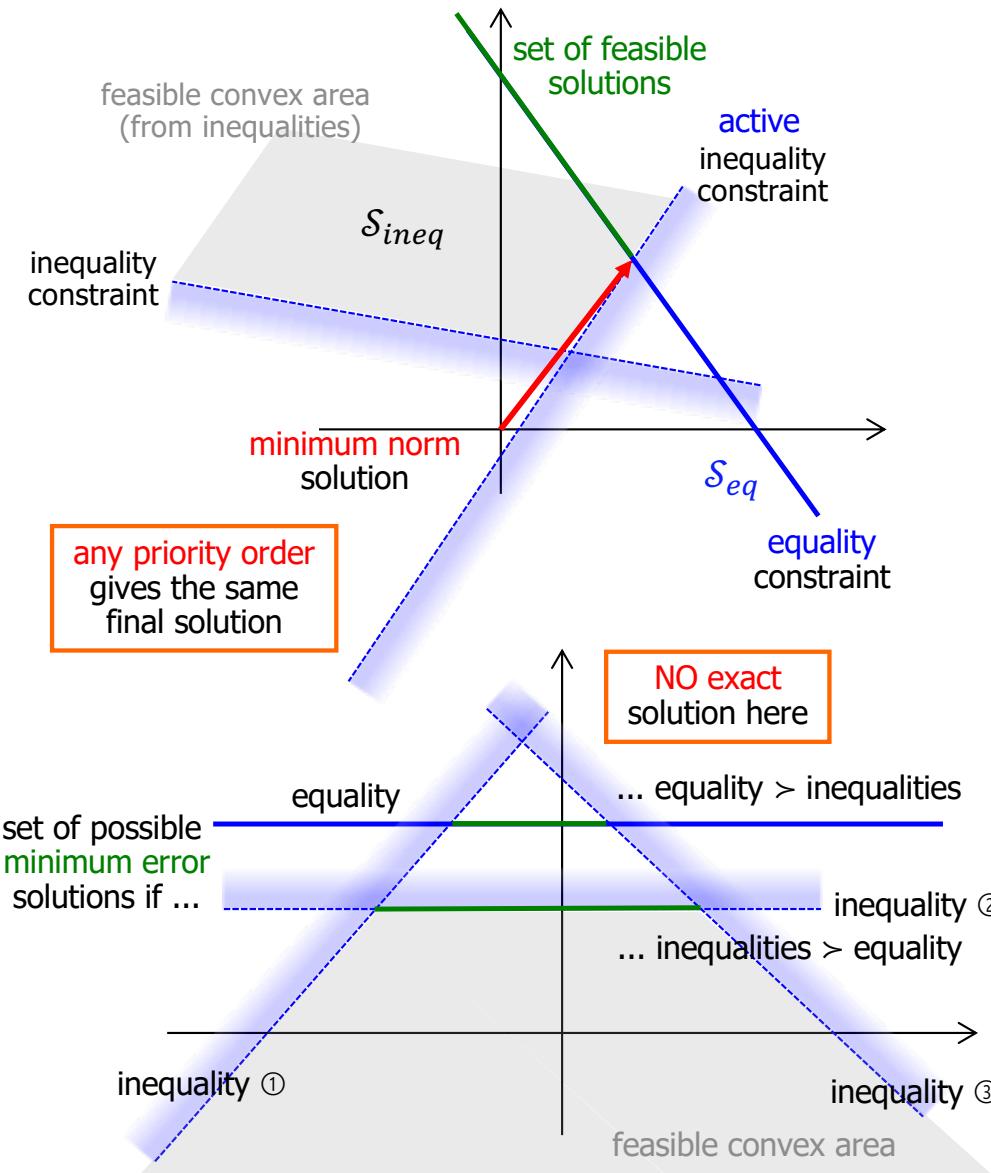
$$\begin{aligned} & \min_{\dot{q} \in \Omega} \frac{1}{2} \|J\dot{q} - \dot{r}\|^2 + \frac{1}{2} \|w\|^2 \\ \text{s.t. } & C\dot{q} - w \leq d \quad w \in \mathbb{R}_+^m \end{aligned}$$

(possibly with prioritization
of constraints)

given $\dot{q}^* \in \mathcal{S}_{ineq}$ $\Rightarrow \mathcal{S}_{ineq} = \Omega \cap \begin{cases} c_j^T \dot{q} \leq d_j, & \text{if } c_j^T \dot{q}^* \leq d_j \\ c_j^T \dot{q} = c_j^T \dot{q}^*, & \text{if } c_j^T \dot{q}^* > d_j \end{cases}$



Equality and inequality linear constraints



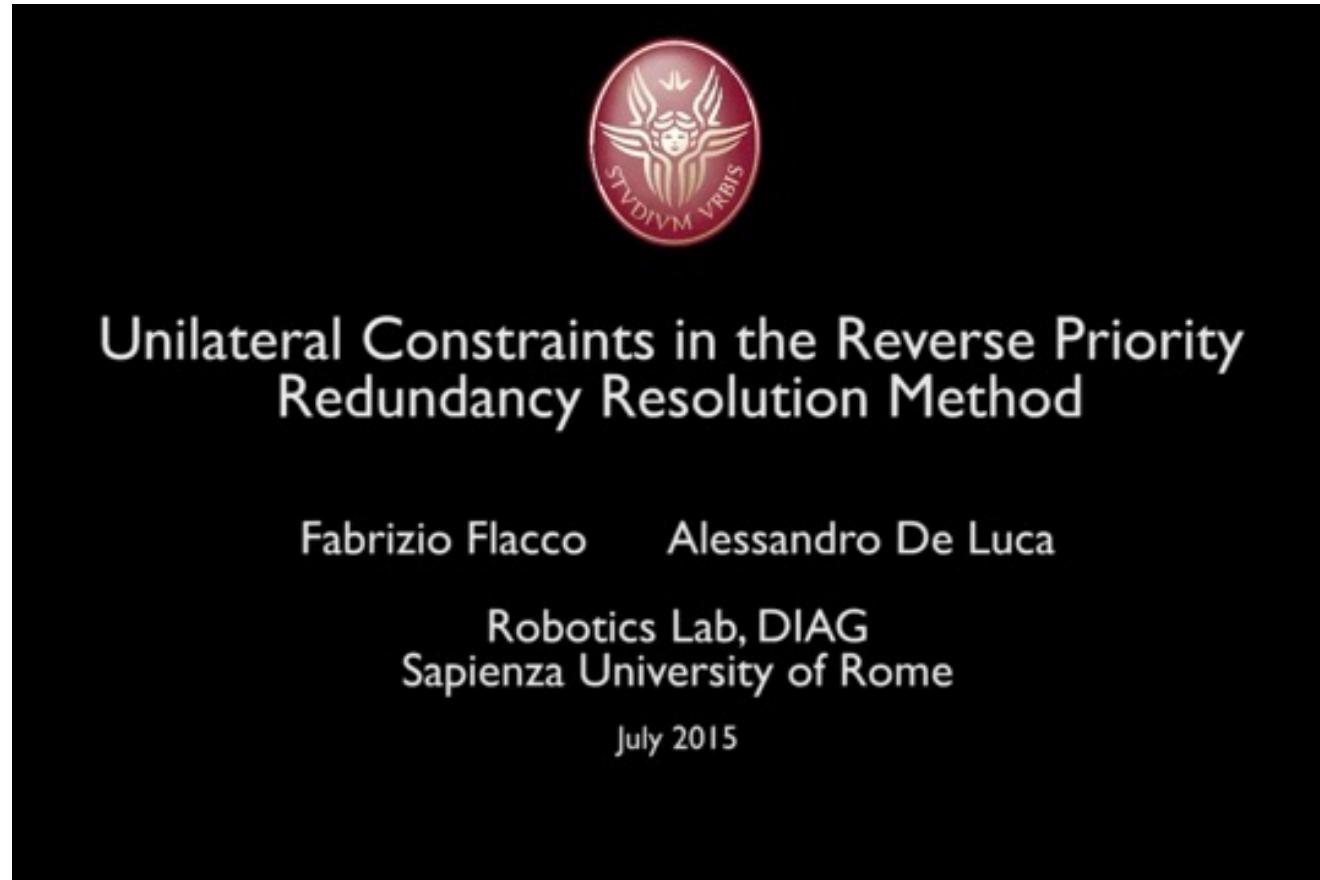


Equality and Inequality Tasks

6R planar robot (simulations) and 7R KUKA LWR (experiment)

- an efficient **task priority** approach, with simultaneous inequality tasks handled as **hard** (cannot be violated) or **soft** (can be relaxed) constraints

video



IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) 2015



Equality and Inequality Tasks

for the high-dof humanoid robot HRP2

- a systematic **task priority** approach, with several simultaneous tasks

video

Prioritizing linear equality and
inequality systems: application to local
motion planning for redundant robots.

*Oussama Kanoun, Florent Lamiraux,
Pierre-Brice Wieber, Fumio Kanehiro,
Eiichi Yoshida and Jean-Paul Laumond*

in **any order** of priority

- avoid the obstacle
- gaze at the object
- reach the object
- ...

while **keeping balance!**



all subtasks are **locally**
expressed by linear
equalities or **inequalities**
(possibly relaxed
when needed)
on **joint velocities**

IEEE Int. Conf. on Robotics and Automation (ICRA) 2009



Inclusion of hard limits in joint space

Saturation in the Null Space (SNS) method

- robot has “limited” capabilities: **hard limits** on joint ranges and/or on joint motion or commands (max velocity, acceleration, torque)
- represented as **box inequalities** that can **never** be violated (at most, **active** constraints or **saturated** commands) – kept separated from “stack” of tasks
- (equality) tasks** are usually executed in full (with priorities, if desired), but can be relaxed (**scaled**) in case of need (i.e., when robot capabilities are used at their limits)



- saturate **one overdriven joint command at a time**, until a feasible and better performing solution is found \Rightarrow **Saturation in the Null Space = SNS**
- on-line** decision: **which joint** commands to saturate and **how**, so that this does not affect task execution
- for tasks that are (certainly) not feasible, SNS **embeds** the selection of a task scaling factor **preserving execution of the task direction** with **minimal scaling**

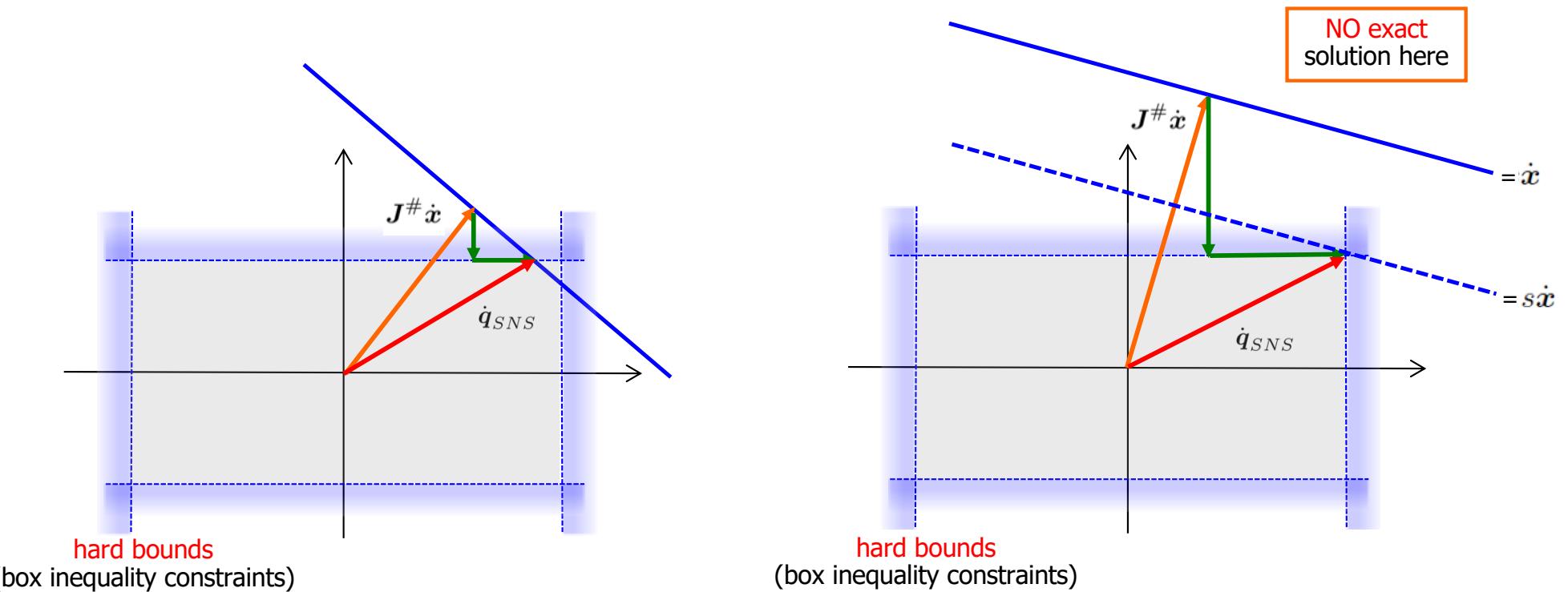
$$\dot{q}_{SNS} = (\mathbf{JW})^\# \mathbf{s}\dot{x} + \left(\mathbf{I} - (\mathbf{JW})^\# \mathbf{J} \right) \dot{q}_N$$

↑ scaling factor ↑ diagonal 0/1 matrix ← contains saturated joint velocities



Geometric view on SNS operation

in the space of joint velocity commands

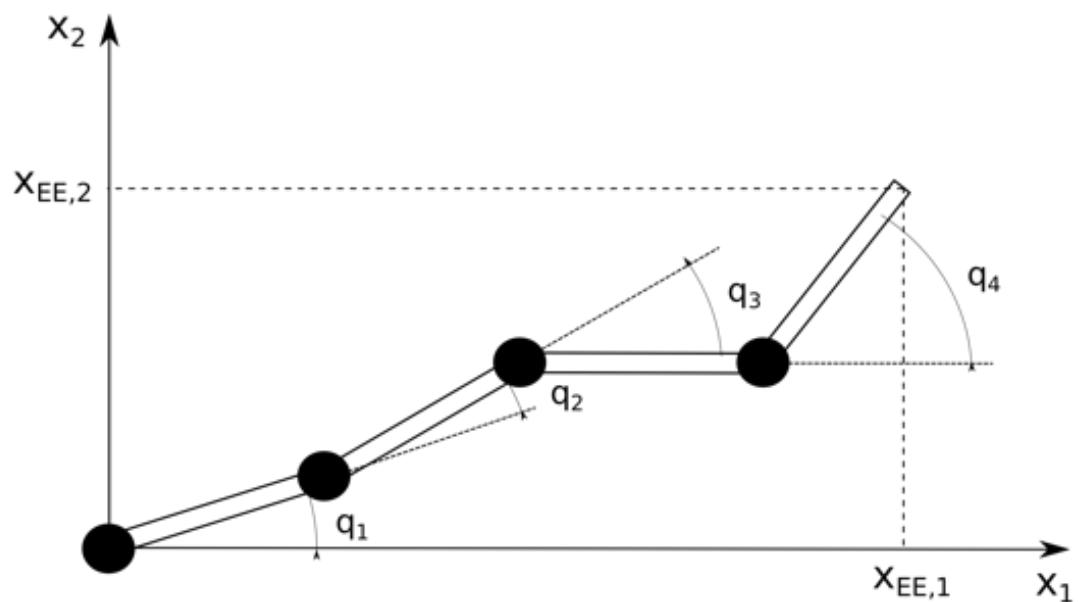


the total correction to the original pseudoinverse solution
is always in the **null space** of the Jacobian (up to task scaling, if present)



Illustrative example - 1

consider a **4R robot** with equal links of unitary length



task Jacobian

$$\mathbf{J}(\mathbf{q}) = \begin{pmatrix} -lS_1 - lS_{12} - lS_{123} - lS_{1234} & -lS_{12} - lS_{123} - lS_{1234} & -lS_{123} - lS_{1234} & -lS_{1234} \\ lC_1 + lC_{12} + lC_{123} + lC_{1234} & lC_{12} + lC_{123} + lC_{1234} & lC_{123} + lC_{1234} & lC_{1234} \end{pmatrix}$$

velocity limits $|\dot{q}_i| \leq V_i, i = 1 \dots 4$

$$V_1 = V_2 = 2 \quad V_3 = V_4 = 4 \text{ [rad/s]}$$

task: end-effector Cartesian position

$$\mathbf{x} = (x_{EE,1} \ x_{EE,2})$$

manipulator configuration

$$\mathbf{q} = (q_1 \ q_2 \ q_3 \ q_4)$$

differential map

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

desired Cartesian velocity $\dot{\mathbf{x}} \in \mathcal{R}^2$

commanded joint velocity $\dot{\mathbf{q}} \in \mathcal{R}^4$



Illustrative example - 2

current configuration $\mathbf{q} = (\pi/2 \quad -\pi/2 \quad \pi/2 \quad -\pi/2)^T$

associated Jacobian $\mathbf{J} = (J_1 \quad J_2 \quad J_3 \quad J_4) = \begin{pmatrix} -2 & -1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{pmatrix}$

desired end-effector velocity $\dot{\mathbf{x}} = (-4 \quad -1.5)^T$

$$\dot{\mathbf{q}}_{PS} = \mathbf{J}^\# \dot{\mathbf{x}} = (2.0 \quad -2.0 \quad 2.4545 \quad -2.1364)^T$$

direct (velocity =) task scaling? $s = 0.8148$

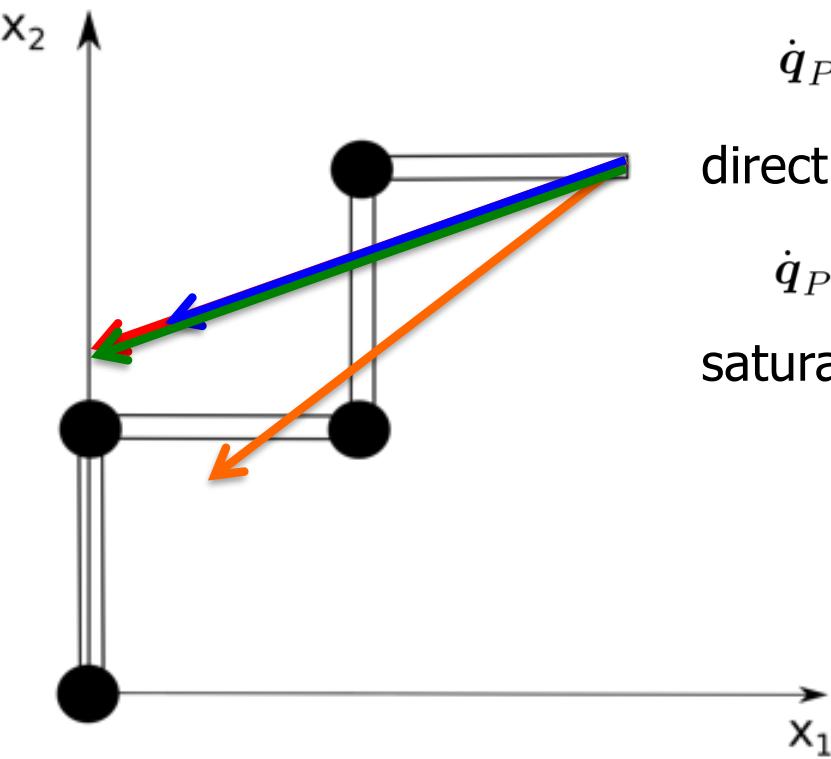
$$\dot{\mathbf{q}}_{PS} = s \mathbf{J}^\# \dot{\mathbf{x}} = (2.0 \quad -1.74 \quad 1.0 \quad -2.74)^T$$

saturating only the most violating velocity? $\dot{q}_1 = V_1 = 2$

$$\dot{\mathbf{x}}_{SNS} = \dot{\mathbf{x}} - J_1 V_1 = (J_2 \quad J_3 \quad J_4) \begin{pmatrix} \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{pmatrix}$$

$$\dot{\mathbf{q}}_{SNS} = \left(V_1 \quad [(\mathbf{J}_2 \quad \mathbf{J}_3 \quad \mathbf{J}_4)^\# \dot{\mathbf{x}}_{SNS}]^T \right)^T$$

$$= (2 \quad -1.8333 \quad 1.8333 \quad -3.6667)^T$$





Joint velocity bounds

joint space
limits

$$\begin{aligned} Q_{min,i} &\leq q_i \leq Q_{max,i} \\ -V_{max,i} &\leq \dot{q}_i \leq V_{max,i} \quad i = 1, \dots, n \\ -A_{max,i} &\leq \ddot{q}_i \leq A_{max,i} \end{aligned}$$

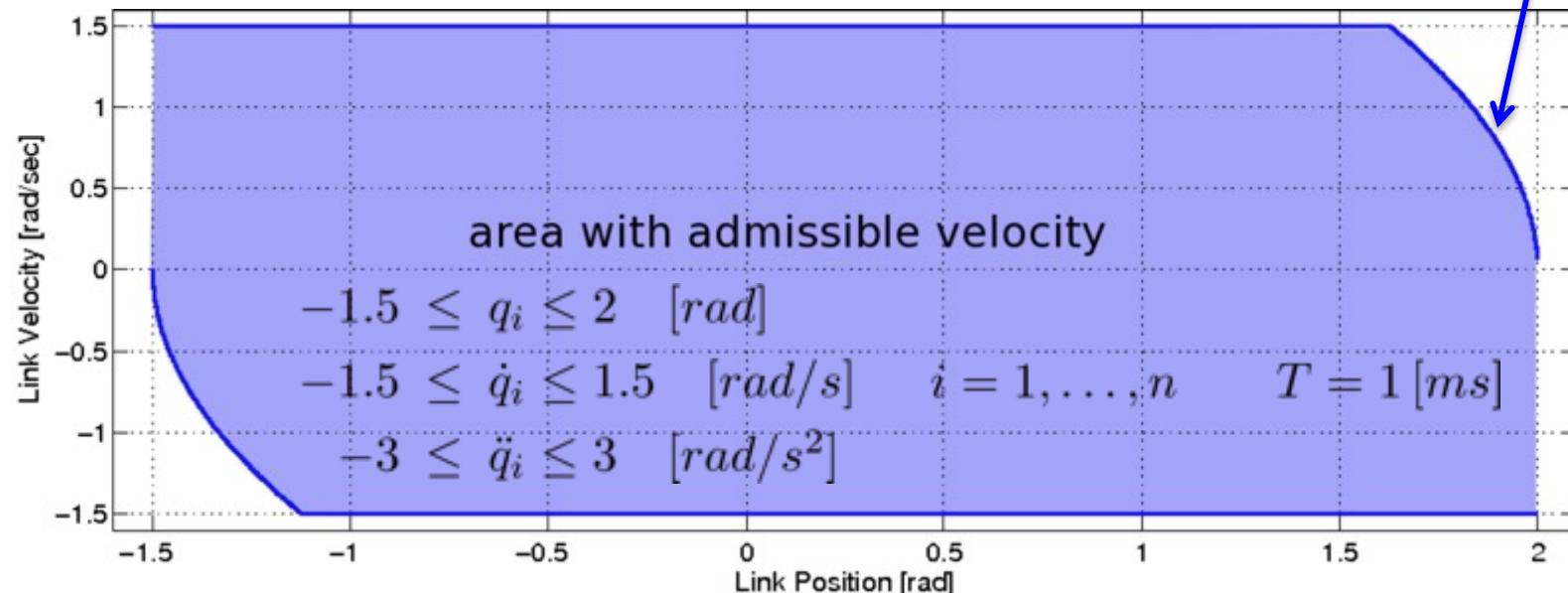
joint velocity bounds

$$\dot{Q}_{min}(t_k) \leq \dot{q} \leq \dot{Q}_{max}(t_k)$$

conversion: control sampling (piece-wise constant velocity commands) + max feasible velocities and decelerations to stay/stop within the joint range

$$\begin{aligned} \dot{Q}_{min,i} &= \max \left\{ \frac{Q_{min,i} - q_{k,i}}{T}, -V_{max,i}, -\sqrt{2A_{max,i}(q_{k,i} - Q_{min,i})} \right\} \\ \dot{Q}_{max,i} &= \min \left\{ \frac{Q_{max,i} - q_{k,i}}{T}, V_{max,i}, \sqrt{2A_{max,i}(Q_{max,i} - q_{k,i})} \right\} \end{aligned}$$

smooth velocity bound "anticipates" the reaching of a hard limit





SNS at velocity level

Algorithm 1

$\mathbf{W} = \mathbf{I}$, $\dot{\mathbf{q}}_N = \mathbf{0}$, $s = 1$, $s^* = 0$

repeat

 limit_exceeded = FALSE

$$\dot{\bar{\mathbf{q}}} = \dot{\mathbf{q}}_N + (\mathbf{J}\mathbf{W})^\# (\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}_N)$$

 if $\left\{ \begin{array}{l} \exists i \in [1:n] : \\ \dot{\bar{q}}_i < \dot{Q}_{min,i} \text{ OR } \dot{\bar{q}}_i > \dot{Q}_{max,i} \end{array} \right\}$ then
 limit_exceeded = TRUE

$$\mathbf{a} = (\mathbf{J}\mathbf{W})^\# \dot{\mathbf{x}}$$

$$\mathbf{b} = \dot{\bar{\mathbf{q}}} - \mathbf{a}$$

 getTaskScalingFactor(\mathbf{a} , \mathbf{b}) (*call Algorithm 2*)

 if {task scaling factor} > s^* then

$s^* = \{\text{task scaling factor}\}$

$$\mathbf{W}^* = \mathbf{W}, \dot{\mathbf{q}}_N^* = \dot{\mathbf{q}}_N$$

 end if

$j = \{\text{the most critical joint}\}$

$$W_{jj} = 0$$

$$\dot{q}_{N,j} = \begin{cases} \dot{Q}_{max,j} & \text{if } \dot{\bar{q}}_j > \dot{Q}_{max,j} \\ \dot{Q}_{min,j} & \text{if } \dot{\bar{q}}_j < \dot{Q}_{min,j} \end{cases}$$

 if $\text{rank}(\mathbf{J}\mathbf{W}) < m$ then

$$s = s^*, \mathbf{W} = \mathbf{W}^*, \dot{\mathbf{q}}_N = \dot{\mathbf{q}}_N^*$$

$$\dot{\bar{\mathbf{q}}} = \dot{\mathbf{q}}_N + (\mathbf{J}\mathbf{W})^\# (s\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}_N)$$

 limit_exceeded = FALSE (*outputs solution*)

 end if

end if

until limit_exceeded = TRUE

$$\dot{\mathbf{q}}_{SNS} = \dot{\bar{\mathbf{q}}}$$

initialization

\mathbf{W} : diagonal matrix with (j, j) element
 = 1 if joint j is enabled
 = 0 if joint j is disabled

$\dot{\bar{\mathbf{q}}}_N$: vector with saturated velocities in correspondence of disabled joints

s : current task scale factor

s^* : largest task scale factor so far



SNS at velocity level

Algorithm 1

$\mathbf{W} = \mathbf{I}$, $\dot{\mathbf{q}}_N = \mathbf{0}$, $s = 1$, $s^* = 0$

repeat

 limit_exceeded = FALSE

$$\dot{\bar{\mathbf{q}}} = \dot{\mathbf{q}}_N + (\mathbf{J}\mathbf{W})^\# (\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}_N) \leftarrow$$

 if $\left\{ \exists i \in [1:n] : \dot{\bar{q}}_i < \dot{Q}_{min,i} \text{ OR } \dot{\bar{q}}_i > \dot{Q}_{max,i} \right\}$ then
 limit_exceeded = TRUE

$$\mathbf{a} = (\mathbf{J}\mathbf{W})^\# \dot{\mathbf{x}}$$

$$\mathbf{b} = \dot{\bar{\mathbf{q}}} - \mathbf{a}$$

 getTaskScalingFactor(\mathbf{a}, \mathbf{b}) (*call Algorithm 2*)

 if {task scaling factor} > s^* then
 $s^* = \{\text{task scaling factor}\}$
 $\mathbf{W}^* = \mathbf{W}$, $\dot{\mathbf{q}}_N^* = \dot{\mathbf{q}}_N$
 end if

$j = \{\text{the most critical joint}\}$

$$W_{jj} = 0$$

$$\dot{q}_{N,j} = \begin{cases} \dot{Q}_{max,j} & \text{if } \dot{\bar{q}}_j > \dot{Q}_{max,j} \\ \dot{Q}_{min,j} & \text{if } \dot{\bar{q}}_j < \dot{Q}_{min,j} \end{cases}$$

 if $\text{rank}(\mathbf{J}\mathbf{W}) < m$ then
 $s = s^*$, $\mathbf{W} = \mathbf{W}^*$, $\dot{\mathbf{q}}_N = \dot{\mathbf{q}}_N^*$
 $\dot{\bar{\mathbf{q}}} = \dot{\mathbf{q}}_N + (\mathbf{J}\mathbf{W})^\# (s\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}_N)$
 limit_exceeded = FALSE (*outputs solution*)
 end if

 end if

until limit_exceeded = TRUE

$$\dot{\mathbf{q}}_{SNS} = \dot{\bar{\mathbf{q}}}$$

compute the **joint velocity** with initialized values

$$\dot{\bar{\mathbf{q}}} = \mathbf{J}^\# \dot{\mathbf{x}}$$

check the joint velocity bounds

compute the **task scaling factor** and the **most critical joint**

if a larger task scaling factor is obtained, save the current solution

disable the most critical joint by forcing it at its saturated velocity



SNS at velocity level

Algorithm 1

$\mathbf{W} = \mathbf{I}$, $\dot{\mathbf{q}}_N = \mathbf{0}$, $s = 1$, $s^* = 0$

repeat

 limit_exceeded = FALSE

$$\dot{\bar{\mathbf{q}}} = \dot{\mathbf{q}}_N + (\mathbf{J}\mathbf{W})^\# (\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}_N)$$

 if $\left\{ \begin{array}{l} \exists i \in [1:n] : \\ \dot{\bar{q}}_i < \dot{Q}_{min,i} \text{ OR } \dot{\bar{q}}_i > \dot{Q}_{max,i} \end{array} \right\}$ then

 limit_exceeded = TRUE

$$\mathbf{a} = (\mathbf{J}\mathbf{W})^\# \dot{\mathbf{x}}$$

$$\mathbf{b} = \dot{\bar{\mathbf{q}}} - \mathbf{a}$$

 getTaskScalingFactor(\mathbf{a} , \mathbf{b}) (*call Algorithm 2*)

 if {task scaling factor} > s^* then

$s^* = \{\text{task scaling factor}\}$

$$\mathbf{W}^* = \mathbf{W}, \dot{\mathbf{q}}_N^* = \dot{\mathbf{q}}_N$$

 end if

$j = \{\text{the most critical joint}\}$

$$W_{jj} = 0$$

$$\dot{q}_{N,j} = \begin{cases} \dot{Q}_{max,j} & \text{if } \dot{\bar{q}}_j > \dot{Q}_{max,j} \\ \dot{Q}_{min,j} & \text{if } \dot{\bar{q}}_j < \dot{Q}_{min,j} \end{cases}$$

 if $\text{rank}(\mathbf{J}\mathbf{W}) < m$ then

$$s = s^*, \mathbf{W} = \mathbf{W}^*, \dot{\mathbf{q}}_N = \dot{\mathbf{q}}_N^*$$

$$\dot{\bar{\mathbf{q}}} = \dot{\mathbf{q}}_N + (\mathbf{J}\mathbf{W})^\# (s\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}_N)$$

 limit_exceeded = FALSE (*outputs solution*)

 end if

 end if

until limit_exceeded = TRUE

$$\dot{\mathbf{q}}_{SNS} = \dot{\bar{\mathbf{q}}}$$

check if task can be accomplished with the remaining **enabled** joints

if NOT, use the parameters that allow the **largest** task scaling factor and **exit**

repeat until at least one joint limit is exceeded (exit if there is none!)



Task scaling factor

Algorithm 2

```

function getTaskScalingFactor( $\mathbf{a}, \mathbf{b}$ )  

for  $i = 1 \rightarrow n$  do  

     $S_{min,i} = (\dot{Q}_{min,i} - b_i) / a_i$        $\dot{Q}_{min,i} \leq a_i s + b_i \leq \dot{Q}_{max,i}$  with  $s \in [0,1]$   

     $S_{max,i} = (\dot{Q}_{max,i} - b_i) / a_i$       restore the correct order of inequalities  

    if  $S_{min,i} > S_{max,i}$  then      (possibly modified by the sign of  $a_i$ )  

        {switch  $S_{min,i}$  and  $S_{max,i}$ }  

    end if  

end for  

 $s_{max} = \min_i \{S_{max,i}\}$       yields the best task scaling factor  

 $s_{min} = \max_i \{S_{min,i}\}$       (i.e., closest to the ideal value = 1) due  

the most critical joint =  $\operatorname{argmin}_i \{S_{max,i}\}$       to the most critical among the currently  

if  $s_{min} > s_{max}$  .OR.  $s_{max} < 0$  .OR.  $s_{min} > 1$  then      enabled joint velocity components  

    task scaling factor = 0      no variation of the scaling factor currently used in  

else      Algorithm 1 is needed (it will keep the previous  $s^*$ )  

    task scaling factor =  $s_{max}$       always take the largest value for task scaling ...  

end if

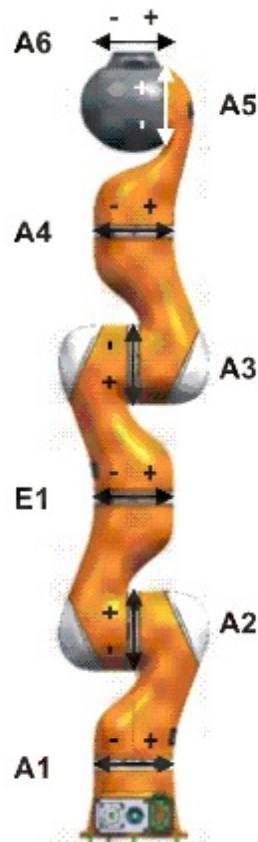
```



Simulation results

Axis	Range of motion, software-limited	Velocity without payload
A1 (J1)	+/-170°	100°/s
A2 (J2)	+/-120°	110°/s
E1 (J3)	+/-170°	100°/s
A3 (J4)	+/-120°	130°/s
A4 (J5)	+/-170°	130°/s
A5 (J6)	+/-120°	180°/s
A6 (J7)	+/-170°	180°/s

7-dof KUKA LWR IV



$$\mathbf{Q}_{max} = (170, 120, 170, 120, 170, 120, 170) \text{ [deg]}$$

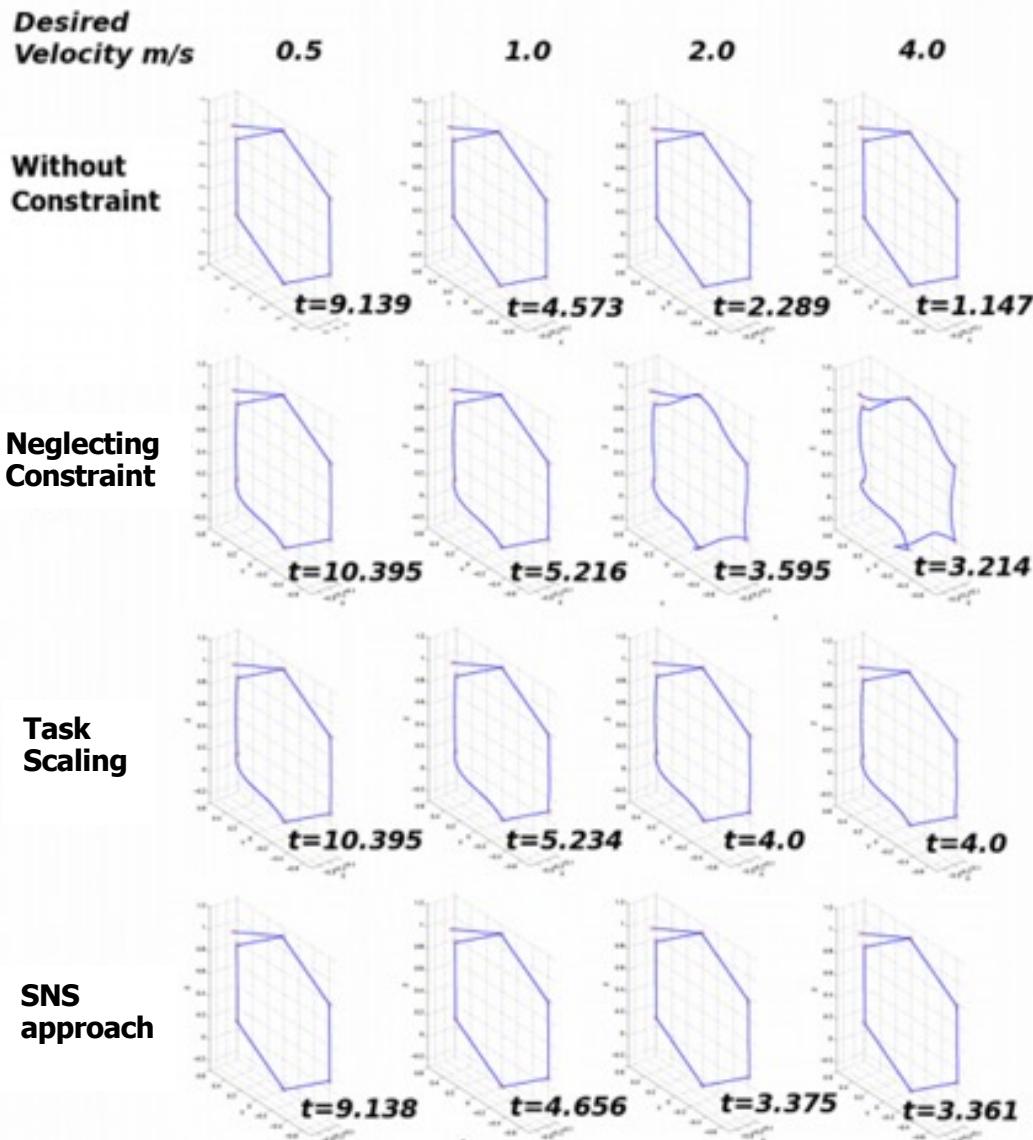
$$\mathbf{V}_{max} = (100, 110, 100, 130, 130, 180, 180) \text{ [deg/s]}$$

$$A_{max,i} = 300 \text{ [deg/s}^2\text{]} \quad \forall i = 1 \dots n$$

$$T = 1 \text{ [ms]}$$



Simulation results



← for increasing V

requested task

move the end-effector through six desired Cartesian positions along linear paths with constant speed V

$$\dot{x} = V \frac{\mathbf{x}_r - \mathbf{x}}{\|\mathbf{x}_r - \mathbf{x}\|}$$

task redundancy degree = $7 - 3 = 4$

robot starts at the configuration

$$\mathbf{q}(0) = (0, 45, 45, 45, 0, 0, 0) \text{ [deg]}$$

(with a small initial approaching phase)



Experimental results

KUKA LWR IV with hard joint-space limits

[video](#)



Control of Redundant Robots under Hard Joint Constraints: Saturation in the Null Space

Fabrizio Flacco Alessandro De Luca Oussama Khatib

Robotics Lab, DIAG
Sapienza Università di Roma

Artificial Intelligence Lab
Stanford University

July 2014

IEEE Transactions on Robotics 2015



Variations of the SNS method

SNS at the **acceleration** command level + consideration of **multiple tasks** with priority
video

Prioritized Multi-Task Motion Control
of Redundant Robots under
Hard Joint Constraints

Attached video to IROS 2012

* F. Flacco *A. De Luca
** O Khatib

*Robotics Laboratory, Università di Roma "La Sapienza"
**Artificial Intelligence Laboratory , Stanford University

IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) 2012



Inclusion of hard Cartesian bounds

- SNS at the **velocity command** level, with hard bounds on joint position, velocity, and acceleration and task scaling factor (just **one task** is considered here ...)
- **additional** (possibly, time-varying) Cartesian box inequalities on position, velocity, and acceleration of ***r* control points** along the structure (including end effector)
- **generalized** treatment of all bounds in a **unified** way (conversions like in **slide #24**)

$$Q_j^{\min} \leq q_j \leq Q_j^{\max}$$

$$V_j^{\min} \leq \dot{q}_j \leq V_j^{\max}$$

$$\Lambda_j^{\min} \leq \ddot{q}_j \leq \Lambda_j^{\max}$$

$$j = 1, \dots, n$$

$$\mathbf{P}_{cp,i}^{\min} \leq \mathbf{p}_{cp,i} \leq \mathbf{P}_{cp,i}^{\max}$$

$$\mathbf{V}_{cp,i}^{\min} \leq \dot{\mathbf{p}}_{cp,i} \leq \mathbf{V}_{cp,i}^{\max}$$

$$\mathbf{\Lambda}_{cp,i}^{\min} \leq \ddot{\mathbf{p}}_{cp,i} \leq \mathbf{\Lambda}_{cp,i}^{\max}$$

$$\mathbf{p}_{cp,i} \in \mathbb{R}^{d_i}$$

$$d_i \in \{1, 2, 3\}$$

$$i = 1, \dots, r$$

generalized vector

$$\mathbf{a} = (\mathbf{q}^T \quad \mathbf{p}_{cp,1}^T \quad \mathbf{p}_{cp,2}^T \quad \dots \quad \mathbf{p}_{cp,r}^T)^T$$

additional processing of $\dot{\mathbf{q}}$ in
Algorithm 1 (rather than by \mathbf{I} only) $\longrightarrow \mathbf{A} = (\mathbf{I} \quad \mathbf{J}_{cp,1}^T \quad \mathbf{J}_{cp,2}^T \quad \dots \quad \mathbf{J}_{cp,r}^T)^T$

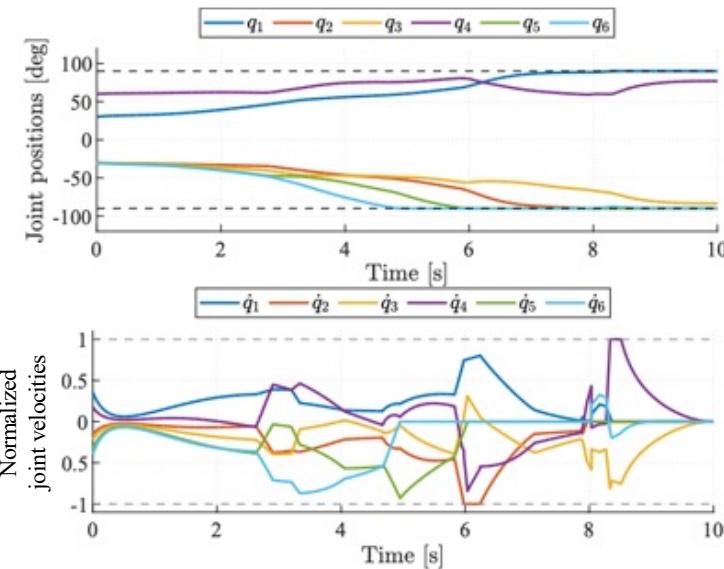
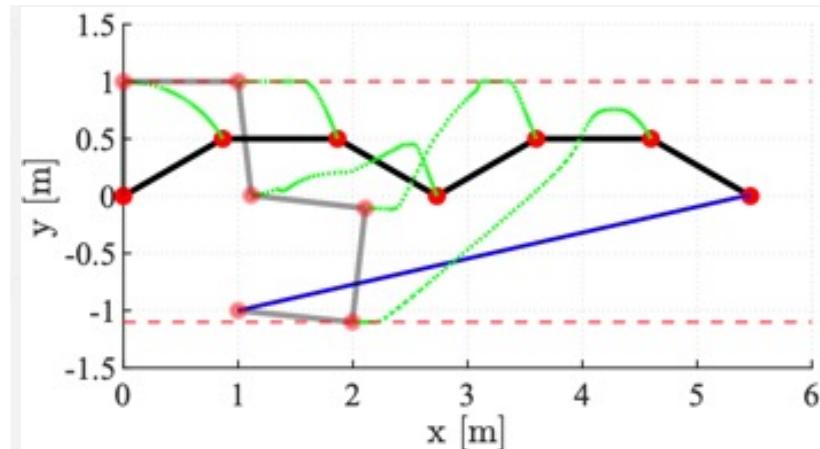
$$\Rightarrow \mathbf{B}_{\min}(t_k) \leq \boxed{\dot{\mathbf{a}}(\mathbf{q}, \dot{\mathbf{q}})} \leq \mathbf{B}_{\max}(t_k) \quad \text{unified joint/Cartesian bounds}$$



Inclusion of hard Cartesian bounds

simulation on a 6R planar manipulator with $r = 5$ control points (at joints from 2 to 6)

video



$$Q_j^{\max} = -Q_j^{\min} = 90 \text{ [deg]}$$

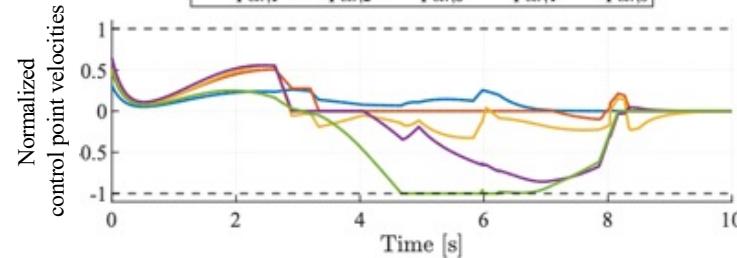
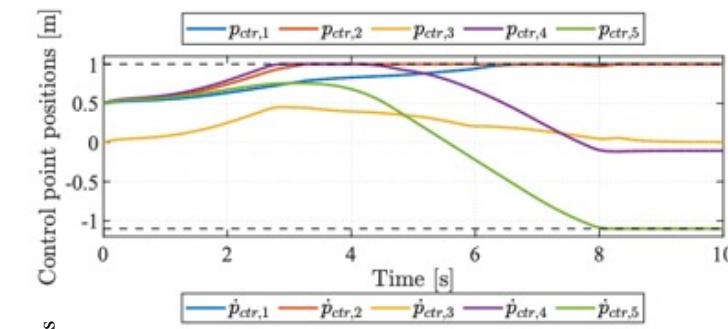
$$V_j^{\max} = -V_j^{\min} = \frac{90}{\pi} \text{ [deg/s]}$$

joint limits on position and velocity ($j = 1, \dots, 6$)

$$P_{cp,i}^{\max,y} = 1, \quad P_{cp,i}^{\min,y} = -1.1 \text{ [m]}$$

$$V_{cp,i}^{\max,y} = 0.5, \quad V_{cp,i}^{\min,y} = -0.5 \text{ [m/s]}$$

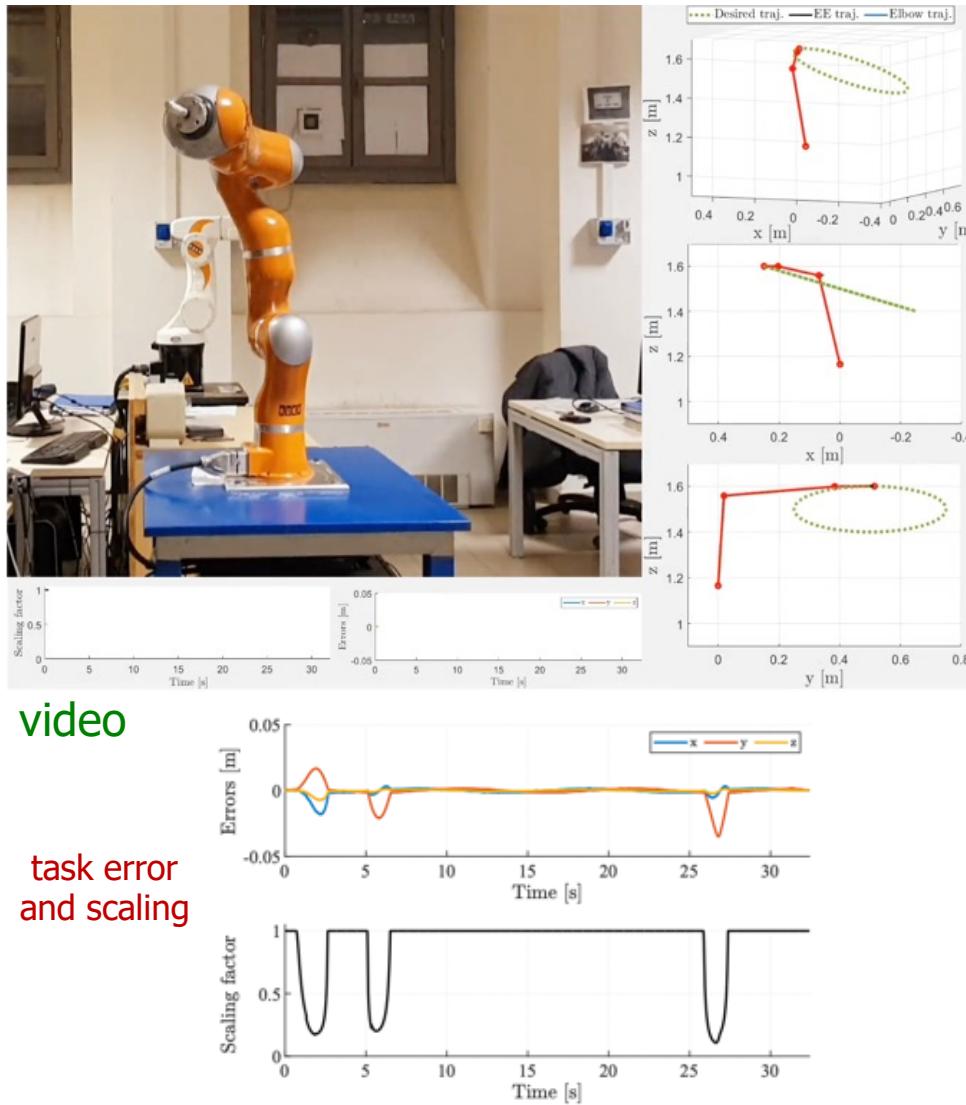
control point limits on position and velocity along y axis ($i = 1, \dots, 5$)





Inclusion of hard Cartesian bounds

experiment #1 on KUKA LWR IV robot with $r = 1$ control point at the robot elbow (with $d_1 = 2$)



joint limits on position, velocity and acceleration

$$\mathbf{Q}^{\max} = -\mathbf{Q}^{\min} = (170 \ 105 \ 170 \ 120 \ 170 \ 85 \ 170)^T \text{ [deg]}$$

$$\mathbf{V}^{\max} = -\mathbf{V}^{\min} = (20 \ 22 \ 20 \ 26 \ 26 \ 36 \ 36)^T \text{ [deg/s]}$$

$$\mathbf{\Lambda}^{\max} = -\mathbf{\Lambda}^{\min} = (30 \ 30 \ 30 \ 30 \ 30 \ 30 \ 30)^T \text{ [deg/s}^2]$$

control point limits on position, velocity and acceleration

$$-0.1 \leq \dot{\mathbf{p}}_{cp_x,1} \leq 0.1, \quad -0.1 \leq \dot{\mathbf{p}}_{cp_y,1} \leq 0.1 \text{ [m/s].}$$

$$-0.5 \leq \ddot{\mathbf{p}}_{cp_x,1} \leq 0.5, \quad -0.5 \leq \ddot{\mathbf{p}}_{cp_y,1} \leq 0.5 \text{ [m/s}^2]$$

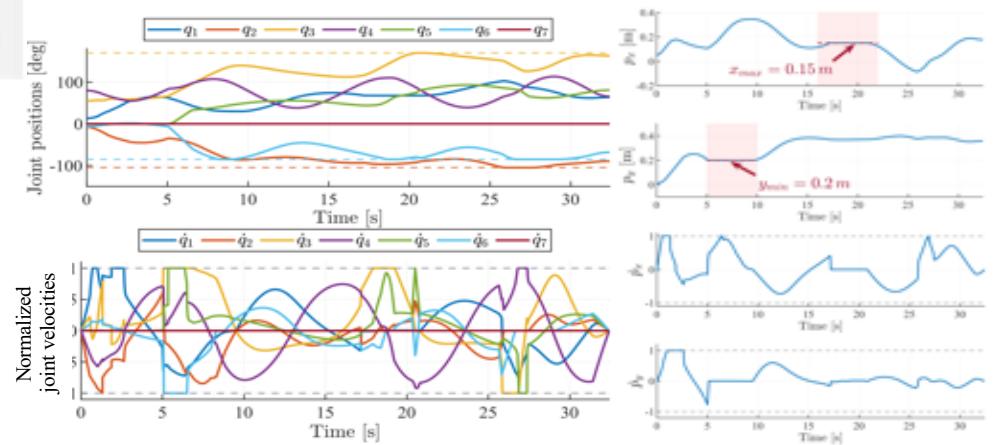
permanent

$$\mathbf{p}_{cp_x,1} \leq 0.15 \text{ [m], } 16 \leq t \leq 22 \text{ [s]}$$

$$\mathbf{p}_{cp_y,1} \leq 0.2 \text{ [m], } 5 \leq t \leq 10 \text{ [s]}$$

(online) time-varying

joint-space and Cartesian control point behaviors



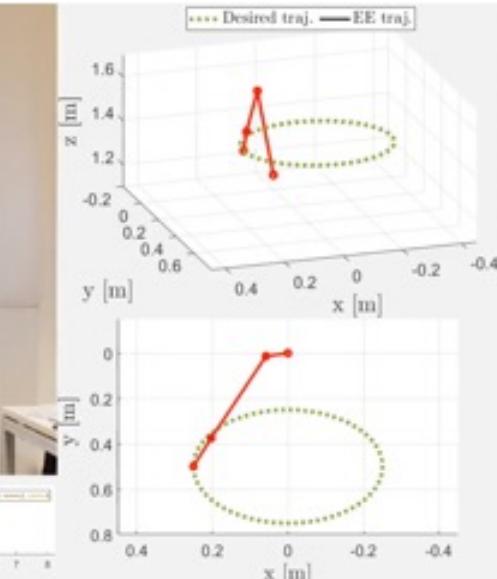


Inclusion of hard Cartesian bounds

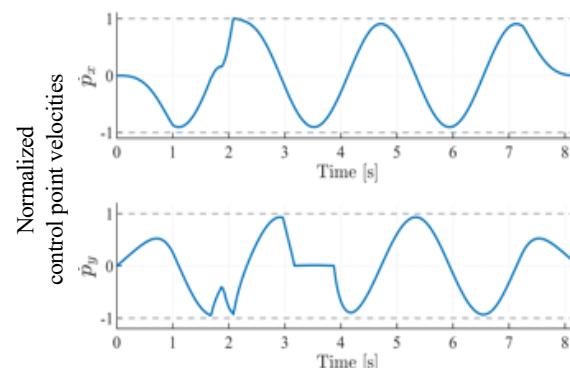
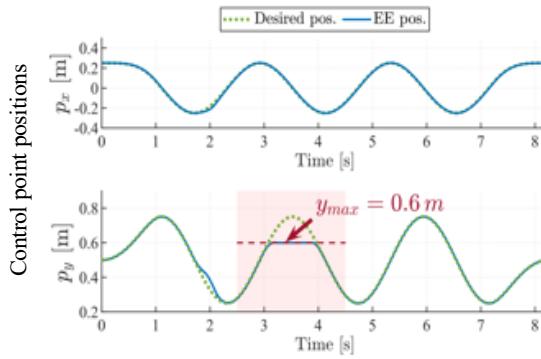
experiment #2 on KUKA LWR IV robot with $r = 1$ control point at the robot elbow (with $d_1 = 2$)



video



circle will be “cut” during second turn!



joint limits on position, velocity and acceleration

$$\mathbf{Q}^{max} = -\mathbf{Q}^{min} = (170 \ 120 \ 170 \ 120 \ 170 \ 120 \ 170)^T \text{ [deg]}$$

$$\mathbf{V}^{max} = -\mathbf{V}^{min} = (100 \ 110 \ 100 \ 130 \ 130 \ 180 \ 180)^T \text{ [deg/s]}$$

$$\mathbf{\Lambda}^{max} = -\mathbf{\Lambda}^{min} = (300 \ 300 \ 300 \ 300 \ 300 \ 300 \ 300)^T \text{ [deg/s}^2]$$

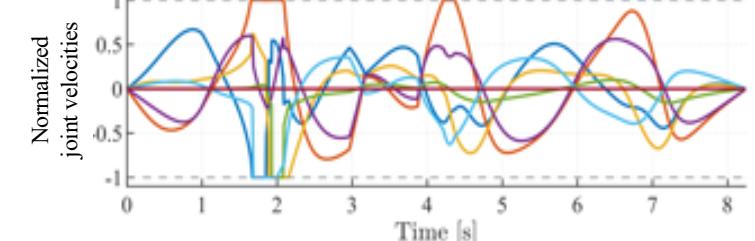
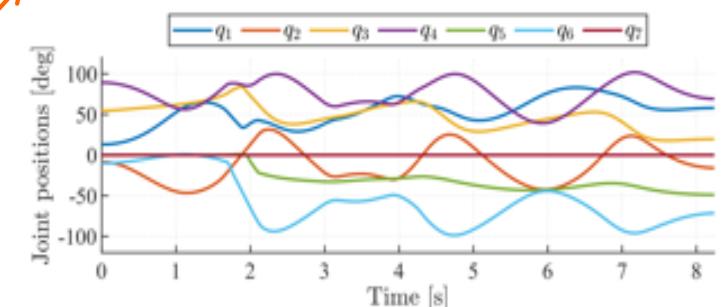
much higher than before \Rightarrow faster motion!

control point limits on position, velocity and acceleration

$$-0.7 \leq \dot{\mathbf{p}}_{cp_x,1} \leq 0.7, \quad -0.7 \leq \dot{\mathbf{p}}_{cp_y,1} \leq 0.7 \text{ [m/s]} \quad \text{permanent}$$

$$-1.5 \leq \ddot{\mathbf{p}}_{cp_x,1} \leq 1.5, \quad -1.5 \leq \ddot{\mathbf{p}}_{cp_y,1} \leq 1.5 \text{ [m/s}^2]$$

$$\mathbf{p}_{cp_y,1} \leq 0.6 \text{ [m]}, \quad 2.5 \leq t \leq 4.5 \text{ [s]} \quad \text{(online) time-varying}$$





Bibliography - 1

- R. Cline, "Representations for the generalized inverse of a partitioned matrix," *J. SIAM*, pp. 588-600, 1964
- T.L. Boullion, P. L. Odell, *Generalized Inverse Matrices*, Wiley-Interscience, 1971
- A. Maciejewski, C. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Int. J. of Robotics Research*, vol. 4, no. 3, pp. 109-117, 1985
- A. Maciejewski, C. Klein, "Numerical filtering for the operation of robotic manipulators through kinematically singular configurations," *J. of Robotic Systems*, vol. 5, no. 6, pp. 527-552, 1988
- Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley, 1991
- B. Siciliano, J.J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," *5th Int. Conf. on Advanced Robotics*, pp. 1211-1216, 1991
- P. Baerlocher, R. Boulic, "Task-priority formulations for the kinematic control of highly redundant articulated structures", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 323-329, 1998
- P. Baerlocher, R. Boulic, "An inverse kinematic architecture enforcing an arbitrary number of strict priority levels," *The Visual Computer*, vol. 6, no. 20, pp. 402-417, 2004
- A. Escande, N. Mansard, P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," *IEEE Int. Conf. on Robotics and Automation*, pp. 3733-3738, 2010
- O. Kanoun, F. Lamiraux, P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. on Robotics*, vol. 27, no. 4, pp. 785-792, 2011
- A. Escande, N. Mansard, P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. Robotics Research*, vol. 33, no. 7, pp. 1006-1028, 2014 ([including software](#), also in <http://hal.archives-ouvertes.fr/hal-00751924>, 26 Dec 2012)
- M.D. Fiore, G. Meli, A. Ziese, B. Siciliano, C. Natale, "A general framework for hierarchical redundancy resolution under arbitrary constraints," *IEEE Trans. on Robotics*, vol. 39, no. 3, pp. 2468-2487, 2023



Bibliography - 2

- A. De Luca, G. Oriolo, "The reduced gradient method for solving redundancy in robot arms," *Robotersysteme*, vol. 7, no. 2, pp. 117-122, 1991
- A. De Luca, G. Oriolo, B. Siciliano, "Robot redundancy resolution at the acceleration level," *Laboratory Robotics and Automation*, vol. 4, no. 2, pp. 97-106, 1992
- A. De Luca, G. Oriolo, "Reconfiguration of redundant robots under kinematic inversion," *Advanced Robotics*, vol. 10, n. 3, pp. 249-263, 1996
- A. De Luca, G. Oriolo, P. Robuffo Giordano, "Kinematic control of nonholonomic mobile manipulators in the presence of steering wheels," *IEEE Int. Conf. on Robotics and Automation*, pp. 1792-1798, 2010
- F. Flacco, A. De Luca, O. Khatib, "Motion control of redundant robots under joint constraints: Saturation in the null space," *IEEE Int. Conf. on Robotics and Automation*, pp. 285-292, 2012
- F. Flacco, A. De Luca, O. Khatib, "Prioritized multi-task motion control of redundant robots under hard joint constraints," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3970-3977, 2012
- F. Flacco, A. De Luca, "Optimal redundancy resolution with task scaling under hard bounds in the robot joint space," *IEEE Int. Conf. on Robotics and Automation*, pp. 3969-3975, 2013
- F. Flacco, A. De Luca, "Fast redundancy resolution for high-dimensional robots executing prioritized tasks under hard bounds in the joint space," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2500-2506, 2013
- F. Flacco, A. De Luca, O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 637-654, 2015
- F. Flacco, A. De Luca, "Unilateral constraints in the Reverse Priority redundancy resolution method," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2564-2571, 2015
- A. Al Khudir, G. Halvorsen, L. Lanari, A. De Luca, "Stable torque optimization for redundant robots using a short preview," *IEEE Robotics and Automation Lett.*, vol 4, no, 2, pp. 2046-2057, 2019
- A. Kazemipour, M. Khatib, K. Al Khudir, A. De Luca, "Motion control of redundant robots with generalised inequality constraints," *3rd Italian Conf. on Robotics and Intelligent Machines*, pp. 138-140, 2021
- A. Kazemipour, M. Khatib, K. Al Khudir, C. Gaz, A. De Luca, "Kinematic control of redundant robots with online handling of variable generalized hard constraints," *IEEE Robotics and Automation Lett.*, vol 7, no, 4, pp. 9279-9286, 2022 ([github repository](#))



Appendix A - Recursive Task Priority

proof of recursive expression for null-space projector

$$P_{A,k} = P_{A,k-1} - (J_k P_{A,k-1})^\# J_k P_{A,k-1}$$

- proof based on a result on pseudoinversion of **partitioned** matrices (Cline: J. SIAM 1964)

$$\begin{pmatrix} A \\ B \end{pmatrix}^\# = \begin{pmatrix} A^\# - TBA^\# & T \end{pmatrix} \quad \begin{aligned} T &= E^\# + X(I - EE^\#) && X \text{ is irrelevant here} \\ E &= B(I - A^\# A) \end{aligned}$$

- (i) $P_{A,k} = I - J_{A,k}^\# J_{A,k} = I - \begin{pmatrix} J_{A,k-1} \\ J_k \end{pmatrix}^\# \begin{pmatrix} J_{A,k-1} \\ J_k \end{pmatrix}$
- (ii) $T = (J_k P_{A,k-1})^\# + X(I - (J_k P_{A,k-1})(J_k P_{A,k-1})^\#)$
- $\begin{aligned} &= I - \left(J_{A,k-1}^\# - TJ_k J_{A,k-1}^\# \quad T \right) \begin{pmatrix} J_{A,k-1} \\ J_k \end{pmatrix} && \text{(i) + (ii)} \Rightarrow \text{Q.E.D.} \\ &= I - J_{A,k-1}^\# J_{A,k-1} + TJ_k J_{A,k-1}^\# J_{A,k-1} - TJ_k \\ &= P_{A,k-1} - TJ_k P_{A,k-1} && \text{if } k\text{-th task is scalar} \\ &J_k = \text{single row } j_k^T \\ &P_{A,k} = P_{A,k-1} - \frac{P_{A,k-1} j_k j_k^T P_{A,k-1}}{\|P_{A,k-1} j_k\|^2} && \text{(Greville formula)} \end{aligned}$



Robotics 2

Dynamic model of robots: Lagrangian approach

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

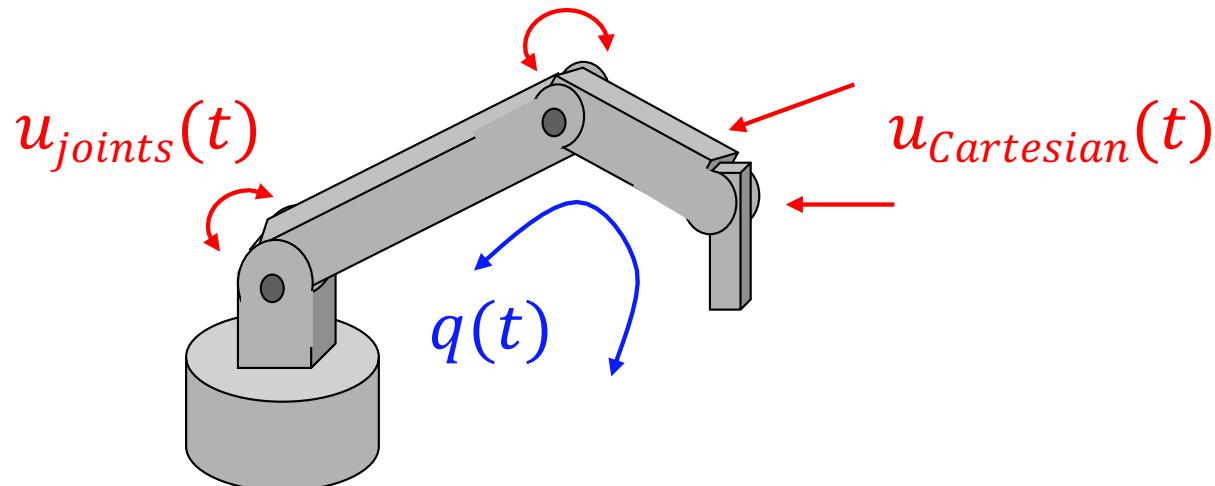




Dynamic model

- provides the **relation** between
both forces and torques
generalized forces $u(t)$ acting on the robot

↑
robot motion, i.e.,
assumed configurations $q(t)$ over time



a system of 2nd order
differential equations

$$\Phi(q, \dot{q}, \ddot{q}) = u$$



Direct dynamics

- direct relation

$$u(t) = \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix} \quad \xrightarrow{\text{red arrow}} \quad \text{Robot Arm Diagram} \quad \xrightarrow{\text{green arrow}} \quad q(t) = \begin{pmatrix} q_1 \\ \vdots \\ q_N \end{pmatrix}$$

input for $t \in [0, T]$ + $q(0), \dot{q}(0)$ resulting motion
initial state at $t = 0$

- experimental solution
 - apply torques/forces with motors and measure joint variables with encoders (with sampling time T_c)
- solution by simulation
 - use dynamic model and integrate numerically the differential equations (with simulation step $T_s \leq T_c$)



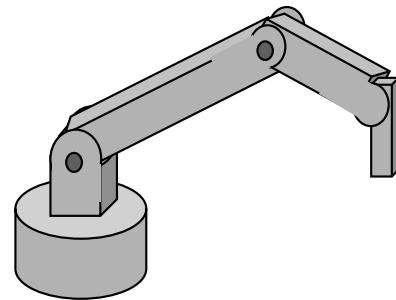
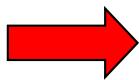
$$\Phi(q, \dot{q}, \ddot{q}) = u$$



Inverse dynamics

- inverse relation

$q_d(t), \dot{q}_d(t), \ddot{q}_d(t)$



$u_d(t)$

desired motion
for $t \in [0, T]$

required input
for $t \in [0, T]$

- experimental solution

- e.g., by repeated motion trials of direct dynamics using $u_k(t)$, with iterative learning of nominal torques updated on trial $k + 1$ based on the error in $[0, T]$ measured in trial k : $\lim_{k \rightarrow \infty} u_k(t) \Rightarrow u_d(t)$

- analytic solution



$$\Phi(q, \dot{q}, \ddot{q}) = u$$

- use dynamic model and compute algebraically the values $u_d(t)$ at every time instant t



Approaches to dynamic modeling

Euler-Lagrange method
(energy-based approach)



Newton-Euler method
(balance of forces/moments)

- dynamic equations in **symbolic**/closed form
 - best for study of dynamic properties and analysis of control schemes
 - many other formal methods based on basic principles in mechanics are available for the derivation of the robot dynamic model:
 - principle of d'Alembert, of Hamilton, of virtual works, Kane's equations ...
- dynamic equations in **numeric**/recursive form
 - best for implementation of control schemes (inverse dynamics in real time)



Euler-Lagrange method (energy-based approach)

basic assumption: the N links in motion are considered as **rigid bodies**
(+ later on, include also **concentrated elasticity** at the joints)

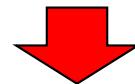
$q \in \mathbb{R}^N$ generalized coordinates (e.g., joint variables, but not only!)

Lagrangian

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q)$$

kinetic energy – potential energy

- principle of least action of Hamilton
- principle of virtual works



Euler-Lagrange
equations

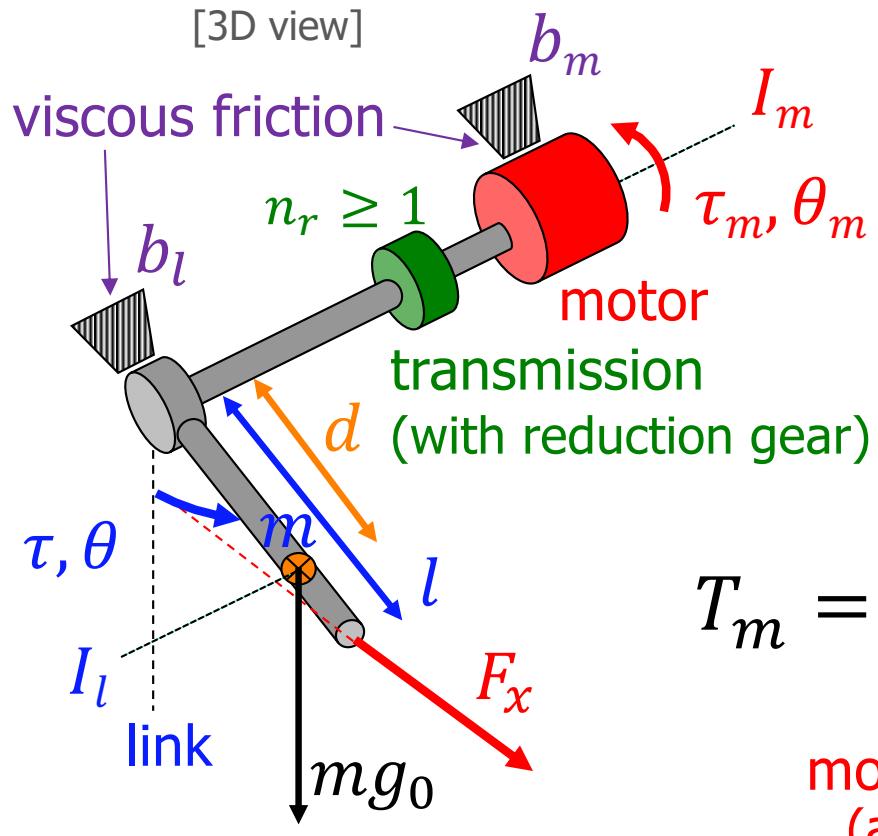
$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = u_i \quad i = 1, \dots, N$$

non-conservative (external or dissipative)
generalized forces performing work on q_i



Dynamics of an actuated pendulum

a first example



$$\dot{\theta}_m = n_r \dot{\theta} \rightarrow \theta_m = n_r \theta + \cancel{\theta_{m0}} = 0$$

$$\tau = n_r \tau_m$$

$q = \theta$ (or $q = \theta_m$)

$$T = T_m + T_l$$

$$T_m = \frac{1}{2} I_m \dot{\theta}_m^2 \quad T_l = \frac{1}{2} (I_l + md^2) \dot{\theta}^2$$

motor inertia (around its spinning axis)

link inertia (around the z-axis through its center of mass ...)

(... around the || axis through its base)

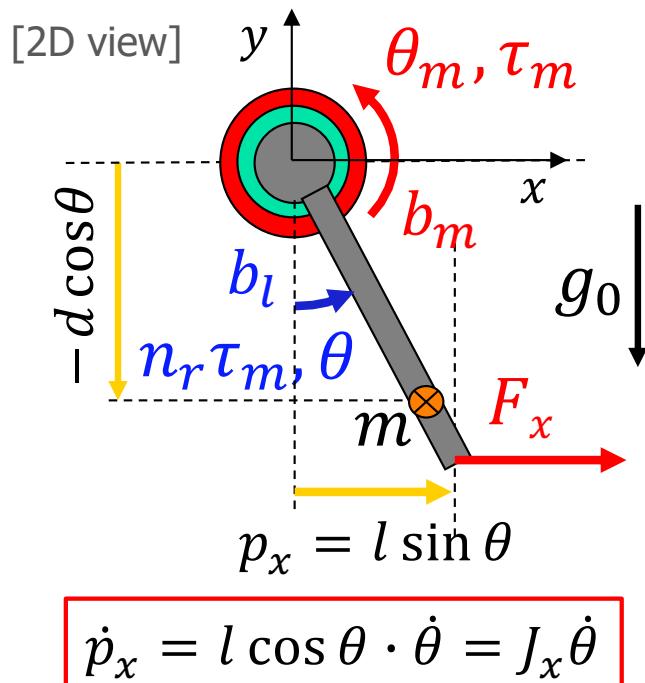
kinetic energy

$$T = \frac{1}{2} (I_l + md^2 + I_m n_r^2) \dot{\theta}^2 = \frac{1}{2} I \dot{\theta}^2$$



Dynamics of an actuated pendulum

(continued)



$$U = U_0 - mg_0 d \cos \theta$$

potential energy

$$L = T - U = \frac{1}{2} I \dot{\theta}^2 + mg_0 d \cos \theta - U_0$$

$$\frac{\partial L}{\partial \dot{\theta}} = I \dot{\theta}$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} = I \ddot{\theta}$$

$$\frac{\partial L}{\partial \theta} = -mg_0 d \sin \theta$$

$$u = n_r \tau_m - b_l \dot{\theta} - n_r b_m \dot{\theta}_m + J_x^T F_x = n_r \tau_m - (b_l + b_m n_r^2) \dot{\theta} + l \cos \theta F_x$$

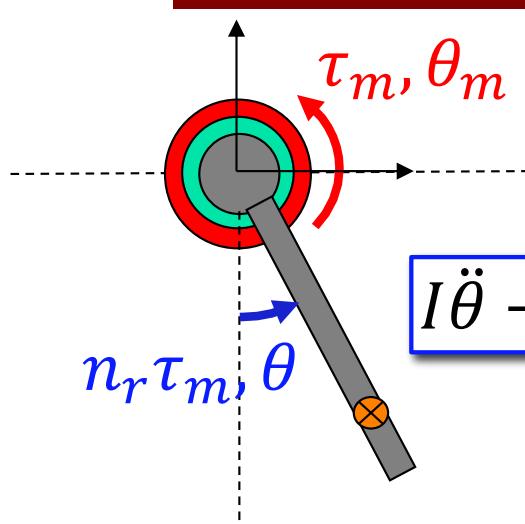
↑
applied or dissipated torques
on motor side are multiplied by n_r
when moved to the link side

equivalent joint torque
due to force F_x applied to
the tip at point p_x

“sum” of
non-conservative
torques



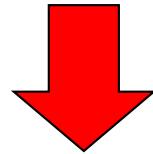
Dynamics of an actuated pendulum (cont)



dynamic model in $q = \theta$

$$I\ddot{\theta} + mg_0d \sin \theta = n_r \tau_m - (b_l + b_m n_r^2) \dot{\theta} + l \cos \theta \cdot F_x$$

dividing by n_r and substituting $\theta = \theta_m/n_r$



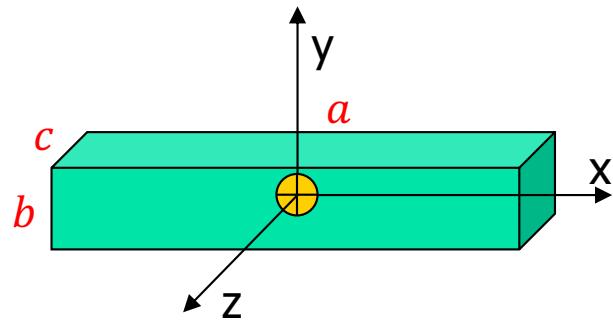
$$\frac{I}{n_r^2} \ddot{\theta}_m + \frac{m}{n_r} g_0 d \sin \frac{\theta_m}{n_r} = \tau_m - \left(\frac{b_l}{n_r^2} + b_m \right) \dot{\theta}_m + \frac{l}{n_r} \cos \frac{\theta_m}{n_r} \cdot F_x$$

dynamic model in $q = \theta_m$



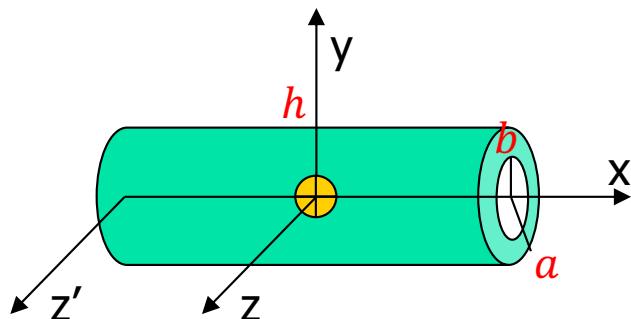
Examples of body inertia matrices

homogeneous bodies of mass m , with axes of symmetry



parallelepiped with sides
 a (length/height), b and c (base)

$$I_c = \begin{pmatrix} I_{xx} & & \\ & I_{yy} & \\ & & I_{zz} \end{pmatrix} = \begin{pmatrix} \frac{1}{12}m(b^2 + c^2) & & \\ & \frac{1}{12}m(a^2 + c^2) & \\ & & \frac{1}{12}m(a^2 + b^2) \end{pmatrix}$$



empty cylinder with length h ,
and external/internal radius a and b

$$I_c = \begin{pmatrix} \frac{1}{2}m(a^2 + b^2) & & \\ & \frac{1}{12}m(3(a^2 + b^2) + h^2) & \\ & & I_{zz} \end{pmatrix} \quad I_{zz} = I_{yy}$$

$$I'_{zz} = I_{zz} + m\left(\frac{h}{2}\right)^2 \quad (\text{parallel}) \text{ axis translation theorem}$$

to compute the inertia around an axis
different from the axis of the CoM

$$I = I_c + m(r^T r \cdot E_{3 \times 3} - rr^T) = I_c + m S^T(r)S(r)$$

body inertia matrix
relative to the CoM

identity
matrix

Homework:
prove last equality

skew-
symmetric

... its generalization:
changes on body inertia matrix
due to a pure translation r of
the reference frame



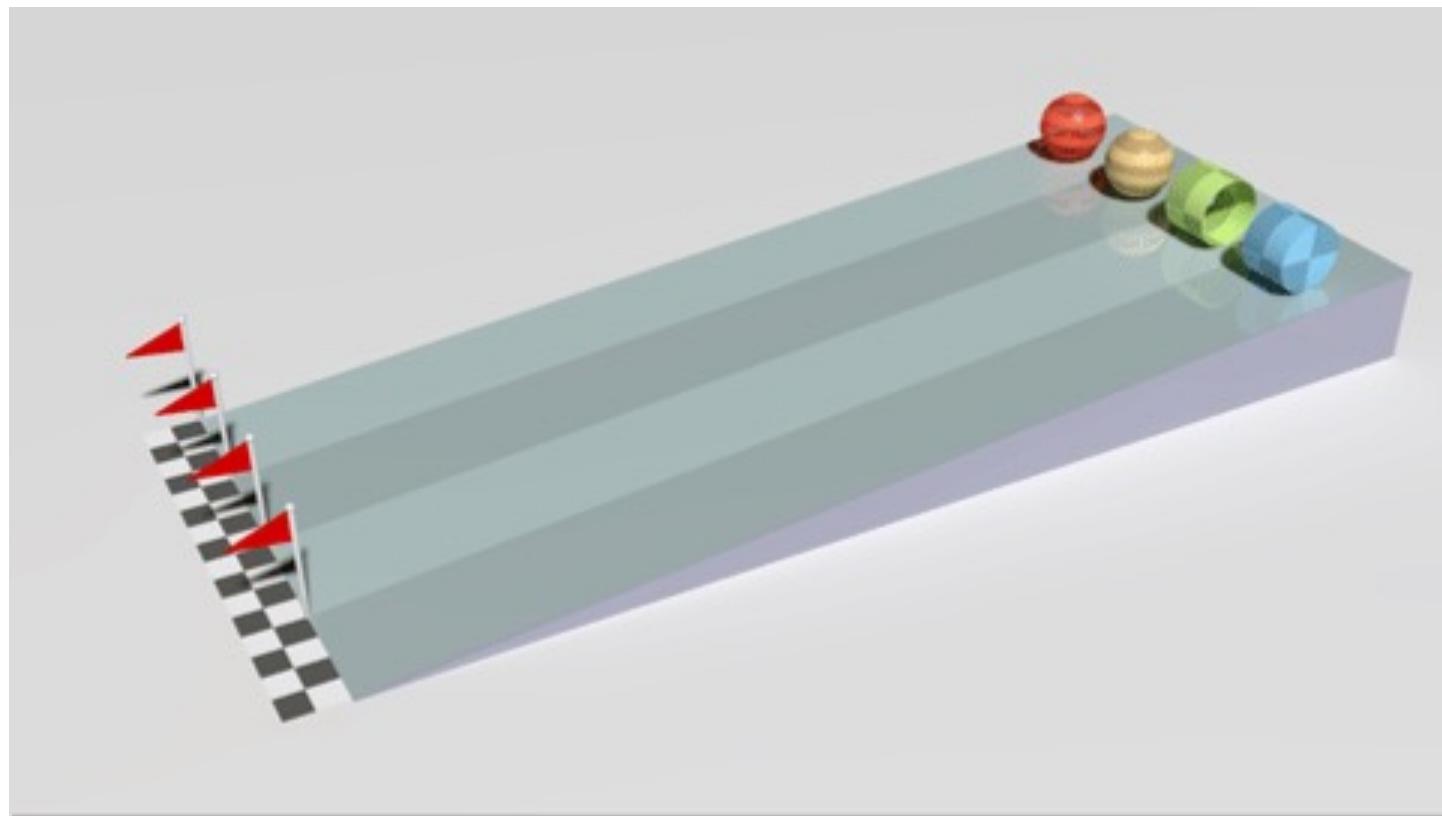
Rolling inertias

4 “circular” bodies
with the same
mass & radius
rolling down
an inclined plane
without slipping



time to reach
the finish line
depends on their
**moment of
inertia**
(about rolling axis!)

https://en.wikipedia.org/wiki/Moment_of_inertia

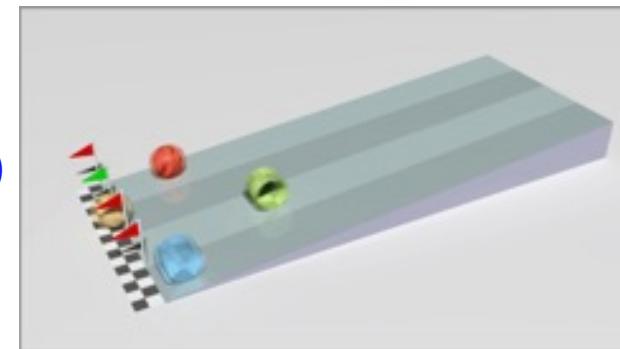


from back to front:

- █ spherical shell
- █ solid sphere
- █ cylindrical ring
- █ solid cylinder

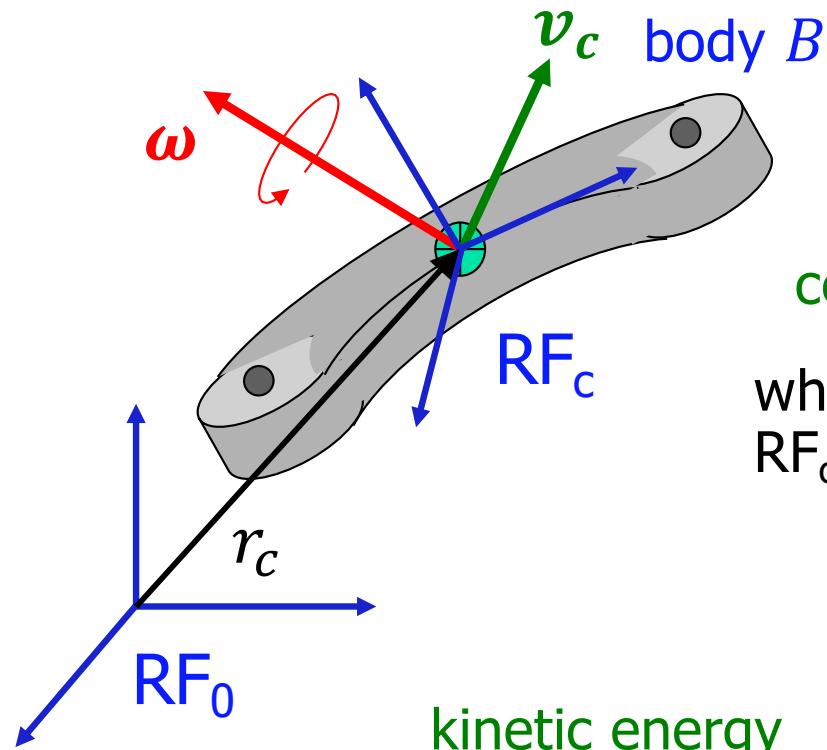


- 3^{rd}
- $1^{\text{st}} \text{ (smallest)}$
- $4^{\text{th}} \text{ (largest)}$
- 2^{nd}





Kinetic energy of a rigid body



kinetic energy

(fundamental)
kinematic relation
for a rigid body

mass density

$$\text{mass } m = \int_B \rho(x, y, z) dx dy dz = \int_B dm$$

position of
center of mass (CoM)

$$r_c = \frac{1}{m} \int_B r dm$$

when all vectors are referred to a body frame
 RF_c attached to the CoM, then

$$r_c = 0 \Rightarrow \int_B r dm = 0$$

$$T = \frac{1}{2} \int_B v^T(x, y, z) v(x, y, z) dm$$

$$v = v_c + \omega \times r = v_c + S(\omega) r$$

↑
skew-symmetric matrix



Kinetic energy of a rigid body (cont)

$$T = \frac{1}{2} \int_B (v_c + S(\omega)r)^T (v_c + S(\omega)r) dm$$

$$= \frac{1}{2} \int_B v_c^T v_c dm + \int_B v_c^T S(\omega) r dm + \frac{1}{2} \int_B r^T S^T(\omega) S(\omega) r dm$$

$$= \frac{1}{2} m v_c^T v_c$$

translational
kinetic energy
(point mass
at CoM)

$$= v_c^T S(\omega) \int_B r dm = 0$$

zero for the property in the slide above

+ rotational
kinetic energy
(of the whole body)

$$= \frac{1}{2} \int_B \omega^T S^T(r) S(r) \omega dm$$

$$= \frac{1}{2} \omega^T \left(\int_B S^T(r) S(r) dm \right) \omega$$

$$= \frac{1}{2} \omega^T I_c \omega$$

Homework
compute "once in your life"
the expression of the
elements of inertia matrix I_c

body inertia matrix
(around the CoM)

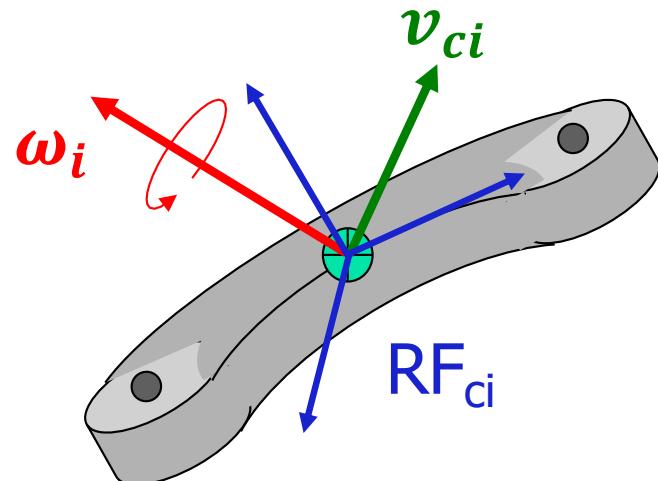
König theorem



Robot kinetic energy

$$T = \sum_{i=1}^N T_i \quad \text{N rigid bodies (+ fixed base)}$$

$$T_i = T_i(q_j, \dot{q}_j; j \leq i) \quad \text{open kinematic chain}$$



i-th link (body)
of the robot

König theorem

$$T_i = \frac{1}{2} m_i v_{ci}^T v_{ci} + \frac{1}{2} \omega_i^T I_{ci} \omega_i$$

absolute velocity
of the center of mass
(CoM)

absolute
angular velocity
of whole body



Kinetic energy of a robot link

$$T_i = \frac{1}{2} m_i v_{ci}^T v_{ci} + \frac{1}{2} \omega_i^T I_{ci} \omega_i$$

ω_i, I_{ci} should be expressed in the **same reference frame**,
but the product $\omega_i^T I_{ci} \omega_i$ is **invariant** w.r.t. any chosen frame

$$\begin{aligned} {}^0\omega_i^T {}^0I_{ci}(q) {}^0\omega_i &= ({}^0R_i(q) {}^i\omega_i)^T {}^0I_{ci}(q) ({}^0R_i(q) {}^i\omega_i) = {}^i\omega_i^T ({}^0R_i^T(q) {}^0I_{ci}(q) {}^0R_i(q)) {}^i\omega_i \\ &= {}^i\omega_i^T {}^iI_{ci} {}^i\omega_i \quad \rightarrow \quad {}^0I_{ci}(q) = {}^0R_i(q) {}^iI_{ci} {}^0R_i^T(q) \end{aligned}$$

in frame RF_{ci} attached to (the center of mass of) link i

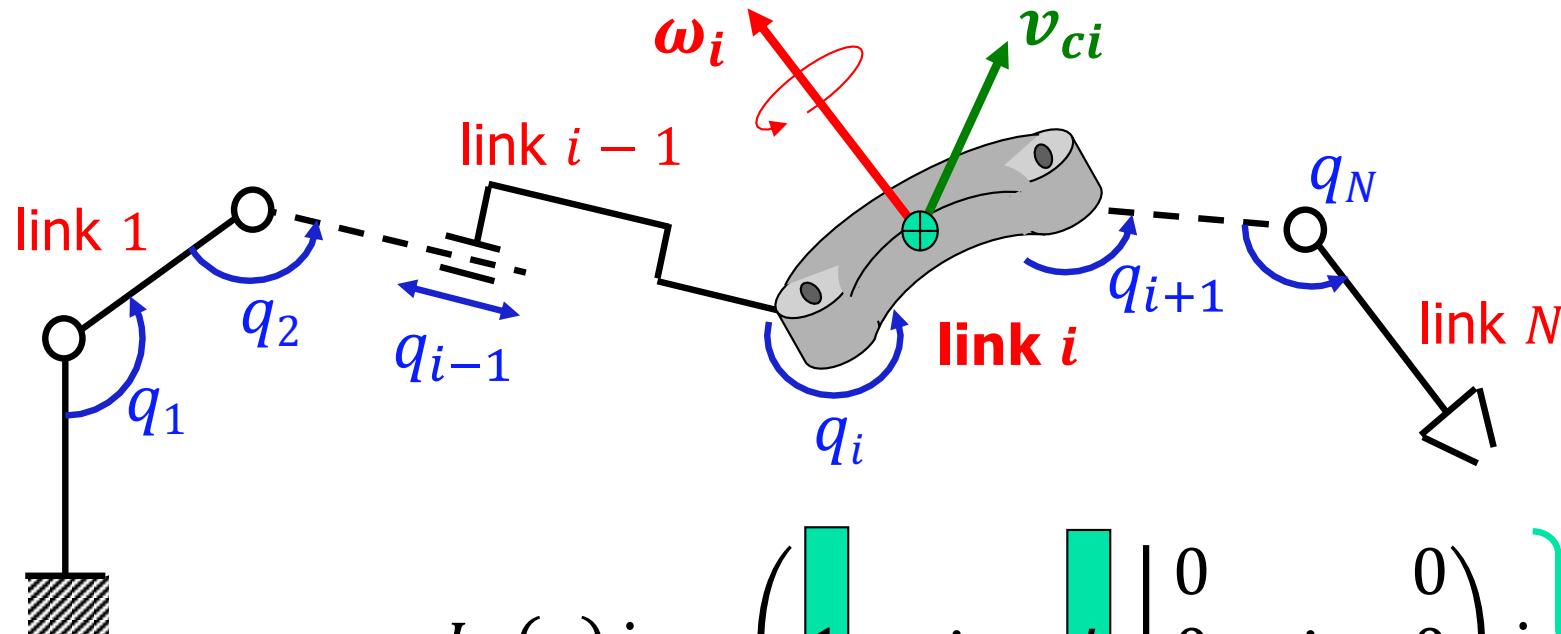
↑
constant!

$${}^iI_{ci} = \begin{pmatrix} \int (y^2 + z^2) dm & - \int xy dm & - \int xz dm \\ & \int (x^2 + z^2) dm & - \int yz dm \\ & & \int (x^2 + y^2) dm \end{pmatrix}$$

symm



Dependence of T from q and \dot{q}



$$v_{ci} = J_{Li}(q)\dot{q} = \left(\begin{array}{c|c|c|c} 1 & \vdots & i & 0 \\ \vdots & & 0 & 0 \\ 0 & & 0 & 0 \end{array} \right) \dot{q} \quad \text{3 rows}$$

(partial) Jacobians
typically expressed in RF₀

$$\omega_i = J_{Ai}(q)\dot{q} = \left(\begin{array}{c|c|c|c} 1 & \vdots & i & 0 \\ \vdots & & 0 & 0 \\ 0 & & 0 & 0 \end{array} \right) \dot{q} \quad \text{3 rows}$$



Final expression of T

$$T = \frac{1}{2} \sum_{i=1}^N (m_i v_{ci}^T v_{ci} + \omega_i^T I_{ci} \omega_i)$$

NOTE 1:
in practice, **NEVER**
use this formula
(or partial Jacobians)
for computing T
 \Rightarrow a better method
is available...

$$= \frac{1}{2} \dot{q}^T \left(\sum_{i=1}^N m_i J_{Li}^T(q) J_{Li}(q) + J_{Ai}^T(q) I_{ci}(q) J_{Ai}(q) \right) \dot{q}$$

constant when ω_i
is expressed in RF_{ci}
else

$$T = \frac{1}{2} \dot{q}^T M(q) \dot{q}$$

$${}^0 I_{ci}(q) = {}^0 R_i(q) {}^i I_{ci} {}^0 R_i^T(q)$$

NOTE 2:
I used previously
the notation $B(q)$
for the robot
inertia matrix ...
(see past exams!)

robot (generalized) inertia matrix

- symmetric
- positive definite, $\forall q \Rightarrow$ **always invertible**



Robot potential energy

assumption: GRAVITY contribution only

$$U = \sum_{i=1}^N U_i \quad \leftarrow \quad N \text{ rigid bodies (+ fixed base)}$$

$$U_i = U_i(q_j; j \leq i) \quad \leftarrow \quad \text{open kinematic chain}$$

$$U_i = -m_i g^T r_{0,ci}$$

{ gravity acceleration vector position of the center of mass of link i }

typically expressed in RF₀

dependence on q

$$\begin{pmatrix} r_{0,ci} \\ 1 \end{pmatrix} = {}^0A_1(q_1) {}^1A_2(q_2) \cdots {}^{i-1}A_i(q_i) \begin{pmatrix} r_{i,ci} \\ 1 \end{pmatrix}$$

constant in RF_i

NOTE: need to work with homogeneous coordinates



Summarizing ...

kinetic
energy

$$T = \frac{1}{2} \dot{q}^T M(q) \dot{q} = \frac{1}{2} \sum_{i,j} m_{ij}(q) \dot{q}_i \dot{q}_j$$

positive definite
quadratic form

$$\boxed{T \geq 0, \\ T = 0 \Leftrightarrow \dot{q} = 0}$$

potential
energy

$$U = U(q)$$

Lagrangian

$$L = T(q, \dot{q}) - U(q)$$

Euler-Lagrange
equations

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = u_k$$

$$k = 1, \dots, N$$

non-conservative (active/dissipative)
generalized forces

performing work on q_k coordinate



Applying Euler-Lagrange equations

(the scalar derivation – see Appendix for vector format)

$$L(q, \dot{q}) = \frac{1}{2} \sum_{i,j} m_{ij}(q) \dot{q}_i \dot{q}_j - U(q)$$

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_j m_{kj} \dot{q}_j \quad \rightarrow \quad \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} = \sum_j m_{kj} \ddot{q}_j + \sum_{i,j} \frac{\partial m_{kj}}{\partial q_i} \dot{q}_i \dot{q}_j$$

(dependences of elements on q are not shown)

$$\frac{\partial L}{\partial q_k} = \frac{1}{2} \sum_{i,j} \frac{\partial m_{ij}}{\partial q_k} \dot{q}_i \dot{q}_j - \frac{\partial U}{\partial q_k}$$

LINEAR terms in ACCELERATION \ddot{q}

QUADRATIC terms in VELOCITY \dot{q}

NONLINEAR terms in CONFIGURATION q



k -th dynamic equation ...

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = u_k$$

$$\sum_j m_{kj} \ddot{q}_j + \sum_{i,j} \left(\frac{\partial m_{kj}}{\partial q_i} - \frac{1}{2} \frac{\partial m_{ij}}{\partial q_k} \right) \dot{q}_i \dot{q}_j + \frac{\partial U}{\partial q_k} = u_k$$

exchanging
“mute” indices i, j

$$\dots + \sum_{i,j} \frac{1}{2} \left(\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k} \right) \dot{q}_i \dot{q}_j + \dots$$

$c_{kij} = c_{kji}$ Christoffel symbols
of the first kind



... and interpretation of dynamic terms

$$\sum_j m_{kj}(q) \ddot{q}_j + \sum_{i,j} c_{kij}(q) \dot{q}_i \dot{q}_j + \frac{\partial U}{\partial q_k} = u_k \quad k = 1, \dots, N$$

INERTIAL CENTRIFUGAL ($i = j$)
 terms and CORIOLIS ($i \neq j$) terms GRAVITY
terms $g_k(q)$

$m_{kk}(q)$ = inertia at joint k when joint k accelerates ($m_{kk} > 0!!$)

$m_{kj}(q)$ = inertia “seen” at joint k when joint j accelerates ($= m_{jk}(q)$)

$c_{kii}(q)$ = coefficient of the centrifugal force at joint k when joint i is moving ($c_{iii} = 0, \forall i$)

$c_{kij}(q)$ = coefficient of the Coriolis force at joint k when joint i and joint j are both moving ($= c_{kji}(q)$)



Robot dynamic model in vector formats

1. $M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u$

extracted
from T

k -th column
of matrix $M(q)$

$$g(q) = \left(\frac{\partial U}{\partial q} \right)^T$$

$$c_k(q, \dot{q}) = \dot{q}^T C_k(q) \dot{q}$$

k -th component
of vector c

$$C_k(q) = \frac{1}{2} \left(\frac{\partial M_k}{\partial q} + \left(\frac{\partial M_k}{\partial q} \right)^T - \frac{\partial M}{\partial q_k} \right)$$

symmetric
matrix!

2. $M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) = u$

NOTE:
the model
is in the form
 $\Phi(q, \dot{q}, \ddot{q}) = u$
as expected

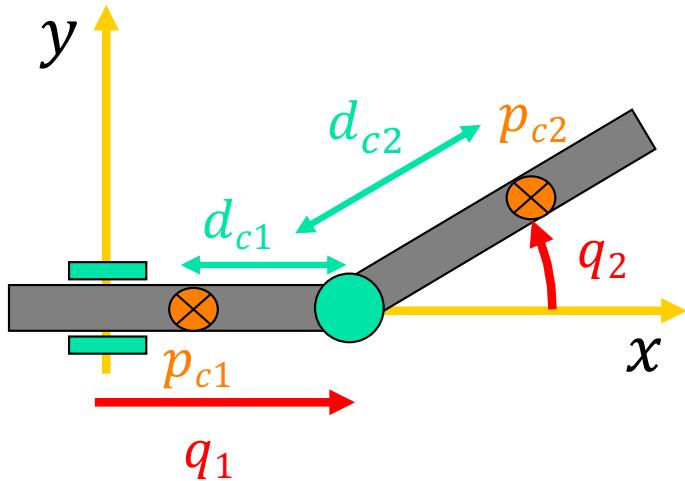
NOT a
symmetric
matrix
in general

$$S_{kj}(q, \dot{q}) = \sum_i c_{kij}(q) \dot{q}_i$$

factorization of c
by S is **not unique!**



Dynamic model of a PR robot



$$T = T_1 + T_2 \quad U = \text{constant} \Rightarrow g(q) \equiv 0 \quad (\text{on horizontal plane})$$

$$p_{c1} = \begin{pmatrix} q_1 - d_{c1} \\ 0 \\ 0 \end{pmatrix} \quad \text{we assume, despite the slide is vertical, that the plane is horizontal} \rightarrow \|v_{c1}\|^2 = \dot{p}_{c1}^T \dot{p}_{c1} = \dot{q}_1^2$$

$$T_1 = \frac{1}{2} m_1 \dot{q}_1^2$$

$$T_2 = \frac{1}{2} m_2 v_{c2}^T v_{c2} + \frac{1}{2} \omega_2^T I_{c2} \omega_2$$

$$p_{c2} = \begin{pmatrix} q_1 + d_{c2} \cos q_2 \\ d_{c2} \sin q_2 \\ 0 \end{pmatrix} \quad \rightarrow \quad v_{c2} = \begin{pmatrix} \dot{q}_1 - d_{c2} \sin q_2 \dot{q}_2 \\ d_{c2} \cos q_2 \dot{q}_2 \\ 0 \end{pmatrix} \quad \omega_2 = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_2 \end{pmatrix} \quad \text{since the body is on the plane the w has a non-null components only along the axis normal to the plane}$$

$$T_2 = \frac{1}{2} m_2 (\dot{q}_1^2 + d_{c2}^2 \dot{q}_2^2 - 2d_{c2} \sin q_2 \dot{q}_1 \dot{q}_2) + \frac{1}{2} I_{c2,zz} \dot{q}_2^2$$

the kinetic energy of the second joint has a quadratic velocity that multiplies the inertia around the CoM of second link + $m_2^2 d_{c2}^2$. That is exactly the parallel axis theorem 24



Dynamic model of a PR robot (cont)

$$M(q) = \begin{pmatrix} m_1 + m_2 & -m_2 d_{c2} \sin q_2 \\ -m_2 d_{c2} \sin q_2 & I_{c2,zz} + m_2 d_{c2}^2 \end{pmatrix}$$

$M_1 \qquad \qquad \qquad M_2$

$$c(q, \dot{q}) = \begin{pmatrix} c_1(q, \dot{q}) \\ c_2(q, \dot{q}) \end{pmatrix}$$

$$c_k(q, \dot{q}) = \dot{q}^T C_k(q) \dot{q}$$

where $C_k(q) = \frac{1}{2} \left(\frac{\partial M_k}{\partial q} + \left(\frac{\partial M_k}{\partial q} \right)^T - \frac{\partial M}{\partial q_k} \right)$

$$C_1(q) = \frac{1}{2} \left(\begin{pmatrix} 0 & 0 \\ 0 & -m_2 d_{c2} \cos q_2 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & -m_2 d_{c2} \cos q_2 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right)$$

c_1 is the centrifugal force that link 1 feels when joint 2 is rotating

$$c_1(q, \dot{q}) = -m_2 d_{c2} \cos q_2 \dot{q}_2^2$$

$$C_2(q) = \frac{1}{2} \left(\begin{pmatrix} 0 & -m_2 d_{c2} \cos q_2 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -m_2 d_{c2} \cos q_2 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ -m_2 d_{c2} \cos q_2 & 0 \end{pmatrix} \right) = 0$$

$$c_2(q, \dot{q}) = 0$$



Dynamic model of a PR robot (cont)

$$M(q)\ddot{q} + c(q, \dot{q}) = u$$



we can use this equation to do direct dynamics, applying u to get q

j1 prismatic $\rightarrow u_1$ force
j2 revolute $\rightarrow u_2$ torque

$$\begin{pmatrix} m_1 + m_2 & -m_2 d_{c2} \sin q_2 \\ -m_2 d_{c2} \sin q_2 & I_{c2,zz} + m_2 d_{c2}^2 \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + \begin{pmatrix} -m_2 d_{c2} \cos q_2 \dot{q}_2^2 \\ 0 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

↑ because the inertia seen when you are accelerating only the last joint, all the rest is fixed, so is a single link

NOTE: the m_{NN} element (here, for $N = 2$) of $M(q)$ is always **constant!**

q_1 is defined w.r.t RF0 that is arbitrary, so an intrinsic, physical property like the inertia cannot depend on $q_1 \rightarrow q_1$ is called cyclic variable

Q1: why is variable q_1 not appearing in $M(q)$? ... this is a **general property!**

Q2: why are Coriolis terms not present?

Q3: when applying a force u_1 , does the second joint accelerate? ... always?

Q4: what is the expression of a factorization matrix S ? ... is it unique here?

Q5: what if the PR robot was moving in a vertical plane? ... just add $g(q)$!



A structural property

Matrix $\dot{M} - 2S$ is skew-symmetric
(when using Christoffel symbols to define matrix S)

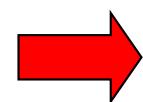
Proof

$$\dot{m}_{kj} = \sum_i \frac{\partial m_{kj}}{\partial q_i} \dot{q}_i \quad 2s_{kj} = \sum_i 2c_{kij} \dot{q}_i = \sum_i \left(\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k} \right) \dot{q}_i$$

$$\rightarrow \dot{m}_{kj} - 2s_{kj} = \sum_i \left(\frac{\partial m_{ij}}{\partial q_k} - \frac{\partial m_{ki}}{\partial q_j} \right) \dot{q}_i = n_{kj}$$

$$n_{jk} = \dot{m}_{jk} - 2s_{jk} = \sum_i \left(\frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{ji}}{\partial q_k} \right) \dot{q}_i = -n_{kj}$$

using the
symmetry of M



$$x^T (\dot{M} - 2S)x = 0, \forall x$$



Energy conservation

- total robot energy

$$E = T + U = \frac{1}{2} \dot{q}^T M(q) \dot{q} + U(q)$$

- its evolution **over time** (using the dynamic model)

$$\begin{aligned}\dot{E} &= \dot{q}^T M(q) \ddot{q} + \frac{1}{2} \dot{q}^T \dot{M}(q) \dot{q} + \frac{\partial U}{\partial q} \dot{q} \\ &= \dot{q}^T (u - S(q, \dot{q}) \dot{q} - g(q)) + \frac{1}{2} \dot{q}^T \dot{M}(q) \dot{q} + \dot{q}^T g(q) \\ &= \dot{q}^T u + \frac{1}{2} \dot{q}^T (\dot{M}(q) - 2S(q, \dot{q})) \dot{q}\end{aligned}$$

here, any factorization of vector c by a matrix S can be used

- if $u \equiv 0$, **total energy is constant** (no dissipation or increase)

$$\dot{E} = 0 \quad \rightarrow \quad \boxed{\dot{q}^T (\dot{M}(q) - 2S(q, \dot{q})) \dot{q} = 0, \forall q, \dot{q}}$$

it is a weaker property than skew-symmetry, as the external velocity \dot{q} in the quadratic form is the **same** inside the two matrices \dot{M} and S

$$\rightarrow \boxed{\dot{E} = \dot{q}^T u}$$

in general, the variation of the total energy is equal to the work of non-conservative forces



Appendix

dynamic model: alternative vector format derivation

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right)^T - \left(\frac{\partial L}{\partial q} \right)^T = u$$

$$L = \frac{1}{2} \dot{q}^T M(q) \dot{q} - U(q)$$

$$M(q) = \begin{pmatrix} M_1(q) & \cdots & M_i(q) & \cdots & M_N(q) \end{pmatrix}$$

$$\left(\frac{\partial L}{\partial \dot{q}} \right)^T = (\dot{q}^T M(q))^T = M(q) \dot{q}$$

$$\sum_{i=1}^N M_i(q) e_i^T$$

$\begin{matrix} (0 & \cdots & 1 & \cdots & 0) \\ \uparrow & & & & \\ i\text{-th position} \end{matrix}$

dyadic expansion

$$\rightarrow \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right)^T = M(q) \ddot{q} + \dot{M}(q) \dot{q} = M(q) \ddot{q} + \sum_{i=1}^N \left(\frac{\partial M_i}{\partial q} \right) \dot{q}_i \dot{q}$$

$$\left(\frac{\partial L}{\partial q} \right)^T = \left(\frac{1}{2} \dot{q}^T \left(\sum_{i=1}^N \frac{\partial M_i}{\partial q} e_i^T \right) \dot{q} - \frac{\partial U}{\partial q} \right)^T = \frac{1}{2} \sum_{i=1}^N \left(\frac{\partial M_i}{\partial q} \right)^T \dot{q}_i \dot{q} - \left(\frac{\partial U}{\partial q} \right)^T$$

special definition
of S matrix giving
skew-symmetry

to $\dot{M} = 2S$

$$\rightarrow M(q) \ddot{q} + \left(\sum_{i=1}^N \left(\frac{\partial M_i}{\partial q} - \frac{1}{2} \left(\frac{\partial M_i}{\partial q} \right)^T \right) \dot{q}_i \right) \dot{q} + \left(\frac{\partial U}{\partial q} \right)^T = u$$

k -th row of matrix S

$$S_k^T(q, \dot{q}) = \dot{q}^T C_k(q)$$

$$\longleftrightarrow S(q, \dot{q}) \text{ (generic)}$$

$$g(q)$$



Robotics 2

Dynamic model of robots: Algorithm for computing kinetic energy

Prof. Alessandro De Luca

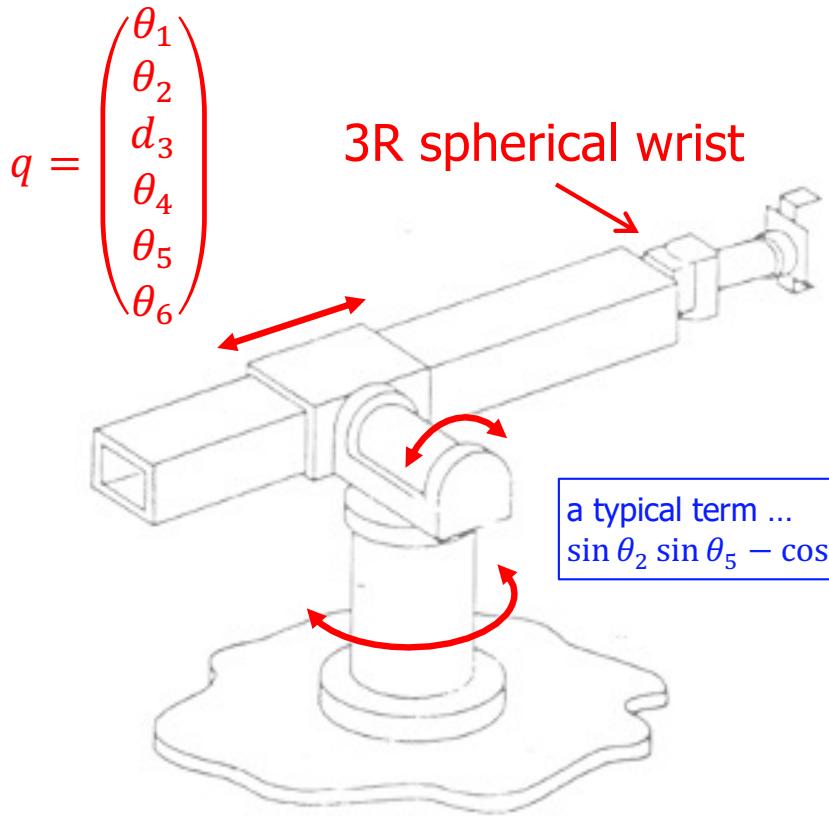
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Complexity of robot inertia terms

element $m_{11}(q)$ of Stanford arm



$$m_{11} = m_1 k_{122}^2$$

$$+ m_2 \left[k_{211}^2 s^2 \theta_2 + k_{233}^2 c^2 \theta_2 + r_2 (2\bar{y}_2 + r_2) \right]$$

$$+ m_3 \left[k_{322}^2 s^2 \theta_2 + k_{333}^2 c^2 \theta_2 + r_3 (2\bar{z}_3 + r_3) s^2 \theta_2 + r_2^2 \right]$$

$$+ m_4 \left\{ \frac{1}{2} k_{411}^2 \left[s^2 \theta_2 (2s^2 \theta_4 - 1) + s^2 \theta_4 \right] + \frac{1}{2} k_{422}^2 (1 + c^2 \theta_2 + s^2 \theta_4) \right.$$

$$\left. + \frac{1}{2} k_{433}^2 \left[s^2 \theta_2 (1 - 2s^2 \theta_4) - s^2 \theta_4 \right] + r_3^2 s^2 \theta_2 + r_2^2 - 2\bar{y}_4 r_3 s^2 \theta_2 + 2\bar{z}_4 (r_2 s \theta_4 + r_3 s \theta_2 c \theta_2 c \theta_4) \right\}$$

$$+ m_5 \left\{ \frac{1}{2} (-k_{511}^2 + k_{522}^2 + k_{533}^2) \left[(s \theta_2 s \theta_5 - c \theta_2 s \theta_4 c \theta_5)^2 + c^2 \theta_4 c^2 \theta_5 \right] \right.$$

$$\left. + \frac{1}{2} (k_{511}^2 - k_{522}^2 + k_{533}^2) (s^2 \theta_4 + c^2 \theta_2 c^2 \theta_4) \right\}$$

$$+ \frac{1}{2} (k_{511}^2 + k_{522}^2 - k_{533}^2) \left[(s \theta_2 c \theta_5 + c \theta_2 s \theta_4 s \theta_5)^2 + c^2 \theta_4 s^2 \theta_5 \right] + r_3^2 s^2 \theta_2 + r_2^2$$

$$+ 2\bar{z}_5 \left[r_3 (s^2 \theta_2 c \theta_5 + s \theta_2 s \theta_4 c \theta_4 s \theta_5) - r_2 c \theta_4 s \theta_5 \right] \right\}$$

$$+ m_6 \left\{ \frac{1}{2} (-k_{611}^2 + k_{622}^2 + k_{633}^2) \left[(s \theta_2 s \theta_5 c \theta_6 - c \theta_2 s \theta_4 c \theta_5 c \theta_6 - c \theta_2 c \theta_4 s \theta_6)^2 + (c \theta_4 c \theta_5 c \theta_6 - s \theta_4 s \theta_6)^2 \right] \right.$$

$$\left. + \frac{1}{2} (k_{611}^2 - k_{622}^2 + k_{633}^2) \left[(c \theta_2 s \theta_4 c \theta_5 s \theta_6 - s \theta_2 s \theta_5 s \theta_6 - c \theta_2 c \theta_4 c \theta_6)^2 + (c \theta_4 c \theta_5 s \theta_6 + s \theta_4 c \theta_6)^2 \right] \right\}$$

$$+ \frac{1}{2} (k_{611}^2 + k_{622}^2 - k_{633}^2) \left[(c \theta_2 s \theta_4 s \theta_5 + s \theta_2 c \theta_5)^2 + c^2 \theta_4 s^2 \theta_5 \right]$$

$$+ \left[r_6 c \theta_2 s \theta_4 s \theta_5 + (r_6 c \theta_5 + r_3) s \theta_2^2 + (r_6 c \theta_4 s \theta_5 - r_2)^2 \right]$$

$$+ 2\bar{z}_6 \left[r_6 (s^2 \theta_2 c^2 \theta_5 + c^2 \theta_4 s^2 \theta_5 + c^2 \theta_2 s^2 \theta_4 s \theta_5 + 2s \theta_2 c \theta_2 s \theta_4 s \theta_5 c \theta_5) \right]$$

$$+ r_3 (s \theta_2 c \theta_2 s \theta_4 s \theta_5 + s^2 \theta_2 c \theta_5) - r_2 c \theta_4 s \theta_5 \right\}$$

... (derived by hand) in JPL
Tech. Memo. 33-669, 1974

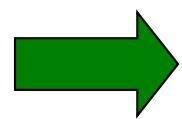
radius of gyration factors k_{ijj}^2 are being used here

for a body of mass m_i and moment of inertia I_j with respect to an axis z_j , the radius of gyration k_{ijj} is the distance of the mass m_i from the same axis, such that $I_j = m_i k_{ijj}^2$



Expression of v_{ci} and ω_i

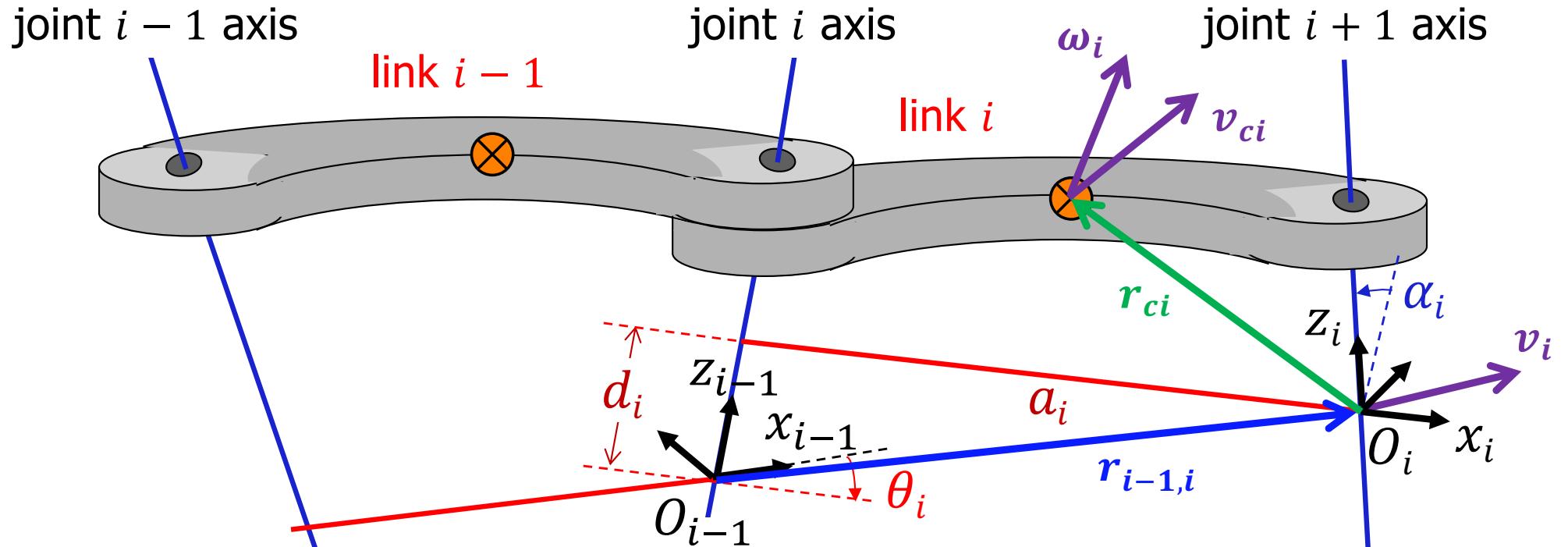
- v_{ci} and ω_i can be written using the relations of the robot differential kinematics (partial Jacobians)
- it is useful however to operate in a **recursive** way, expressing each vector quantity related to link i in the “**moving**” **frame** RF_i attached to link i (with the notation **i vector _{i}**)
 - particularly convenient when using algebraic/symbolic manipulation languages (Matlab Symbolic Toolbox, Maple, Mathematica, ...) for computing the kinetic energy of a (open chain) robot arm, when the number of joints increases (e.g., for $N \geq 4$)



Moving Frames



Recall: D-H frames



$${}^{i-1}A_i(q_i) = \begin{pmatrix} \cos \theta_i & -\cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cos \theta_i & -\sin \alpha_i \cos \theta_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Moving Frames algorithm

$$\text{velocity of center of mass of link } i \rightarrow v_{ci} = v_i + \omega_i \times r_{ci} \leftarrow \text{position of center of mass of link } i \text{ w.r.t. } O_i$$

↑ velocity of O_i
(origin of RF_i) ↑ angular velocity of link i

set $\sigma_i = \begin{cases} 0 & \text{revolute joint} \\ 1 & \text{prismatic joint} \end{cases}$

$${}^i\omega_i = {}^{i-1}R_i^T(q_i) [{}^{i-1}\omega_{i-1} + (1 - \sigma_i)\dot{q}_i {}^{i-1}z_{i-1}] = {}^{i-1}R_i^T(q_i) {}^{i-1}\omega_i$$

z-axis of RF_{i-1}

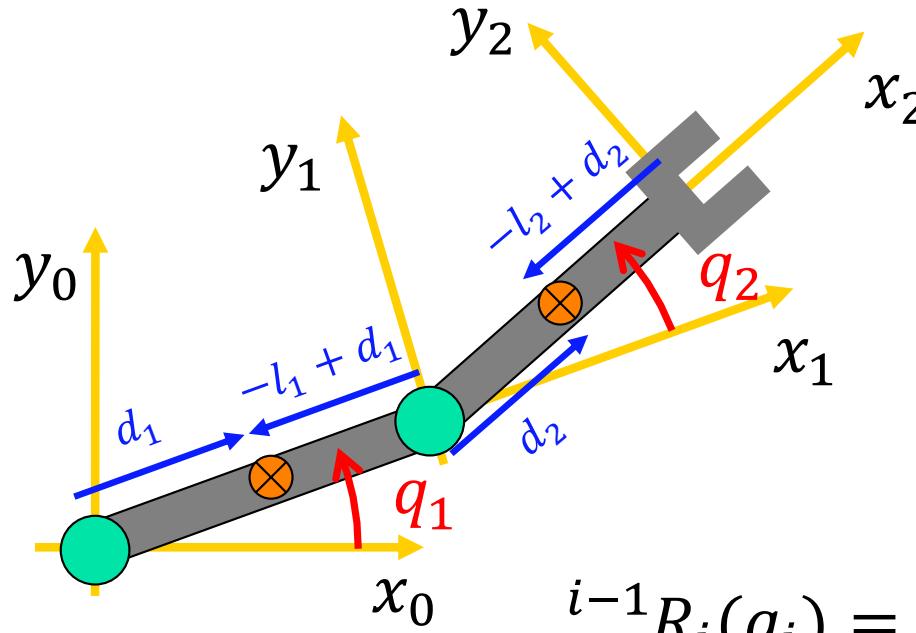
$${}^i v_i = {}^{i-1}R_i^T(q_i) [{}^{i-1}v_{i-1} + \sigma_i \dot{q}_i {}^{i-1}z_{i-1} + {}^{i-1}\omega_i \times {}^{i-1}r_{i-1,i}]$$

... = ${}^i\omega_i$ already computed *... = ${}^i r_{i-1,i}$ (constant, if joint i is revolute!)*



Dynamic model of a 2R robot

application of the algorithm



$$g = \begin{pmatrix} 0 \\ -g_0 \\ 0 \end{pmatrix}$$

$$g_0 = 9.81$$

$${}^{i-1}R_i(q_i) = \begin{pmatrix} c_i & -s_i & 0 \\ s_i & c_i & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$${}^i r_{ci} = \begin{pmatrix} -l_i + d_i \\ 0 \\ 0 \end{pmatrix}$$

assumption: center of mass of each link is on its kinematic axis

initialization: $i = 0$

$${}^0\omega_0 = 0$$

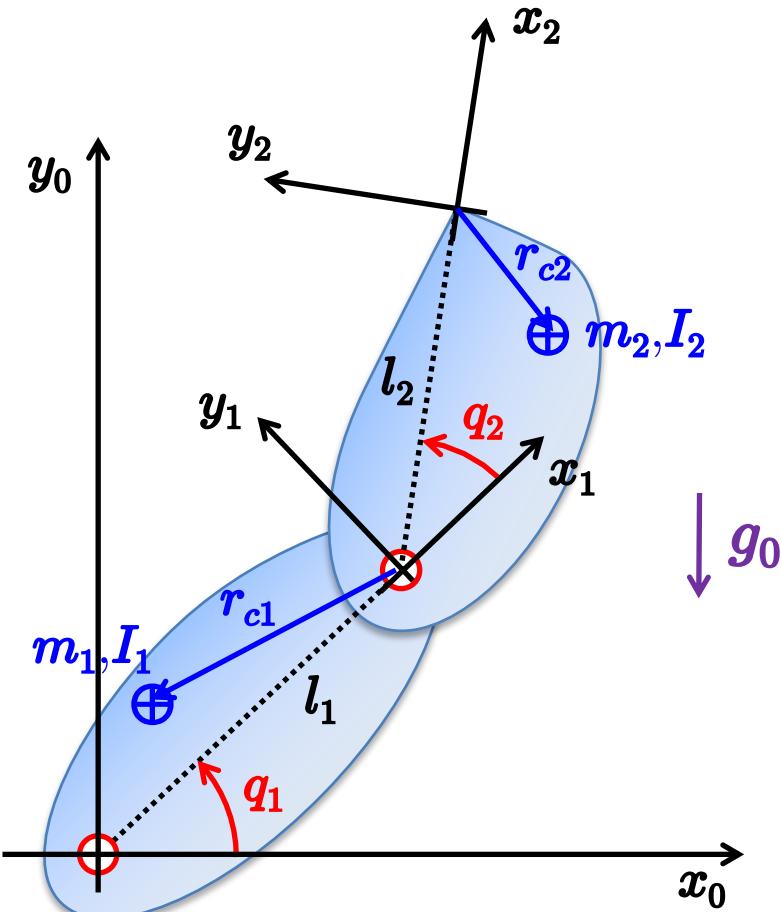
$${}^0v_0 = 0$$



Dynamic model of a 2R robot

what if the CoM is not on the kinematic axis ...

see **Robotics 2 Midterm 2021** (14 April)



$${}^{i-1}R_i(q_i) = \begin{pmatrix} c_i & -s_i & 0 \\ s_i & c_i & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$${}^i r_{ci} = \begin{pmatrix} r_{ci,x} \\ r_{ci,y} \\ 0 \end{pmatrix}$$

may also work
in 2D (planar case)
 ${}^{i-1}\bar{R}_i(q_i)$, ${}^i\bar{r}_{ci}$

$$\begin{aligned} T_1 &= \frac{1}{2} m_1 \|v_{c1}\|^2 + \frac{1}{2} \omega_1^T \mathbf{I}_1 \omega_1 \\ &= \frac{1}{2} m_1 \left((l_1 + r_{c1,x})^2 + r_{c1,y}^2 \right) \dot{q}_1^2 + \frac{1}{2} I_1 \dot{q}_1^2 \end{aligned}$$

$${}^0 p_{c2} = \begin{pmatrix} l_1 c_1 \\ l_1 s_1 \end{pmatrix} + {}^0 \bar{R}_2(q_1, q_2) \begin{pmatrix} l_2 + r_{c2,x} \\ r_{c2,y} \end{pmatrix}$$

$${}^0 \bar{R}_2(q_1, q_2) = \begin{pmatrix} c_{12} & -s_{12} \\ s_{12} & c_{12} \end{pmatrix} \quad \text{etc ...}$$



First step (link 1)

$i = 1$

$${}^1\omega_1 = {}^0R_1^T(q_1) \left[{}^0\omega_0 + \dot{q}_1 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \end{pmatrix}$$

$${}^1v_1 = {}^0R_1^T(q_1) \left[{}^0v_0 + \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \end{pmatrix} \times \begin{pmatrix} l_1 c_1 \\ l_1 s_1 \\ 0 \end{pmatrix} \right] = {}^0R_1^T(q_1) \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \end{pmatrix} \times {}^0R_1^T(q_1) \begin{pmatrix} l_1 c_1 \\ l_1 s_1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \end{pmatrix} \times \begin{pmatrix} l_1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ l_1 \dot{q}_1 \\ 0 \end{pmatrix}$$

there is not dependence on q_1

$${}^1v_{c1} = \begin{pmatrix} 0 \\ l_1 \dot{q}_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \end{pmatrix} \times \begin{pmatrix} -l_1 + d_1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ d_1 \dot{q}_1 \\ 0 \end{pmatrix}$$



Kinetic energy of link 1

$${}^1\omega_1 = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \end{pmatrix}$$

$${}^1v_{c1} = \begin{pmatrix} 0 \\ d_1 \dot{q}_1 \\ 0 \end{pmatrix}$$



$$T_1 = \frac{1}{2} m_1 d_1^2 \dot{q}_1^2 + \frac{1}{2} I_{c1,zz} \dot{q}_1^2 = \frac{1}{2} (I_{c1,zz} + m_1 d_1^2) \dot{q}_1^2$$

the actual inertia around the rotation axis
of the first joint (parallel axis theorem)



Second step (link 2)

$$i = 2$$

$${}^2\omega_2 = {}^1R_2^T(q_2) \left[{}^1\omega_1 + \dot{q}_2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{pmatrix}$$

$${}^2v_2 = {}^1R_2^T(q_2) \left[{}^1v_1 + \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{pmatrix} \times \begin{pmatrix} l_2 c_2 \\ l_2 s_2 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} l_1 s_2 \dot{q}_1 \\ l_1 c_2 \dot{q}_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{pmatrix} \times \begin{pmatrix} l_2 \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} l_1 s_2 \dot{q}_1 \\ l_1 c_2 \dot{q}_1 + l_2(\dot{q}_1 + \dot{q}_2) \\ 0 \end{pmatrix}$$



Kinetic energy of link 2

$$i = 2$$

$${}^2v_{c2} = {}^2v_2 + \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{pmatrix} \times \begin{pmatrix} -l_2 + d_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} l_1 s_2 \dot{q}_1 \\ l_1 c_2 \dot{q}_1 + d_2(\dot{q}_1 + \dot{q}_2) \\ 0 \end{pmatrix}$$



$$T_2 = \frac{1}{2} m_2 \left(l_1^2 \dot{q}_1^2 + d_2^2 (\dot{q}_1 + \dot{q}_2)^2 + 2l_1 d_2 c_2 \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \right)$$

$$+ \frac{1}{2} I_{c2,zz} (\dot{q}_1 + \dot{q}_2)^2$$



Robot inertia matrix $M(q)$

$$T = T_1 + T_2 = \frac{1}{2} (\dot{q}_1 \quad \dot{q}_2)^T \begin{pmatrix} m_{11}(q) & m_{12}(q) \\ m_{21}(q) & m_{22} \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix}$$

$$\begin{aligned} m_{11}(q) &= I_{c1,zz} + m_1 d_1^2 + I_{c2,zz} + m_2 d_2^2 + m_2 l_1^2 + 2m_2 l_1 d_2 c_2 \\ &= a_1 + 2a_2 \cos q_2 \end{aligned}$$

$$m_{12}(q) = m_{21}(q) = I_{c2,zz} + m_2 d_2^2 + m_2 l_1 d_2 c_2 = a_3 + a_2 \cos q_2$$

$$m_{22} = I_{c2,zz} + m_2 d_2^2 = a_3$$

NOTE: introduction of **dynamic coefficients** a_i is a convenient **regrouping** of the dynamic parameters (more on this later → [linear parametrization of dynamics](#))



Centrifugal and Coriolis terms

$$C_1(q) = \frac{1}{2} \left(\frac{\partial M_1}{\partial q} + \left(\frac{\partial M_1}{\partial q} \right)^T - \frac{\partial M}{\partial q_1} \right) = \frac{1}{2} \left(\begin{pmatrix} 0 & -2a_2 s_2 \\ 0 & -a_2 s_2 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -2a_2 s_2 & -a_2 s_2 \end{pmatrix} \right)$$
$$= \begin{pmatrix} 0 & -a_2 s_2 \\ -a_2 s_2 & -a_2 s_2 \end{pmatrix} \quad \rightarrow \quad c_1(q, \dot{q}) = -a_2 s_2 (\dot{q}_2^2 + 2\dot{q}_1 \dot{q}_2)$$

$$C_2(q) = \frac{1}{2} \left(\frac{\partial M_2}{\partial q} + \left(\frac{\partial M_2}{\partial q} \right)^T - \frac{\partial M}{\partial q_2} \right) = \dots = \begin{pmatrix} a_2 s_2 & 0 \\ 0 & 0 \end{pmatrix}$$



$$c_2(q, \dot{q}) = a_2 s_2 \dot{q}_1^2$$



Gravity terms

$$U_1 = -m_1 g^T r_{0,c1} = -m_1 \begin{pmatrix} 0 & -g_0 & 0 \end{pmatrix} \begin{pmatrix} * \\ d_1 s_1 \\ * \end{pmatrix} = m_1 g_0 d_1 s_1$$

$$U_2 = -m_2 g^T r_{0,c2} = m_2 g_0 (l_1 s_1 + d_2 s_{12})$$

$$U = U_1 + U_2$$

$$g(q) = \left(\frac{\partial U}{\partial q} \right)^T = \begin{pmatrix} g_0(m_1 d_1 c_1 + m_2 l_1 c_1 + m_2 d_2 c_{12}) \\ g_0 m_2 d_2 c_{12} \end{pmatrix} = \begin{pmatrix} a_4 c_1 + a_5 c_{12} \\ a_5 c_{12} \end{pmatrix}$$

There are only 5 dynamics coefficients that matters: a1 to a5



Dynamic model of a 2R robot

$$(a_1 + 2a_2c_2)\ddot{q}_1 + (a_2c_2 + a_3)\ddot{q}_2 - a_2s_2(\dot{q}_2^2 + 2\dot{q}_1\dot{q}_2) + a_4c_1 + a_5c_{12} = u_1$$

$$(a_2c_2 + a_3)\ddot{q}_1 + a_3\ddot{q}_2 + a_2s_2\dot{q}_1^2 + a_5c_{12} = u_2$$

$a_2 = m_2^2 I_1^2 d_2 \rightarrow d_2$ is the only one that can go to 0: the second link has the CoM on the joint axis 2
 if $a_2 = 0$ it means that the inertia matrix is constant \rightarrow no coriolis centrifugal term.

moreover if $d_2 = 0 \rightarrow a_5 = 0$ so also the gravity will have no effect since the CoM is on the joint axis

Q1: is it $a_2 = 0$ possible? ...physical interpretation? ...consequences?

$a_5 = 0$ if second link balanced ($d_2 = 0$). $a_4 = 0$ when $d_1 = -I_1$, so the CoM balances its own mass and the mass of the link that is beyond (GUARDA FINE VIDEO LECTURE 09)

Q2: is it $a_4 = a_5 = 0$ possible as well? ...physical interpretation?

Q3: based on the expressions of the dynamic coefficients a_1, a_2, a_3 , check that the robot inertia matrix is **always** positive definite, and in particular that the diagonal elements are **always** positive ($\forall q$)

Q4: provide two different matrices S' and S'' for the factorization of the quadratic velocity terms, respectively **satisfying** and **not satisfying** the skew-symmetry of $\dot{M} - 2S$



Robotics 2

Dynamic model of robots: Analysis, properties, extensions, uses

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

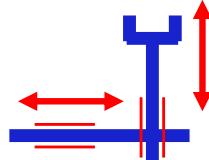


SAPIENZA
UNIVERSITÀ DI ROMA



Analysis of inertial couplings

- Cartesian robot

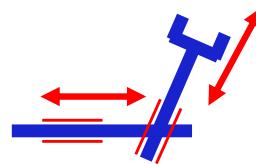


m_{11} and m_{22} are constant

$$M = \begin{pmatrix} m_{11} & 0 \\ 0 & m_{22} \end{pmatrix}$$

the fact that the matrix is diagonal shows no inertia coupling between the two joints: if we give a force to the first joint only the first joint accelerates and the same is valid for the second joint.

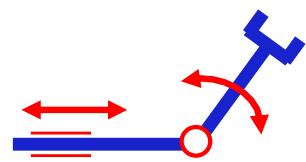
- Cartesian “skew” robot



$$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{pmatrix}$$

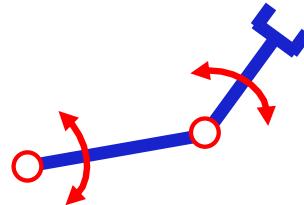
there is a diagonal coupling. However the matrix is still constant, so no coriolis effect

- PR robot



$$M = \begin{pmatrix} m_{11} & m_{12}(q_2) \\ m_{12}(q_2) & m_{22} \end{pmatrix}$$

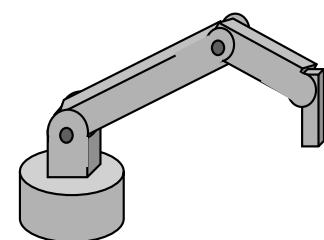
- 2R robot



$$M = \begin{pmatrix} m_{11}(q_2) & m_{12}(q_2) \\ m_{12}(q_2) & m_{22} \end{pmatrix}$$

- 3R articulated robot

(under simplifying assumptions on the CoMs)



$$M = \begin{pmatrix} m_{11}(q_2, q_3) & 0 & 0 \\ 0 & m_{22}(q_3) & m_{23}(q_3) \\ 0 & m_{23}(q_3) & m_{33} \end{pmatrix}$$

if you apply force on the first joint only the first joint will accelerate
if you apply force on the second and third joint only the second and third joint will accelerate and the first not



Analysis of gravity term

- absence of gravity
 - constant U_g (motion on horizontal plane)
 - applications in remote space
 - static balancing
 - distribution of masses (including motors)
 - mechanical compensation
 - articulated system of springs
 - closed kinematic chains
- $\left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \quad \rightarrow g(q) \approx 0$





Bounds on dynamic terms

- for an open-chain (serial) manipulator, there always exist positive real constants k_0 to k_7 such that, for **any** value of q and \dot{q}

$$k_0 \leq \|M(q)\| \leq k_1 + k_2\|q\| + k_3\|q\|^2 \quad \text{inertia matrix}$$

$$\|S(q, \dot{q})\| \leq (k_4 + k_5\|q\|) \|\dot{q}\| \quad \text{factorization matrix of Coriolis/centrifugal terms}$$

$$\|g(q)\| \leq k_6 + k_7\|q\| \quad \text{gravity vector}$$

- if the robot has only **revolute** joints, these simplify to

$$k_0 \leq \|M(q)\| \leq k_1 \quad \|S(q, \dot{q})\| \leq k_4\|\dot{q}\| \quad \|g(q)\| \leq k_6$$

(the same holds true with bounds $q_{i,min} \leq q_i \leq q_{i,max}$ on prismatic joints)

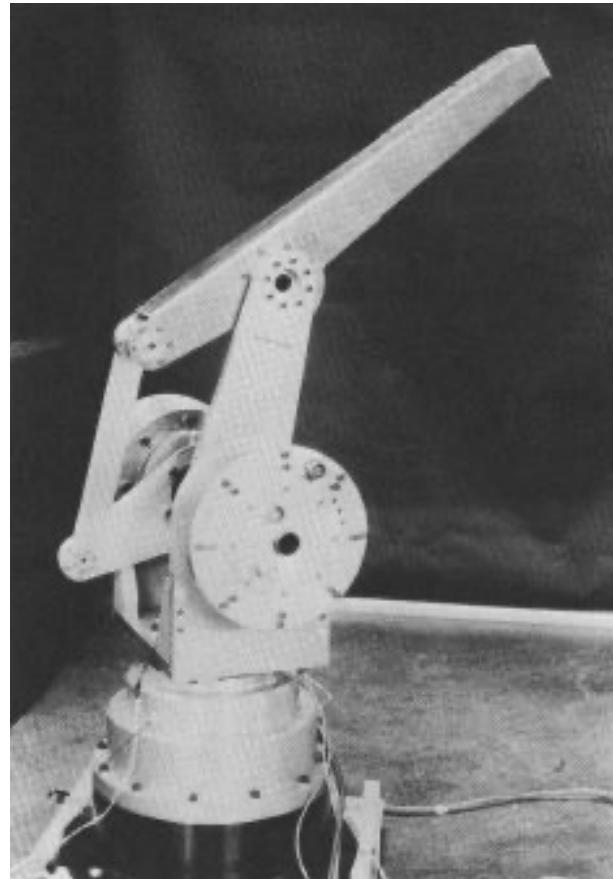
NOTE: norms are either for vectors or for matrices (induced norms)



Robots with closed kinematic chains - 1



Comau Smart NJ130

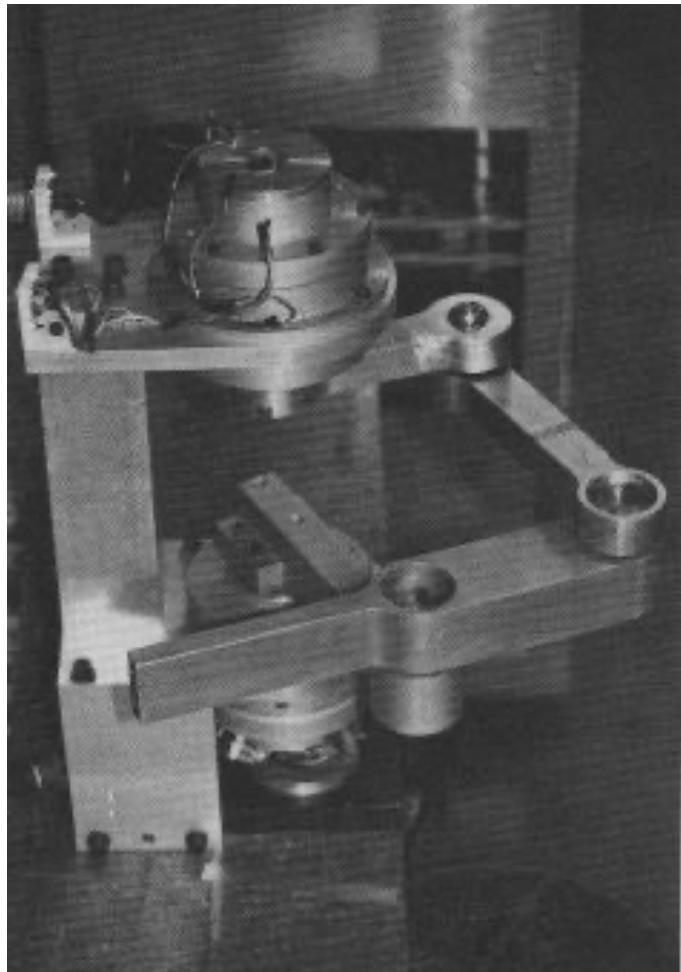


MIT Direct Drive Mark II and Mark III

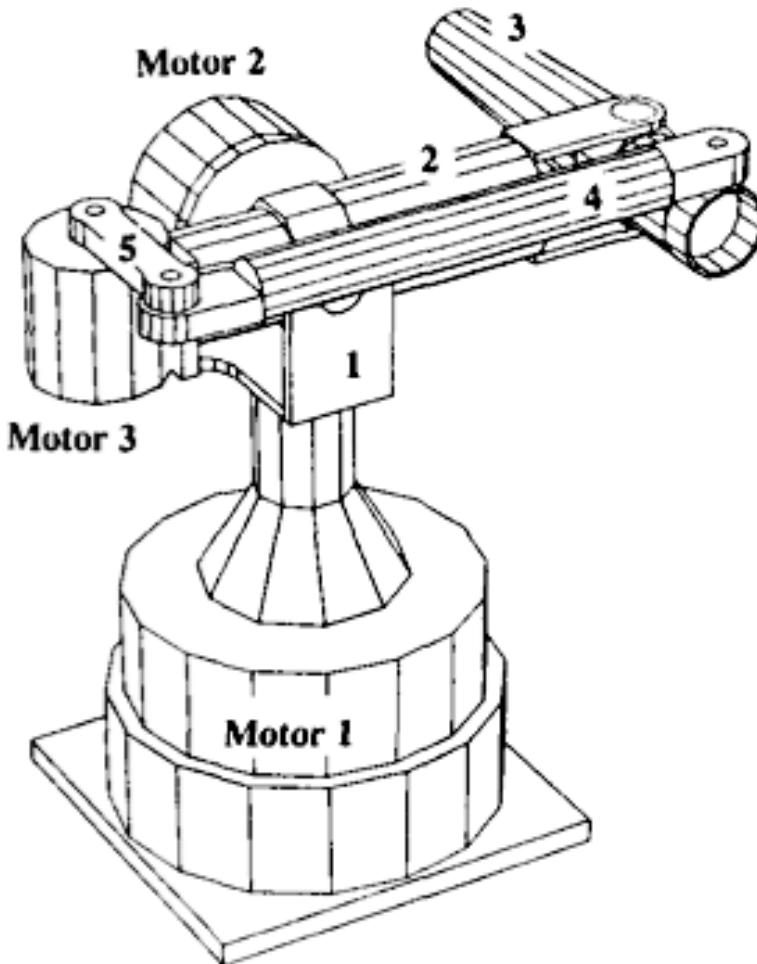




Robots with closed kinematic chains - 2



MIT Direct Drive Mark IV
(**planar** five-bar linkage)

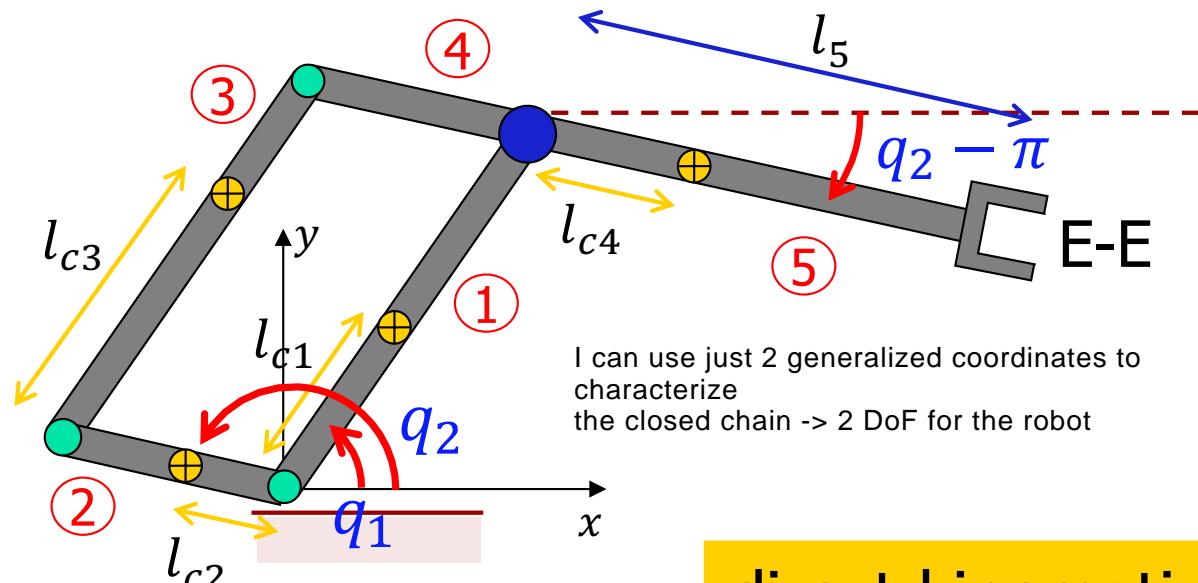


UMinneapolis Direct Drive Arm
(**spatial** five-bar linkage)



Robot with parallelogram structure

(planar) kinematics and dynamics



center of mass:
arbitrary l_{ci}

parallelogram:

$$l_1 = l_3$$

$$l_2 = l_4$$

direct kinematics

$$p_{EE} = \begin{pmatrix} l_1 c_1 \\ l_1 s_1 \end{pmatrix} + \begin{pmatrix} l_5 \cos(q_2 - \pi) \\ l_5 \sin(q_2 - \pi) \end{pmatrix} = \begin{pmatrix} l_1 c_1 \\ l_1 s_1 \end{pmatrix} - \begin{pmatrix} l_5 c_2 \\ l_5 s_2 \end{pmatrix}$$

position of center of masses

$$p_{c1} = \begin{pmatrix} l_{c1} c_1 \\ l_{c1} s_1 \end{pmatrix} \quad p_{c2} = \begin{pmatrix} l_{c2} c_2 \\ l_{c2} s_2 \end{pmatrix} \quad p_{c3} = \begin{pmatrix} l_2 c_2 \\ l_2 s_2 \end{pmatrix} + \begin{pmatrix} l_{c3} c_1 \\ l_{c3} s_1 \end{pmatrix} \quad p_{c4} = \begin{pmatrix} l_1 c_1 \\ l_1 s_1 \end{pmatrix} - \begin{pmatrix} l_{c4} c_2 \\ l_{c4} s_2 \end{pmatrix}$$



Kinetic energy

linear/angular velocities

$$v_{c1} = \begin{pmatrix} -l_{c1}s_1 \\ l_{c1}c_1 \end{pmatrix} \dot{q}_1 \quad v_{c3} = \begin{pmatrix} -l_{c3}s_1 \\ l_{c3}c_1 \end{pmatrix} \dot{q}_1 + \begin{pmatrix} -l_2s_2 \\ l_2c_2 \end{pmatrix} \dot{q}_2 \quad \omega_1 = \omega_3 = \dot{q}_1$$

$$v_{c2} = \begin{pmatrix} -l_{c2}s_2 \\ l_{c2}c_2 \end{pmatrix} \dot{q}_2 \quad v_{c4} = \begin{pmatrix} -l_1s_1 \\ l_1c_1 \end{pmatrix} \dot{q}_1 + \begin{pmatrix} l_{c4}s_2 \\ -l_{c4}c_2 \end{pmatrix} \dot{q}_2 \quad \omega_2 = \omega_4 = \dot{q}_2$$

Note: a (planar) 2D notation is used here!

T_i

$$T_1 = \frac{1}{2}m_1 l_{c1}^2 \dot{q}_1^2 + \frac{1}{2}I_{c1,zz} \dot{q}_1^2 \quad T_2 = \frac{1}{2}m_2 l_{c2}^2 \dot{q}_2^2 + \frac{1}{2}I_{c2,zz} \dot{q}_2^2$$

$$T_3 = \frac{1}{2}m_3(l_2^2 \dot{q}_2^2 + l_{c3}^2 \dot{q}_1^2 + 2l_2 l_{c3} c_{2-1} \dot{q}_1 \dot{q}_2) + \frac{1}{2}I_{c3,zz} \dot{q}_1^2$$

$$T_4 = \frac{1}{2}m_4(l_1^2 \dot{q}_1^2 + l_{c4}^2 \dot{q}_2^2 - 2l_1 l_{c4} c_{2-1} \dot{q}_1 \dot{q}_2) + \frac{1}{2}I_{c4,zz} \dot{q}_2^2$$



Robot inertia matrix

$$T = \sum_{i=1}^4 T_i = \frac{1}{2} \dot{q}^T M(q) \dot{q}$$

$$M(q) = \begin{pmatrix} I_{c1,zz} + m_1 l_{c1}^2 + I_{c3,zz} + m_3 l_{c3}^2 + m_4 l_1^2 & \text{symm} \\ (m_3 l_2 l_{c3} - m_4 l_1 l_{c4}) c_{2-1} & I_{c2,zz} + m_2 l_{c2}^2 + I_{c4,zz} + m_4 l_{c4}^2 + m_3 l_2^2 \end{pmatrix}$$

structural condition
in mechanical design

$$m_3 l_2 l_{c3} = m_4 l_1 l_{c4} \quad (*)$$

this condition doesn't occurs in serial manipulator. If you set this condition the inertia matrix becomes diagonal and constant.



$M(q)$ diagonal and **constant** \Rightarrow centrifugal and Coriolis terms $\equiv 0$

mechanically **DECOUPLED** and **LINEAR**
dynamic model (up to the gravity term $g(q)$)



$$\begin{pmatrix} M_{11} & 0 \\ 0 & M_{22} \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

big advantage for the design of motion control laws!



Potential energy and gravity terms

from the y -components of vectors p_{ci}

U_i

$$U_1 = m_1 g_0 l_{c1} s_1$$

$$U_2 = m_2 g_0 l_{c2} s_2$$

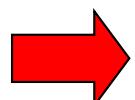
$$U_3 = m_3 g_0 (l_2 s_2 + l_{c3} s_1) \quad U_4 = m_4 g_0 (l_1 s_1 - l_{c4} s_2)$$

$$U = \sum_{i=1}^4 U_i$$

$$g(q) = \left(\frac{\partial U}{\partial q} \right)^T = \begin{pmatrix} g_0(m_1 l_{c1} + m_3 l_{c3} + m_4 l_1) c_1 \\ g_0(m_2 l_{c2} + m_3 l_2 - m_4 l_{c4}) c_2 \end{pmatrix} = \begin{pmatrix} g_1(q_1) \\ g_2(q_2) \end{pmatrix}$$

gravity components are **always** "decoupled"

in addition,
when (*) holds



$$\begin{aligned} m_{11} \ddot{q}_1 + g_1(q_1) &= u_1 \\ m_{22} \ddot{q}_2 + g_2(q_2) &= u_2 \end{aligned}$$

u_i are
(non-conservative) torques
performing work on q_i

further structural conditions in the mechanical design lead to $g(q) \equiv 0!!$



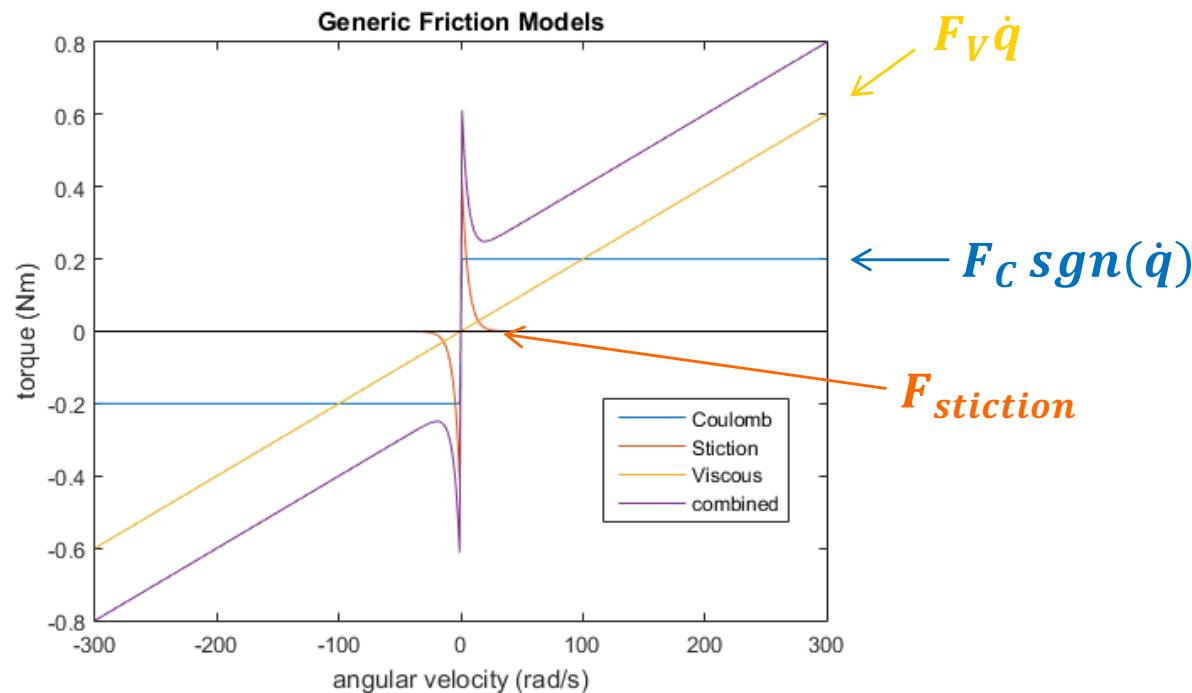
Adding dynamic terms ...

- 1) dissipative phenomena due to friction at the joints/transmissions
 - viscous, Coulomb, stiction, Stribeck, LuGre (dynamic)...
 - local effects at the joints
 - difficult to model in general, except for:

$$u_{V,i} = -F_{V,i} \dot{q}_i$$

$$u_{C,i} = -F_{C,i} \operatorname{sgn}(\dot{q}_i)$$

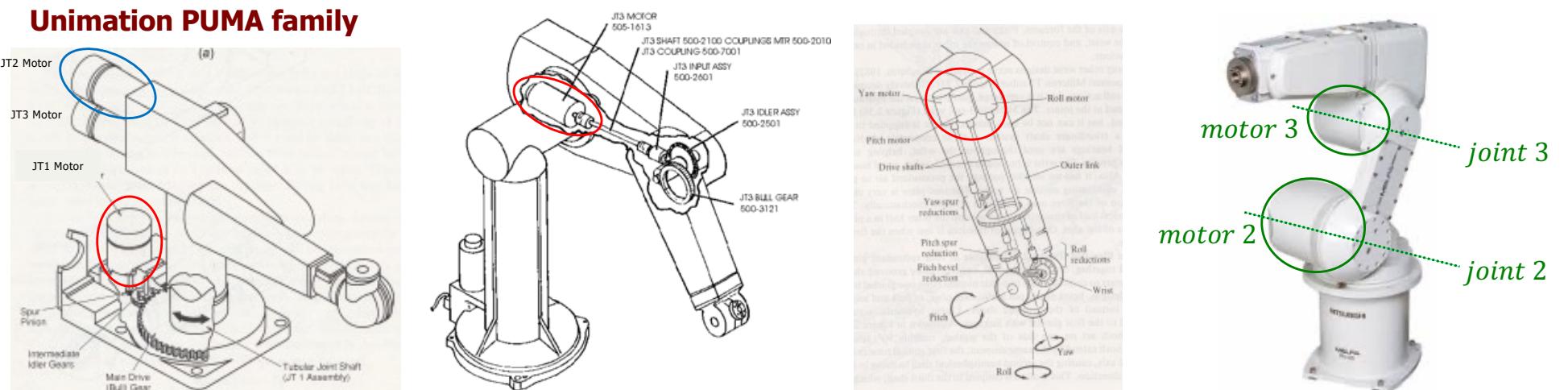
in general:
 $u_{diss}^T \dot{q} < 0$
 (component-wise too)





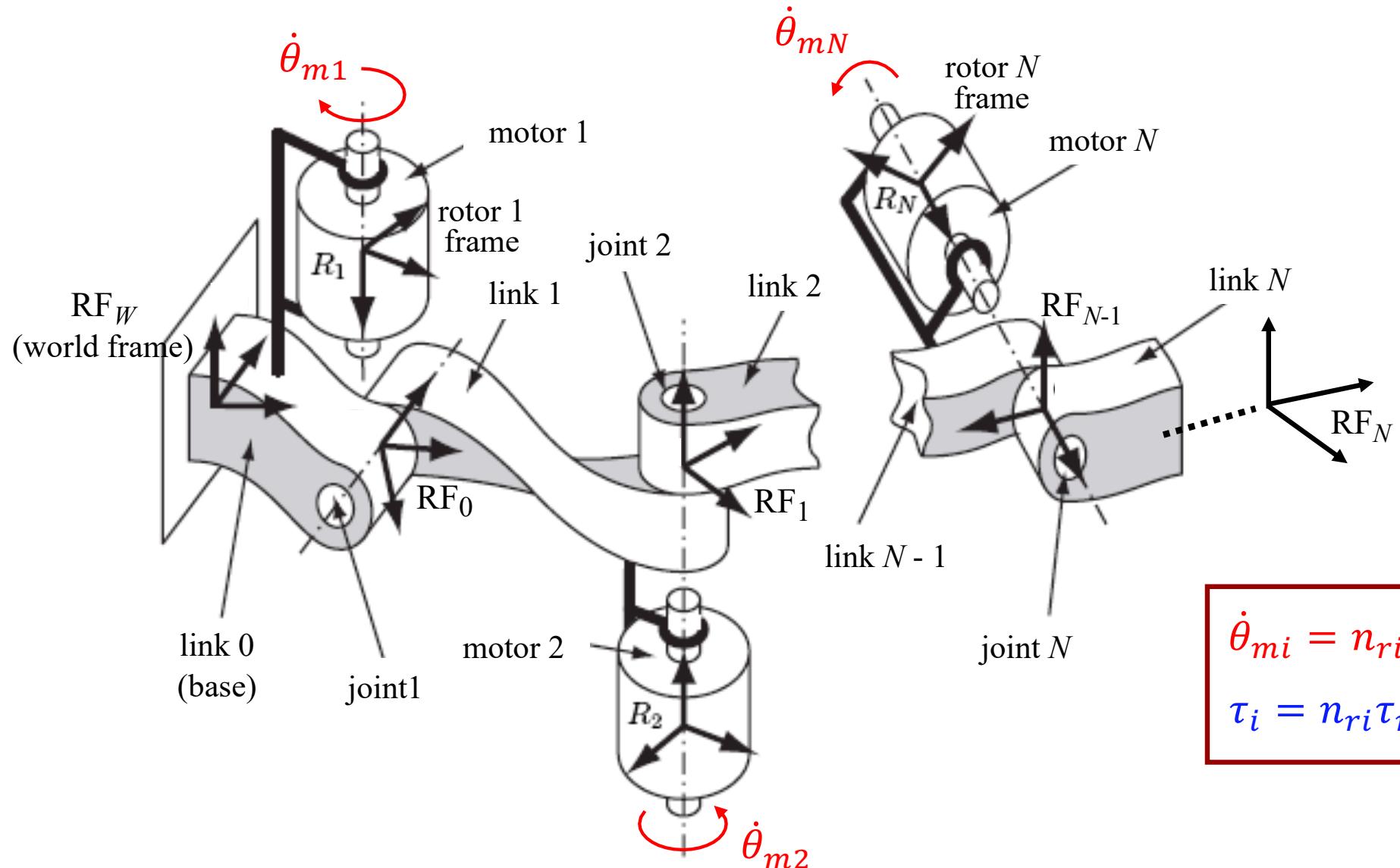
Adding dynamic terms ...

- 2) inclusion of electrical **actuators** (as additional rigid bodies)
- motor i mounted on link $i - 1$ (or **before**), with very few **exceptions**
 - often with its spinning **axis aligned with joint axis i**
 - (balanced) **mass** of motor included in total mass of carrying link
 - (rotor) **inertia** is to be **added** to robot kinetic energy
 - transmissions with **reduction gears** (often, large reduction ratios)
 - in some cases, multiple motors cooperate in moving multiple links: use a **transmission coupling** matrix Γ (with off-diagonal elements)





Placement of motors along the chain



$$\dot{\theta}_{mi} = n_{ri} \dot{\theta}_i$$

$$\tau_i = n_{ri} \tau_{mi}$$



Resulting dynamic model

- simplifying assumption: in the **rotational** part of the kinetic energy, only the “spinning” rotor velocity is considered

$$T_{mi} = \frac{1}{2} I_{mi} \dot{\theta}_{mi}^2 = \frac{1}{2} I_{mi} n_{ri}^2 \dot{q}_i^2 = \frac{1}{2} B_{mi} \dot{q}_i^2 \quad T_m = \sum_{i=1}^N T_{mi} = \frac{1}{2} \dot{q}^T B_m \dot{q}$$

the fact that the rotor, mounted on joint i , has an angular velocity because the previous link is rotating is neglected
diagonal, > 0

- including all added terms, the robot dynamics becomes

$$(M(q) + B_m)\ddot{q} + c(q, \dot{q}) + g(q) + \underbrace{F_V \dot{q} + F_C \operatorname{sgn}(\dot{q})}_{\substack{F_V > 0, F_C > 0 \\ \text{diagonal}}} = \tau$$

moved to the left ...
motor torques (after reduction gears)

↓ constant → does NOT contribute to c

- scaling by the reduction gears, looking from the motor side

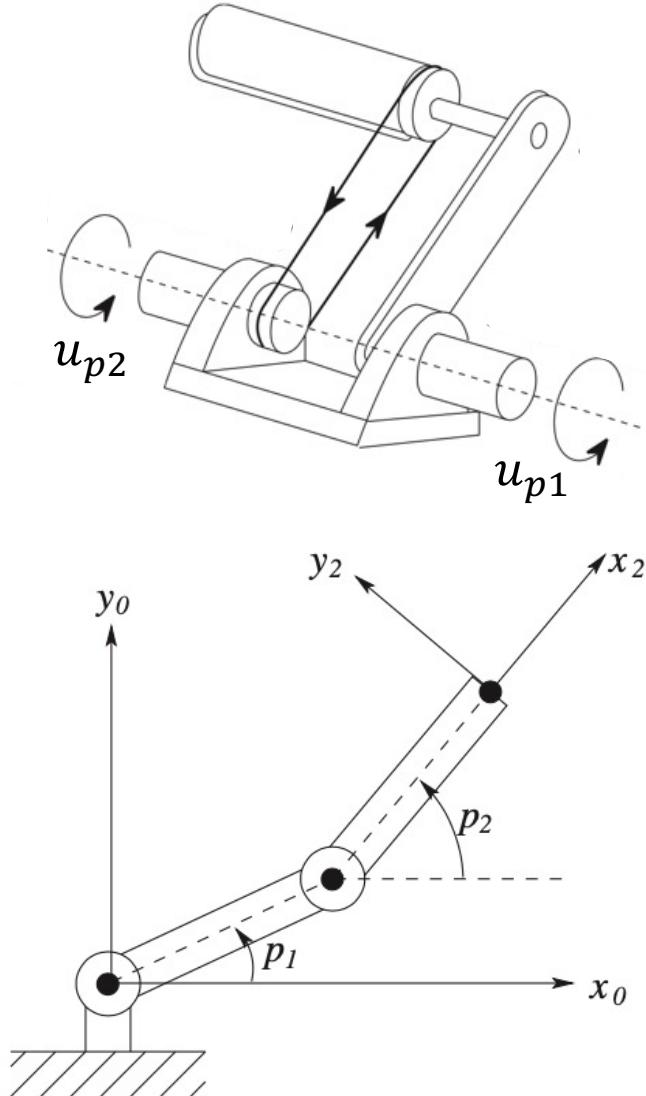
$$\left(I_m + \operatorname{diag} \left\{ \frac{m_{ii}(q)}{n_{ri}^2} \right\} \right) \ddot{\theta}_m + \operatorname{diag} \left\{ \frac{1}{n_{ri}} \right\} \left(\sum_{j=1}^N \bar{M}_j(q) \ddot{q}_j + f(q, \dot{q}) \right) = \tau_m$$

motor torques (before reduction gears)
except the terms m_{jj}



Special actuation and associated coordinates

planar 2R robot with remotely driven forearm



- motor 1 moves link 1 by p_1
- motor 2 **at the base** moves the **absolute** angle p_2 of link 2
- derive the dynamic model **from scratch** using the p coordinates

$$M(p)\ddot{p} + c(p, \dot{p}) + g(p) = u_p$$

$$M(p) = \begin{pmatrix} a_1 - a_3 & a_2 c_{2-1} \\ a_2 c_{2-1} & a_3 \end{pmatrix}$$

$$c(p, \dot{p}) = \begin{pmatrix} -a_2 s_{2-1} \dot{p}_2^2 \\ a_2 s_{2-1} \dot{p}_1^2 \end{pmatrix} \quad \text{no more Coriolis forces!}$$

$$g(p) = \begin{pmatrix} a_4 c_1 \\ a_5 c_2 \end{pmatrix}$$

$$c_1 = \cos p_1 \quad c_2 = \cos p_2$$

$$c_{2-1} = \cos(p_2 - p_1) \quad s_{2-1} = \sin(p_2 - p_1)$$



Including joint elasticity

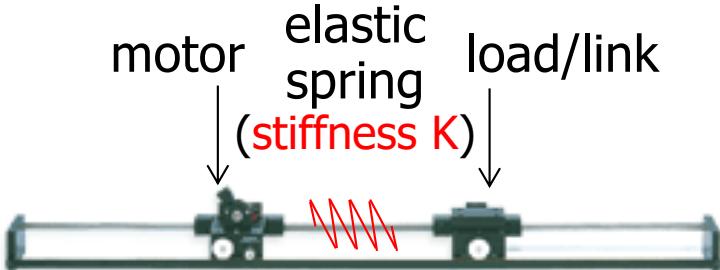
- in **industrial** robots, use of motion transmissions based on
 - belts
 - harmonic drives
 - long shaftsintroduces **flexibility** between actuating motors (input) and driven links (output)
- in **research** robots, **compliance** in transmissions is introduced on purpose for **safety** (human collaboration) and/or **energy efficiency**
 - actuator relocation by means of (compliant) cables and pulleys
 - harmonic drives and lightweight (but rigid) link design
 - redundant (macro-mini or parallel) actuation, with elastic couplings
- in both cases, flexibility is modeled as **concentrated at the joints**
- in most cases, assuming small joint deformation (**elastic domain**)



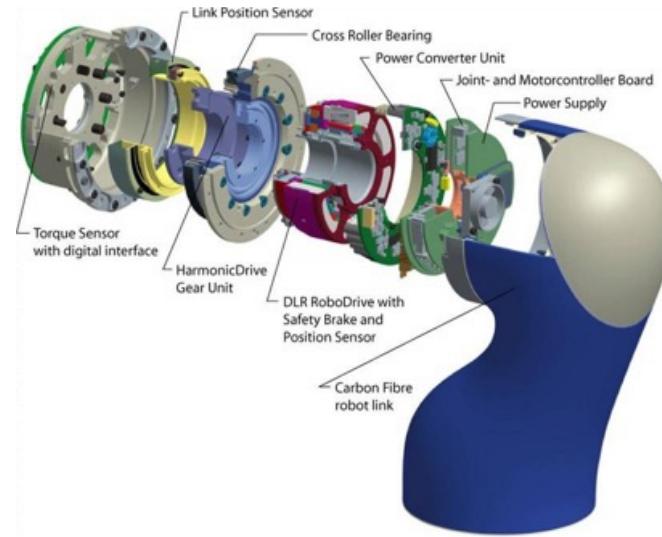
Robots with joint elasticity



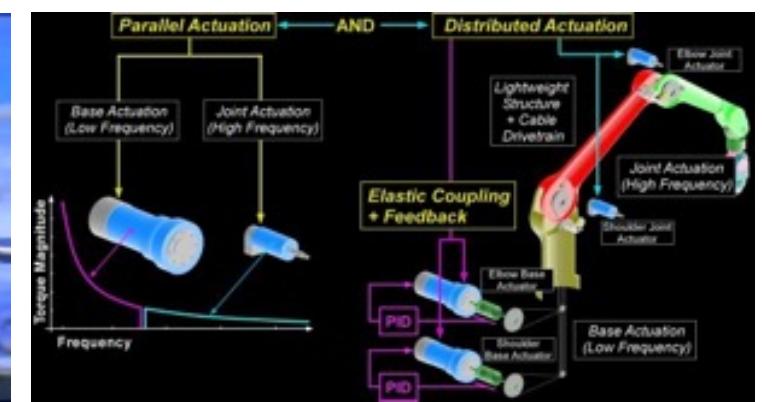
Dexter
with cable transmissions



Quanser Flexible Joint
(1-dof linear, educational)



DLR LWR-III
with harmonic drives



Stanford DECMMA
with micro-macro actuation



Dynamic model of robots with elastic joints

- introduce $2N$ generalized coordinates
 - $q = N$ link positions
 - $\theta = N$ motor positions (after reduction, $\theta_i = \theta_{mi}/n_{ri}$)

- add **motor kinetic energy** T_m to that of the links $T_q = \frac{1}{2} \dot{q}^T M(q) \dot{q}$

$$T_{mi} = \frac{1}{2} I_{mi} \dot{\theta}_{mi}^2 = \frac{1}{2} I_{mi} n_{ri}^2 \dot{\theta}_i^2 = \frac{1}{2} B_{mi} \dot{\theta}_i^2 \quad T_m = \sum_{i=1}^N T_{mi} = \frac{1}{2} \dot{\theta}^T B_m \dot{\theta}$$

diagonal, > 0

- add **elastic potential energy** U_e to that due to gravity $U_g(q)$

- K = matrix of **joint stiffness** (diagonal, > 0)

$$U_{ei} = \frac{1}{2} K_i \left(q_i - \left(\frac{\theta_{mi}}{n_{ri}} \right) \right)^2 = \frac{1}{2} K_i (q_i - \theta_i)^2 \quad U_e = \sum_{i=1}^N U_{ei} = \frac{1}{2} (q - \theta)^T K (q - \theta)$$

- apply **Euler-Lagrange** equations w.r.t. (q, \dot{q})

$2N$ 2nd-order differential equations

$$\left\{ \begin{array}{l} M(q)\ddot{q} + c(q, \dot{q}) + g(q) + K(q - \theta) = 0 \\ B_m \ddot{\theta} + K(\theta - q) = \tau \end{array} \right. \begin{array}{l} \text{no external torques} \\ \text{performing work on } q \end{array}$$

if theta goes to q and K to infinity inf⁰ but somehow it disappear, and summing up the two equations we obtain the previous model, the one without elasticity



Use of the dynamic model

inverse dynamics

- given a **desired trajectory** $q_d(t)$
 - twice differentiable ($\exists \ddot{q}_d(t)$)
 - possibly obtained from a task/Cartesian trajectory $r_d(t)$, by (differential) kinematic inversion
- the **input torque** needed to execute this motion (in **free space**) is
$$\tau_d = (M(q_d) + B_m)\ddot{q}_d + c(q_d, \dot{q}_d) + g(q_d) + F_V\dot{q}_d + F_C \operatorname{sgn}(\dot{q}_d)$$
(in **contact**, with an external wrench) ... $- J_{ext}^T(q_d) \mathbf{F}_{ext,d}$

- useful also for control (e.g., nominal feedforward)
- however, this way of performing the algebraic computation ($\forall t$) is **not efficient** when using the Lagrangian modeling approach
 - symbolic terms grow much longer, quite rapidly for larger N N is DoF
 - in real time, numerical computation is based on **Newton-Euler method**



State equations direct dynamics

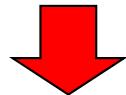
Lagrangian
dynamic model

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u$$

N differential
2nd order
equations

defining the vector of state variables as $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} q \\ \dot{q} \end{pmatrix} \in \mathbb{R}^{2N}$

state equations



$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -M^{-1}(x_1)[c(x_1, x_2) + g(x_1)] \end{pmatrix} + \begin{pmatrix} 0 \\ M^{-1}(x_1) \end{pmatrix} u$$

$$= f(x) + G(x)u$$

$\uparrow \quad \uparrow$
 $2N \times 1 \quad 2N \times N$

The input appears in a linear fashion: outside the non-linear function of x . Is very convenient.

$2N$ differential
1st order
equations

another choice...

$$\tilde{x} = \begin{pmatrix} q \\ M(q)\dot{q} \end{pmatrix}$$

generalized
momentum

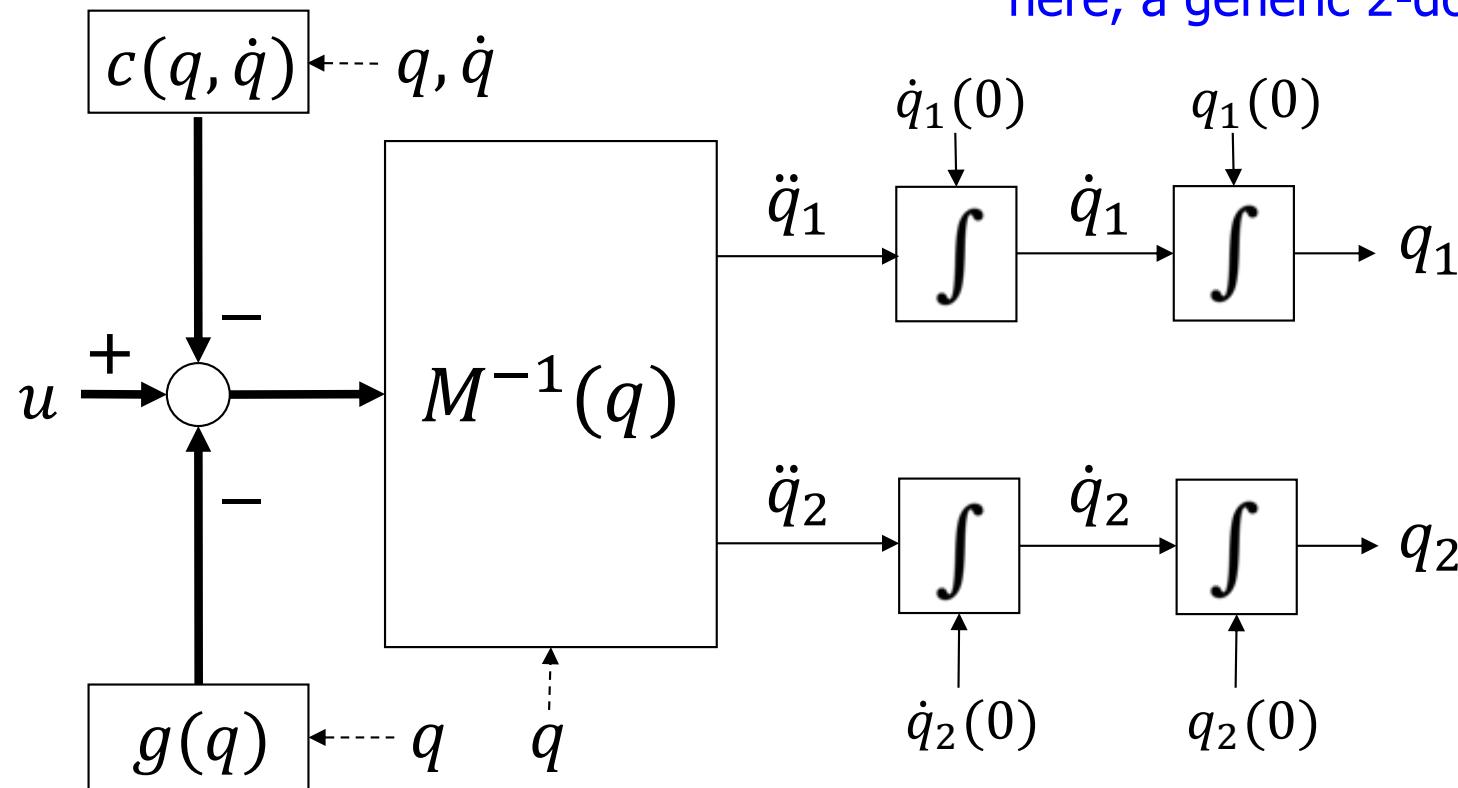
$\dot{\tilde{x}} = \dots$ (do it as exercise)



Dynamic simulation

Simulink
block
scheme

input torque
command
(open-loop
or in
feedback)



here, a generic 2-dof robot

including "inv(M)"

- initialization (dynamic coefficients and initial state)
- calls to (user-defined) Matlab functions for the evaluation of model terms
- choice of a numerical integration method (and of its parameters)

e.g., 4th-order Runge-Kutta (ode45)



Approximate linearization

- we can derive a **linear** dynamic model of the robot, which is valid **locally** around a given operative condition
 - useful for analysis, design, and **gain tuning** of linear (or of the linear part of) control laws
 - approximation by Taylor series expansion, up to the first order
 - linearization around a (constant) **equilibrium state** or along a (nominal, time-varying) **equilibrium trajectory**
 - usually, we work with (nonlinear) state equations; for mechanical systems, it is more convenient to directly use the **2nd order model**
 - same result, but easier derivation

velocity zero since you don't want to move away

equilibrium torque that balances gravity

$$\text{equilibrium state } (q, \dot{q}) = (q_e, 0) [\ddot{q} = 0] \quad \rightarrow \quad g(q_e) = u_e$$

$$\text{equilibrium trajectory } (q, \dot{q}) = (q_d(t), \dot{q}_d(t)) [\ddot{q} = \ddot{q}_d(t)]$$

$$\rightarrow M(q_d)\ddot{q}_d + c(q_d, \dot{q}_d) + g(q_d) = u_d$$



Linearization at an equilibrium state

- variations around an equilibrium state

$$q = q_e + \Delta q \quad \dot{q} = \cancel{\dot{q}_e} + \dot{\Delta q} = \dot{\Delta q} \quad \ddot{q} = \cancel{\ddot{q}_e} + \ddot{\Delta q} = \ddot{\Delta q} \quad u = u_e + \Delta u$$

- keeping into account the quadratic dependence of c terms on velocity (thus, neglected around the zero velocity)

$\Delta q * \Delta q = \Delta q^2$ is an infinitesimal of higher order than Δq so is neglectable. Since c terms have Δq^2 we neglect the c terms

$$M(q_e) \ddot{\Delta q} + g(q_e) + \underbrace{\frac{\partial g}{\partial q} \Big|_{q=q_e}}_{G(q_e)} \Delta q + o(\|\Delta q\|, \|\dot{\Delta q}\|) = u_e + \Delta u$$

infinitesimal terms
of second or higher order

- in state-space format, with $\Delta x = \begin{pmatrix} \Delta q \\ \dot{\Delta q} \end{pmatrix}$

$$\dot{\Delta x} = \begin{pmatrix} 0 & I \\ -M^{-1}(q_e)G(q_e) & 0 \end{pmatrix} \Delta x + \begin{pmatrix} 0 \\ M^{-1}(q_e) \end{pmatrix} \Delta u = A \Delta x + B \Delta u$$



Linearization along a trajectory

- variations around an equilibrium trajectory

$$q = q_d + \Delta q \quad \dot{q} = \dot{q}_d + \dot{\Delta q} \quad \ddot{q} = \ddot{q}_d + \ddot{\Delta q} \quad u = u_d + \Delta u$$

- developing to 1st order the terms in the dynamic model ...

$$M(q_d + \Delta q)(\ddot{q}_d + \ddot{\Delta q}) + c(q_d + \Delta q, \dot{q}_d + \dot{\Delta q}) + g(q_d + \Delta q) = u_d + \Delta u$$

$$M(q_d + \Delta q) \cong M(q_d) + \sum_{i=1}^N \frac{\partial M_i}{\partial q} \Big|_{q=q_d} e_i^T \Delta q \quad \text{↓ } i\text{-th row of the identity matrix}$$

$$g(q_d + \Delta q) \cong g(q_d) + G(q_d) \Delta q$$

$$c(q_d + \Delta q, \dot{q}_d + \dot{\Delta q}) \cong c(q_d, \dot{q}_d) + \underbrace{\frac{\partial c}{\partial q} \Big|_{\substack{q=q_d \\ \dot{q}=\dot{q}_d}}}_{C_1(q_d, \dot{q}_d)} \Delta q + \underbrace{\frac{\partial c}{\partial \dot{q}} \Big|_{\substack{q=q_d \\ \dot{q}=\dot{q}_d}}}_{C_2(q_d, \dot{q}_d)} \dot{\Delta q}$$



Linearization along a trajectory (cont)

- after simplifications ...

$$M(q_d)\ddot{\Delta q} + C_2(q_d, \dot{q}_d)\dot{\Delta q} + D(q_d, \dot{q}_d, \ddot{q}_d)\Delta q = \Delta u$$

with

$$D(q_d, \dot{q}_d, \ddot{q}_d) = G(q_d) + C_1(q_d, \dot{q}_d) + \sum_{i=1}^N \frac{\partial M_i}{\partial q} \Big|_{q=q_d} \ddot{q}_d e_i^T$$

- in state-space format

$$\begin{aligned}\dot{\Delta x} &= \begin{pmatrix} 0 & I \\ -M^{-1}(q_d)D(q_d, \dot{q}_d, \ddot{q}_d) & -M^{-1}(q_d)C_2(q_d, \dot{q}_d) \end{pmatrix} \Delta x \\ &\quad + \begin{pmatrix} 0 \\ M^{-1}(q_d) \end{pmatrix} \Delta u = A(t) \Delta x + B(t) \Delta u\end{aligned}$$

a linear, but **time-varying** system!!



Coordinate transformation

$$q \in \mathbb{R}^N$$

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = M(q)\ddot{q} + n(q, \dot{q}) = u_q \quad 1$$

if we wish/need to use a **new** set of generalized coordinates p

$$p \in \mathbb{R}^N$$

$$p = f(q)$$



$$q = f^{-1}(p)$$

by duality

(principle of virtual work)

$$\dot{p} = \frac{\partial f}{\partial q} \dot{q} = J(q)\dot{q}$$



$$\dot{q} = J^{-1}(q)\dot{p}$$

$$u_q = J^T(q)u_p$$

$$\ddot{p} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$$



$$\ddot{q} = J^{-1}(q)(\ddot{p} - \dot{J}(q)J^{-1}(q)\dot{p})$$

1

$$M(q)J^{-1}(q)\ddot{p} - M(q)J^{-1}(q)\dot{J}(q)J^{-1}(q)\dot{p} + n(q, \dot{q}) = J^T(q)u_p$$

$$J^{-T}(q) \cdot$$

pre-multiplying the whole equation...



Robot dynamic model after coordinate transformation

$$J^{-T}(q)M(q)J^{-1}(q)\ddot{p} + J^{-T}(q)(n(q, \dot{q}) - M(q)J^{-1}(q)\dot{J}(q)J^{-1}(q)\dot{p}) = u_p$$

$$q \rightarrow p$$

for actual computation,
these inner substitutions
are not strictly necessary

$$(q, \dot{q}) \rightarrow (p, \dot{p})$$

$$\rightarrow M_p(p)\ddot{p} + c_p(p, \dot{p}) + g_p(p) = u_p$$

non-conservative
generalized forces
performing work on p

$$M_p = J^{-T} M J^{-1}$$

symmetric,
positive definite
(out of singularities)

$$g_p = J^{-T} g$$

$$c_p = J^{-T}(c - M J^{-1} \dot{J} J^{-1} \dot{p}) = J^{-T} c - M_p \dot{J} J^{-1} \dot{p}$$

\dot{J} is a function of p and \dot{p} so
quadratic dependence of \dot{p}
**quadratic
dependence on \dot{p}**

$$c_p(p, \dot{p}) = S_p(p, \dot{p}) \dot{p} \quad \dot{M}_p - 2S_p \quad \text{skew-symmetric}$$

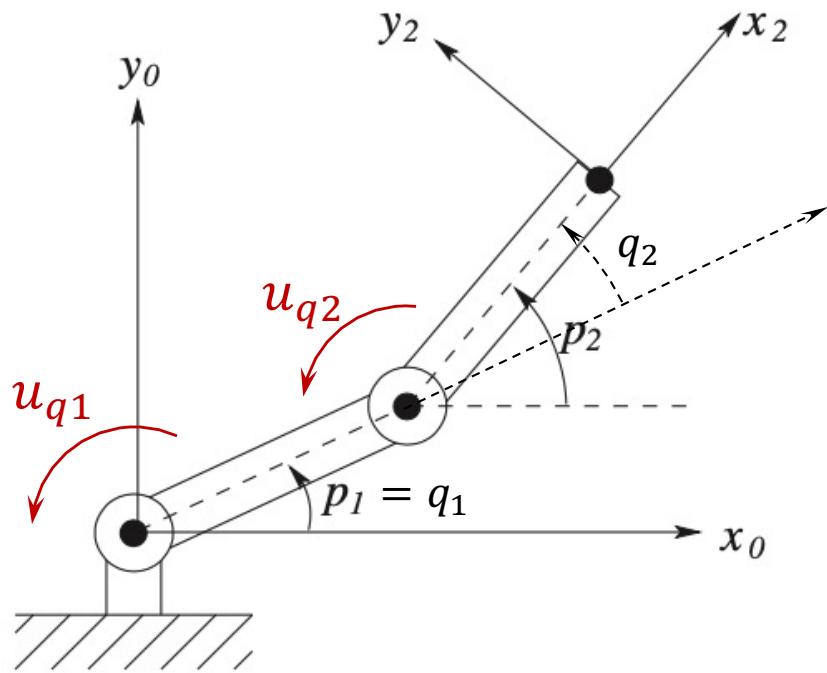
when $p = \text{E-E pose}$, this is the robot **dynamic model** in **Cartesian coordinates**

NOTE: in this case, we have implicitly assumed than $M = N$ (no redundancy!)



Example of coordinate transformation

planar 2R robot using absolute coordinates



- motor 1 at joint 1, motor 2 at joint 2
- in place of DH angles q , use the **absolute** angles $p_1 = q_1$ and $p_2 = q_1 + q_2$

$$p = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} q = J q \quad \text{a linear transformation}$$

$$J^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \quad J^{-T} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$$

- from $M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u_q$
obtained with DH **relative** coordinates

blue terms are the same found in a direct way in slide #15

$$M_p(p) = J^{-T} M J^{-1} = \begin{pmatrix} a_1 - a_3 & a_2 c_{2-1} \\ a_2 c_{2-1} & a_3 \end{pmatrix} \quad g_p(p) = J^{-T} g = \begin{pmatrix} a_4 c_1 \\ a_5 c_2 \end{pmatrix}$$

$$c_p(p, \dot{p}) = J^{-T} c = \begin{pmatrix} -a_2 s_{2-1} \dot{p}_2^2 \\ a_2 s_{2-1} \dot{p}_1^2 \end{pmatrix} \quad u_p = J^{-T} u_q = \begin{pmatrix} u_{q1} - u_{q2} \\ u_{q2} \end{pmatrix}$$



Robot dynamic model in the task/Cartesian space, with redundancy

dynamic model in the joint space

$$M(q)\ddot{q} + n(q, \dot{q}) = \tau$$

$$\boxed{\begin{aligned} q &\in \mathbb{R}^N \\ r = f(q) &\in \mathbb{R}^M \\ M &< N \end{aligned}}$$

second-order task kinematics

$$\ddot{r} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$$

J is full rank = M

1) isolate the joint acceleration from the dynamics $\rightarrow \ddot{q} = M^{-1}(q) (\tau - n(q, \dot{q}))$

2) decompose the joint torques in two complementary spaces

$$\tau = J^T(q)F + (I - J^T(q)H(q))\tau_0$$

$$\in \mathcal{R}(J^T)$$

torques coming from
generalized forces F
in the task space ...

H is a generalized inverse of J^T

$$\in \mathcal{N}(J^T H)$$

$$J^T H J^T = J^T$$

... and joint torques $\tau_0 \notin \mathcal{R}(J^T)$

$$\rightarrow \tau_0 = J^T(q)F, \forall F \in \mathbb{R}^M \Rightarrow (I - J^T(q)H(q))J^T(q)F = 0$$

3) substitute 1) and 2) in the differential task kinematics

$$\rightarrow \ddot{r} = J(q)M^{-1}(q) \left(J^T(q)F + (I - J^T(q)H(q))\tau_0 - n(q, \dot{q}) \right) + \dot{J}(q)\dot{q}$$

4) isolate on the right-hand side the generalized forces F in the task space ...



Robot dynamic model in the task/Cartesian space, with redundancy

→
$$(J(q)M^{-1}(q)J^T(q))^{-1}\ddot{r} = \textcolor{red}{F} + (J(q)M^{-1}(q)J^T(q))^{-1}(J(q)M^{-1}(q)((I - J^T(q)\textcolor{blue}{H}(q))\tau_0 - n(q, \dot{q})) + j(q)\dot{q})$$

5) choose as generalized inverse $\textcolor{blue}{H} = (JM^{-1}J^T)^{-1}JM^{-1} = (J_M^\#)^T$, i.e., the transpose of the **inertia-weighted pseudoinverse** of the task Jacobian (see block of slides #2)

→ in this way, the joint torque component τ_0 will **NOT** affect the task acceleration \ddot{r}

$$(J(q)M^{-1}(q)J^T(q))^{-1}\ddot{r} = F + (J(q)M^{-1}(q)J^T(q))^{-1}(j(q)\dot{q} - J(q)M^{-1}(q)n(q, \dot{q}))$$

6) the resulting (M –dimensional) **task dynamics** is then

$$M_r(q)\ddot{r} + n_r(q, \dot{q}) = F \dots + F_{ext}$$

with

$$\begin{aligned} M_r(q) &= (J(q)M^{-1}(q)J^T(q))^{-1} \text{ task inertia matrix} \\ n_r(q, \dot{q}) &= M_r(q)(J(q)M^{-1}(q)n(q, \dot{q}) - j(q)\dot{q}) \end{aligned}$$

external forces can be added on the rhs of the equations in a **dynamically consistent** way!

for $M = N$, these terms are identical to slide #27

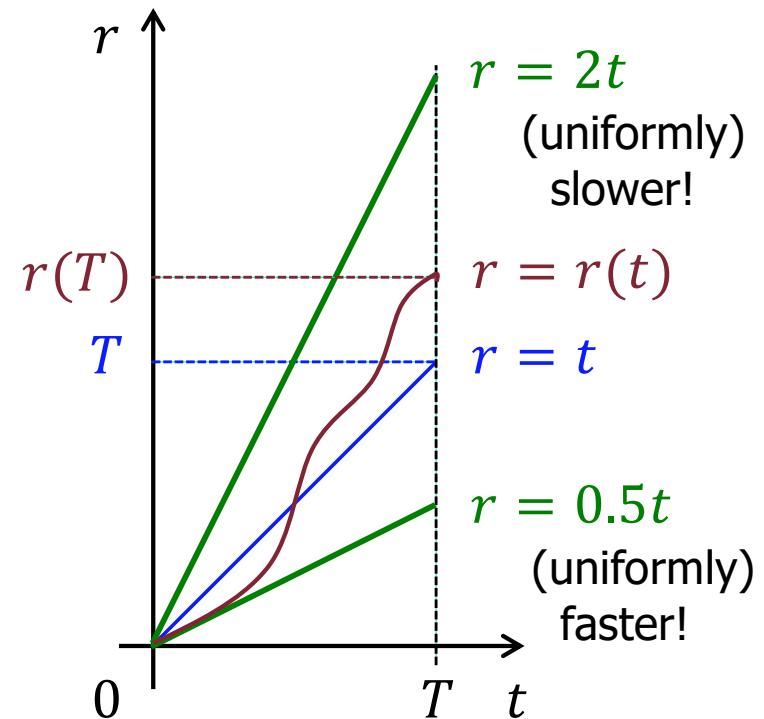
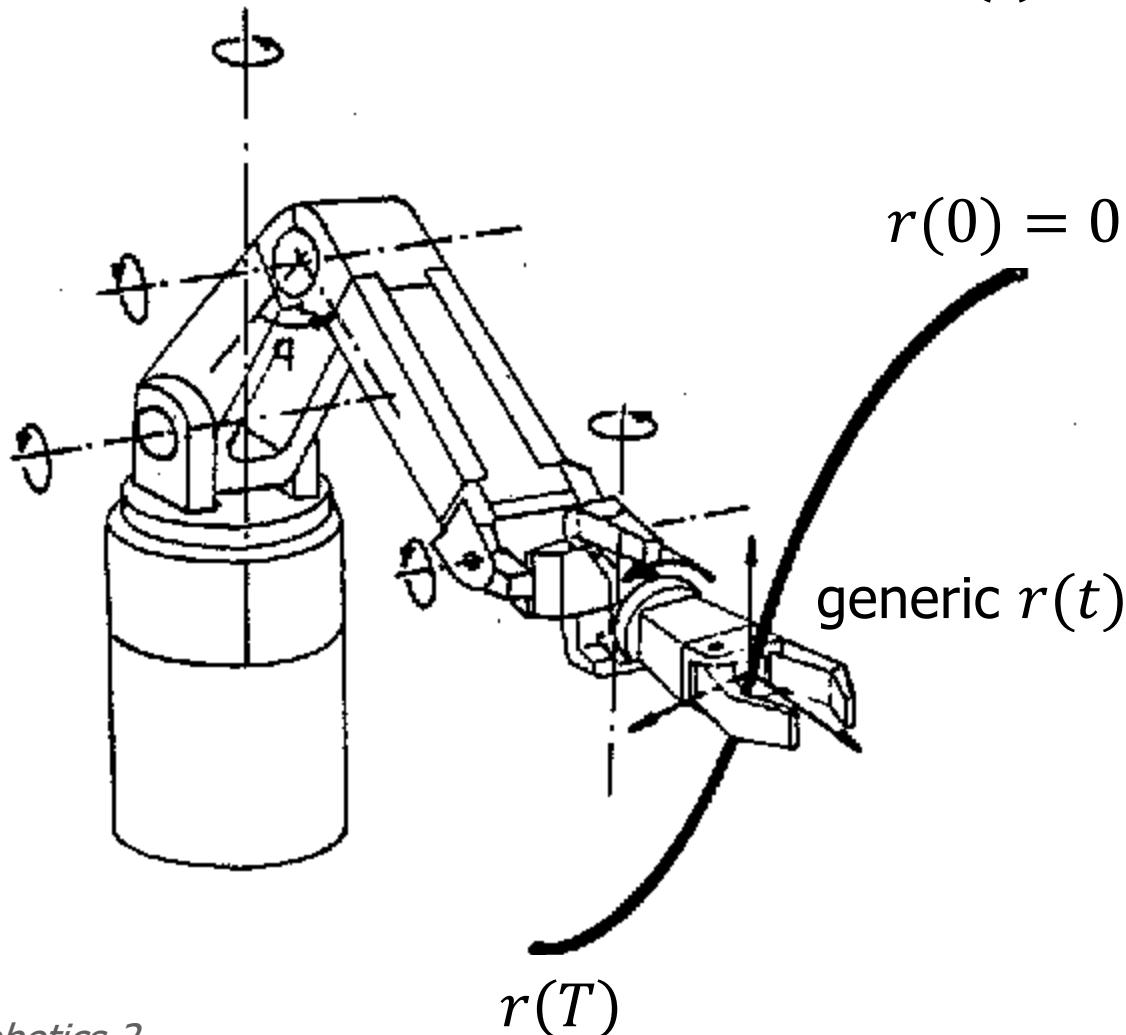
7) an additional ($N - M$)-dimensional second-order dynamics is needed to describe the full robot!



Dynamic scaling of trajectories

uniform time scaling of motion

- given a smooth **original trajectory** $q_d(t)$ of motion for $t \in [0, T]$
 - suppose to **rescale** time as $t \rightarrow r(t)$ (a strictly **increasing** function of t)





Dynamic scaling of trajectories

uniform time scaling of motion

- in the new time scale, the scaled trajectory $q_s(r)$ satisfies

$$q_d(t) = q_s(r(t)) \rightarrow \dot{q}_d(t) = \frac{dq_d}{dt} = \frac{dq_s}{dr} \frac{dr}{dt} = q'_s(r) \dot{r}(t)$$

same path executed
(at different instants of time)

$$\ddot{q}_d(t) = \frac{d\dot{q}_d}{dt} = \left(\frac{dq'_s}{dr} \frac{dr}{dt} \right) \dot{r} + q'_s \frac{d\dot{r}}{dt} = q''_s(r) \dot{r}^2(t) + q'_s(r) \ddot{r}(t)$$

- uniform scaling of the trajectory occurs when $r(t) = kt$

$$\dot{q}_d(t) = kq'_s(kt) \quad \ddot{q}_d(t) = k^2 q''_s(kt)$$

Q: what is the new input torque needed to execute the scaled trajectory?
(suppose dissipative terms can be neglected)



Dynamic scaling of trajectories

inverse dynamics under uniform time scaling

- the new torque could be recomputed through the inverse dynamics, for every $r = kt \in [0, T_s] = [0, kT]$ along the **scaled** trajectory, as

$$\tau_s(kt) = M(q_s)q_s'' + c(q_s, q_s') + g(q_s)$$

- however, being the dynamic model **linear** in the acceleration and **quadratic** in the velocity, it is

$$\begin{aligned} \tau_d(t) &= M(q_d)\ddot{q}_d + c(q_d, \dot{q}_d) + g(q_d) = M(q_s)k^2q_s'' + c(q_s, kq_s') + g(q_s) \\ &= k^2(M(q_s)q_s'' + c(q_s, q_s')) + g(q_s) = k^2(\tau_s(kt) - g(q_s)) + g(q_s) \end{aligned}$$

- thus, saving separately the total torque $\tau_d(t)$ and gravity torque $g_d(t)$ in the inverse dynamics computation along the **original** trajectory, the **new input torque** is obtained **directly** as

$$\boxed{\tau_s(kt) = \frac{1}{k^2}(\tau_d(t) - g(q_d(t))) + g(q_d(t))}$$

$k > 1$: slow down
 \Rightarrow reduce torque
 $k < 1$: speed up
 \Rightarrow increase torque

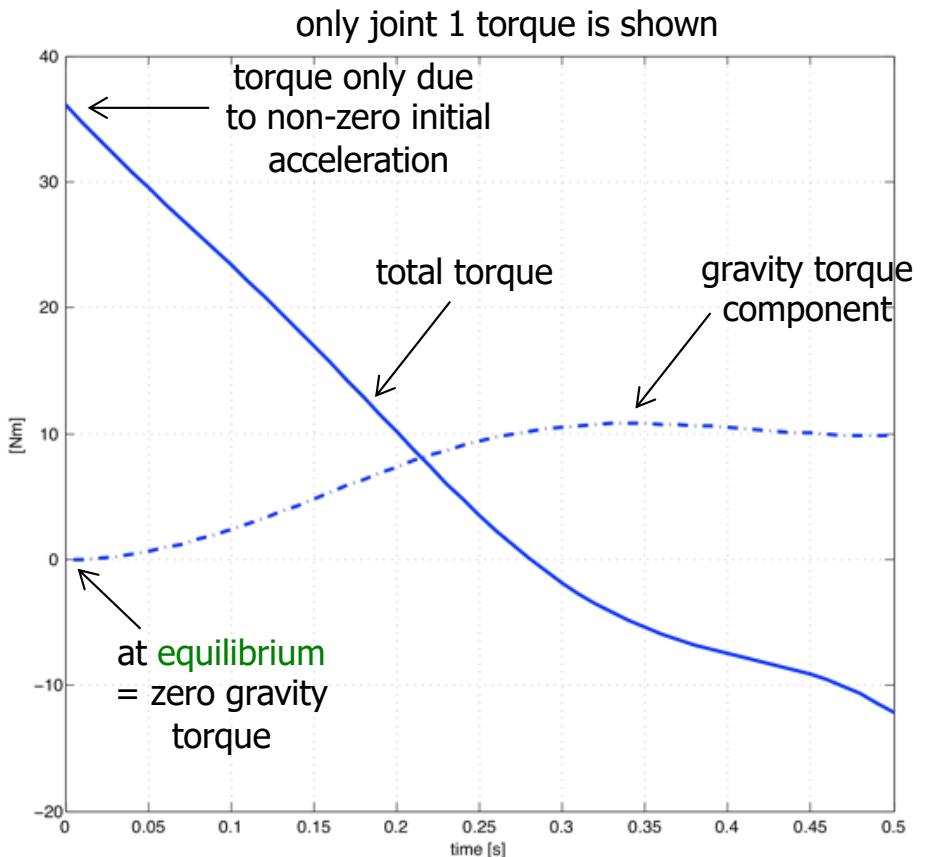
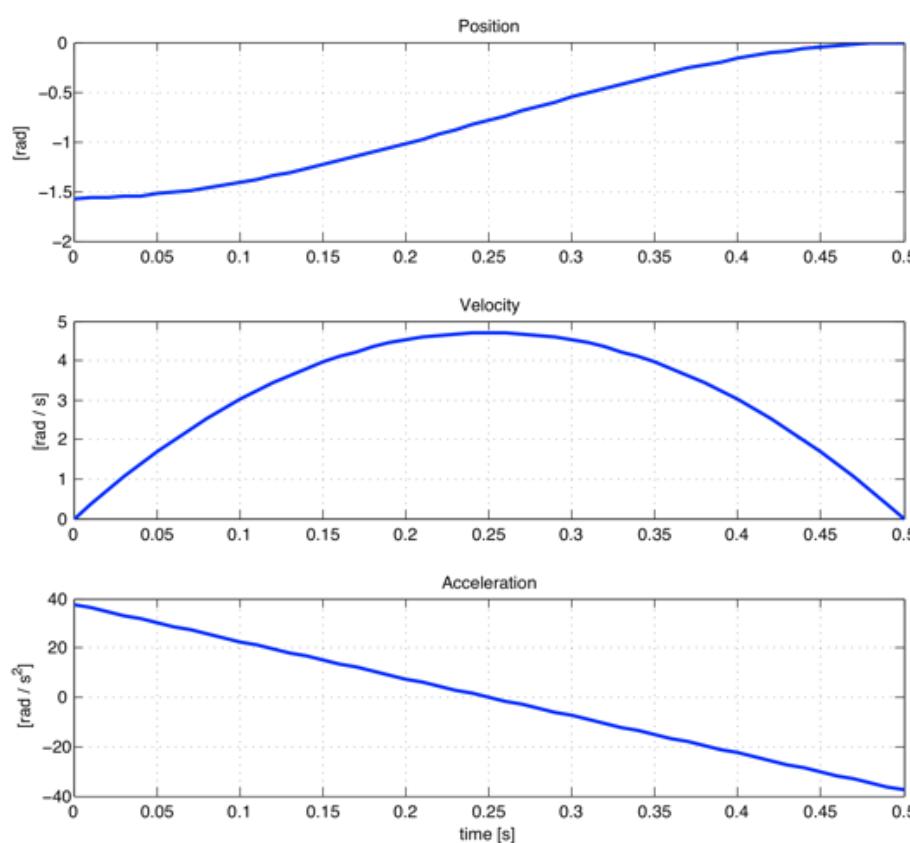
gravity term (only position-dependent): does **NOT** scale!



Dynamic scaling of trajectories

numerical example

- rest-to-rest motion with cubic polynomials for planar 2R robot under gravity (from downward **equilibrium** to horizontal link 1 & upward vertical link 2)
- original trajectory lasts $T = 0.5$ s (but say, it violates the torque limit at joint 1)

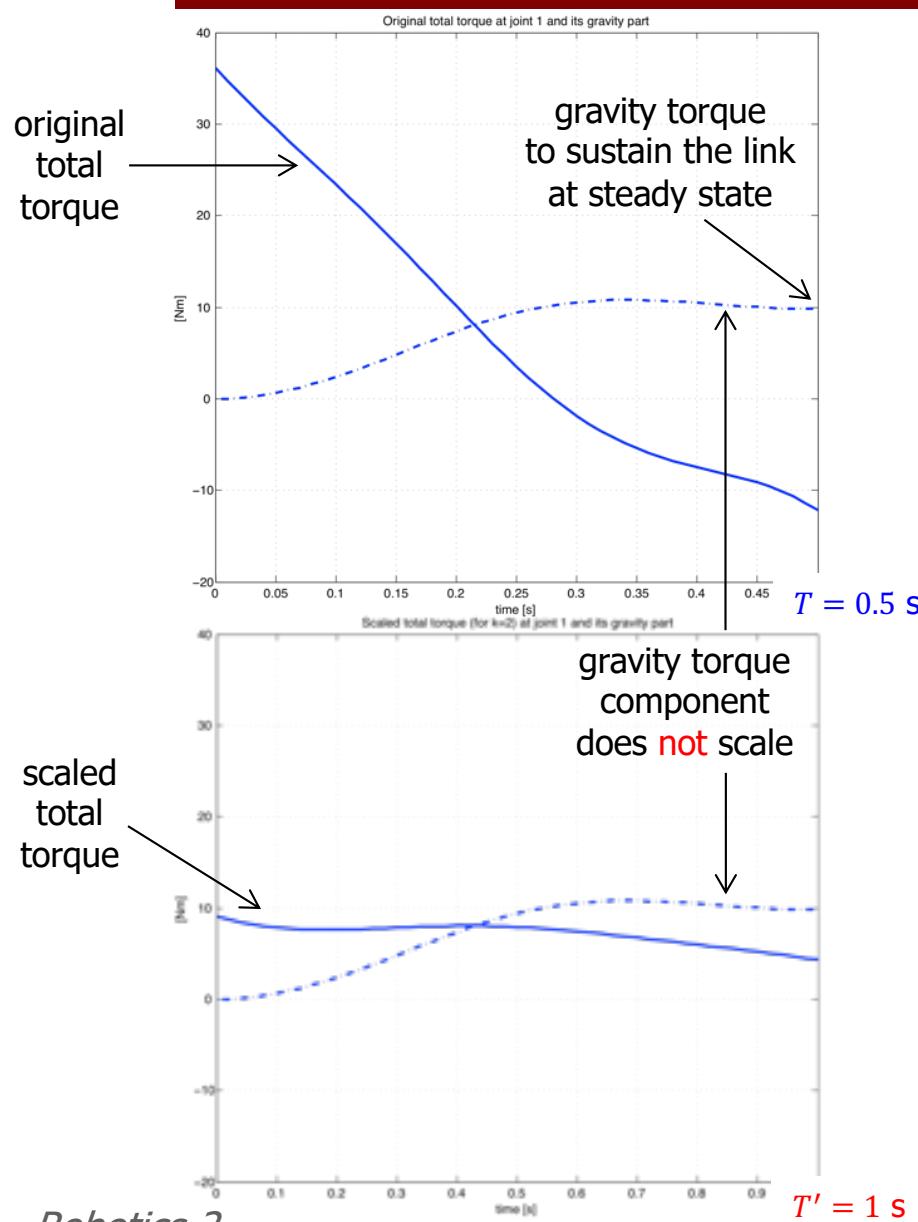


for **both** joints

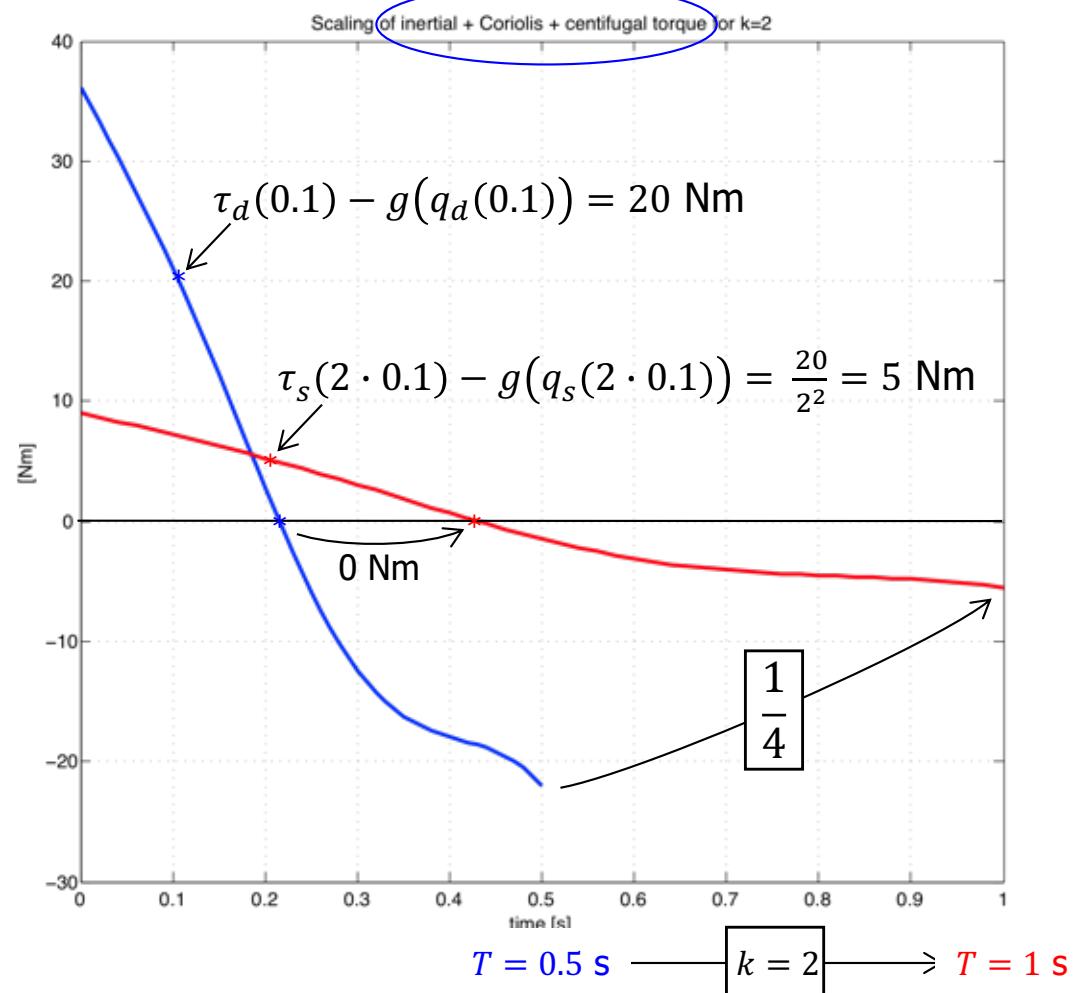


Dynamic scaling of trajectories

numerical example



- scaling with $k = 2$ (slower) $\rightarrow T' = 1 \text{ s}$

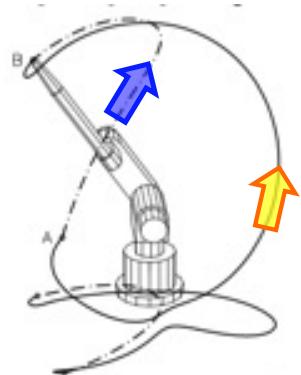




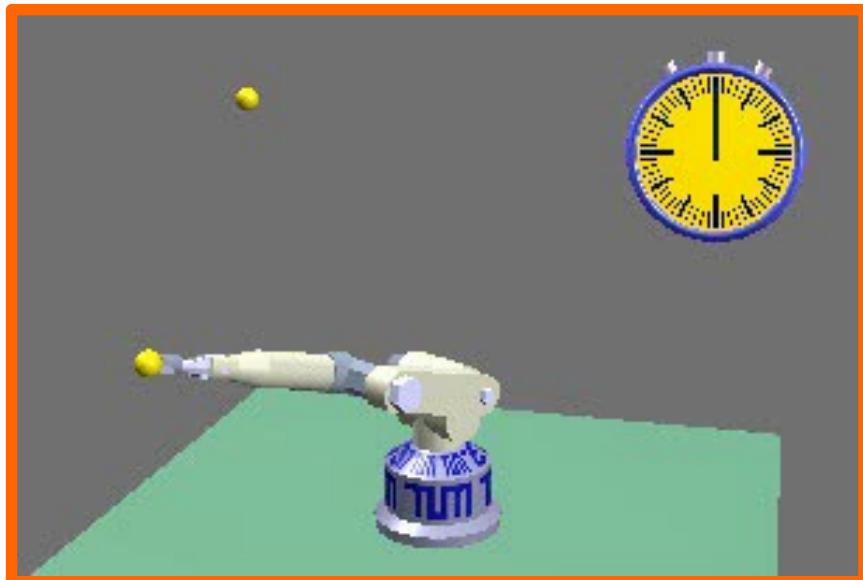
Optimal point-to-point robot motion

considering the dynamic model

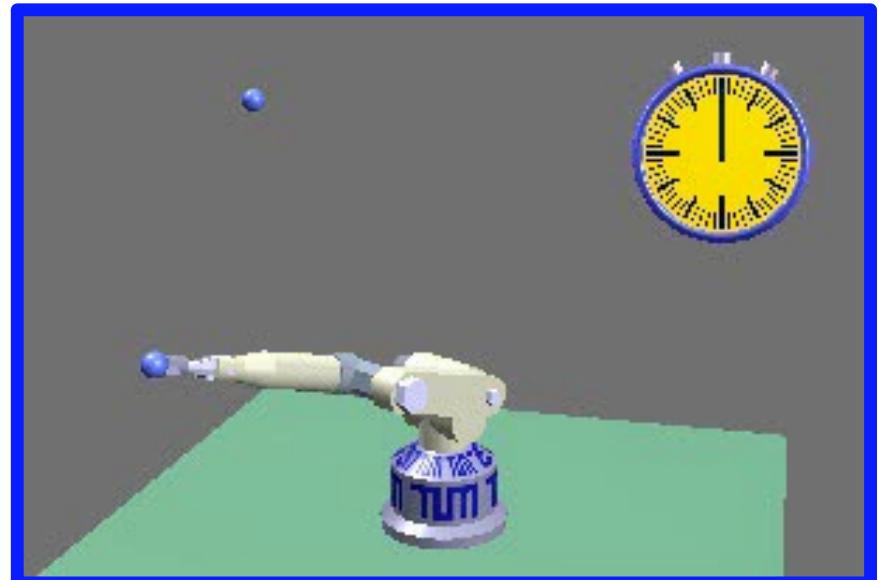
- given the initial ($\Rightarrow A$) and final ($\Rightarrow B$) robot configurations (at rest) and the actuator torque bounds, find
 - the **minimum-time** T_{\min} motion
 - the (global/integral) **minimum-energy** E_{\min} motionand the associated **command torques** needed to execute them
- a complex nonlinear optimization problem solved **numerically**



video



$T_{\min} = 1.32 \text{ s}, E = 306$



$T = 1.60 \text{ s}, E_{\min} = 6.14$



Robotics 2

Linear parametrization and identification of robot dynamics

Prof. Alessandro De Luca

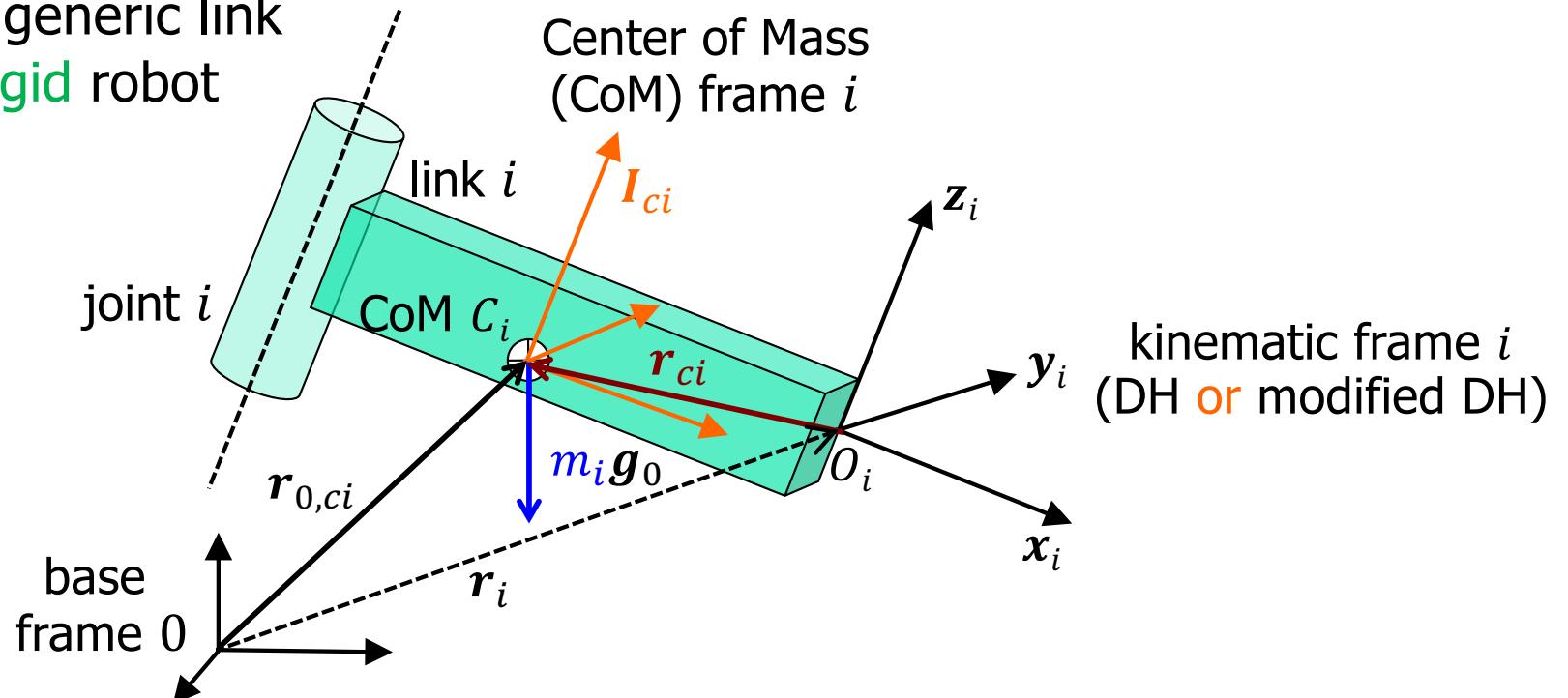
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Dynamic parameters of robot links

- consider a generic link of a **fully rigid** robot



- each link is characterized by $\begin{bmatrix} m_i & \mathbf{r}_{ci} = \begin{pmatrix} r_{xi} \\ r_{yi} \\ r_{zi} \end{pmatrix} & \mathbf{I}_{ci} = \begin{pmatrix} I_{ci,xx} & I_{ci,xy} & I_{ci,xz} \\ I_{ci,yx} & I_{ci,yy} & I_{ci,yz} \\ \text{symm} & I_{ci,zx} & I_{ci,zz} \end{pmatrix} \end{bmatrix}$
- however, robot dynamics depends **only on some** of these parameters and possibly in a **nonlinear** way (e.g., via the combination $I_{ci,zz} + m_i r_{xi}^2$)



Dynamic parameters of robots

- both the kinetic energy and the gravity potential energy can be rewritten so that a **new** set of dynamic parameters appears **only in a linear way**
 - need to re-express link inertia and CoM position in (any) **known** kinematic frame attached to the link (same orientation as the barycentric frame)
- fundamental kinematic relation

$$\boldsymbol{v}_{ci} = \boldsymbol{v}_i + \boldsymbol{\omega}_i \times \boldsymbol{r}_{ci} = \boldsymbol{v}_i + S(\boldsymbol{\omega}_i) \boldsymbol{r}_{ci} = \boldsymbol{v}_i - S(\boldsymbol{r}_{ci}) \boldsymbol{\omega}_i$$

- kinetic energy of link i

$$\begin{aligned}
 T_i &= \frac{1}{2} m_i \boldsymbol{v}_{ci}^T \boldsymbol{v}_{ci} + \frac{1}{2} \boldsymbol{\omega}_i^T \boldsymbol{I}_{ci} \boldsymbol{\omega}_i \quad \Leftrightarrow \text{"reversing" König theorem now ...} \\
 &= \frac{1}{2} m_i (\boldsymbol{v}_i - S(\boldsymbol{r}_{ci}) \boldsymbol{\omega}_i)^T (\boldsymbol{v}_i - S(\boldsymbol{r}_{ci}) \boldsymbol{\omega}_i) + \frac{1}{2} \boldsymbol{\omega}_i^T \boldsymbol{I}_{ci} \boldsymbol{\omega}_i \\
 &= \frac{1}{2} m_i \boldsymbol{v}_i^T \boldsymbol{v}_i + \frac{1}{2} \boldsymbol{\omega}_i^T \underbrace{(\boldsymbol{I}_{ci} + m_i S^T(\boldsymbol{r}_{ci}) S(\boldsymbol{r}_{ci}))}_{\substack{\text{Steiner theorem}}} \boldsymbol{\omega}_i - \boldsymbol{v}_i^T S(m_i \boldsymbol{r}_{ci}) \boldsymbol{\omega}_i
 \end{aligned}$$

Steiner theorem → $\boldsymbol{I}_i = \begin{pmatrix} I_{i,xx} & I_{i,xy} & I_{i,xz} \\ & I_{i,yy} & I_{i,yz} \\ \text{symm} & & I_{i,zz} \end{pmatrix}$



Standard dynamic parameters of robots

- gravitational potential energy of link i

$$U_i = -m_i g_0^T r_{0,ci} = -m_i g_0^T (r_i + r_{ci}) = -m_i g_0^T r_i - g_0^T (m_i r_{ci})$$

- by expressing vectors and matrices in frame i , both T_i and U_i will be **linear** in the set of 10 (constant) **standard parameters** $\boldsymbol{\pi}_i \in \mathbb{R}^{10}$

$$T_i = \frac{1}{2} m_i^i v_i^T v_i + m_i^i r_{ci}^T S(v_i) \omega_i + \frac{1}{2} \omega_i^T I_i \omega_i$$

$$U_i = -m_i g_0^T r_i - g_0^T R_i (m_i^i r_{ci})$$

mass of link i
(0-th order moment)
mass \times CoM
position of link i
(1-st order moment)
inertia of link i
(2-nd order moment)

$$\boldsymbol{\pi}_i = \begin{pmatrix} m_i \\ m_i^i r_{ci} \\ \text{vect}\{I_i\} \end{pmatrix} = (m_i \quad m_i^i r_{ci,x} \quad m_i^i r_{ci,y} \quad m_i^i r_{ci,z} \quad iI_{i,xx} \quad iI_{i,xy} \quad iI_{i,xz} \quad iI_{i,yy} \quad iI_{i,yz} \quad iI_{i,zz})^T$$

- since the E-L equations involve only **linear** operations on T and U , also the robot dynamic model is linear in the standard parameters $\boldsymbol{\pi} \in \mathbb{R}^{10N}$



Linearity in the dynamic parameters

- using a $N \times 10N$ regression matrix Y_π that depends only on **kinematic** quantities, the robot dynamic equations can always be rewritten **linearly** in the **standard dynamic parameters** as

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = Y_\pi(q, \dot{q}, \ddot{q}) \pi = u$$

$$\pi^T = (\pi_1^T \quad \pi_2^T \quad \dots \quad \pi_N^T)$$

- the open kinematic chain structure of the manipulator implies that the i -th dynamic equation can depend only on the dynamic parameters of links from i to $N \Rightarrow Y_\pi$ has a **block upper triangular** structure

$$Y_\pi(q, \dot{q}, \ddot{q}) = \begin{pmatrix} Y_{11} & Y_{12} & \cdots & Y_{1N} \\ 0 & Y_{22} & \cdots & Y_{2N} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & Y_{NN} \end{pmatrix}$$

with **row** vectors
 Y_{ij} of size 1×10

Property: element m_{ij} of $M(q)$ depends at most on (q_{k+1}, \dots, q_N) , with $k = \min\{i, j\}$, and at most on the dynamic parameters of links h to N , with $h = \max\{i, j\}$



Linearity in the dynamic coefficients

- many standard parameters do not appear (“play no role”) in the dynamic model of a given robot \Rightarrow the associated **columns of Y_π are 0!**
- some standard parameters may appear only in fixed combinations with others \Rightarrow the associated **columns of Y_π are linearly dependent!**
- one can isolate $p \ll 10N$ **groups** of parameters π (associated to p **functionally independent** columns Y_{indep} of Y_π) and partition matrix Y_π in two blocks, the second containing dependent (or zero) columns as $Y_{dep} = Y_{indep}T$, for a suitable $p \times (10N - p)$ constant matrix T

$$\begin{aligned}
 Y_\pi(q, \dot{q}, \ddot{q}) \pi &= (Y_{indep} \quad Y_{dep}) \begin{pmatrix} \pi_{indep} \\ \pi_{dep} \end{pmatrix} = (Y_{indep} \quad Y_{indep}T) \begin{pmatrix} \pi_{indep} \\ \pi_{dep} \end{pmatrix} \\
 &= Y_{indep}(\pi_{indep} + T\pi_{dep}) = \boxed{Y(q, \dot{q}, \ddot{q}) a}
 \end{aligned}$$

- these grouped parameters are called **dynamic coefficients** $a \in \mathbb{R}^p$, “the only that matter” in robot dynamics (= **base parameters** by W. Khalil)
- the **minimal number p** of dynamic coefficients that is needed can also be checked numerically (see \rightarrow identification)



Linear parametrization of robot dynamics

it is **always** possible to rewrite the dynamic model in the form

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = Y(q, \dot{q}, \ddot{q}) a = u$$

regression matrix \downarrow a = vector of dynamic coefficients \downarrow
 $N \times p$ \uparrow $p \times 1$ \uparrow

e.g., the **heuristic** grouping (found by inspection) on the planar 2R robot

$$\begin{pmatrix} \ddot{q}_1 & c_2(2\ddot{q}_1 + \ddot{q}_2) - s_2(\dot{q}_2^2 + 2\dot{q}_1\dot{q}_2) & 0 \\ 0 & c_2\ddot{q}_1 + s_2\dot{q}_1^2 & c_1\ddot{q}_2 + \ddot{q}_1 + \ddot{q}_2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$\begin{aligned} a_1 &= I_{c1,zz} + m_1 d_1^2 + I_{c2,zz} + m_2 d_2^2 + m_2 l_1^2 & a_2 &= m_2 l_1 d_2 \\ a_3 &= I_{c2,zz} + m_2 d_2^2 & a_4 &= g_0(m_1 d_1 + m_2 l_1) \\ a_5 &= g_0 m_2 d_2 & \end{aligned}$$

Note: 4 more coefficients are added when including the coefficients $F_{V,i}$ and $F_{C,i}$ of viscous and Coulomb friction at the joints ("decoupled" terms appearing only in i -th equation, for $i = 1, 2$)



Linear parametrization of a 2R planar robot ($N = 2$)

- being the kinematics known (i.e., l_1 and g_0), the number of dynamic coefficients can be reduced since we can merge the two coefficients
 $a_2 = m_2 l_1 d_2 \quad \& \quad a_5 = g_0 m_2 d_2 \quad \Rightarrow \quad a_2 = m_2 d_2$ (factoring out l_1 and g_0)
- therefore, after regrouping, $\textcolor{green}{p = 4}$ dynamic coefficients are sufficient

$$\begin{pmatrix} \ddot{q}_1 & l_1 c_2(2\ddot{q}_1 + \ddot{q}_2) - l_1 s_2(\dot{q}_2^2 + 2\dot{q}_1 \dot{q}_2) + g_0 c_{12} \\ 0 & l_1(c_2 \ddot{q}_1 + s_2 \dot{q}_1^2) + g_0 c_{12} \end{pmatrix} \begin{pmatrix} \ddot{q}_2 \\ \dot{q}_1 + \ddot{q}_2 \\ 0 \end{pmatrix} = Y a = u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$a_1 = I_{c1,zz} + m_1 d_1^2 + I_{c2,zz} + m_2 d_2^2 + m_2 l_1^2 \quad a_3 = I_{c2,zz} + m_2 d_2^2$$

$$a_2 = m_2 d_2 \quad a_4 = m_1 d_1 + m_2 l_1$$

- this (minimal) linear parametrization of robot dynamics is **not unique**, both in terms of the chosen set of dynamic coefficients $\textcolor{blue}{a}$ and for the associated regression matrix Y
- a systematic procedure for its derivation would be preferable



Linear parametrization of a 2R planar robot ($N = 2$)

- as alternative to the previous heuristic method, apply the **general procedure**
 - $10N = 20$ **standard parameters π** are defined for the two links
 - from the assumptions made on CoM locations, **only 5** such parameters actually appear, namely (with ${}^i r_{ci,x} = -l_i + d_i$)

$$\text{link 1: } m_1 d_1 \quad I_{1,zz} = I_{c1,zz} + m_1 d_1^2 \quad \begin{matrix} \pi_1 \\ \pi_2 \end{matrix}$$

$$\text{link 2: } m_2 \quad m_2 d_2 \quad I_{2,zz} = I_{c2,zz} + m_2 d_2^2 \quad \begin{matrix} \pi_3 \\ \pi_4 \\ \pi_5 \end{matrix}$$



- in the 2×5 matrix Y_π , the 3rd column (associated to m_2) is $Y_{\pi 3} = Y_{\pi 1} l_1 + Y_{\pi 2} l_1^2$
- after regrouping/reordering, **$p = 4$ dynamic coefficients** are again sufficient

$$\begin{pmatrix} g_0 c_1 & \ddot{q}_1 & l_1 c_2 (2\ddot{q}_1 + \ddot{q}_2) - l_1 s_2 (\dot{q}_2^2 + 2\dot{q}_1 \dot{q}_2) + g_0 c_{12} & \ddot{q}_1 + \ddot{q}_2 \\ 0 & 0 & l_1 (c_2 \ddot{q}_1 + s_2 \dot{q}_1^2) + g_0 c_{12} & \ddot{q}_1 + \ddot{q}_2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = Y a = u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$a_1 = m_1 d_1 + \boxed{m_2} l_1 \quad a_2 = I_{1,zz} + \boxed{m_2} l_1^2 = (I_{c1,zz} + m_1 d_1^2) + m_1 l_1^2 \quad a_3 = m_2 d_2$$

$$a_4 = I_{2,zz} = I_{c2,zz} + m_2 d_2^2$$

- determining a **minimal parameterization** (i.e., minimizing p) is important for
 - experimental identification of dynamic coefficients
 - adaptive/robust control design in the presence of uncertain parameters



Identification of dynamic coefficients

- in order to “use” the model, one needs to know the numeric values of the robot **dynamic coefficients**
 - robot manufacturers provide at most only a few principal dynamic parameters (e.g., link masses)
- **estimates** can be found with CAD tools (e.g., assuming uniform mass)
- friction coefficients are (slowly) varying over time
 - lubrication of joints/transmissions
- for an added payload (attached to the E-E)
 - a change in the 10 dynamic parameters of last link ...
 - ... implies a variation of (almost) all robot dynamic coefficients!
- preliminary **identification experiments** are needed
 - robot **in motion** (dynamic issues, not just static or geometric ones!)
 - **only** the robot dynamic **coefficients** can be identified (and **not all** the link standard parameters!)



Identification experiments

1. choose a motion trajectory $q_d(t)$ that is sufficiently “exciting”, i.e.,
 - explores the robot workspace and involves all components in the robot dynamic model
 - is periodic, with multiple frequency components
2. execute this motion (approximately) by means of a control law
 - taking advantage of any available information on the robot model
 - often $u = K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})$ (PD, no model information used)
3. measure q (encoders) in n_c time instants (and, if available, also \dot{q})
 - joint velocity \dot{q} and acceleration \ddot{q} can be estimated later off line by numerical differentiation (use of non-causal filters is feasible)
4. with such measures/estimates, evaluate the regression matrix Y (on the left) and use the applied commands u (on the right) in the expression

$$Y(q(t_k), \dot{q}(t_k), \ddot{q}(t_k)) a = u(t_k) \quad k = 1, \dots, n_c$$



Least Squares (LS) identification

- set up the system of **linear** equations

$$n_c \times N \begin{pmatrix} Y(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y(q(t_{n_c}), \dot{q}(t_{n_c}), \ddot{q}(t_{n_c})) \end{pmatrix} a = \begin{pmatrix} u(t_1) \\ \vdots \\ u(t_{n_c}) \end{pmatrix} \quad \leftrightarrow \quad \bar{Y}a = \bar{u}$$

- sufficiently “exciting” trajectories, large enough number of samples ($n_c \times N \gg p$), and their suitable selection/position guarantee that **rank(\bar{Y}) = p** (full column rank)
- solution by **pseudoinversion** of matrix \bar{Y}

$$a = \bar{Y}^\# \bar{u} = (\bar{Y}^T \bar{Y})^{-1} \bar{Y}^T \bar{u} \quad (\in \mathbb{R}^p)$$

- one can also use a **weighted** pseudoinverse, to take into account different levels of noise in the collected measures

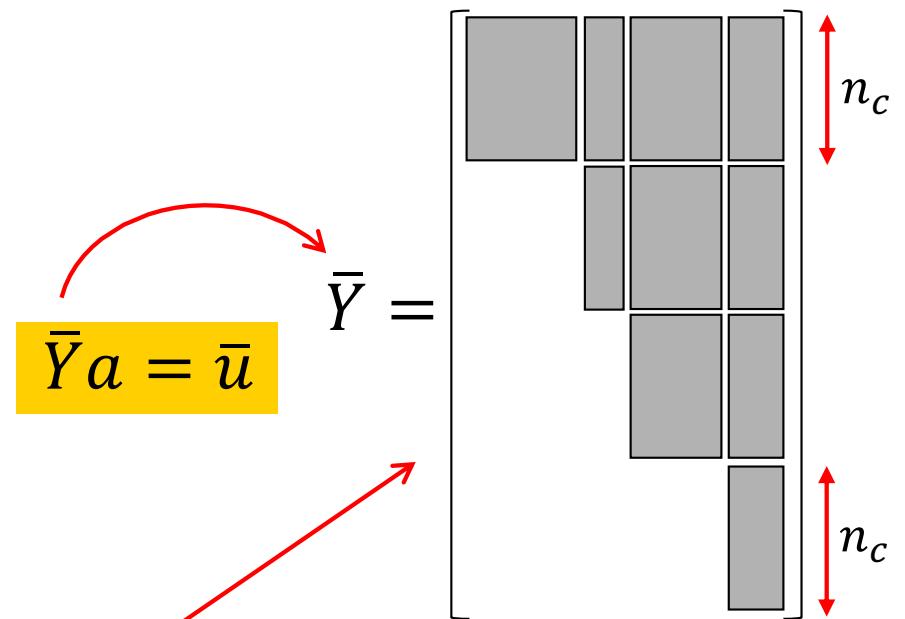


Additional remarks on LS identification

- it is convenient to preserve the **block (upper) triangular structure** of the regression matrix, by “stacking” all time evaluations **in row by row sequence** of the original Y matrix

$$N \times \left(\begin{array}{c} Y_1(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y_1(q(t_{n_c}), \dot{q}(t_{n_c}), \ddot{q}(t_{n_c})) \\ Y_2(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y_2(q(t_{n_c}), \dot{q}(t_{n_c}), \ddot{q}(t_{n_c})) \\ \vdots \\ Y_N(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y_N(q(t_{n_c}), \dot{q}(t_{n_c}), \ddot{q}(t_{n_c})) \end{array} \right) a = \left(\begin{array}{c} u_1(t_1) \\ \vdots \\ u_1(t_{n_c}) \\ u_2(t_1) \\ \vdots \\ u_2(t_{n_c}) \\ \vdots \\ u_N(t_1) \\ \vdots \\ u_N(t_{n_c}) \end{array} \right)$$

$$\bar{Y}a = \bar{u}$$

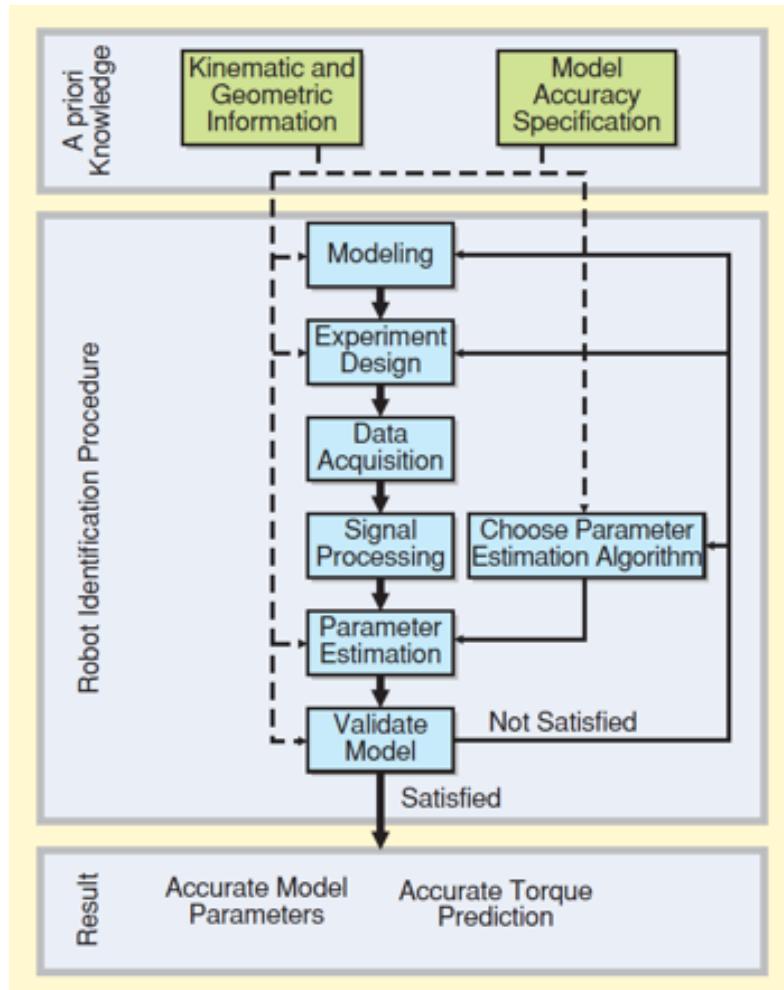


- numerical** check of full column rank is more robust \Leftrightarrow **rank = p** (# of col's)

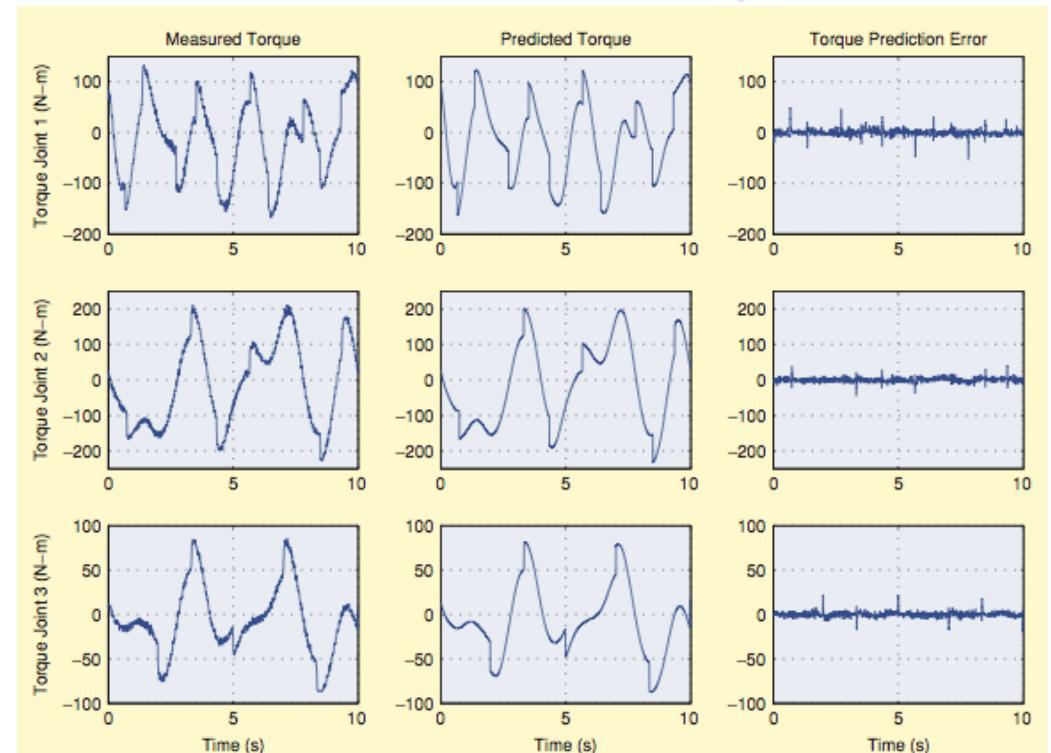
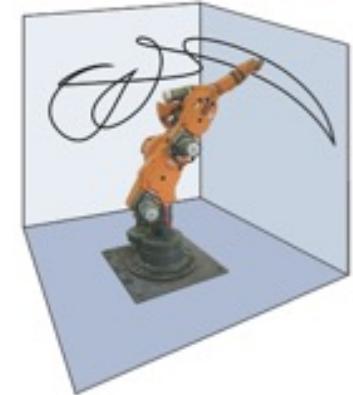
- further practical hints
 - outlier data** can be eliminated in advance (when building Y)
 - if sufficiently rich **friction** models are not included in Ya , **discard the data** collected at joint velocities **close to zero**



Summary on dynamic identification



KUKA IR 361
robot and
optimal
excitation
trajectory



J. Swevers, W. Verdonck, and J. De Schutter:
"Dynamic model identification for industrial robots"
IEEE Control Systems Mag., Oct 2007



Dynamic identification of KUKA LWR4

video

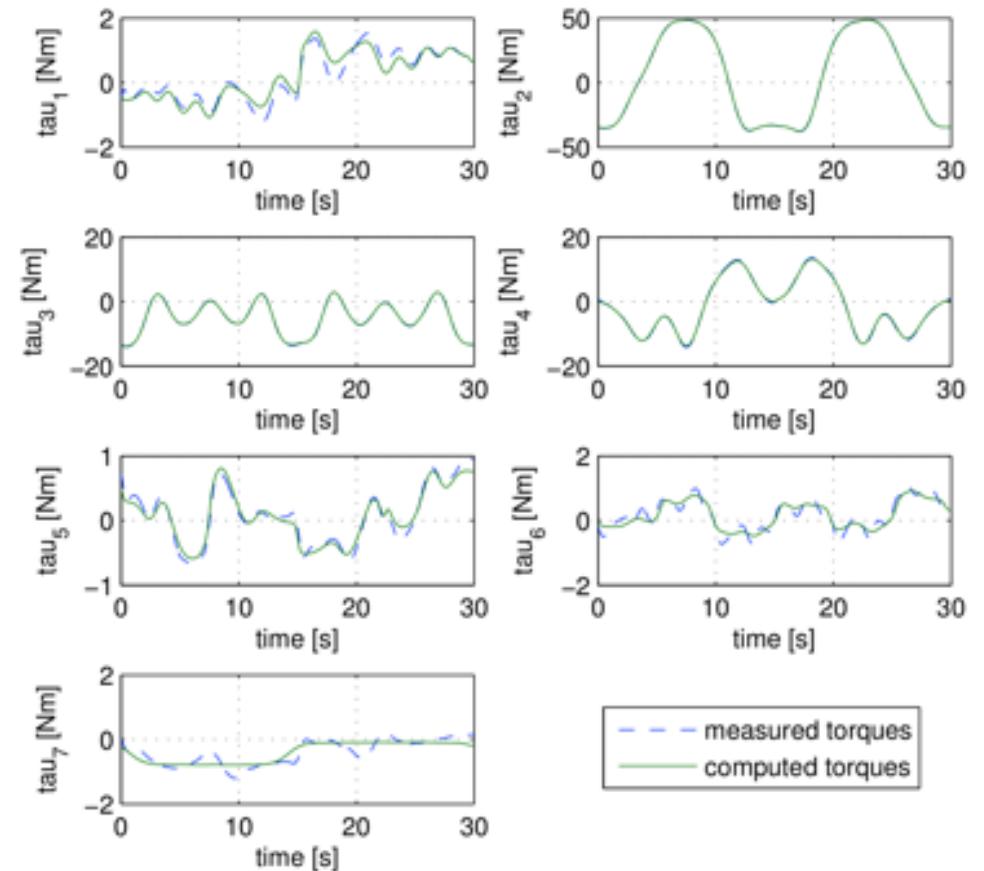


data acquisition for identification

dynamic coefficients: 30 inertial, 12 for gravity

C. Gaz, F. Flacco, A. De Luca:

"Identifying the dynamic model used by the KUKA LWR:
A reverse engineering approach", IEEE ICRA 2014



validation after identification (for all 7 joints):
on new desired trajectories, compare
torques computed with the identified model
and torques measured by joint torque sensors



Identification of LWR4 gravity terms

using the linear parametrization, gravity terms can also be identified **separately**

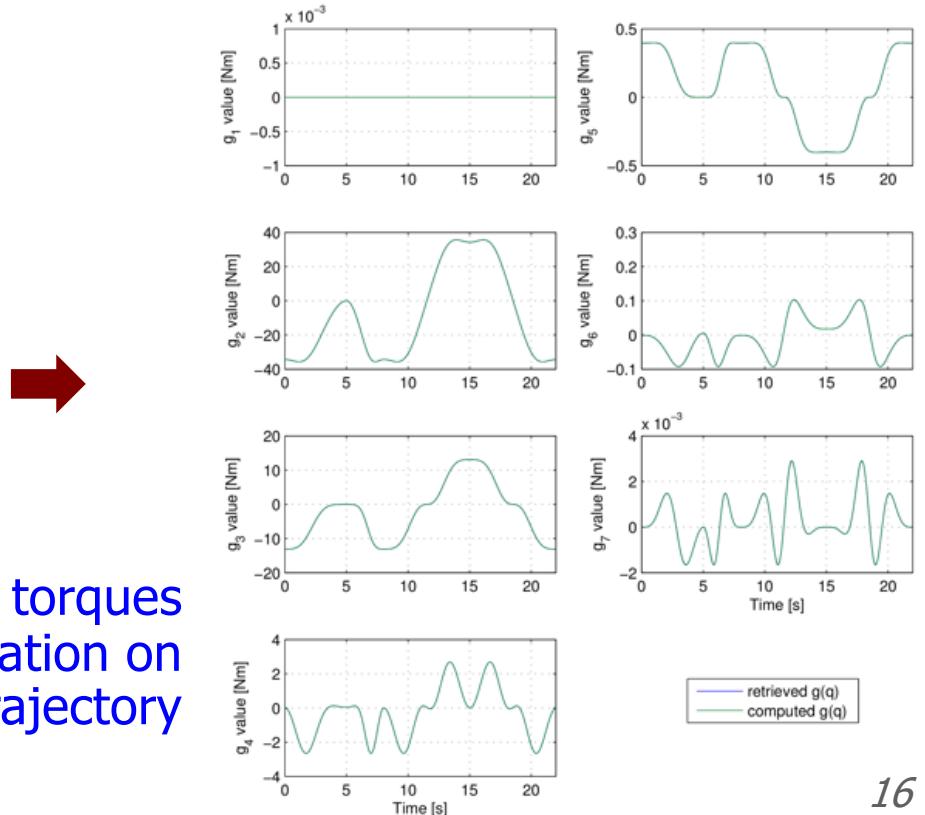
$$\boldsymbol{\pi}_g = \begin{pmatrix} c_{7y}m_7 \\ c_{7x}m_7 \\ c_{6x}m_6 \\ c_{6z}m_6 + c_{7z}m_7 \\ c_{5z}m_5 - c_{6y}m_6 \\ c_{5x}m_5 \\ c_{5y}m_5 + c_{4z}m_4 + d_2(m_5 + m_6 + m_7) \\ c_{4x}m_4 \\ c_{4y}m_4 + c_{3z}m_3 \\ c_{2x}m_2 \\ c_{3x}m_3 \\ c_{2z}m_2 - c_{3y}m_3 + d_1(m_3 + m_4 + m_5 + m_6 + m_7) \end{pmatrix}$$

some small values may also be discarded ...

$$\hat{\boldsymbol{\pi}}_g = \begin{pmatrix} 9.5457 \times 10^{-4} \\ -2.9826 \times 10^{-4} \\ 8.3524 \times 10^{-4} \\ 0.0286 \\ -0.0407 \\ -6.5637 \times 10^{-4} \\ 1.334 \\ -0.0035 \\ -4.7258 \times 10^{-4} \\ 0.0014 \\ 9.4532 \times 10^{-4} \\ 3.4568 \end{pmatrix}$$

$$\mathbf{g}(\mathbf{q}) = \mathbf{Y}_g(\mathbf{q})\boldsymbol{\pi}_g$$

symbolic expressions of gravity-related dynamic coefficients



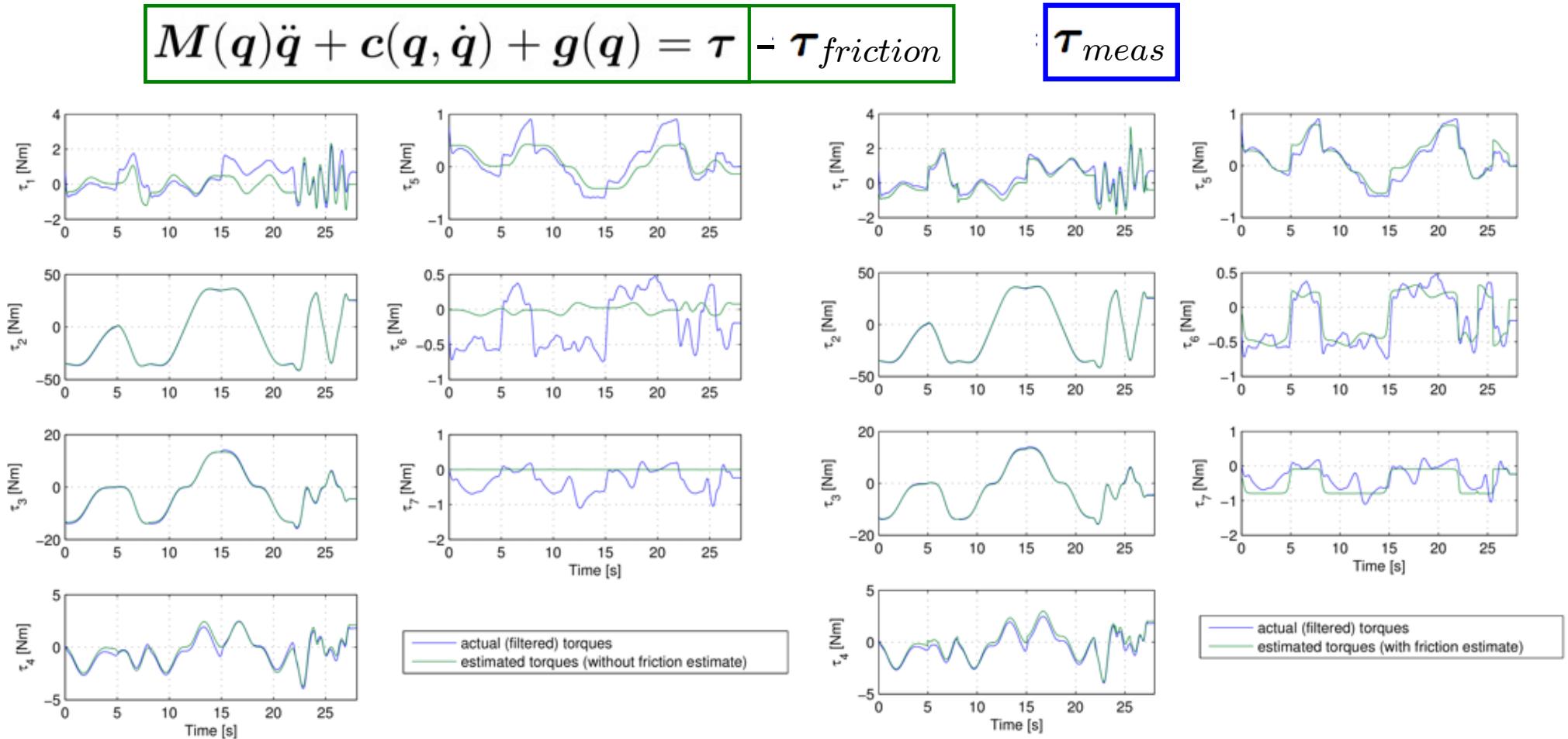
numerical values identified through experiments

gravity joint torques prediction/evaluation on new validation trajectory



Role of friction in identification

KUKA LWR4 dynamic model estimation vs. joint torque sensor measurement



without the use of a joint friction model

including an identified joint friction model

$$\tau_{f,j}(\dot{q}_j) = \frac{\varphi_{1,j}}{1 + e^{-\varphi_{2,j}(\dot{q}_j + \varphi_{3,j})}} - \frac{\varphi_{1,j}}{1 + e^{-\varphi_{2,j}\varphi_{3,j}}}$$



Dynamic identification of KUKA LWR4



using more dynamic robot motions for model identification

J. Hollerbach, W. Khalil, M. Gautier: "Ch. 6: Model Identification", Springer Handbook of Robotics (2nd Ed), 2016
free access to multimedia extension: <http://handbookofrobotics.org>



Adding a payload to the robot

- in several industrial applications, changes in the robot payload are often needed
 - using different tools for various technological operations such as polishing, welding, grinding, ...
 - pick-and-place tasks of objects having unknown mass
- what is the rule of change for dynamic parameters when there is an additional payload?
 - do we obtain again a linearly parameterized problem?
 - does this property rely on some specific choice of reference frames (e.g., conventional or modified D-H)?

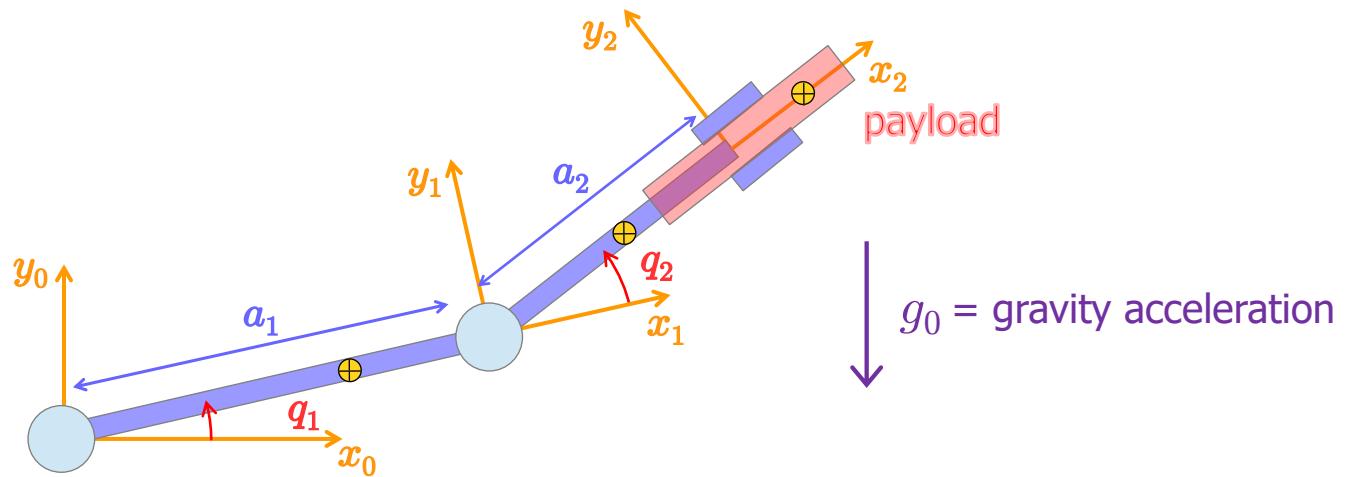


Rule of change in dynamic parameters

- only the dynamic parameters of the link where a load is added will change (typically, added to the last one –link n – as payload)
 - last link dynamic parameters: m_n (mass), $\mathbf{c}_n = (c_{nx} \ c_{ny} \ c_{nz})^T$ (center of mass), \mathbf{I}_n (inertia tensor expressed **w.r.t. frame n**)
 - payload dynamic parameters: m_L (mass), $\mathbf{c}_L = (c_{Lx} \ c_{Ly} \ c_{Lz})^T$ (center of mass), \mathbf{I}_L (inertia tensor expressed **w.r.t. frame n**)
- mass $m_n \rightarrow m_n + m_L$
- center of mass $c_{ni}m_n \rightarrow \frac{c_{ni}m_n + c_{Li}m_L}{m_n + m_L} (m_n + m_L) = c_{ni}m_n + c_{Li}m_L$
(weighted average) where $i = x, y, z$
- inertia tensor $\mathbf{I}_n \rightarrow \mathbf{I}_n + \mathbf{I}_L$ valid **only if** tensors are expressed w.r.t. the **same** reference frame (i.e., frame n)!
- **linear** parametrization is preserved with any kinematic convention (the parameters of the last link will always appear in the form shown above)



Example: 2R planar robot with payload



unloaded robot dynamics $Y\pi = \tau$

$$\pi = \begin{pmatrix} \frac{1}{2} (m_2 a_2^2 + I_{2zz}) + a_2 c_{2x} m_2 \\ c_{2x} m_2 + a_2 m_2 \\ c_{2y} m_2 \\ \frac{1}{2} (I_{1zz} + a_1^2 m_1 + a_1^2 m_2) + a_1 c_{1x} m_1 \\ c_{1x} m_1 + a_1 m_1 + a_1 m_2 \\ c_{1y} m_1 \end{pmatrix} \rightarrow \pi^L =$$

loaded robot dynamics $Y\pi^L = \tau^L$

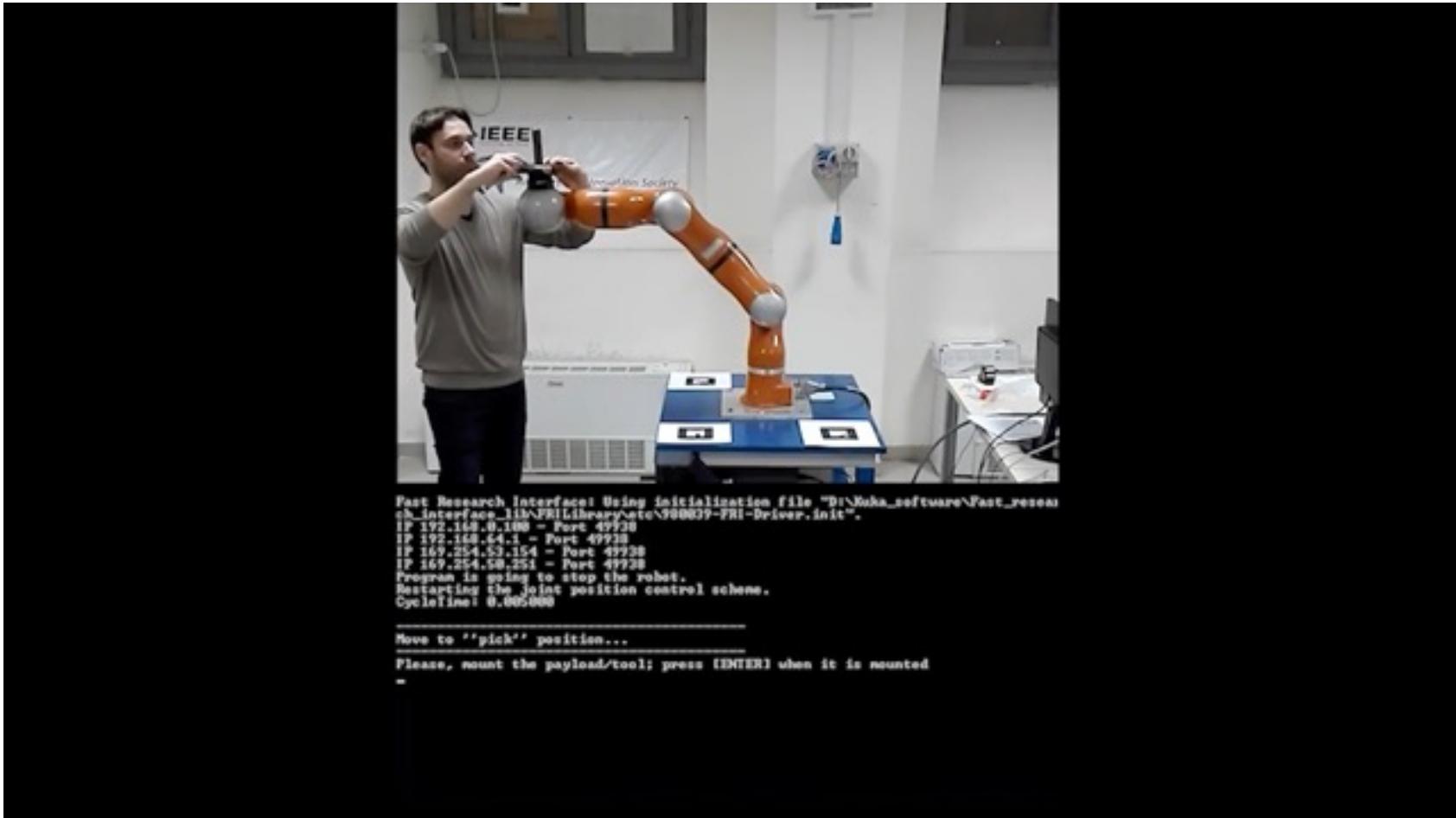
$$\begin{pmatrix} \frac{1}{2} (a_2^2 (m_2 + m_L) + I_{2zz} + I_{Lzz}) + a_2 (c_{2x} m_2 + c_{Lx} m_L) \\ c_{2x} m_2 + c_{Lx} m_L + a_2 (m_2 + m_L) \\ c_{2y} m_2 + c_{Ly} m_L \\ \frac{1}{2} (I_{1zz} + a_1^2 m_1 + a_1^2 (m_2 + m_L)) + a_1 c_{1x} m_1 \\ c_{1x} m_1 + a_1 m_1 + a_1 (m_2 + m_L) \\ c_{1y} m_1 \end{pmatrix}$$

Note 1: position of the center of mass of the two links and of the payload may also be asymmetric

Note 2: link inertia & center of mass are expressed in the DH kinematic frame attached to the link
(e.g., I_{2zz} is the inertia of the second link around the axis z_2)



Validation on the KUKA LWR4 robot



C. Gaz, A. De Luca: "Payload estimation based on identified coefficients of robot dynamics – with an application to collision detection" IEEE IROS 2017, Vancouver, September 2017



Bibliography

- J. Swevers, W. Verdonck, J. De Schutter, "Dynamic model identification for industrial robots," *IEEE Control Systems Mag.*, vol. 27, no. 5, pp. 58–71, 2007
- J. Hollerbach, W. Khalil, M. Gautier, "Model Identification," *Springer Handbook of Robotics (2nd Ed)*, pp. 113-138, 2016
- C. Gaz, F. Flacco, A. De Luca, "Identifying the dynamic model used by the **KUKA LWR**: A reverse engineering approach," *IEEE Int. Conf. on Robotics and Automation*, pp. 1386-1392, 2014
- C. Gaz, F. Flacco, A. De Luca, "Extracting feasible robot parameters from dynamic coefficients using nonlinear optimization methods," *IEEE Int. Conf. on Robotics and Automation*, pp. 2075-2081, 2016
- C. Gaz, A. De Luca, "Payload estimation based on identified coefficients of robot dynamics – with an application to collision detection," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3033-3040, 2017
- C. Gaz, E. Magrini, A. De Luca, "A **model-based** residual approach for human-robot collaboration during manual polishing operations," *Mechatronics*, vol. 55, pp. 234-247, 2018
- C. Gaz, M. Cognetti, A. Oliva, P. Robuffo Giordano, A. De Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Lett.*, vol. 4, no. 4, pp. 4147-4154, 2019

KUKA
LWR4 (7R)



Robotics 2



Universal Robot
UR10 (6R)



Franka Emika
Panda (7R)



Robotics 2

Dynamic model of robots: Newton-Euler approach

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Approaches to dynamic modeling

(reprise)

energy-based approach (Euler-Lagrange)



Newton-Euler method (balance of forces/momenta)

- multi-body robot seen as a whole
 - constraint (internal) reaction forces between the links are automatically eliminated: in fact, they do not perform work
 - closed-form (symbolic) equations are directly obtained
 - best suited for study of dynamic properties and **analysis** of control schemes
-
- dynamic equations written separately for each link/body
 - mainly used for **inverse dynamics in real time**
 - equations are evaluated in a **numeric** and **recursive** way
 - best for **synthesis** (=implementation) of model-based control schemes
 - by eliminating the internal reaction forces and performing back-substitution of all expressions, we get dynamic equations in closed-form (identical to Euler-Lagrange!)



Derivative of a vector in a moving frame

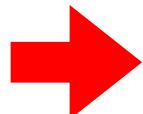
... from velocity to acceleration

$${}^0v_i = {}^0R_i {}^i v_i$$

$${}^0\dot{R}_i = S({}^0\omega_i) {}^0R_i$$

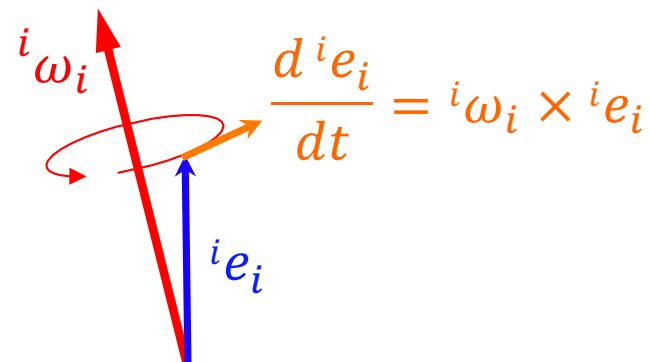
$${}^0\dot{v}_i = {}^0a_i = {}^0R_i {}^i a_i = {}^0R_i {}^i \dot{v}_i + {}^0\dot{R}_i {}^i v_i$$

$$= {}^0R_i {}^i \dot{v}_i + {}^0\omega_i \times {}^0R_i {}^i v_i = {}^0R_i ({}^i \dot{v}_i + {}^i\omega_i \times {}^i v_i)$$



$${}^i a_i = {}^i \dot{v}_i + {}^i\omega_i \times {}^i v_i$$

derivative of a “unit” vector
in a moving frame





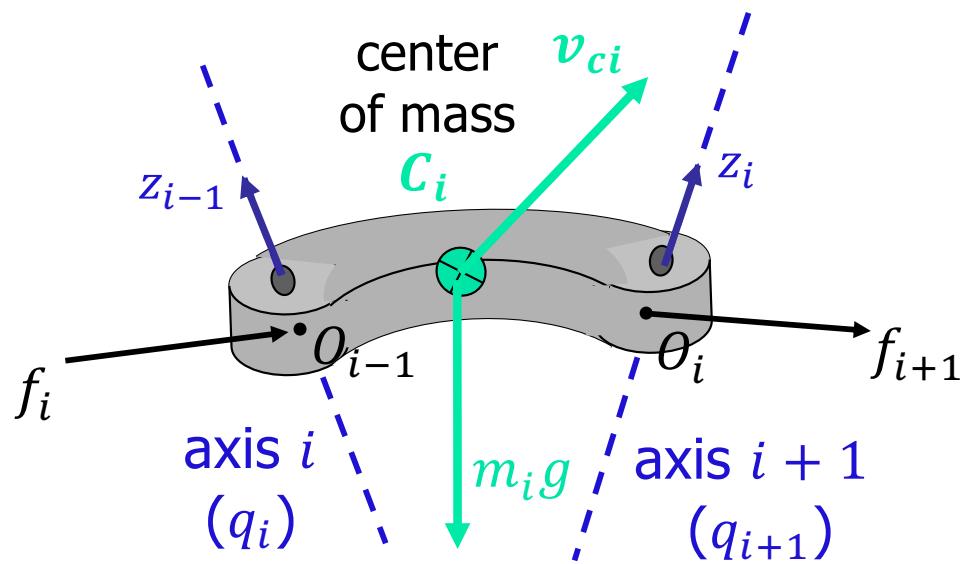
Dynamics of a rigid body

- **Newton** dynamic equation
 - balance: sum of forces = variation of **linear** momentum
$$\sum f_i = \frac{d}{dt} (m v_c) = m \dot{v}_c$$
- **Euler** dynamic equation
 - balance: sum of moments = variation of **angular** momentum
$$\begin{aligned}\sum \mu_i &= \frac{d}{dt} (I \omega) = I \dot{\omega} + \frac{d}{dt} (R \bar{I} R^T) \omega = I \dot{\omega} + (\dot{R} \bar{I} R^T + R \bar{I} \dot{R}^T) \omega \\ &= I \dot{\omega} + S(\omega) R \bar{I} R^T \omega + R \bar{I} R^T S^T(\omega) \cancel{\omega} = I \dot{\omega} + \omega \times I \omega\end{aligned}$$
- principle of **action and reaction**
 - forces/momenta: applied **by** body *i* to body *i* + 1
= – applied **by** body *i* + 1 **to** body *i*



Newton-Euler equations - 1

link i



FORCES

f_i force applied from link $i - 1$ on link i

f_{i+1} force applied from link i on link $i + 1$

$m_i g$ gravity force

all vectors expressed in the same RF (better in RF_i ...)

Newton equation

$$f_i - f_{i+1} + m_i g = m_i a_{ci}$$

linear acceleration of C_i



Newton-Euler equations - 2

link i

MOMENTS

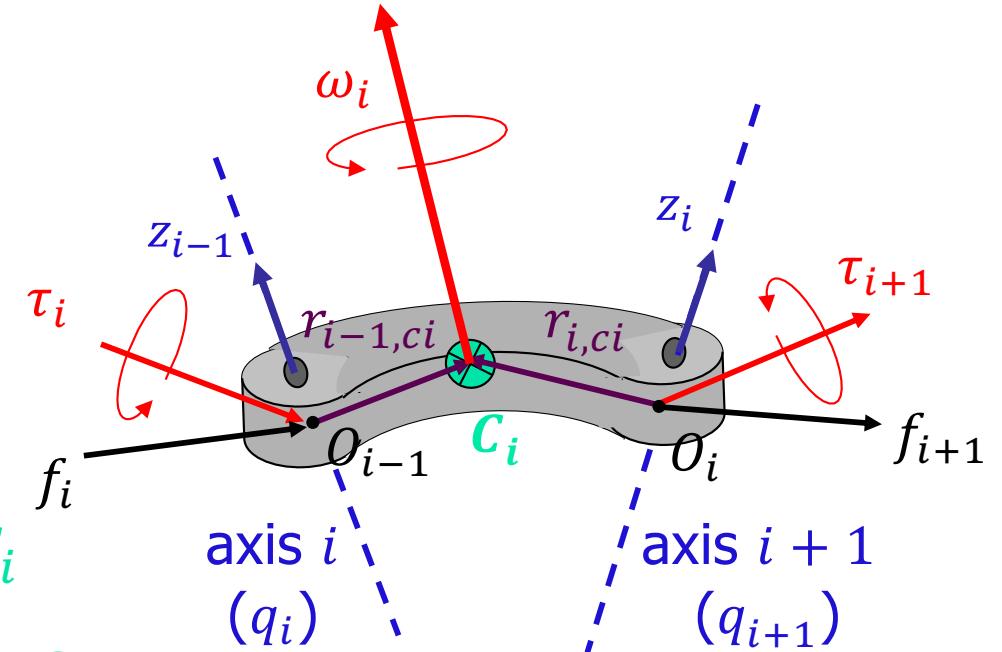
τ_i moment applied
from link $(i - 1)$ on link i

τ_{i+1} moment applied
from link i on link $(i + 1)$

$f_i \times r_{i-1,ci}$ moment due to f_i w.r.t. C_i

$-f_{i+1} \times r_{i,ci}$ moment due to $-f_{i+1}$ w.r.t. C_i

Euler equation



gravity force gives
no moment at C_i

all vectors expressed in
the same RF (... RF_i !!)

$$\tau_i - \tau_{i+1} + f_i \times r_{i-1,ci} - f_{i+1} \times r_{i,ci} = I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i)$$

E

angular acceleration of body i



Forward recursion

Computing velocities and accelerations

- “moving frames” algorithm (as for velocities in Lagrange)
- for simplicity, only revolute joints here
(see **textbook** for the more general treatment)

initializations

$${}^i\omega_i = {}^{i-1}R_i^T [{}^{i-1}\omega_{i-1} + \dot{q}_i {}^{i-1}z_{i-1}]$$

${}^0\omega_0$

$${}^i\dot{\omega}_i = {}^{i-1}R_i^T [{}^{i-1}\dot{\omega}_{i-1} + \ddot{q}_i {}^{i-1}z_{i-1}] + {}^{i-1}\dot{R}_i^T [{}^{i-1}\omega_{i-1} + \dot{q}_i {}^{i-1}z_{i-1}]$$

$$\text{AR} = {}^{i-1}R_i^T [{}^{i-1}\dot{\omega}_{i-1} + \ddot{q}_i {}^{i-1}z_{i-1} + \dot{q}_i {}^{i-1}\omega_{i-1} \times {}^{i-1}z_{i-1}]$$

${}^0\dot{\omega}_0$

$${}^i a_i = {}^{i-1}R_i^T {}^{i-1}a_{i-1} + {}^i\dot{\omega}_i \times {}^i r_{i-1,i} + {}^i\omega_i \times ({}^i\omega_i \times {}^i r_{i-1,i})$$

${}^0a_0 - {}^0g$

$${}^i a_{ci} = {}^i a_i + {}^i\dot{\omega}_i \times {}^i r_{i,ci} + {}^i\omega_i \times ({}^i\omega_i \times {}^i r_{i,ci})$$

the gravity force term can be skipped in Newton equation, if added here



Backward recursion

Computing forces and moments

from $N_i \rightarrow$ to N_{i-1} eliminated, if inserted
in forward recursion ($i=0$)

initializations

$${}^i f_i = {}^i R_{i+1} {}^{i+1} f_{i+1} + m_i ({}^i a_{ci} - \cancel{{}^i g}) \quad \leftarrow f_{N+1} \quad \tau_{N+1}$$

F/MR

$$\begin{aligned} {}^i \tau_i = & {}^i R_{i+1} {}^{i+1} \tau_{i+1} + ({}^i R_{i+1} {}^{i+1} f_{i+1}) \times {}^i r_{i,ci} - {}^i f_i \times ({}^i r_{i-1,i} + {}^i r_{i,ci}) \\ & + {}^i I_i {}^i \dot{\omega}_i + {}^i \omega_i \times {}^i I_i {}^i \omega_i \end{aligned}$$

from $E_i \rightarrow$ to E_{i-1}

at each recursion step, the two vector equations ($N_i + E_i$) at joint i provide a wrench $(f_i, \tau_i) \in \mathbb{R}^6$: this contains ALSO **reaction forces/moment**s at the joint axis \Rightarrow to be “projected” along/around this axis to produce **work**

FP

$$u_i = \begin{cases} {}^i f_i^T {}^i z_{i-1} + F_{vi} \dot{q}_i \\ {}^i \tau_i^T {}^i z_{i-1} + F_{vi} \dot{q}_i \end{cases}$$

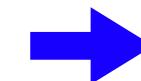
generalized forces

(in rhs of Euler-Lagrange eqs)

for prismatic joint

for revolute joint

add any dissipative term
(here, viscous friction only)



**N scalar
equations**
at the end



Comments on Newton-Euler method

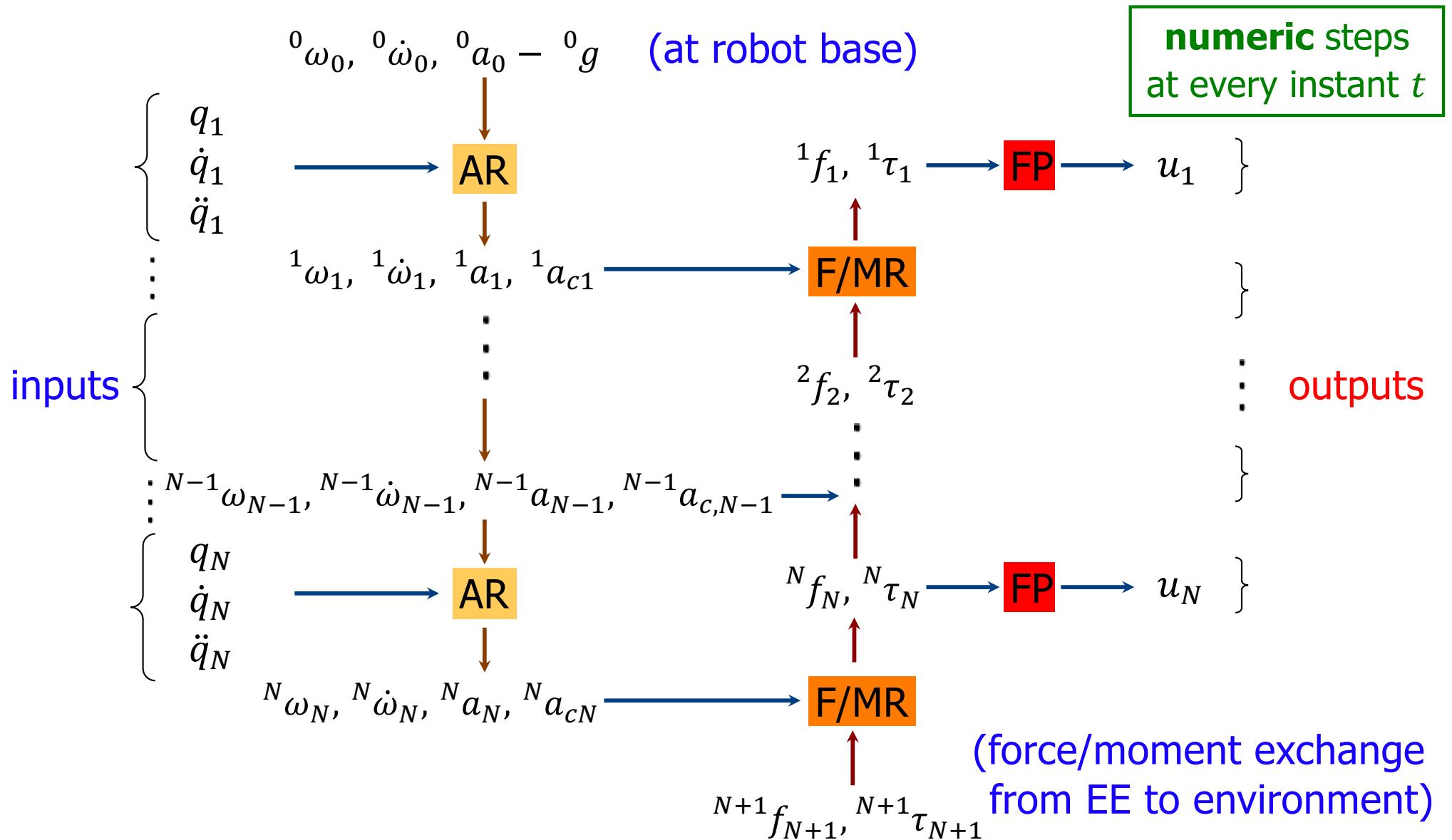
- the previous forward/backward recursive formulas can be evaluated in symbolic or numeric form
 - **symbolic**
 - substituting expressions in a recursive way
 - at the end, a closed-form dynamic model is obtained, which is identical to the one obtained using Euler-Lagrange (or any other) method
 - there is **no** special convenience in using N-E in this way ...
 - **numeric**
 - substituting numeric values (numbers!) at each step
 - **computational complexity** of each step remains constant ⇒ grows **in a linear fashion** with the number **N** of joints (**$O(N)$**)
 - strongly recommended for real-time use, especially when the number **N** of joints **is large**

In the lagrange method you have to derive the model for the specific robot, here you just use the numerical tool that is general for each robot



Newton-Euler algorithm

efficient computational scheme for inverse dynamics





Matlab (or C) script

general routine $NE_\alpha(\arg_1, \arg_2, \arg_3)$

- data file (of a specific robot)
 - number N and types $\sigma = \{0,1\}^N$ of joints (revolute/prismatic)
 - table of DH kinematic parameters
 - list of **ALL** dynamic parameters of the links (and of the motors)
- input
 - vector parameter $\alpha = \{{}^0g, 0\}$ (presence or absence of gravity)
 - three ordered **vector arguments**
 - typically, samples of joint **position, velocity, acceleration** taken from a desired trajectory
- output
 - generalized force u for the **complete** inverse dynamics
 - ... or **single terms** of the dynamic model

assuming **no** interaction
with the environment
 $(f_{N+1} = \tau_{N+1} = 0)$



Examples of output

- complete inverse dynamics

$$u = NE^0 g(q_d, \dot{q}_d, \ddot{q}_d) = M(q_d)\ddot{q}_d + c(q_d, \dot{q}_d) + g(q_d) = u_d$$

- gravity term

$$u = NE^0 g(q, 0, 0) = g(q)$$

- centrifugal and Coriolis term

$$u = NE_0(q, \dot{q}, 0) = c(q, \dot{q})$$

- i -th column of the inertia matrix

$$u = NE_0(q, 0, e_i) = M_i(q)$$

e_i = i -th column
of identity matrix

- generalized momentum

$$u = NE_0(q, 0, \dot{q}) = M(q)\dot{q} = p$$



A further example of output

- factorization of centrifugal and Coriolis term

$$u = NE_0(q, \dot{q}, 0) = c(q, \dot{q}) = S(q, \dot{q})\dot{q}$$

- for later use, what about a “mixed” velocity term?

$$S(q, \dot{q})\dot{q}_r \Leftrightarrow \begin{cases} u = NE_0(q, \dot{q}_r, 0) = S(q, \dot{q}_r)\dot{q}_r \\ u = NE_0(q, e_i \dot{q}_{ri}, 0) = S_i(q, e_i \dot{q}_{ri})\dot{q}_{ri} \end{cases} \text{ no good!}$$

- $S(q, \dot{q})\dot{q}_r = S(q, \dot{q}_r)\dot{q}$, when using Christoffel symbols
- $S(q, \dot{q} + \dot{q}_r)(\dot{q} + \dot{q}_r) = S(q, \dot{q})\dot{q} + S(q, \dot{q}_r)\dot{q}_r + 2S(q, \dot{q})\dot{q}_r$

$$\Rightarrow u = \frac{1}{2}(NE_0(q, \dot{q} + \dot{q}_r, 0) - NE_0(q, \dot{q}, 0) - NE_0(q, \dot{q}_r, 0)) \\ = S(q, \dot{q})\dot{q}_r \quad (\text{i.e., with 3 calls of standard NE algorithm})$$

[Kawasaki et al., IEEE T-RA 1996]



Modified NE algorithm

modified routine $\widehat{NE}_\alpha(\arg_1, \arg_2, \arg_3, \arg_4)$ with 4 arguments

[De Luca, Ferrajoli, ICRA 2009]

$$\widehat{NE}_\alpha(x, y, y, z) = NE_\alpha(x, y, z) \quad \text{consistency property}$$

e.g., $u = \widehat{NE} \circ g(q, 0, 0, 0) = NE \circ g(q, 0, 0) = g(q)$

$$u = \widehat{NE}_0(q, \dot{q}, \dot{q}, 0) = NE_0(q, \dot{q}, 0) = c(q, \dot{q}) = S(q, \dot{q})\dot{q}$$

$\Rightarrow u = \widehat{NE}_0(q, \dot{q}, \dot{q}_r, 0) = S(q, \dot{q})\dot{q}_r$ with $\dot{M} - 2S$ skew-symmetric

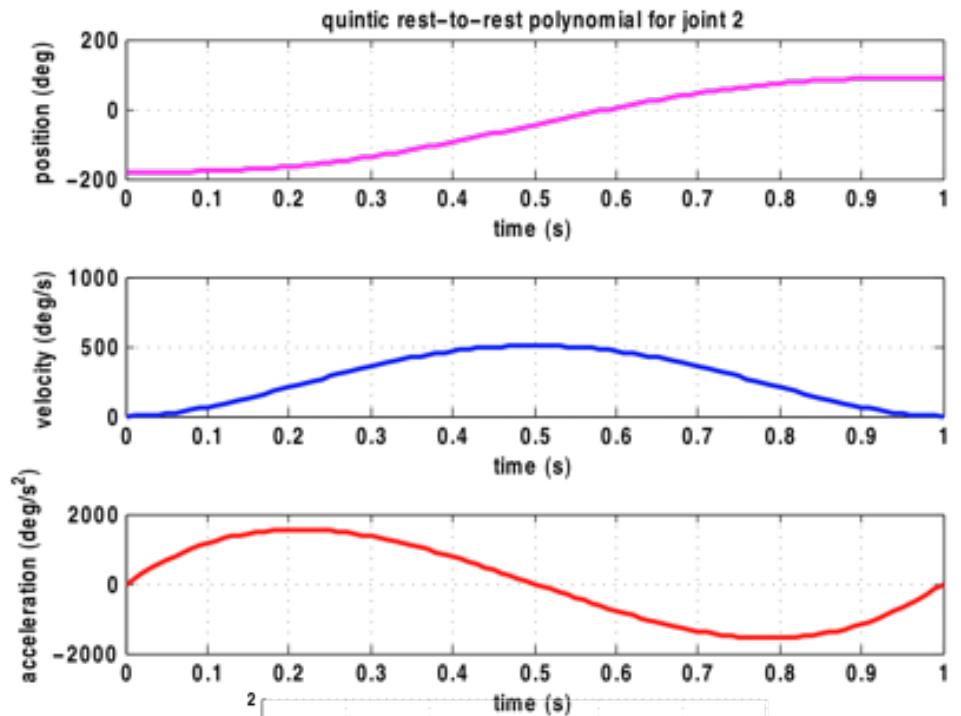
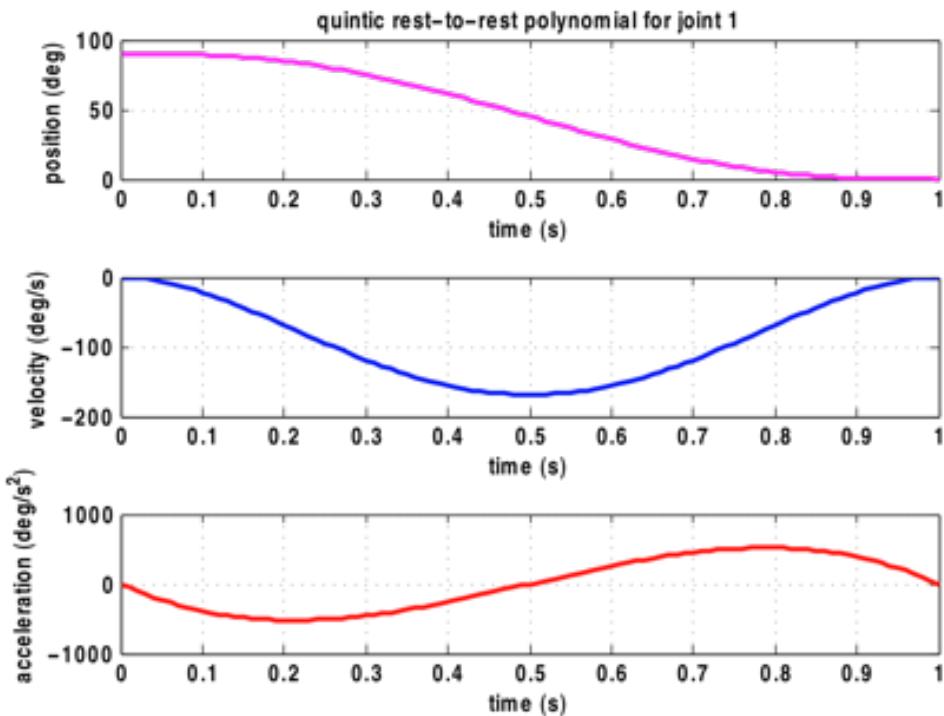
(i.e., with 1 call of modified NE algorithm)

$\Rightarrow u = \widehat{NE}_0(q, \dot{q}, e_i, 0) = S_i(q, \dot{q})$

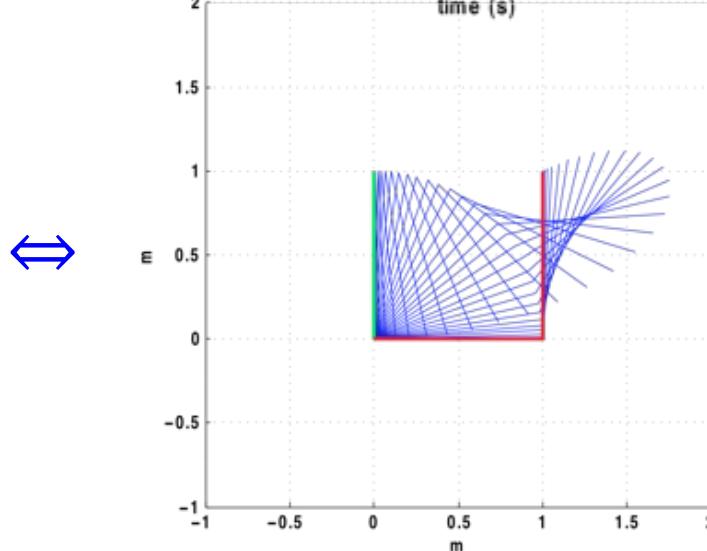
(i.e., the full matrix S satisfying the skew-symmetry of $\dot{M} - 2S$ with N calls of the modified NE algorithm)



Inverse dynamics of a 2R planar robot



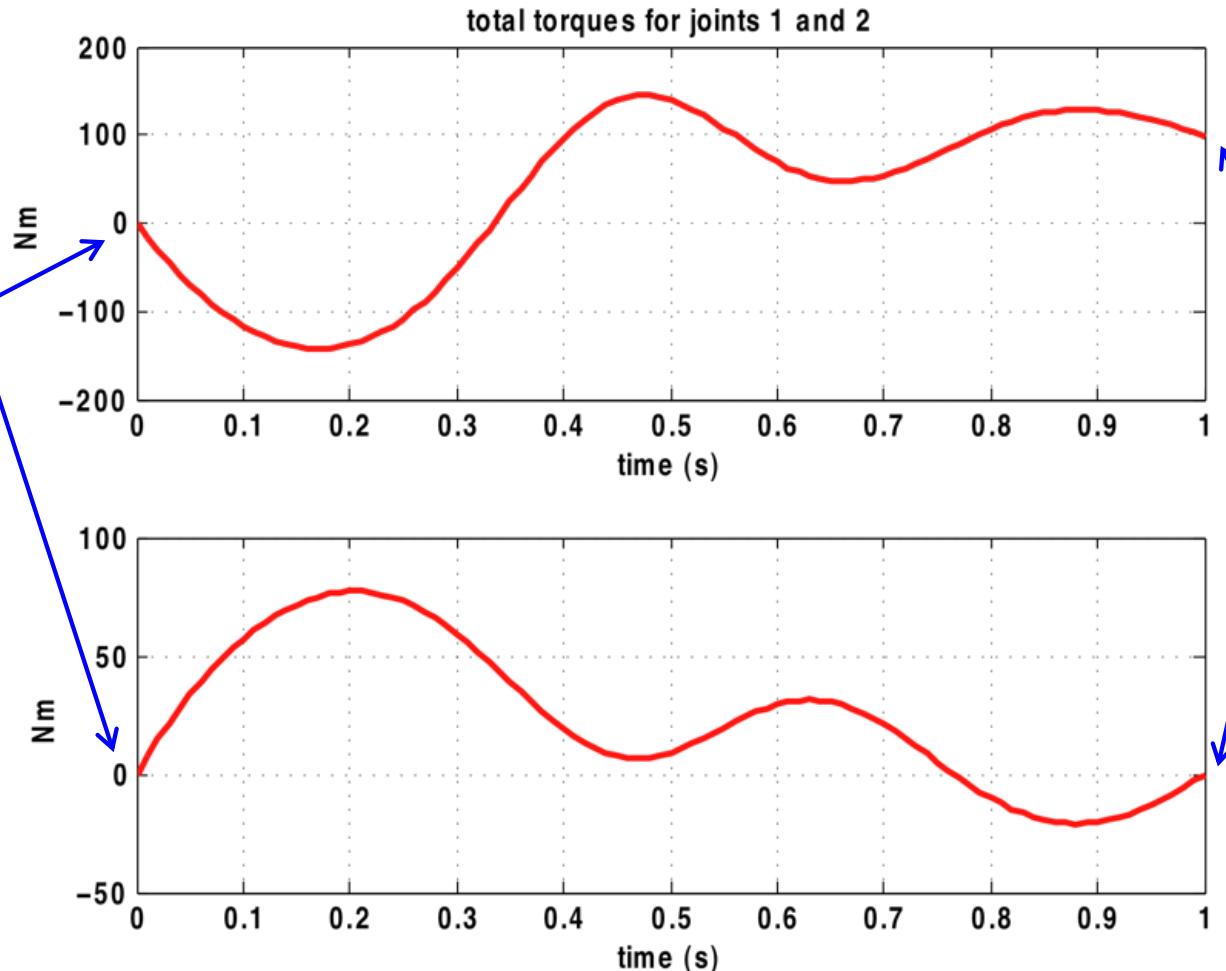
desired (smooth) joint motion:
quintic polynomials for q_1, q_2 with
zero vel/acc boundary conditions
from $(90^\circ, -180^\circ)$ to $(0^\circ, 90^\circ)$ in $T = 1$ s





Inverse dynamics of a 2R planar robot

zero
initial torques =
free equilibrium
configuration
+
zero initial
accelerations



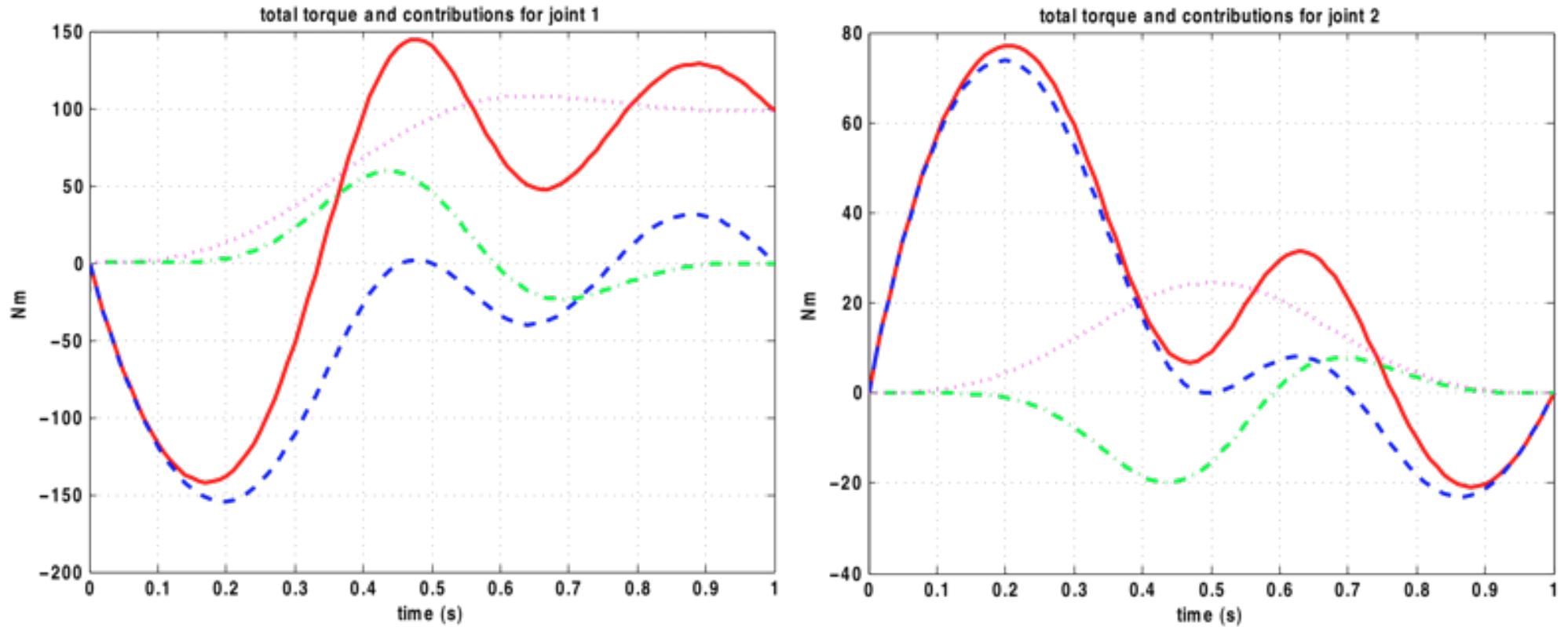
final torques
 $u_1 \neq 0, u_2 = 0$
balance
link weights
in final ($0^\circ, 90^\circ$)
configuration

motion in vertical plane (under gravity)

both links are thin rods of uniform mass $m_1 = 10 \text{ kg}$, $m_2 = 5 \text{ kg}$



Inverse dynamics of a 2R planar robot



torque contributions at the two joints for the desired motion

— = total, --- = inertial
- - - = Coriolis/centrifugal, = gravitational



Use of NE routine for simulation

direct dynamics

- numerical integration, at **current** state (q, \dot{q}) , of
$$\ddot{q} = M^{-1}(q)[u - (c(q, \dot{q}) + g(q))] = M^{-1}(q)[u - n(q, \dot{q})]$$

- Coriolis, centrifugal, and gravity terms

$$n = NE^o_g(q, \dot{q}, 0) \quad \text{complexity } O(N)$$

- i -th column of the inertia matrix, for $i = 1, \dots, N$

$$M_i = NE_0(q, 0, e_i) \quad O(N^2)$$

- numerical inversion of inertia matrix

$$InvM = \text{inv}(M) \quad O(N^3)$$

but with small coefficient

- given u , integrate acceleration computed as

$$\ddot{q} = InvM * [u - n] \quad \rightarrow \quad \begin{array}{l} \text{new state } (q, \dot{q}) \\ \text{and repeat over time ...} \end{array}$$



Robotics 2

Introduction to Control

Prof. Alessandro De Luca

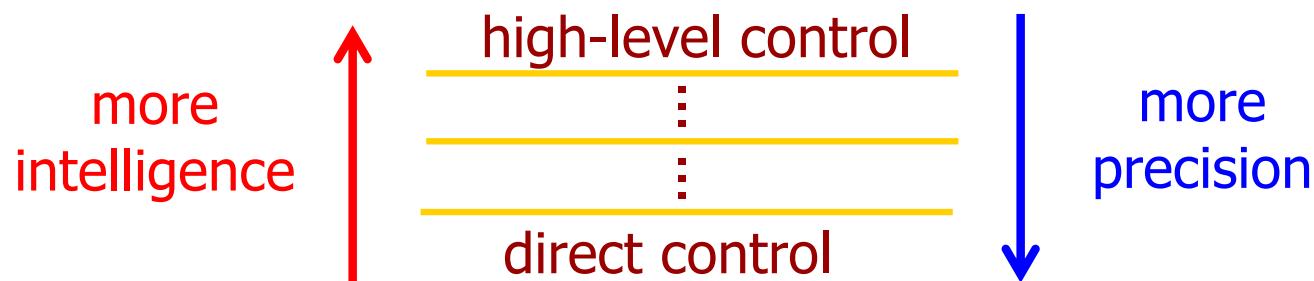
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





What do we mean by robot control?

- different **level** of definitions may be given to robot control
 - successfully complete a **task** or **work program**
 - accurate execution of a **motion trajectory**
 - zeroing a **positioning error**
- ⇒ control system unit has a **hierarchical** internal structure



- different time scales in the various control levels: lowest $\leq 1 \text{ ms}$, higher levels up to **seconds**
- different but cooperating models, objectives, methods



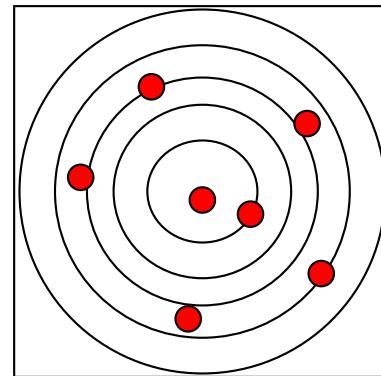
Evaluation of control performance

- **quality** of execution in **nominal** conditions
 - velocity/speed of task completion
 - accuracy/repeatability (in **static** and **dynamic** terms)
 - energy requirements
 - ⇒ improvements also thanks to **models** (software!)
- **robustness** in **perturbed/uncertain** conditions
 - adaptation to changing environments
 - high repeatability despite disturbances, changes of parameters, uncertainties, modeling errors
 - ⇒ can be improved by a generalized use of **feedback**, using more **sensor information**
 - ⇒ **learn** through repeated robot trials/human experience

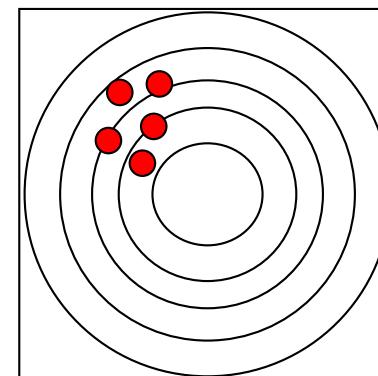


Static positioning accuracy and repeatability

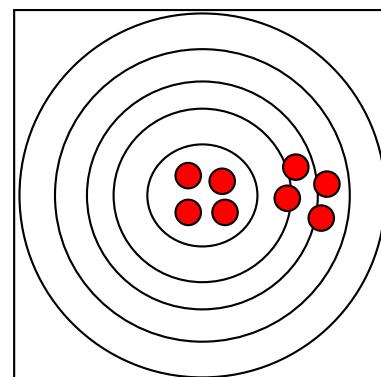
poor accuracy
poor repeatability



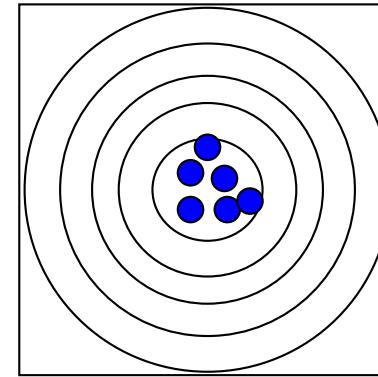
poor accuracy
good repeatability



good accuracy
poor repeatability



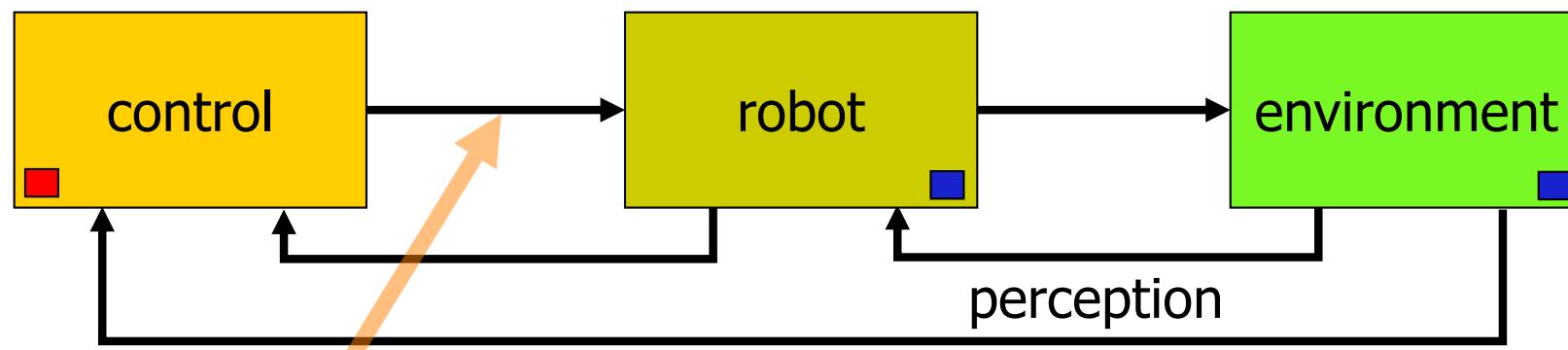
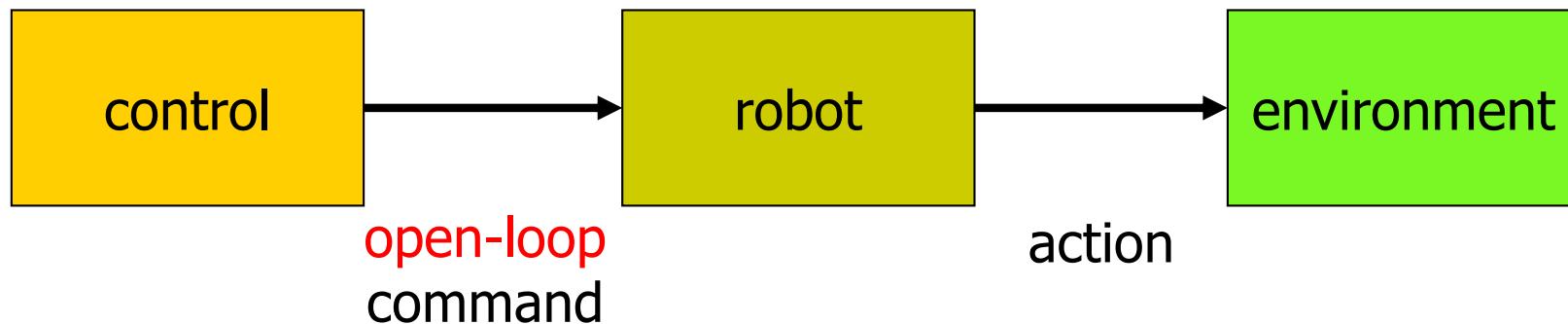
good accuracy
good repeatability



what about "dynamic" accuracy on (test or selected) motion trajectories?



Basic control schemes



combination of
feedforward and
feedback commands

closed-loop commands

■ METHODS ■ MODELS



Control schemes and uncertainty

- **feedback control**
 - insensitivity to mild disturbances, small variations of parameters, and different initial conditions
- **robust control**
 - tolerates relatively large uncertainties of known range
- **adaptive control**
 - improves performance online, adapting the control law to unknown range of uncertainties and/or large (but slow) parameter variations
- **intelligent control**
 - performance improved based on trials/experience: **LEARNING**
 - autonomous search and change of internal structure for optimizing system behavior: **SELF-ORGANIZING**

uncertainty on parametric values



IDENTIFICATION

... on the system structure



...



Limits in control of industrial robots - 1

- from a **functional** viewpoint
 - “closed” control architectures, relatively difficult to interface with external programs and sensing devices for **hard real-time** operation
 - need of some expertise for programming and handling of exceptions
 - ⇒ introducing easy/more intuitive user (multi-modal) interfaces
- at the **higher** level
 - open-loop task command generation
 - ⇒ exteroceptive sensory feedback absent or very loose, with low capability of autonomous reasoning
- at the **intermediate** level
 - limited consideration of advanced kinematic and dynamic issues
 - ⇒ e.g., singularity robustness: solved on a case-by-case basis
 - ⇒ task redundancy: no automatic use of the extra degrees of freedom



Limits in control of industrial robots - 2

- at the **lower (direct)** level
 - reduced execution speed ("control bandwidth")
 - ⇒ typically, heavy mechanical structures
 - reduced dynamic accuracy on fast motion trajectories
 - ⇒ standard: use of kinematic control + PID only
 - problems with dry friction and backlash at the joints
 - compliance in the robot structure
 - ⇒ flexible transmissions
(belts, harmonic drives, long shafts)
 - ⇒ large structures or relatively lightweight links

now **desired**
for safe
physical
Human-Robot
Interaction



- need to include better **dynamic models** and model-based **control laws**
- handled, e.g., using **direct-drive** actuators or online friction **compensation**



Example of robot positioning

- low damped vibrations due to joint elasticity



without modeling
and explicit
control of
joint elasticity

- 6R KUKA KR-15/2 robot (235 kg), with 15 kg payload



Advanced robot control laws

- deeper mathematical/physical analysis and modeling of robot components (**model-based** approach)
- schemes using various control loops at different/multiple hierarchical levels (**feedback**) and with additional sensors
 - visual servoing
 - force/torque sensors for interaction control
 - ...
- “new” methods
 - integration of (open-loop/feedforward) **motion planning** and **feedback control** aspects (e.g., sensor-based planning)
 - fast (sensor-based) re-planning
 - model predictive control (with preview)
 - **learning** (iterative, by imitation, skill transfer, ...)
 - ...



Example of visual-based control

- human-obstacle collision avoidance



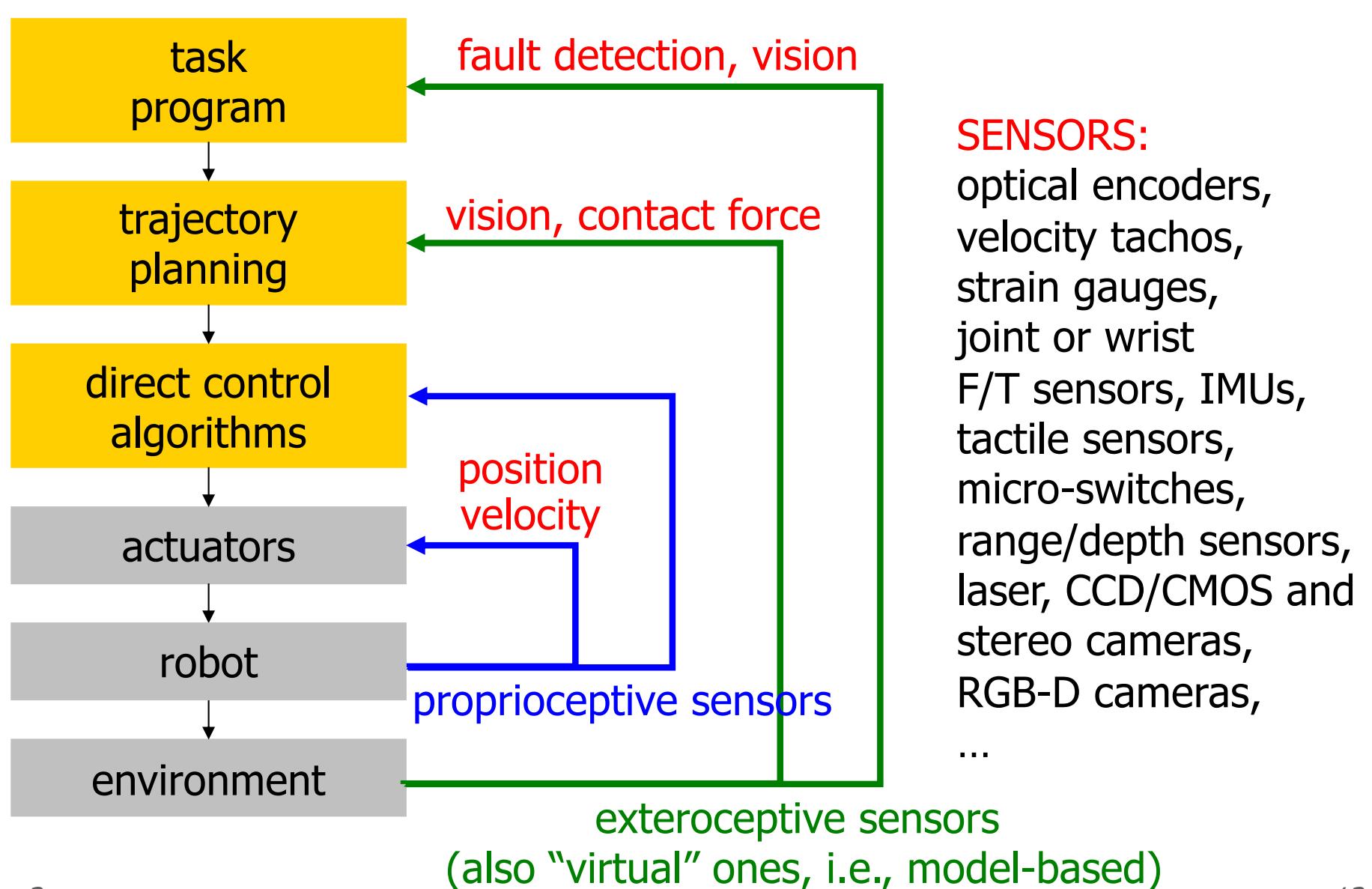
video

- 3R SoftArm prototype with McKibben actuators (Univ. of Pisa) using **repulsive force field** built from stereo camera information



Functional structure of a control unit

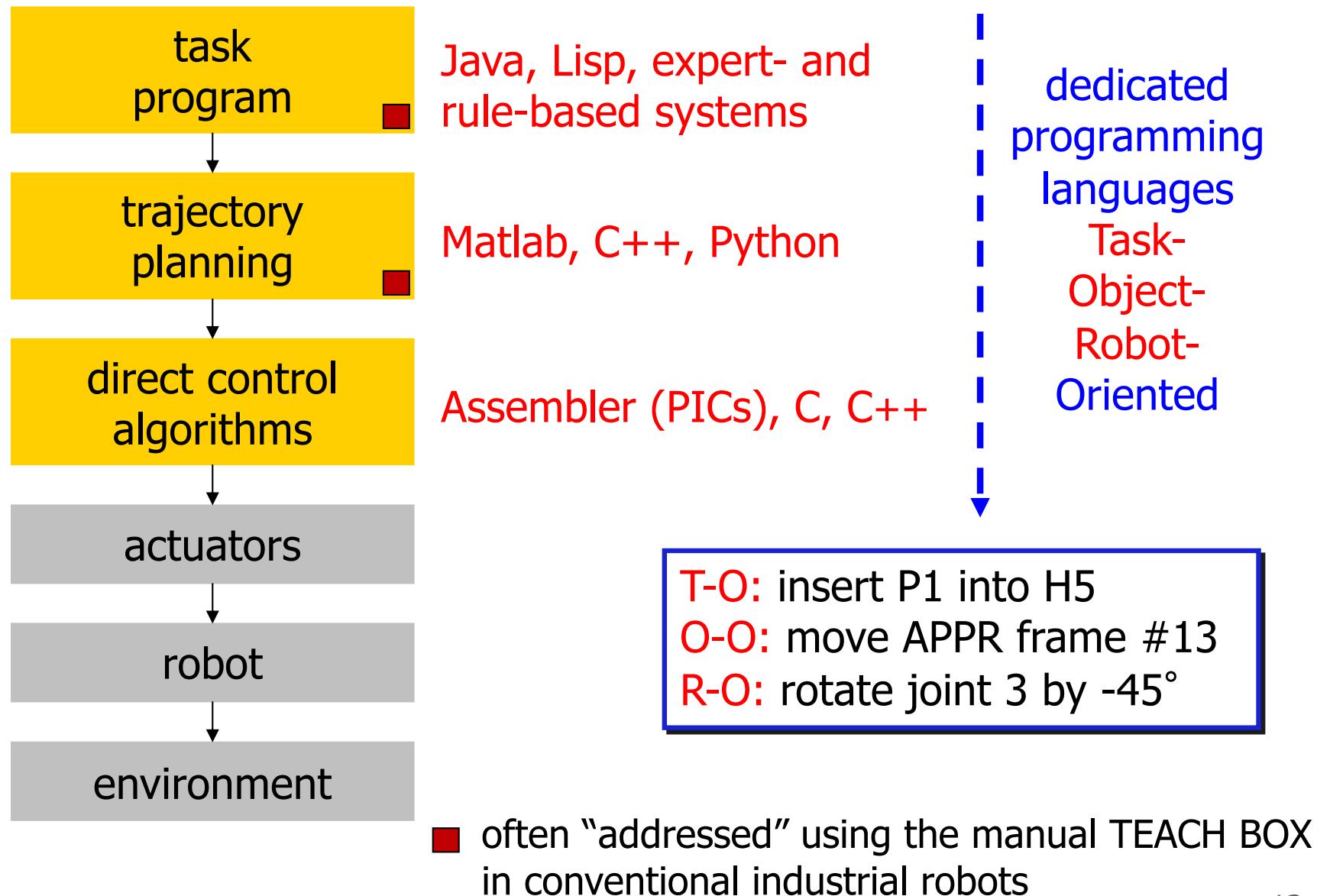
sensor measurements





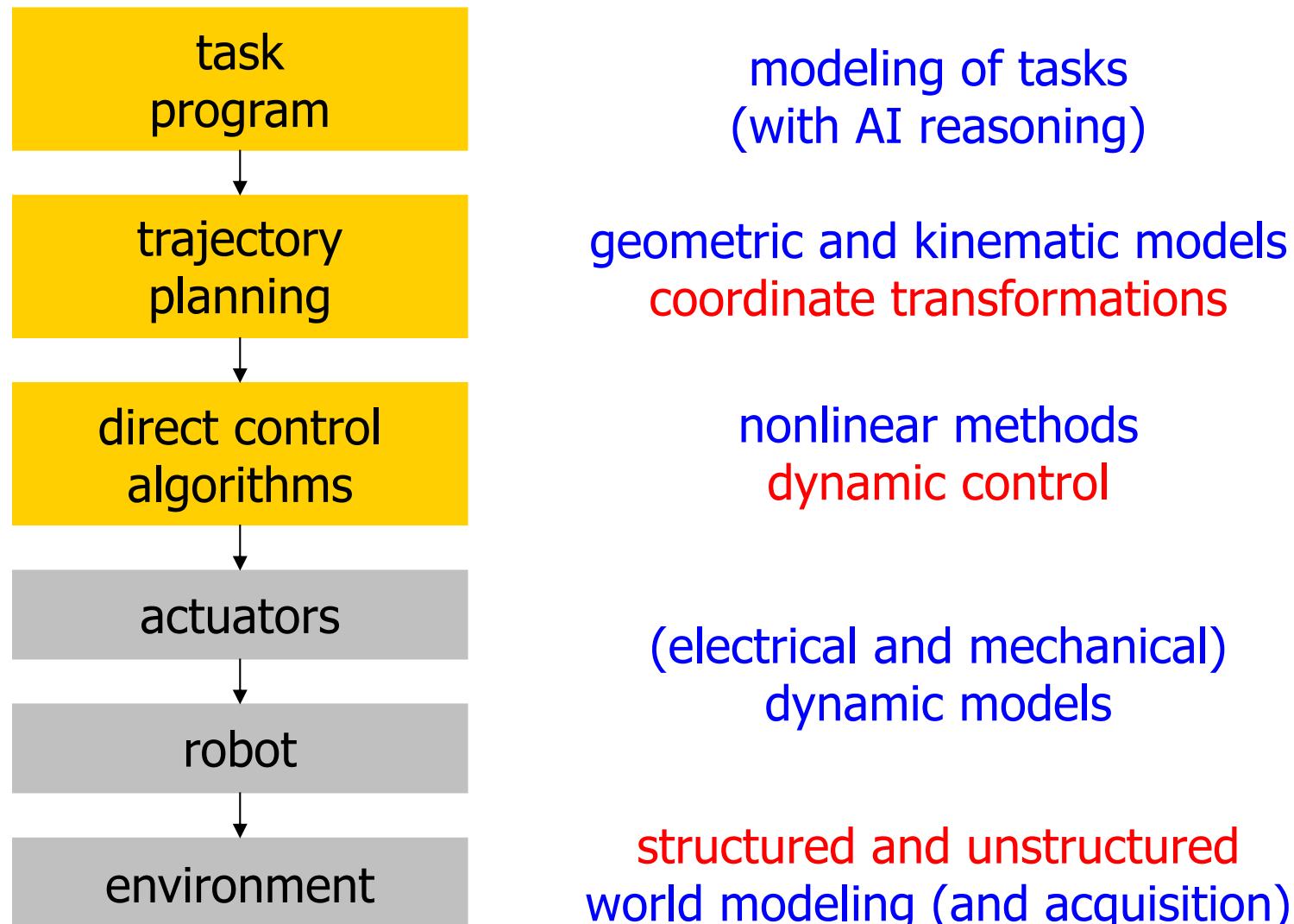
Functional structure of a control unit

programming languages



Functional structure of a control unit

modeling issues

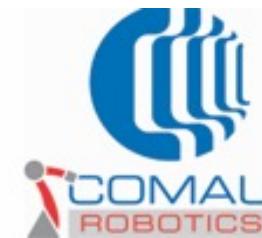




Industrial robot programming languages

- ABB Rapid
- COMAU PDL2
- FANUC Karel
- KUKA KRL
- MITSUBISHI MELFA
- UNIVERSAL ROBOTS RoboDK
- ...

ABB



FANUC

KUKA

MITSUBISHI

 **UNIVERSAL ROBOTS**

Robot control/research software

(last updated in April 2024)



- a (partial) list of **open source** robot software
 - for simulation and/or real-time control
 - for interfacing with devices and sensors
 - research oriented

Player/Stage playerstage.sourceforge.net ⇒ github.com/rtv/stage

- **Stage:** in origin, a networked Linux/MacOS X robotics server acting as abstraction layer to support a variety of hardware ⇒ now a 2(.5)D mobile robot standalone simulation environment
- **Gazebo:** 3D robot simulator (**ODE** physics engine and **OpenGL** rendering), now an independent project ⇒ gazebosim.org



CoppeliaSim (was V-REP; edu version available) www.coppeliarobotics.com

- each object/model controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution
- controllers written in C/C++, Python, Java, Matlab, ...





Robot control/research software (cont'd)

Robotics System Toolbox (license for Sapienza)



- tools/algorithms for simulation of kinematics, dynamics, trajectory planning, control of serial manipulators, mobile robots and humanoids
- library of robots, scene and map creation, Gazebo interface ...

QUT Robot Academy petercorke.com



- free software for robotics and for vision; includes the Robotics Toolbox ([release 10](#)) and the Machine Vision Toolbox (release 4) for MATLAB

ROS (Robot Operating System) ros.org



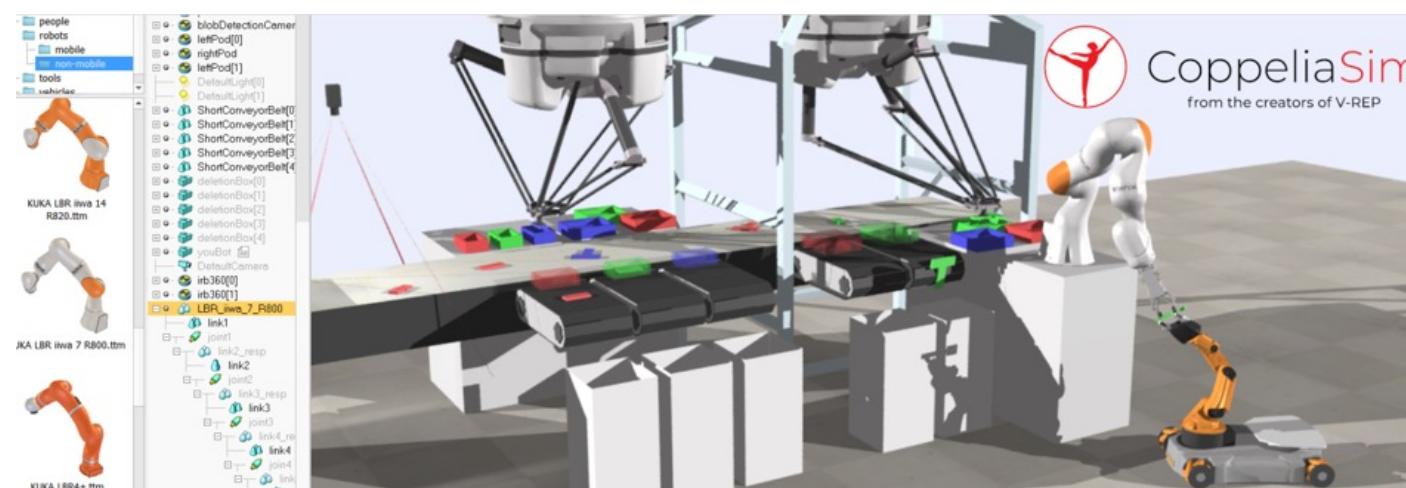
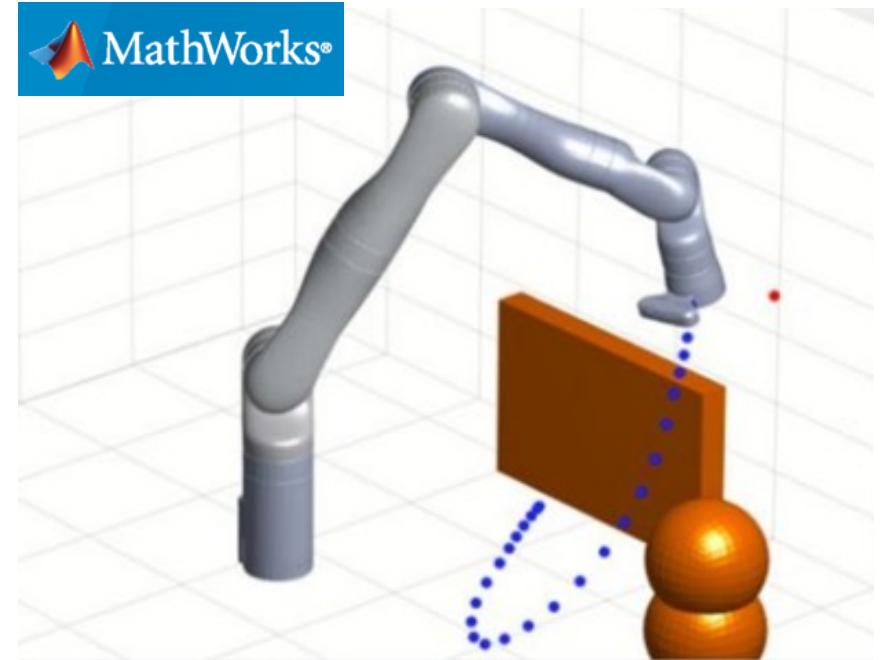
- **middleware** with hardware abstraction, device drivers, libraries, visualizers, message-passing, package management (now **ROS 2**)
- “nodes”: executable code (in Python, C++) running with a publish/subscribe communication style
- drivers, tools, state-of-the-art algorithms ... (all open source)

PyRobotics (Python API) pypi.org/project/pyRobotics (v1.8 in 2015)





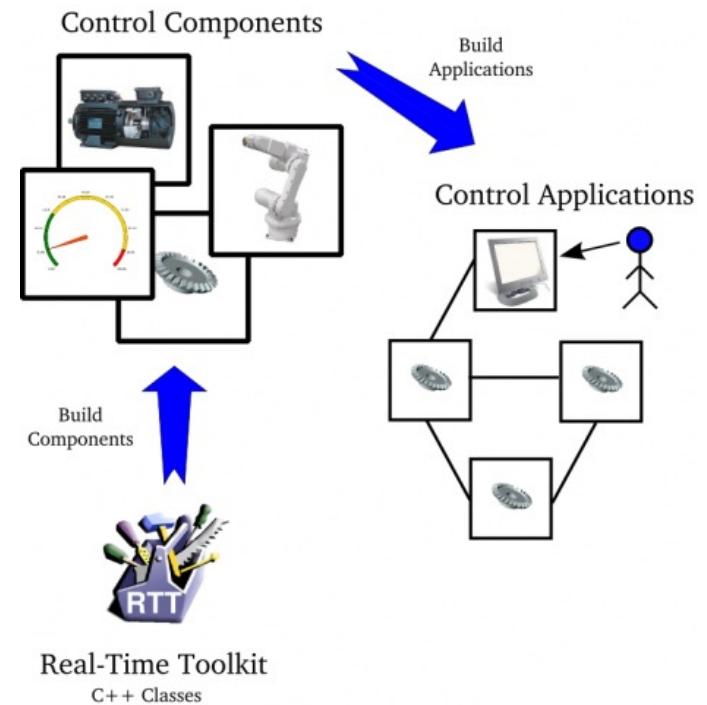
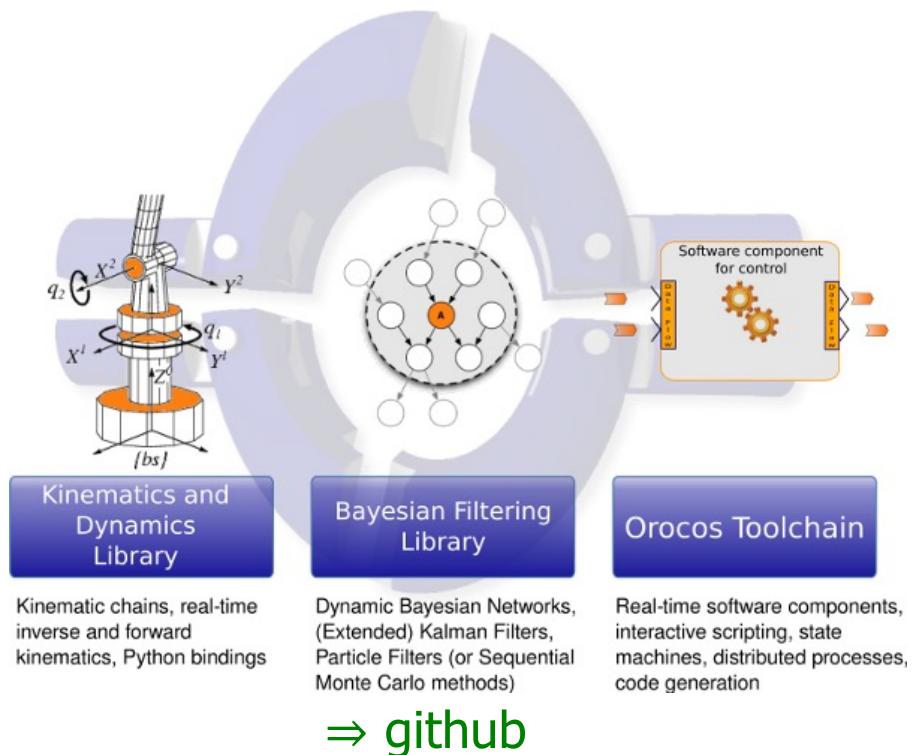
Robot control/research software





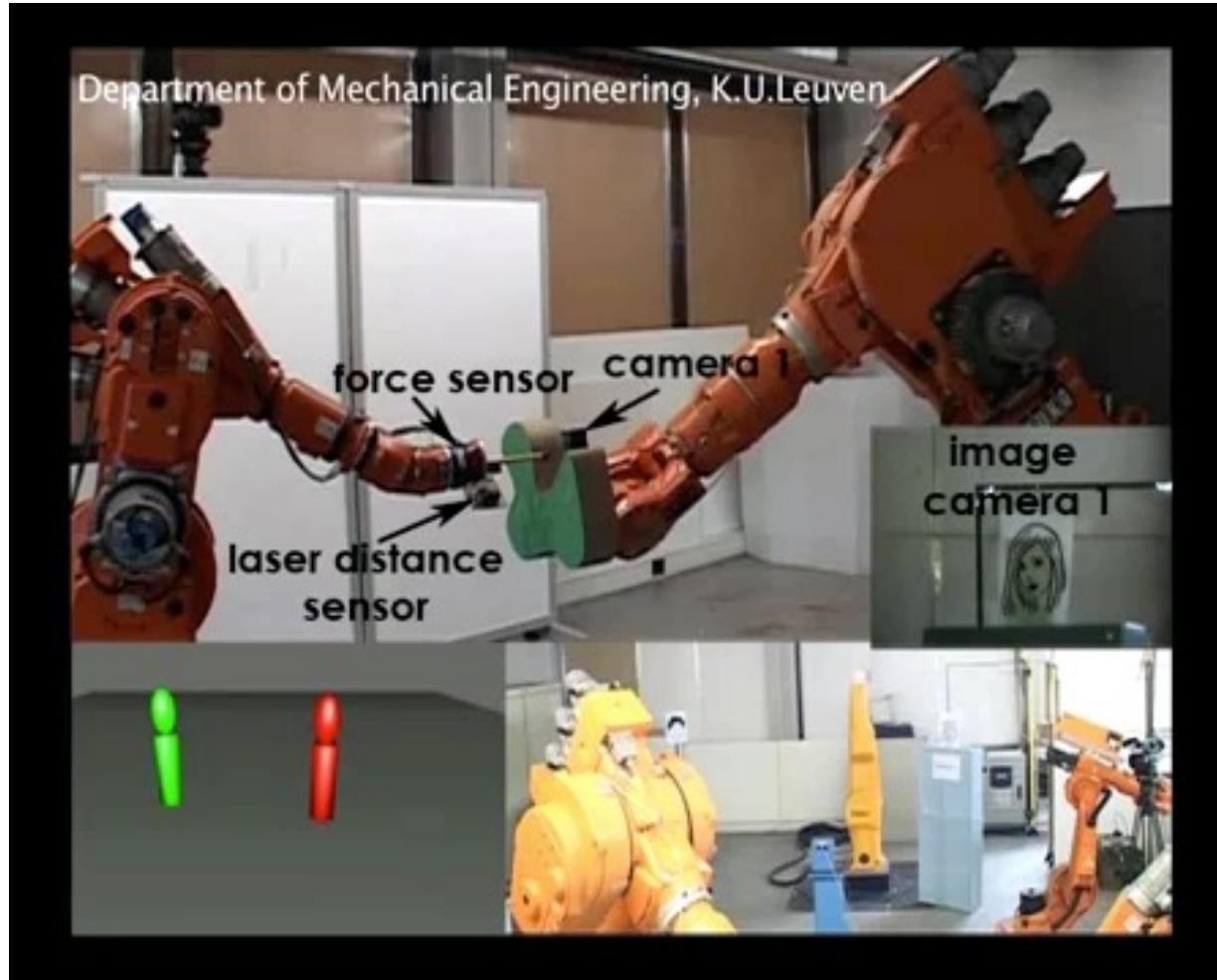
OROCOS control software

- OROCOS (Open RObot COntrol Software) orocos.org
 - open-source, portable C++ libraries for robot control
 - Real-Time Toolkit (for Linux, MacOS X, Windows Visual Studio)
 - supports CORBA for distributed network computing and ROS interface
 - (user-defined) application libraries





Example application using OROCOS



multi-sensor fusion for multi-robot manipulation
in a human populated environment (KU Leuven)



Summarizing ...

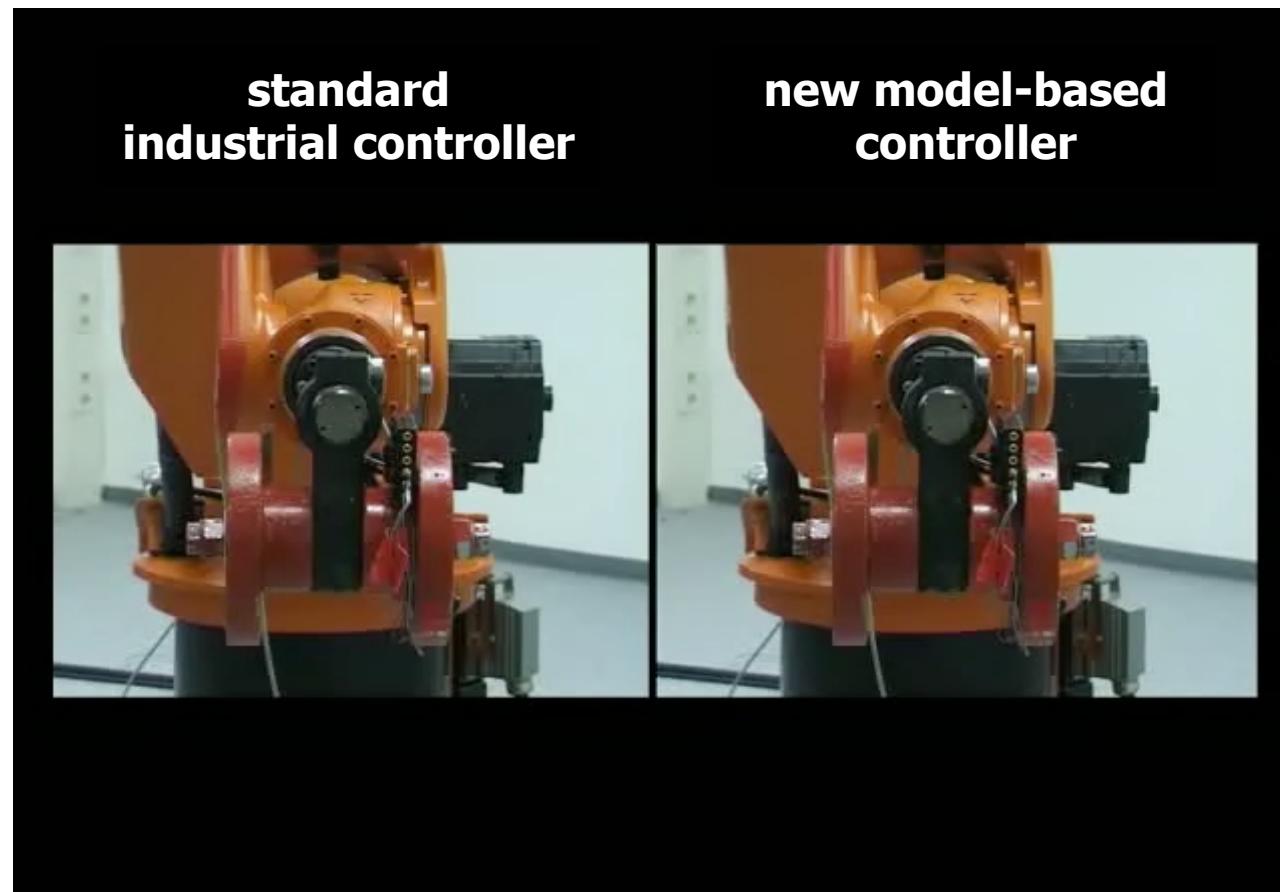
- to **improve performance** of robot controllers
 1. more complete **modeling** (kinematics and **dynamics**)
 2. introduction of **feedback** throughout all hierarchical levels
- **dynamic control** at low level allows in principle
 1. much **higher accuracy** on generic motion trajectories
 2. **larger velocity** in task execution with same accuracy
- interplay between **control, mechanics, electronics**
 1. able to control accurately also **lightweight/compliant** robots
 2. full utilization of task-related **redundancy**
 3. smart **mechanical design** can reduce control efforts (e.g., closed kinematic chains simplifying robot inertia matrix)
 4. **actuators** with higher dynamic performance (e.g., direct drives) and/or including controlled variable stiffness

advanced applications should justify additional costs
(e.g., laser cutting with 10g accelerations, safe human-robot interaction)



Benefits of model-based control

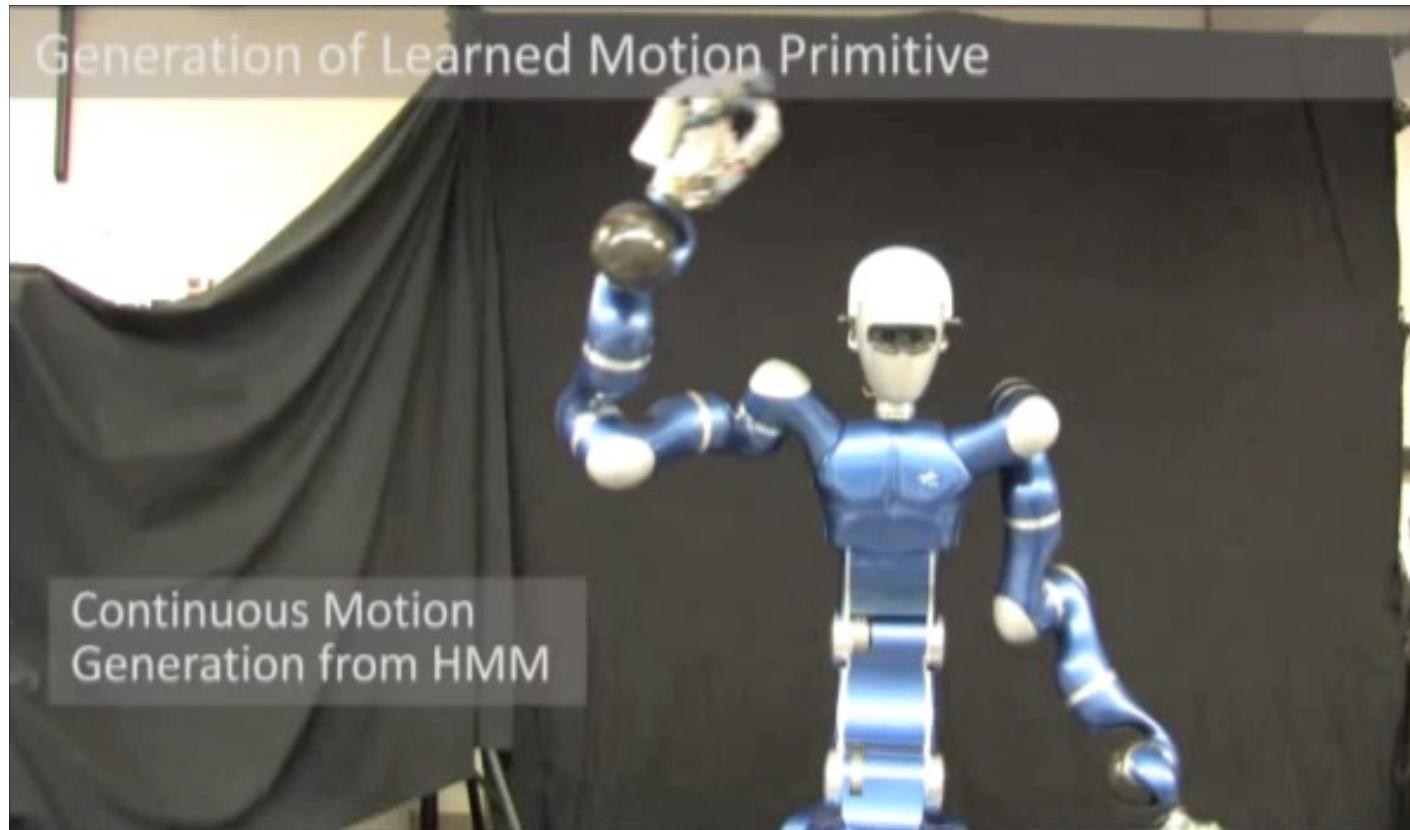
- trajectory tracking task: comparison between standard industrial and new model-based controller





Robot learning by imitation

- learning from human motion primitives (imitation)
- motion refinement by kinesthetic teaching (with impedance control)



@TUM, Munich (D. Lee, C. Ott), for the EU SAPHARI project



Using visual or depth sensor feedback

Stanford University
Artificial Intelligence Laboratory

Robust Visual Servo Control Using
the Reflexxes Motion Libraries

<http://cs.stanford.edu/groups/manips>

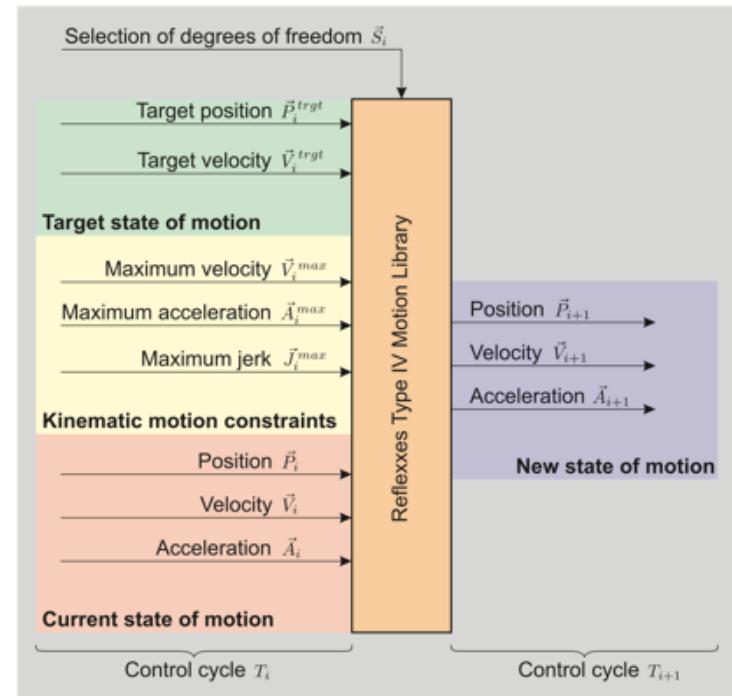
Stanford University
Artificial Intelligence Laboratory

Università di Roma "Sapienza"
Robotics Laboratory

Collision Avoidance Using
the Reflexxes Motion Libraries

video

- robust visual or depth (Kinect) feedback for motion tracking



- collision avoidance schemes
(here, redundancy w.r.t. an E-E task)

video



Panoramic view of control laws

- problems & methods for robot manipulators that will be considered
(control command is always a **joint torque**, if not **else** specified)

type of task	definition of error	joint space	Cartesian space	task space
free motion	means that we don't care how we got from one initial configuration to one desired equilibrium configuration regulation for instance pick and place operations are regulation technique	PD, PID, gravity compensation, iterative learning	PD with gravity compensation	visual servoing (kinematic scheme)
	trajectory tracking	feedback linearization, inverse dynamics + PD, passivity-based control, robust/adaptive control	feedback linearization	
contact motion (with force exchange)		-	impedance control (with variants), admittance control (kinematic scheme)	hybrid force-velocity control

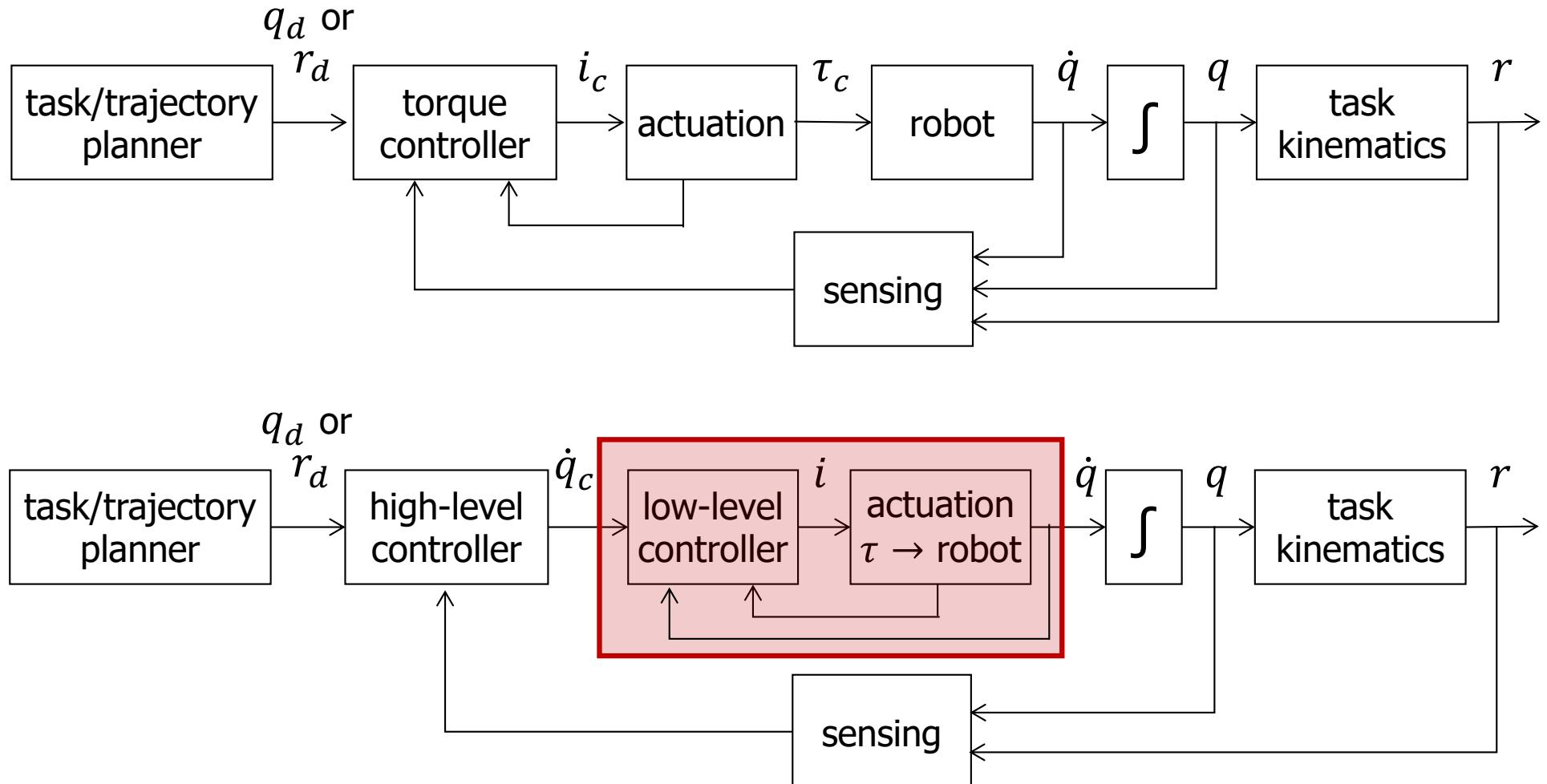


Dynamic or kinematic control laws

- torque-controlled robots
 - issue **current** commands $i = i_c$ (with $\tau_c = K_i i_c$) to drive the (electrical) motors, based on information on the **dynamic** model
 - often, a low-level (analog) current loop is present to enforce the execution of the desired command
 - may use a **torque** measure τ_J (by joint torque sensors) to do the same, in case of joint/transmission elasticity (with $\tau_J = K(\theta - q)$)
 - best suited for high dynamic performance and 'transparent' control of interaction forces
- position/motion-controlled robots
 - issue **kinematic** commands: velocity $\dot{q} = \dot{q}_c$, acceleration $\ddot{q} = \ddot{q}_c$, or their integrated/micro-interpolated version $q = q_c$
 - references for a **low-level** direct loop at high frequency ($T_c \cong 400 \mu\text{s}!$)



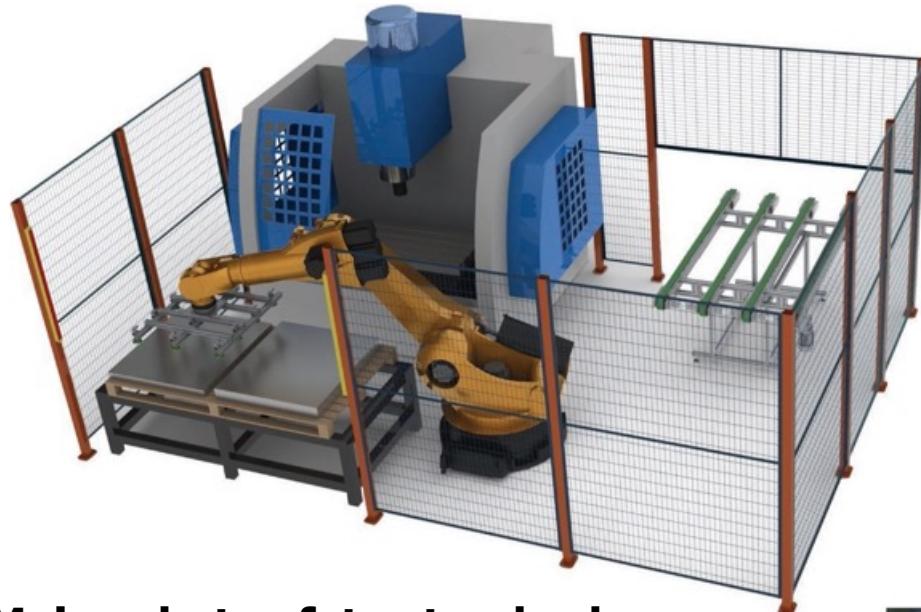
Torque- vs. position-controlled robots



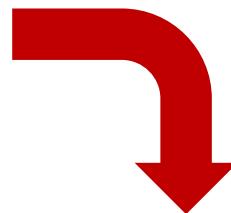
- both modes may be present even in the same robotic system



HRI in industrial settings



non-collaborative robots:
safety fences are required to
prevent harming human operators

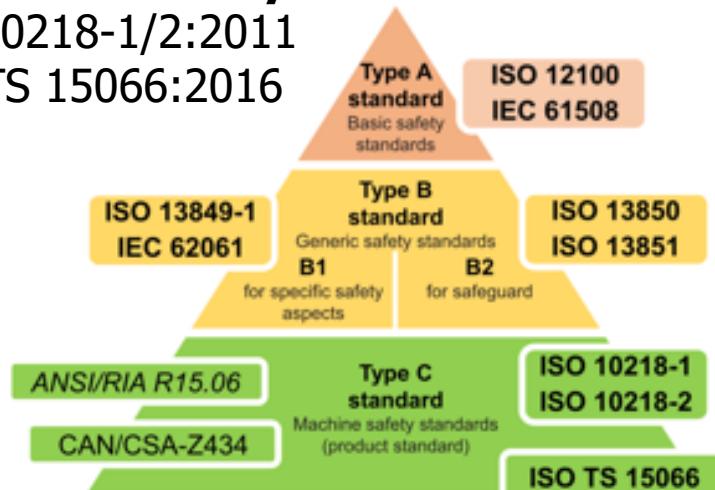


collaborative robots:
allow human workers to
stand in their proximity and
work together on the same task

Main robot safety standards

ISO 10218-1/2:2011

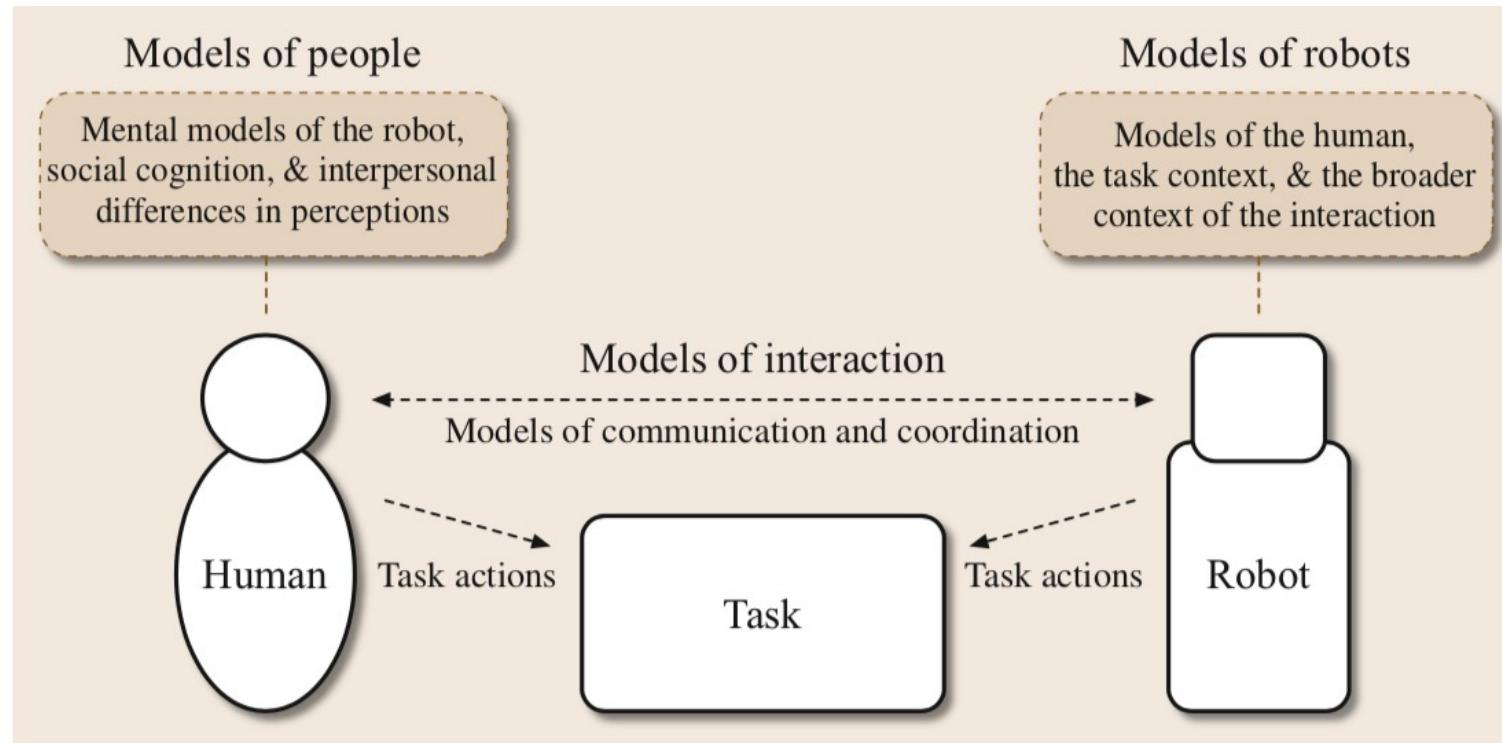
ISO/TS 15066:2016





Human-Robot Interaction taxonomy

- **cognitive** (cHRI) vs. **physical** (pHRI) Human-Robot Interaction
- cHRI models of humans, of robots, and of the interaction itself
 - dialog-based, intention- and activity-based, simulation-theoretic models



B. Mutlu, N. Roy, S. Sabanovic: *Ch. 71, Springer Handbook of Robotics, 2016*



Human-Robot Interaction taxonomy

- pHRI planned and controlled robot behaviors: 3-layer architecture

Safety

lightweight mechanical design
compliance at robot joints

**collision detection
and safe reaction**

Coexistence

robot and human sharing
the same workspace

collision avoidance
no need of physical contact

Collaboration

contactless, e.g., gestures
or voice commands

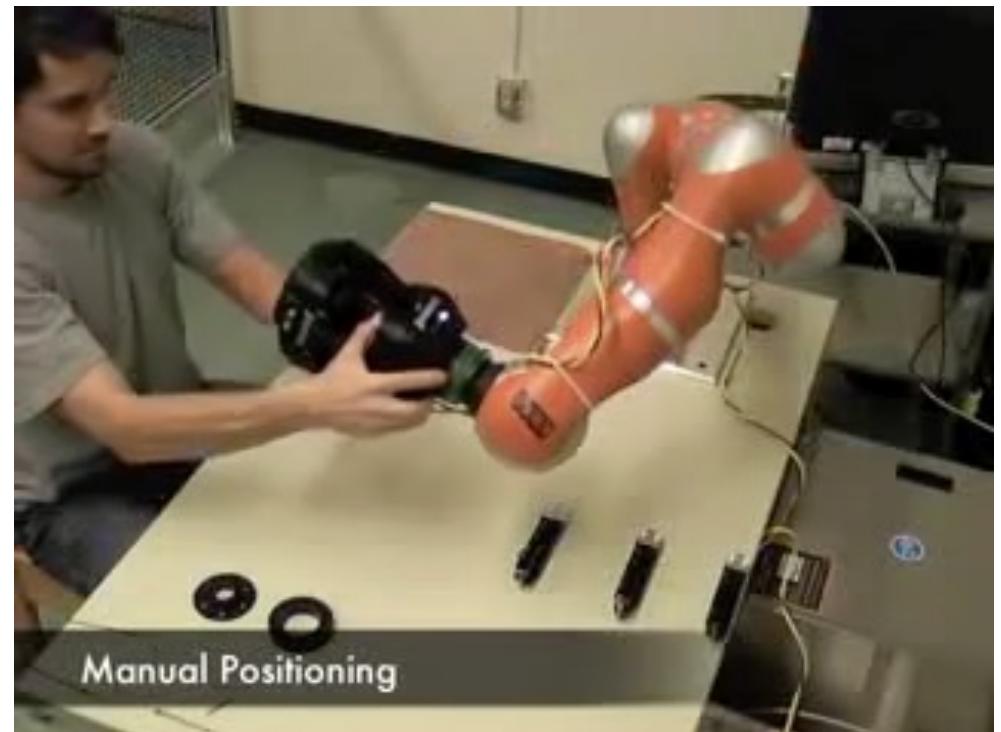
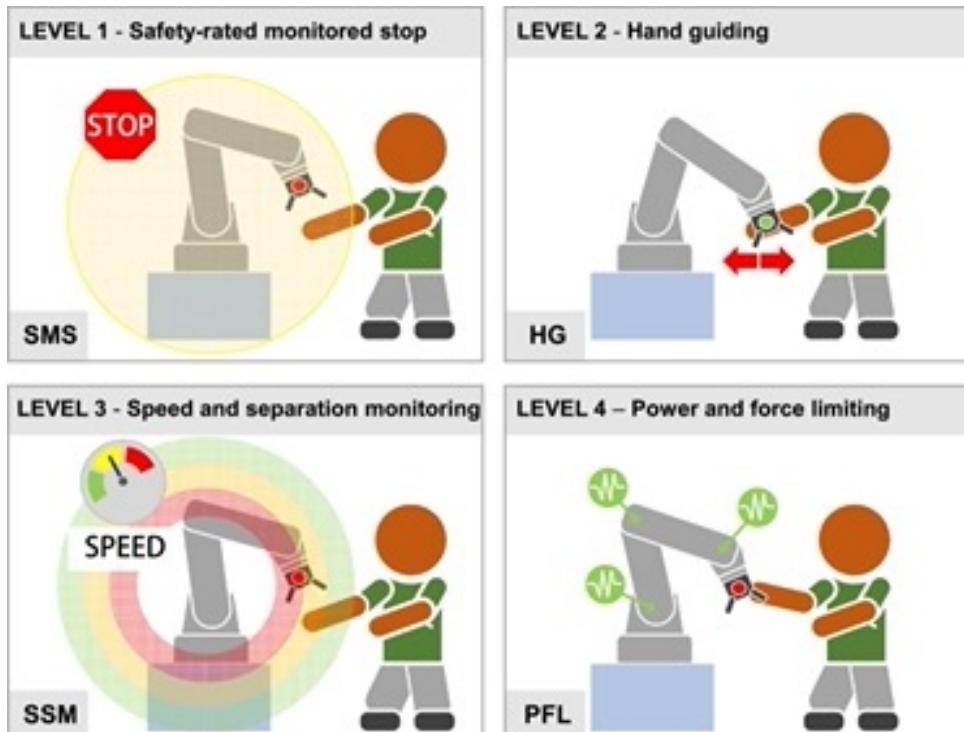
with intentional contact and
coordinated exchange of forces

A. De Luca, F. Flacco: *IEEE BioRob Conference, 2012*



Human-Robot Collaboration

- the different possible levels of pHRI are represented also within ISO safety standards (from safe coexistence to safe collaboration)



V. Villani et al.: *Mechatronics*, 2018

[video](#)



Robotics 2

Regulation in the Joint Space

(with an introduction to stability)

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Equilibrium states of a robot

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$$

$$\rightarrow \dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -M^{-1}(x_1)[c(x_1, x_2) + g(x_1)] \end{pmatrix} + \begin{pmatrix} 0 \\ M^{-1}(x_1) \end{pmatrix} u$$

↑
= 0 = $f(x) + G(x_1)u$

$$x_e \text{ unforced equilibrium} \quad (u = 0) \quad \rightarrow \quad f(x_e) = 0 \quad \rightarrow \quad \begin{cases} x_{e2} = 0 \\ g(x_{e1}) = 0 \end{cases}$$

$$x_e \text{ forced equilibrium} \quad (u = u(x)) \quad \rightarrow \quad f(x_e) + G(x_{e1})u(x_e) = 0 \quad \rightarrow \quad \begin{cases} x_{e2} = 0 \\ u(x_e) = g(x_{e1}) \end{cases}$$

all equilibrium states of mechanical systems have zero velocity!

joint torques must balance gravity at the equilibrium!



Stability of dynamical systems

definitions - 1

$$\dot{x} = f(x)$$

e.g., a closed-loop system
(under feedback control)

x_e **equilibrium:** $f(x_e) = 0$

(sometimes we consider as equilibrium state
 $x_e = 0$, e.g., when using errors as variables)

stability of x_e

$\forall \varepsilon > 0, \exists \delta_\varepsilon > 0: \|x(t_0) - x_e\| < \delta_\varepsilon \Rightarrow \|x(t) - x_e\| < \varepsilon, \forall t \geq t_0$

asymptotic stability of x_e

stability +

$\exists \delta > 0: \|x(t_0) - x_e\| < \delta \Rightarrow \|x(t) - x_e\| \rightarrow 0, \text{ for } t \rightarrow \infty$

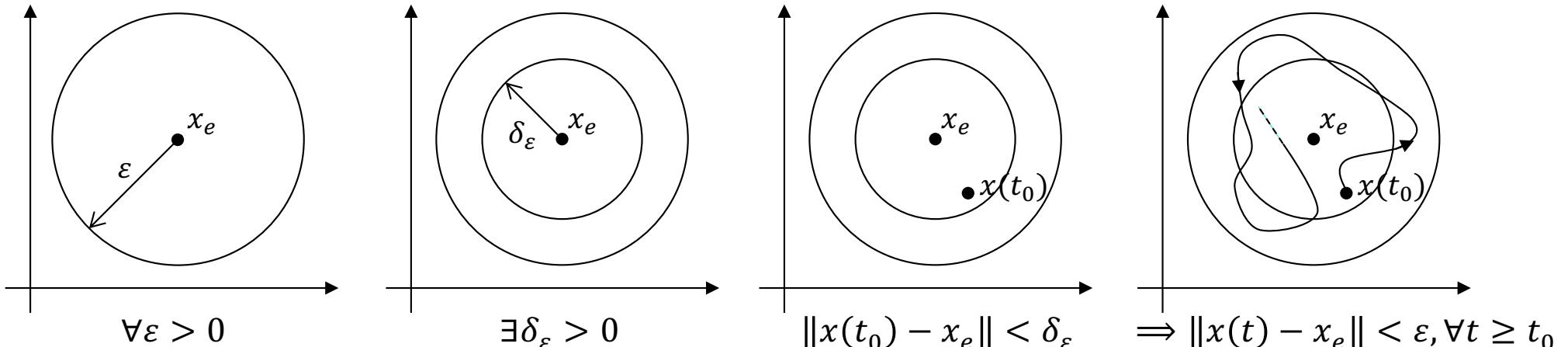
asymptotic stability may become **global ($\forall \delta > 0$, finite)**

note: these are definitions of stability “in the sense of Lyapunov”

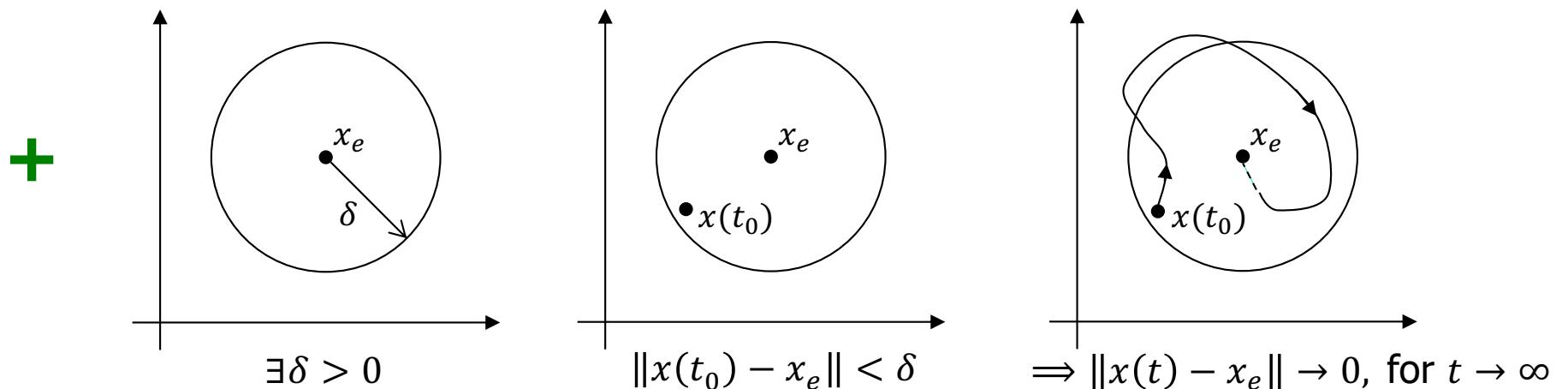


Stability vs. asymptotic stability

whiteboard...



equilibrium state x_e is **stable**



equilibrium state x_e is **asymptotically stable**



Stability of dynamical systems

definitions - 2

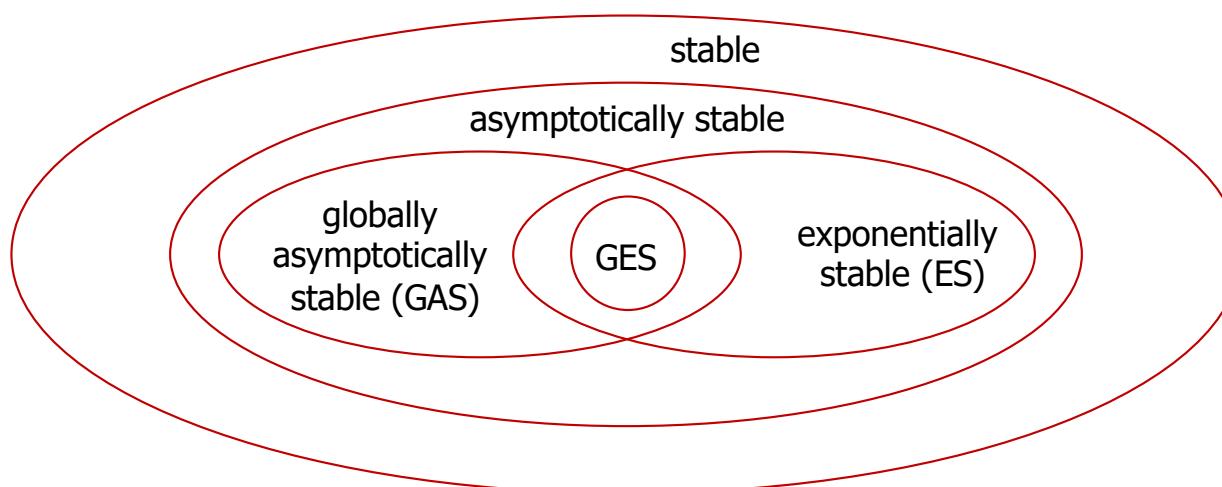
exponential stability of x_e

exponential rate λ

$$\exists \delta, c, \lambda > 0: \|x(t_0) - x_e\| < \delta \Rightarrow \|x(t) - x_e\| \leq ce^{-\lambda(t-t_0)} \|x(t_0) - x_e\|$$

- allows to estimate the time needed to "approximately" converge: for $c = 1$, in $t - t_0 = 3 \times$ the **time constant** $\tau = 1/\lambda$, the initial error is reduced to 5%
- typically, this is a **local** property only (within some maximum **finite** radius δ)
⇒ such "domain of attraction" is hard to be estimated accurately

taxonomy
of stability
definitions



a **necessary**
condition for
 x_e to be GAS
is that
it is the **only**
equilibrium state
of the system



The need for analysis and criteria

whiteboard...

a nonlinear system $\dot{x} = f(x)$ in \mathbb{R}^2 two equilibria $f(x_e) = 0$

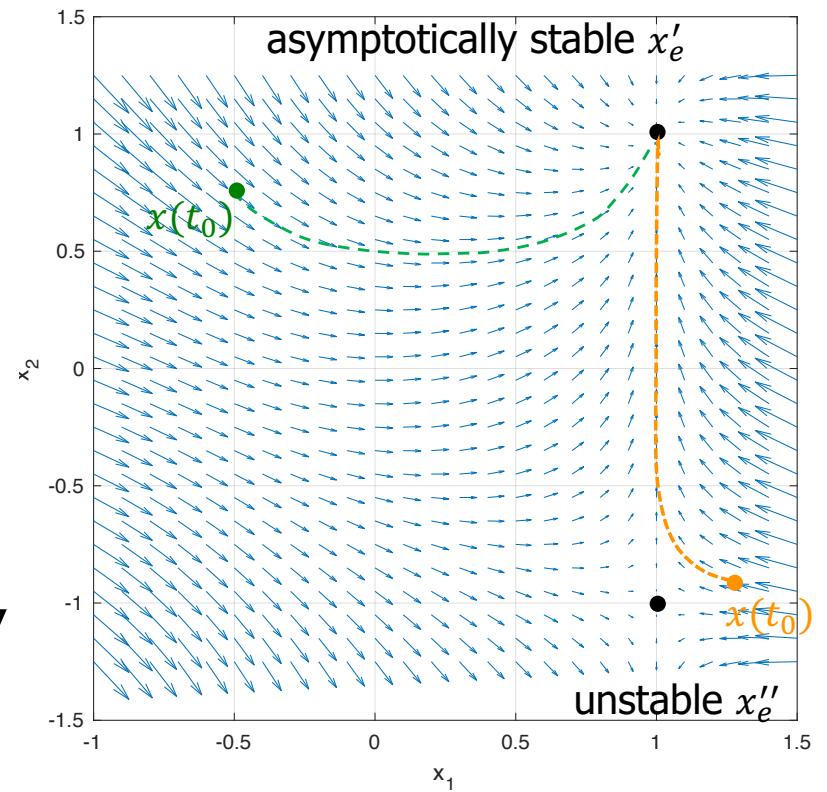
$$\begin{cases} \dot{x}_1 = 1 - x_1^3 \\ \dot{x}_2 = x_1 - x_2^2 \end{cases} \quad \rightarrow$$

$$x'_e = (1, 1), \quad x''_e = (1, -1)$$

to assess (asymptotic) stability [or not] of equilibria, do we need to compute all system trajectories, starting from all possible initial states $x(t_0)$?



rather, we may be able to just look at the time evolution of a scalar function V , evaluated analytically along the state trajectories of the system (even in \mathbb{R}^n !)





Stability of dynamical systems

results - 1

Lyapunov candidate

$V(x): \mathbb{R}^n \rightarrow \mathbb{R}$ such that

$$V(x_e) = 0, V(x) > 0, \forall x \neq x_e$$

positive
definite
function

typically, **quadratic** (e.g., $\frac{1}{2}(x - x_e)^T P(x - x_e)$ with level surfaces = ellipsoids)
may also be a **local** candidate only ($\forall x \neq x_e: \|x - x_e\| < \delta$)

sufficient condition of stability

$\exists V$ candidate: $\dot{V}(x) \leq 0$, along the trajectories of $\dot{x} = f(x)$

negative
semi-definite
function

sufficient condition of asymptotic stability

$\exists V$ candidate: $\dot{V}(x) < 0$, along the trajectories of $\dot{x} = f(x)$

negative
definite
function

sufficient condition of instability

$\exists V$ candidate: $\dot{V}(x) > 0$, along the trajectories of $\dot{x} = f(x)$



Stability of dynamical systems

results - 2

LaSalle Theorem

if $\exists V$ candidate: $\dot{V}(x) \leq 0$ along the trajectories of $\dot{x} = f(x)$

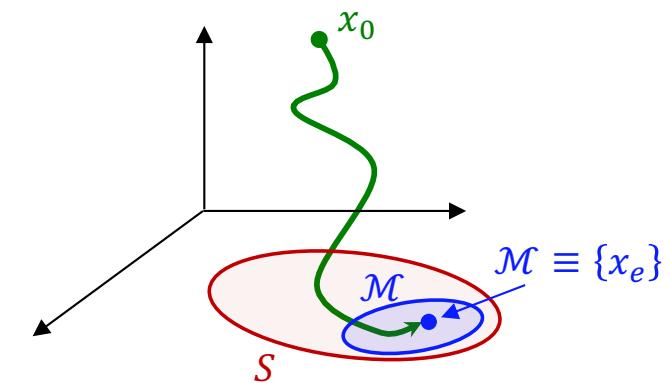
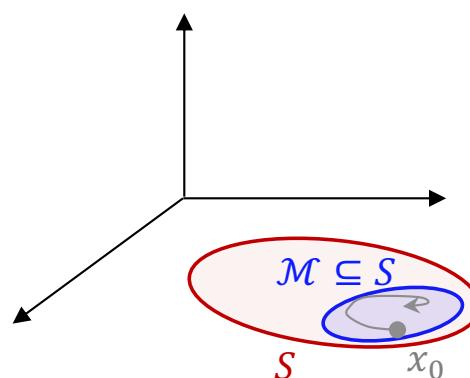
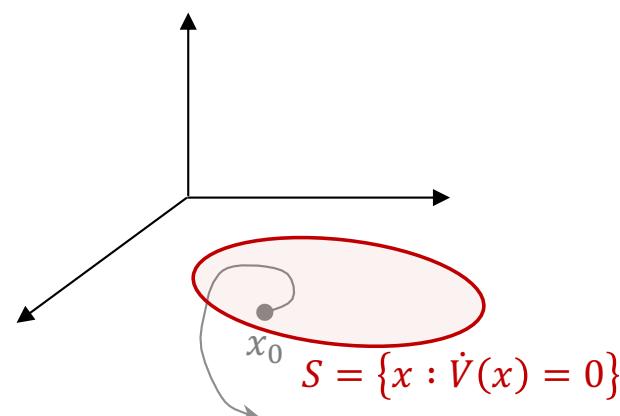


then system trajectories asymptotically converge to the largest invariant set $\mathcal{M} \subseteq S = \{x \in \mathbb{R}^n : \dot{V}(x) = 0\}$

\mathcal{M} is invariant if $x(t_0) \in \mathcal{M} \Rightarrow x(t) \in \mathcal{M}, \forall t \geq t_0$

Corollary

$\mathcal{M} \equiv \{x_e\} \rightarrow$ asymptotic stability





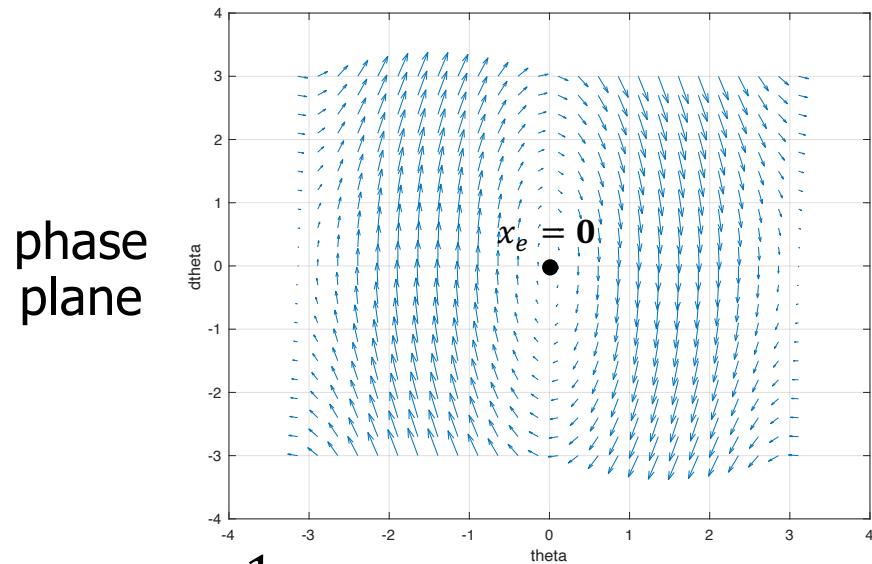
Bird-eye view on Lyapunov analysis

whiteboard...

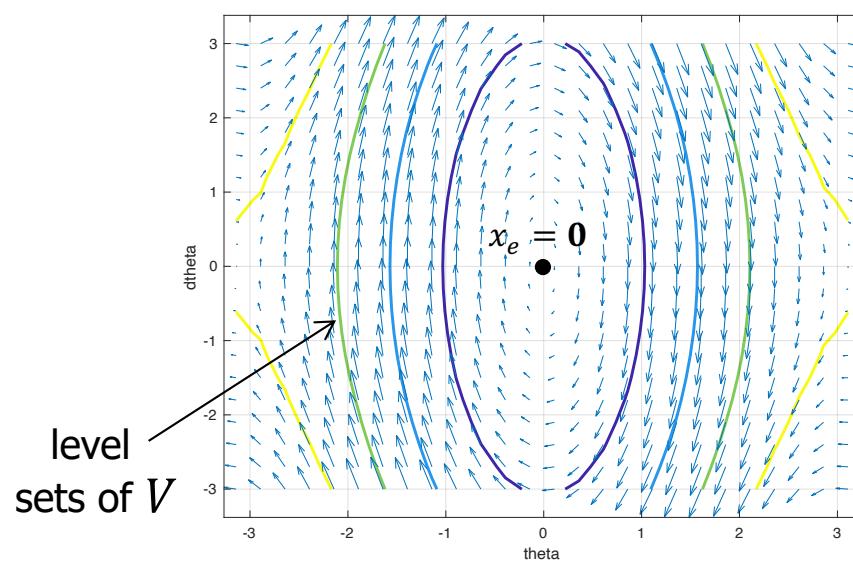
a mass m at the end of an unforced (passive) pendulum of length l

$$ml^2\ddot{\theta} + b\dot{\theta} + mlg_0 \sin \theta = 0 \Rightarrow x = (x_1, x_2) = (\theta, \dot{\theta}) \in \mathbb{R}^2 \Rightarrow \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\left(\frac{g_0}{l}\right) \sin x_1 - \left(\frac{b}{ml^2}\right) x_2 \end{cases}$$

lower equilibrium at $\theta_e = 0$



$$V = E = \frac{1}{2} ml^2 \dot{\theta}^2 + mlg_0 (1 - \cos \theta) \geq 0$$



$$V = 0 \Leftrightarrow x_e = (\theta_e, \dot{\theta}_e) = (0,0)$$

$$\dot{V} = \dot{\theta}(ml^2\ddot{\theta} + mlg_0 \sin \theta) = -b\dot{\theta}^2 \leq 0 \Rightarrow \text{stability of equilibrium } x_e = 0 \text{ (... at least!)}$$

$$\Rightarrow \text{use LaSalle: } \dot{V} = 0 \Leftrightarrow \dot{\theta} = 0 \Rightarrow \ddot{\theta} = -\left(\frac{g_0}{l}\right) \sin \theta \neq 0 \text{ unless } \theta = \theta_e = 0 \text{ (or } \pi!)$$

\Rightarrow local asymptotic stability



Stability of dynamical systems

results - 3

- previous results are also valid for **periodic** time-varying systems

$$\dot{x} = f(x, t) = f(x, t + T_p) \Rightarrow V(x, t) = V(x, t + T_p)$$

- for general **time-varying** systems (e.g., in robot trajectory tracking control)

$$\dot{x} = f(x, t)$$

Barbalat Lemma

if i) a function $V(x, t)$ is lower bounded

ii) $\dot{V}(x, t) \leq 0$

then $\Rightarrow \exists \lim_{t \rightarrow \infty} V(x, t)$ (but this does **not** imply that $\lim_{t \rightarrow \infty} \dot{V}(x, t) = 0$)

if in addition iii) $\ddot{V}(x, t)$ is bounded

then $\Rightarrow \lim_{t \rightarrow \infty} \dot{V}(x, t) = 0$

Corollary

if a Lyapunov candidate $V(x, t)$ satisfies Barbalat Lemma along the trajectories of $\dot{x} = f(x, t)$, then the conclusions of LaSalle Theorem hold



Stability of dynamical systems

additional definition and result (for robust control)

“practical” stability of a set S

$$\exists T(x(t_0), S) \in \mathbb{R}: x(t) \in S, \forall t \geq t_0 + T(x(t_0), S)$$

a finite time

also known as u.u.b. stability

⇒ trajectories $x(t)$ are “uniformly ultimately bounded” (use in robust control)

sufficient condition of u.u.b. stability of a set S

$\exists V$ candidate: i) S is a level set of V for a given c_0

$$S = S(c_0) = \{x \in \mathbb{R}^n: V(x) \leq c_0\}$$

ii) $\dot{V}(x) < 0$ along trajectories of $\dot{x} = f(x), x \notin S$



Stability of linear systems

time-invariant case

$$\dot{x} = Ax$$

$x_e = 0$ is always an equilibrium state

- I. asymptotic stability
- II. global asymptotic stability
- III. exponential stability
- IV. $\sigma(A) \subset \mathbb{C}^-$ (all eigenvalues of A have negative real part)
- V. $\forall Q > 0$ (positive definite), $\exists! P > 0$: $A^T P + PA = -Q$
Lyapunov equation $\Rightarrow \frac{1}{2}x^T Px$ is a Lyapunov candidate

ALL EQUIVALENT !!

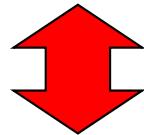
if $x_e = 0$ is an asymptotically stable equilibrium,
then it is necessarily the unique equilibrium



Stability of the linear approximation

Let $\Delta x = x - x_e$ and let $\dot{\Delta x} = \frac{df}{dx} \Big|_{x=x_e} (x - x_e) = A \Delta x$ be the linear approximation of $\dot{x} = f(x)$ around the equilibrium x_e

A asymptotically stable ($\sigma(A) \subset \mathbb{C}^-$)



the original nonlinear system is
exponentially stable at the origin

this is only a **local** result
(used also to estimate the domain of attraction)



PD control

(proportional + derivative action on the error)

robot

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u$$

goal: asymptotic stabilization (= **regulation**)
of the closed-loop equilibrium state

$$q = q_d, \dot{q} = 0$$



possibly obtained from kinematic inversion: $q_d = f^{-1}(r_d)$

control law

$$u = K_P(q_d - q) - K_D\dot{q}$$

$K_P > 0, K_D > 0$ (positive definite), symmetric



Asymptotic stability with PD control

Theorem 1

In the absence of gravity ($g(q) \equiv 0$), the robot state $(q_d, 0)$ under the given PD joint control law is **globally asymptotically stable**

Proof

let $e = q_d - q$ (q_d constant)

Lyapunov candidate

$$V = \frac{1}{2} \dot{q}^T M(q) \dot{q} + \frac{1}{2} e^T K_P e \geq 0$$

$$V = 0 \Leftrightarrow e = \dot{e} = 0$$

$$\begin{aligned} \dot{V} &= \dot{q}^T M \ddot{q} + \frac{1}{2} \dot{q}^T \dot{M} \dot{q} - e^T K_P \dot{q} = \dot{q}^T \left(u - \underbrace{S \dot{q} + \frac{1}{2} \dot{M} \dot{q}}_{= 0, \text{ due to energy conservation}} \right) - e^T K_P \dot{q} \\ &= \cancel{\dot{q}^T K_P e} - \cancel{\dot{q}^T K_D \dot{q}} - e^T \cancel{K_P \dot{q}} = -\dot{q}^T K_D \dot{q} \leq 0 \quad (K_D > 0, \text{ symmetric}) \end{aligned}$$

up to here, we proved
stability only

but $\dot{V} = 0 \Leftrightarrow \dot{q} = 0$ continues ...



Asymptotic stability with PD control

$\dot{V} = 0 \Leftrightarrow \dot{q} = 0$ LaSalle \rightarrow system trajectories converge to the largest invariant set of states \mathcal{M} where $\dot{q} \equiv 0$ (that is $\dot{q} = \ddot{q} = 0$)

$$\dot{q} = 0 \rightarrow \underbrace{M(q)\ddot{q} = K_P e}_{\text{closed-loop dynamics}} \rightarrow \ddot{q} = \underbrace{M^{-1}(q)K_P e}_{\text{invertible}}$$

$$\dot{q} = 0, \ddot{q} = 0 \Leftrightarrow e = 0$$

→ the only invariant state in $\dot{V} = 0$ is given by $q = q_d, \dot{q} = 0$

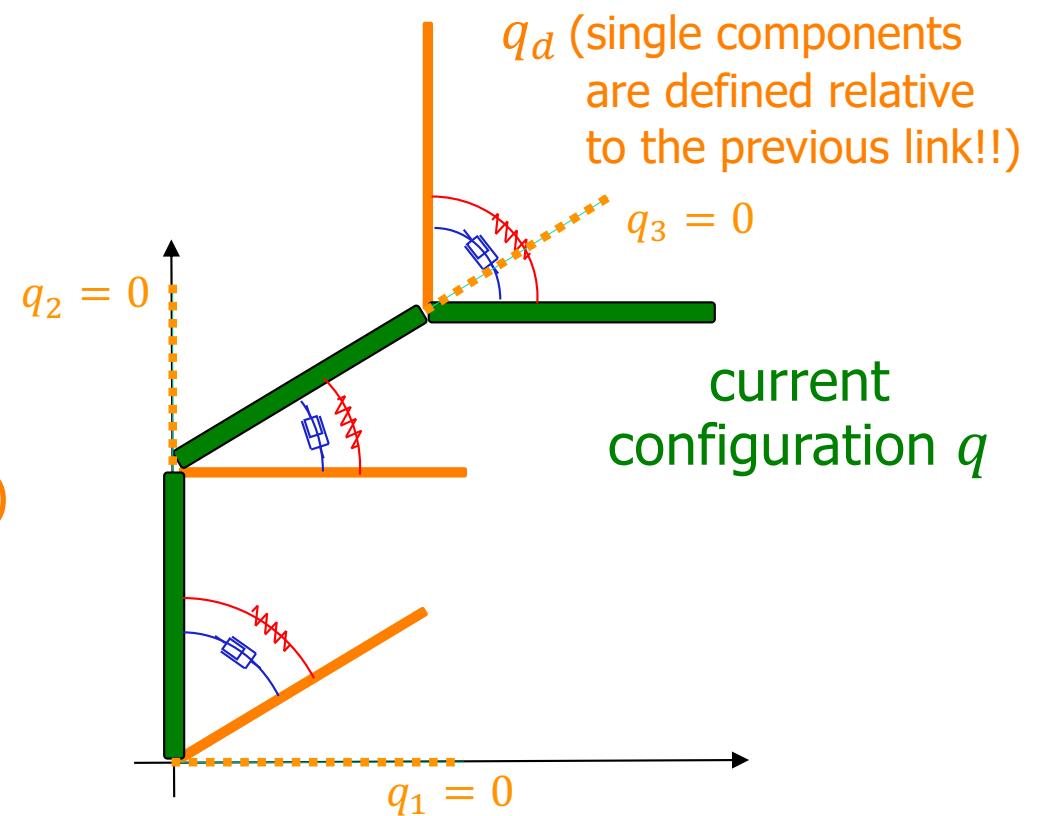
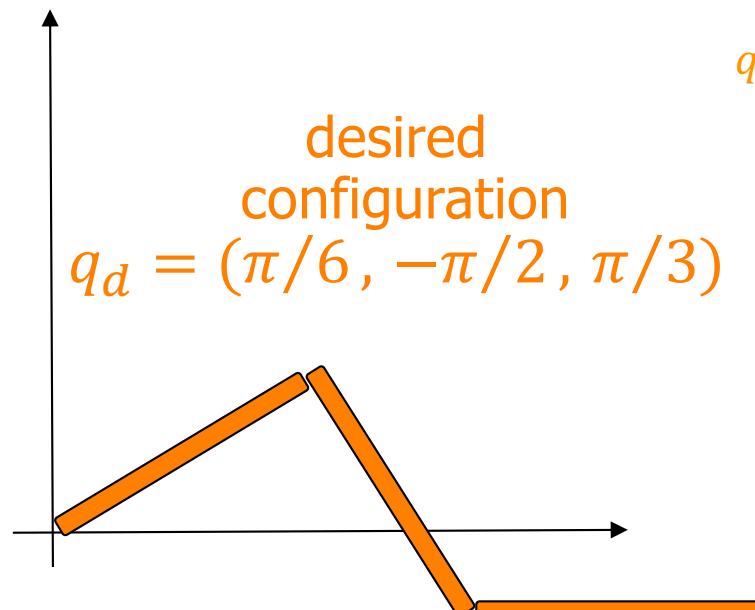
note: typically, $K_P = \text{diag}\{k_{Pi}\}$, $K_D = \text{diag}\{k_{Di}\}$,
→ decentralized linear control (**local** to each joint)



Mechanical interpretation

- for diagonal positive definite gain matrices K_P and K_D (thus, with positive diagonal elements), such values correspond to stiffness of “virtual” **springs** and viscosity of “virtual” **dampers** placed at the joints

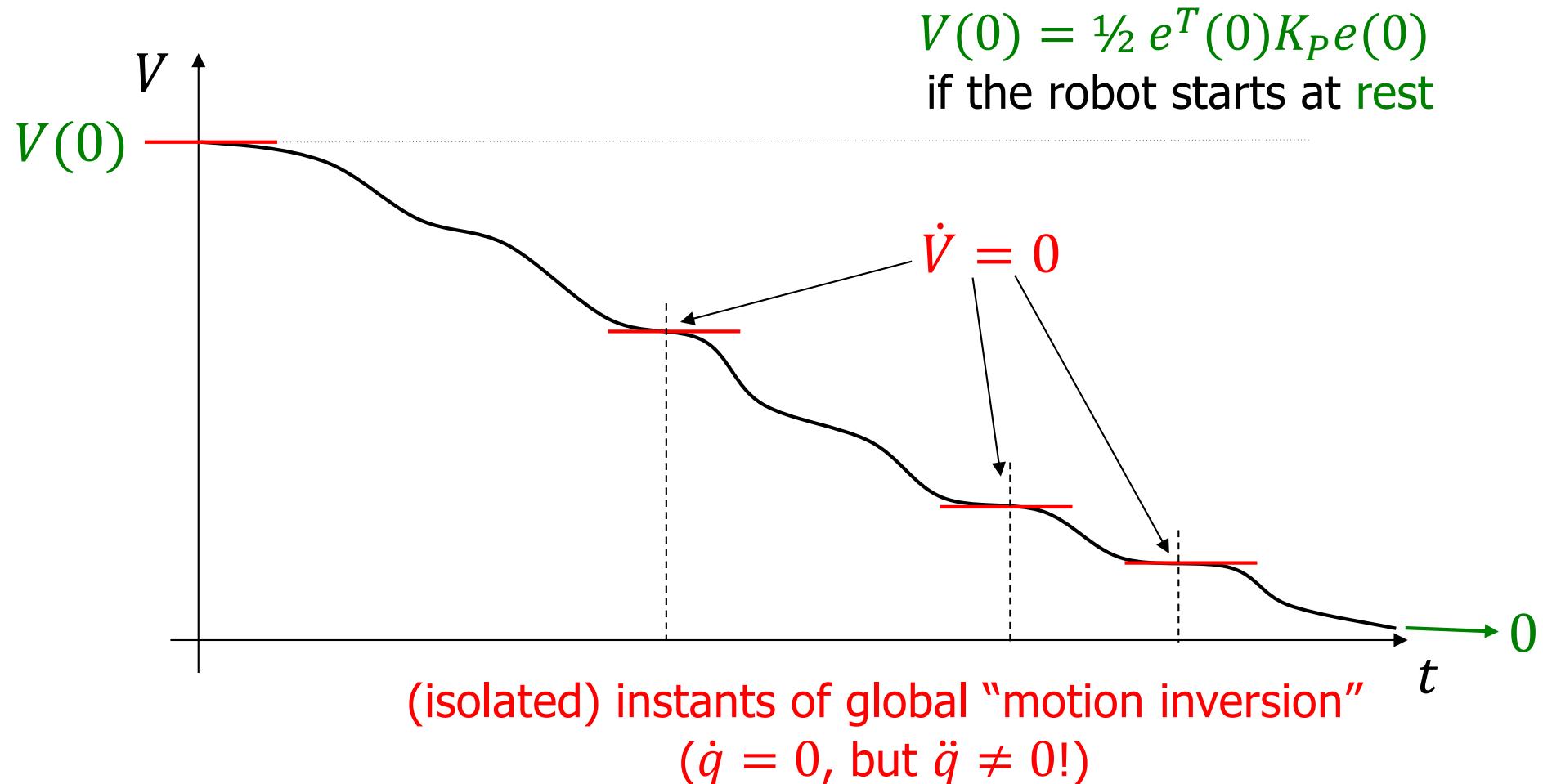
stiffness $k_{Pi} > 0$
 viscosity $k_{Di} > 0$





Plot of the Lyapunov function V

- time evolution of the Lyapunov candidate





Comments on PD control - 1

- choice of control gains affects robot evolution during transients and practical settling times
 - hard to define values that are “optimal” in the whole workspace
 - “full” K_P and K_D gain matrices allow to assign desired eigenvalues to the linear approximation of the robot dynamics around the final desired state $(q_d, 0)$
- when (joint) viscous friction is present, the derivative term in the control law is not strictly necessary
 - $-F_V \dot{q}$ in the robot model acts similarly to $-K_D \dot{q}$ in the control law, but the latter can be modulated at will
- in the absence of tachometers, the actual realization of the derivative term in the feedback law requires some processing of joint position data measured by digital encoders (or analog resolvers/potentiometers)



Comments on PD control - 2

- analog or digital implementation of derivative action in the control law when only position is measured at the joints (e.g., through encoders)

continuous-time
control law (design)

$$u(t) = K_P e(t) + K_D \dot{e}(t)$$

$$e = q_d - q, \dot{e} = -\dot{q}$$



representation in
the Laplace domain

$$u(s) = (K_P + K_D s) e(s)$$



not realizable as such
(non-proper transfer function)

$$u(s) = \left(K_P + \frac{K_D s}{1 + \tau s} \right) e(s)$$

derivative action limited
in bandwidth (up to $\omega \leq 1/\tau$)

transformation in
the Zeta-domain
(e.g., via backward
differentiation rule on
samples, every T_c sec)

$$u(z) = \left(K_P + K_D \frac{1 - z^{-1}}{T_c} \right) e(z)$$



$$u(z) = \left(K_P + K_D \frac{\frac{1 - z^{-1}}{T_c}}{1 + \tau \frac{1 - z^{-1}}{T_c}} \right) e(z)$$



discrete-time
implementations

$$u_k = K_P e_k + K_D \frac{e_k - e_{k-1}}{T_c}$$

both realizable

$$u_k = K_P e_k + \frac{K_D}{\tau + T_c} (e_k - e_{k-1}) + \frac{\tau}{\tau + T_c} (u_{k-1} - K_P e_{k-1})$$



Inclusion of gravity

- in the presence of gravity, the same previous arguments (and proof) show that the control law

$$u = K_P(q_d - q) - K_D \dot{q} + g(q)$$

$$K_P > 0, K_D > 0$$

will make the equilibrium state $(q_d, 0)$ globally asymptotically stable (**nonlinear cancellation of gravity**)

- if gravity is **not** cancelled or only **approximately** cancelled

$$u = K_P(q_d - q) - K_D \dot{q} + \hat{g}(q) \quad \hat{g}(q) \neq g(q)$$

it is $q \rightarrow q^* \neq q_d, \dot{q} \rightarrow 0$, with a **steady-state** position error

- q^* is not unique in general, except when K_P is chosen large enough
- explanation in terms of linear systems: there is **no integral action** before the point of access of the **constant “disturbance”** acting on the system



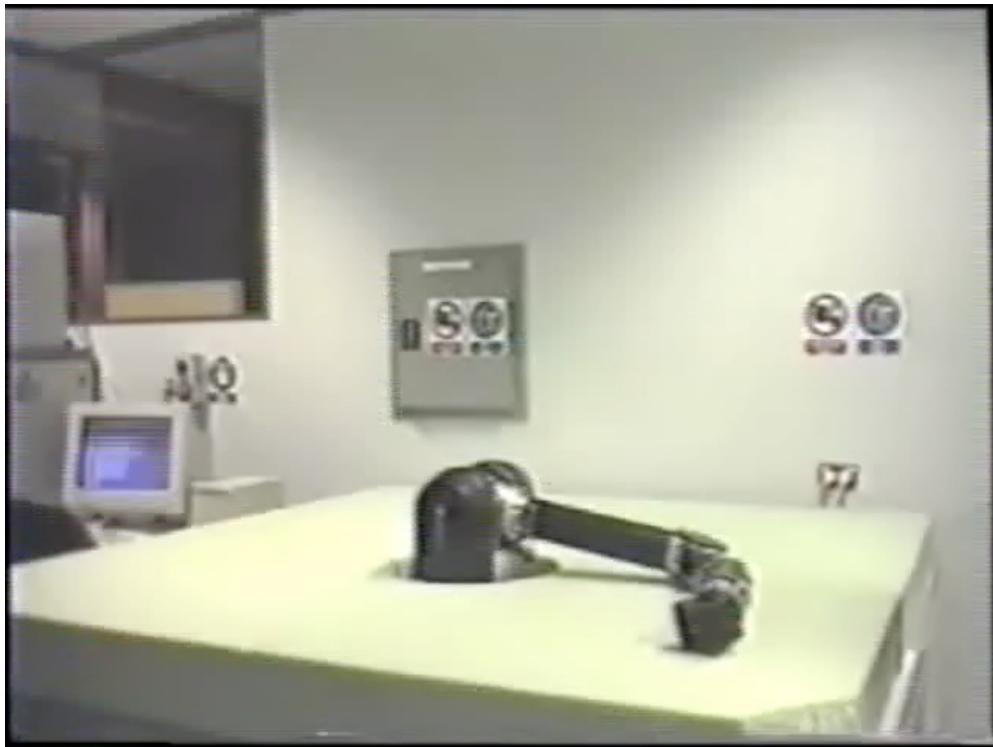
Approximate cancellation of gravity

WAM Barrett
(with some viscous friction)

$$\hat{g}(q) = g(q)$$

$$u = \hat{g}(q)$$

$$\hat{g}(q) \neq g(q)$$



two-part video

<http://handbookofrobotics.org/view-chapter/69/videodetails/611>

PD control + constant gravity compensation



if the robot potential energy $U(q)$ is bounded for all q , then its partial derivative $g(q)$ is also **bounded** everywhere and the following **structural property** holds

finite

$$\exists \alpha > 0: \left\| \frac{\partial^2 U}{\partial q^2} \right\| = \left\| \frac{\partial g}{\partial q} \right\| \leq \alpha, \forall q$$

consequence

$$\|g(q) - g(q_d)\| \leq \alpha \|q - q_d\|$$

note: induced
norm of
a matrix

$$\|A\| = \sqrt{\lambda_{\max}(A^T A)} \triangleq A_M \geq A_m \triangleq \sqrt{\lambda_{\min}(A^T A)}$$

LINEAR CONTROL law

$$u = K_P(q_d - q) - K_D \dot{q} + g(q_d)$$

$K_P, K_D > 0$
symmetric

linear feedback + **constant** feedforward



More on the basic assumption ...

(in PD control + gravity compensation)

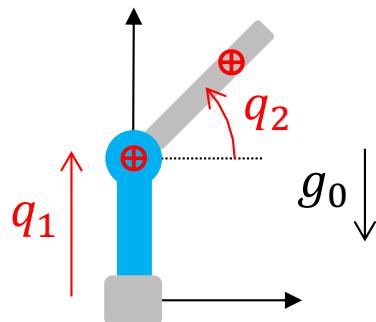
when is the (non-zero) gravity term $g(q)$ bounded for all q ?

- the robot has **all revolute** joints
 - all terms in $U(q)$ and thus in $g(q)$ are trigonometric (bounded)
- the robot has both types of joints, but **no prismatic variables** in $g(q)$
 - potential energy $U(q)$ may still be unbounded!
- **all prismatic** joints of the robot have a **limited range**
 - ... ok, but one should take these limits into account in the control analysis

$$U = g_0(m_1 q_1 + m_2(q_1 + d_{c2} \sin q_2)) + U_0$$

$$g(q) = g_0 \left(\frac{(m_1 + m_2)}{m_2 d_{c2} \cos q_2} \right)$$

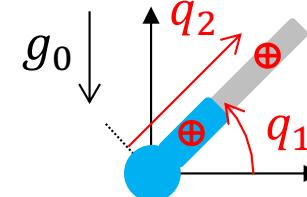
$$\left\| \frac{\partial g}{\partial q} \right\| \leq \alpha = m_2 d_{c2} g_0$$



PR robot

$$U = g_0(m_1 d_{c1} + m_2 q_2) \sin q_1 + U_0$$

$$g(q) = g_0 \left(\frac{(m_1 d_{c1} + m_2 q_2) \cos q_1}{m_2 \sin q_1} \right)$$



RP robot

$$\left\| \frac{\partial g}{\partial q} \right\| \leq ??$$



PD control + constant gravity compensation

stability analysis

Theorem 2

If $K_{P,m} > \alpha$, the state $(q_d, 0)$ of the robot under joint-space PD control + constant gravity compensation at q_d is **globally asymptotically stable**

Proof

1.

$(q_d, 0)$ is the unique closed-loop equilibrium state

in fact, for $\dot{q} = 0$ and $\ddot{q} = 0$, it is $K_P e = g(q) - g(q_d)$

which can hold only for $q = q_d$, because when $q \neq q_d$

$$\|K_P e\| \geq K_{P,m} \|e\| > \alpha \|e\| \geq \|g(q) - g(q_d)\|$$

$$M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) = K_P e - K_D \dot{q} + g(q_d)$$



PD control + constant gravity compensation

stability analysis

with $e = q_d - q$, $g(q) = \left(\frac{\partial U}{\partial q}\right)^T$, consider as Lyapunov candidate

$$V = \frac{1}{2} \dot{q}^T M(q) \dot{q} + \frac{1}{2} e^T K_P e + U(q) - U(q_d) + e^T g(q_d)$$

2.

V is convex in \dot{q} and e , and zero only for $e = \dot{q} = 0$

$$\left(\frac{\partial V}{\partial \dot{q}}\right)^T = M(q) \dot{q} = 0 \text{ only for } \dot{q} = 0$$

$$\frac{\partial^2 V}{\partial \dot{q}^2} = M(q) > 0$$



$(q_d, 0)$ is a
global minimum
of $V \geq 0$

$$\left(\frac{\partial V|_{\dot{q}=0}}{\partial e}\right)^T = K_P e - \left(\frac{\partial U}{\partial q}\right)^T + g(q_d) = K_P e + g(q_d) - g(q) = 0$$

$$\frac{\partial e}{\partial q} = -I$$

only for $q = q_d$

$$\frac{\partial^2 V|_{\dot{q}=0}}{\partial e^2} = K_P + \frac{\partial^2 U}{\partial q^2} > 0, \text{ since } \|K_P\| = K_{P,M} \geq K_{P,m} > \alpha$$



PD control + constant gravity compensation

stability analysis

differentiating

$$V = \frac{1}{2} \dot{q}^T M(q) \dot{q} + \frac{1}{2} e^T K_P e + U(q) - U(q_d) + e^T g(q_d)$$

$$\begin{aligned}\dot{V} &= \dot{q}^T \left(M(q) \ddot{q} + \frac{1}{2} \dot{M}(q) \dot{q} \right) - e^T K_P \dot{q} + \frac{\partial U(q)}{\partial q} \dot{q} - \dot{q}^T g(q_d) \\ &= \dot{q}^T \left(u - S(q, \dot{q}) \dot{q} + \frac{1}{2} \dot{M}(q) \dot{q} - g(q) \right) - e^T K_P \dot{q} + \dot{q}^T (g(q) - g(q_d)) \\ &= \cancel{\dot{q}^T K_P e} - \cancel{\dot{q}^T K_D \dot{q}} + \dot{q}^T (g(q_d) - g(q)) - \cancel{e^T K_P \dot{q}} + \cancel{\dot{q}^T (g(q) - g(q_d))} \\ &= -\dot{q}^T K_D \dot{q} \leq 0\end{aligned}$$

for $\dot{V} = 0$ ($\Leftrightarrow \dot{q} = 0$), we have in the **closed-loop** system

$$M(q) \ddot{q} + g(q) = K_P e + g(q_d)$$

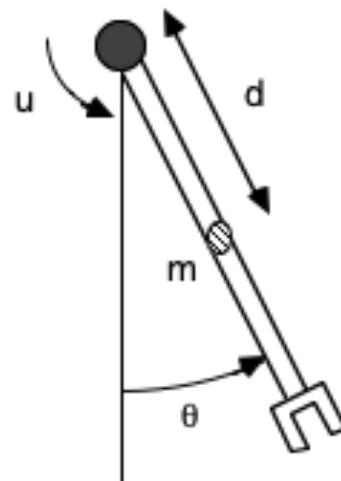
$$\rightarrow \ddot{q} = M^{-1}(q)(K_P e + g(q_d) - g(q)) = 0 \Leftrightarrow e = 0$$

by LaSalle Theorem, the thesis follows





Example of a single-link robot stability analysis



task: regulate the link position to the upward equilibrium

$$\theta_d = \pi \rightarrow g(\theta_d) = 0$$

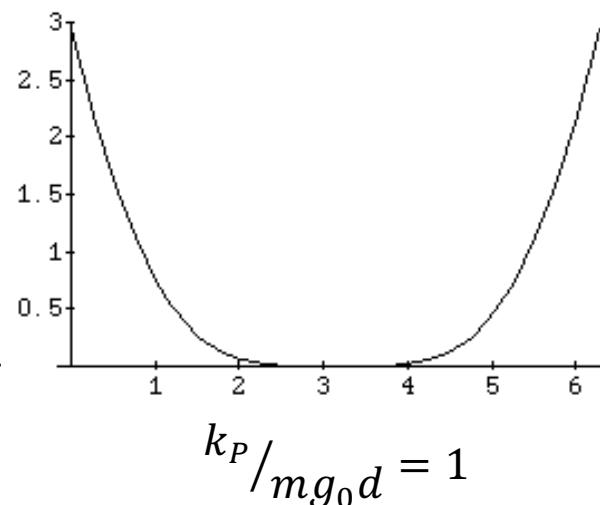
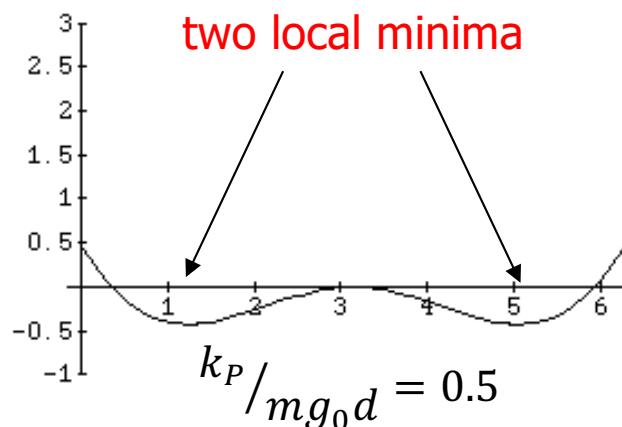
PD control + constant gravity compensation (here, zero!)

$$u = k_P(\pi - \theta) - k_D \dot{\theta}$$

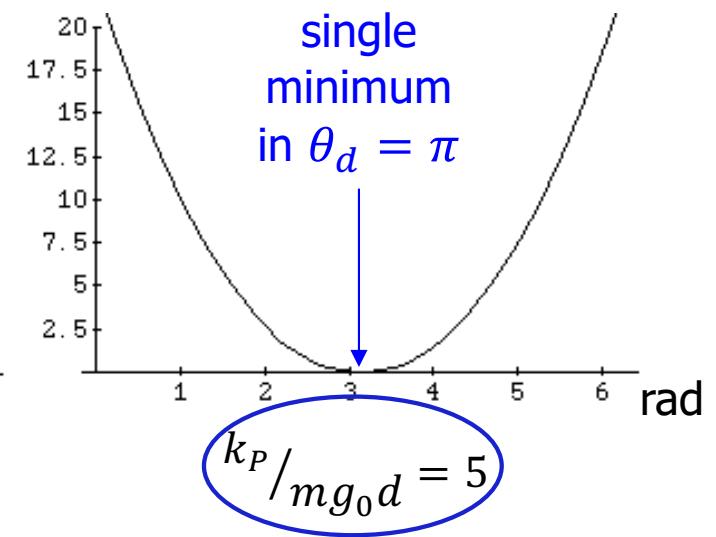
by Theorem 2, it is sufficient (here, also necessary*) to choose

$$k_P > \alpha = mg_0d, \quad k_D > 0$$

$$I\ddot{\theta} + mg_0d \sin \theta = u$$



plots of $V(\theta)$ (for $\dot{\theta} = 0$)

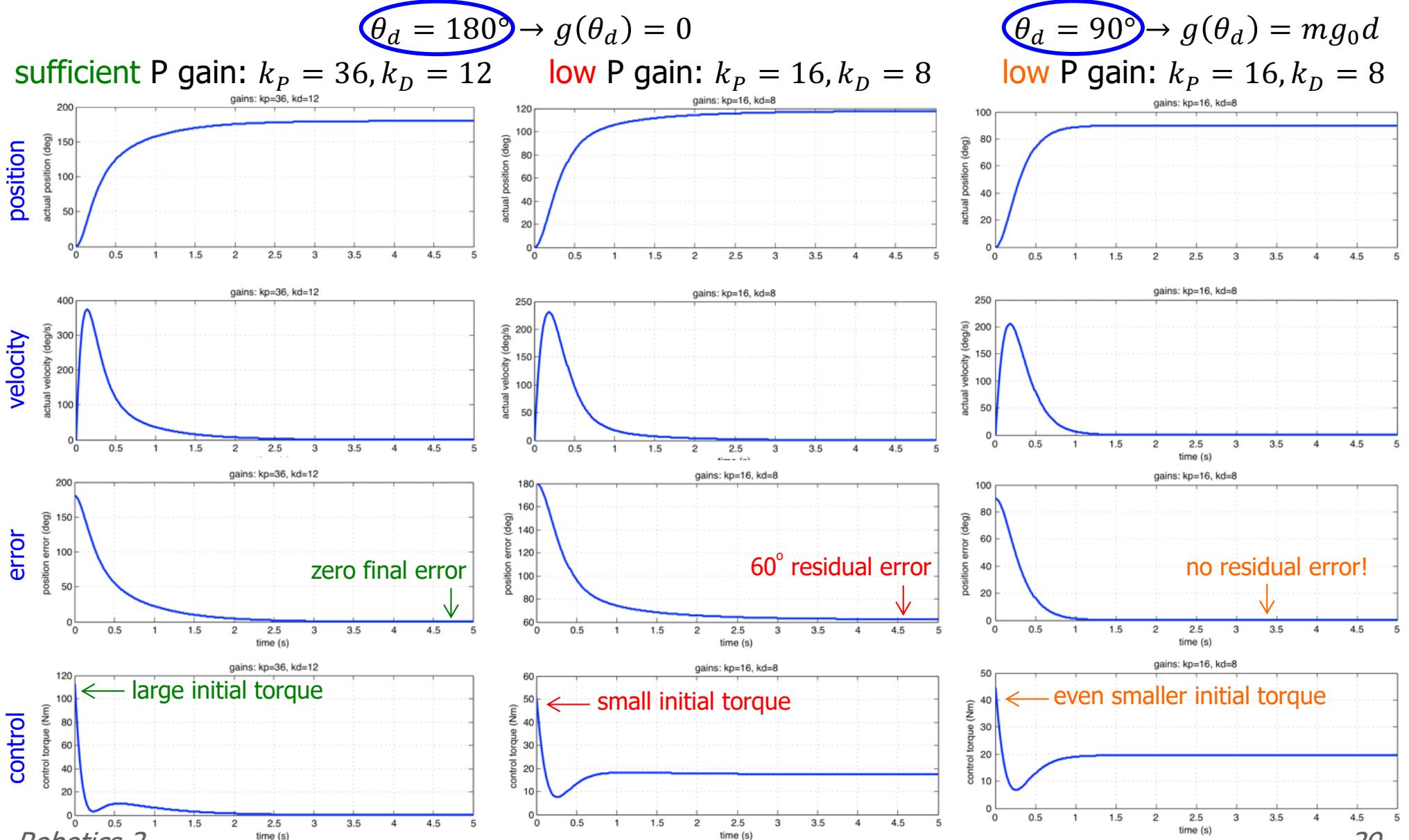


* by a local analysis of the linear approximation at π



Example of a single-link robot

simulations with data: $I = 0.9333$, $mg_0d = 19.62$ ($= \alpha$)





Approximate gravity compensation

the **approximate** control law

$$u = K_P(q_d - q) - K_D \dot{q} + \hat{g}(q_d)$$

leads, under similar hypotheses, to a closed-loop equilibrium q^*

- its uniqueness is not guaranteed (unless K_P is large enough)
- for $K_P \rightarrow \infty$, one has $q^* \rightarrow q_d$

conclusion: in the presence of gravity, the previous regulation control laws require an **accurate knowledge** of the **gravity term** in the dynamic model
to guarantee the zeroing of the position error
(since we can only use “finite” control gains \Rightarrow in practice, not too large)



PID control

- in **linear systems**, the addition of an integral control action is used to eliminate a constant error in the step response at steady state
- in **robots**, a PID may be used to recover such a position error due to an incomplete (or absent) gravity compensation/cancellation

→ the control law

$$u(t) = K_P(q_d - q(t)) + K_I \int_0^t (q_d - q(\tau)) d\tau - K_D \dot{q}(t)$$

- is **independent** from any robot dynamic model term
- **if** the desired closed-loop equilibrium is asymptotically stable under PID control, the integral term is “loaded” at **steady state** to the value

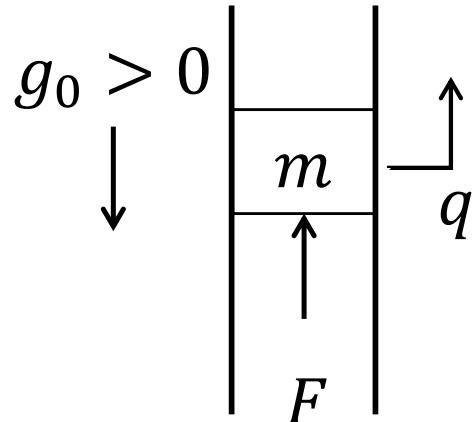
$$K_I \int_0^\infty (q_d - q(\tau)) d\tau = g(q_d)$$

- however, one can show only **local** asymptotic stability of this law, i.e., for $q(0) \in \Delta(q_d)$, under **complex conditions** on K_P, K_I, K_D and $e(0)$



Linear example with PID control

whiteboard...



$$m\ddot{q} + mg_0 = F$$

(no friction)

$$e(t) = q_d - q(t)$$

$$\dot{e}(t) = -\dot{q}(t)$$

$$F = k_P(q_d - q) - k_D\dot{q}$$

(PD \Rightarrow steady-state error $e = q_d - \bar{q}$, with $\bar{q} = q_d - \frac{mg_0}{k_P}$)

$$F = k_P(q_d - q) - k_D\dot{q} + mg_0$$

(PD + gravity cancellation \Rightarrow regulation $\forall k_P > 0, k_D > 0$)

$$F = k_P(q_d - q) - k_D\dot{q} + k_I \int_0^t (q_d - q(\tau)) d\tau$$

(PID \Rightarrow regulation $\forall k_I > 0, k_D > 0, k_P > \frac{mk_I}{k_D} > 0$)

with global exponential stability!

Laplace domain analysis: $e(s) = \mathcal{L}[e(t)]$, $d(s) = \mathcal{L}[mg_0]$ + Routh criterion

$$\frac{e(s)}{d(s)} = W_d(s) = \frac{s}{ms^3 + k_D s^2 + k_P s + k_I}$$

3	m	k_P	
2	k_D	k_I	
1	$(k_D k_P - m k_I)/k_D$		
0	k_I		



Saturated PID control

- more in general, one can prove **global** asymptotic stability of $(q_d, 0)$, under **lower bound limitations** for K_P, K_I, K_D (depending on suitable “bounds” on the terms in the dynamic model), for a **nonlinear PID law**

$$u(t) = K_P(q_d - q(t)) + K_I \int_0^t \Phi(q_d - q(\tau)) d\tau - K_D \dot{q}$$

where $\Phi(q_d - q)$ is a **saturation-type** function, such as

$$\Phi(x) = \begin{cases} \sin x, & |x| \leq \pi/2 \\ 1, & x > \pi/2 \\ -1, & x < -\pi/2 \end{cases} \quad \text{or} \quad \Phi(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

(see paper by R. Kelly, IEEE TAC, 1998; available as extra material on the course web)



Limits of robot regulation controllers

- response times needed for reaching the desired steady state are not easily predictable in advance
 - depend heavily on robot dynamics, on PD/PID gains, on the required total displacement, and on the interested area of the robot workspace
 - integral term (when present) needs some time to “unload” itself from the error history accumulated during transients
 - large initial errors are stored in the integral term
 - anti-windup schemes stop the integration when commands saturate
 - ... an intuitive explanation for the success of “saturated” PID law
- control efforts in the few first instants of motion typically exceed by far those required at steady state
 - especially for high positional gains
 - may lead to saturation (hard nonlinearity) of robot actuators



Regulation in industrial robots

- in industrial robots, the planner generates a **reference trajectory** $q_r(t)$ even when the task requires **only** positioning/regulation of the robot
 - “smooth” enough, with a user-defined **transfer time** T
 - reference trajectory interpolates initial and final desired position

$$q_r(0) = q(0) \quad q_r(t \geq T) = q_d$$

- $q_r(t)$ is used within a control law of the form

$$u = K_P(q_r(t) - q) + K_D(\dot{q}_r(t) - \dot{q}) + g(q)$$

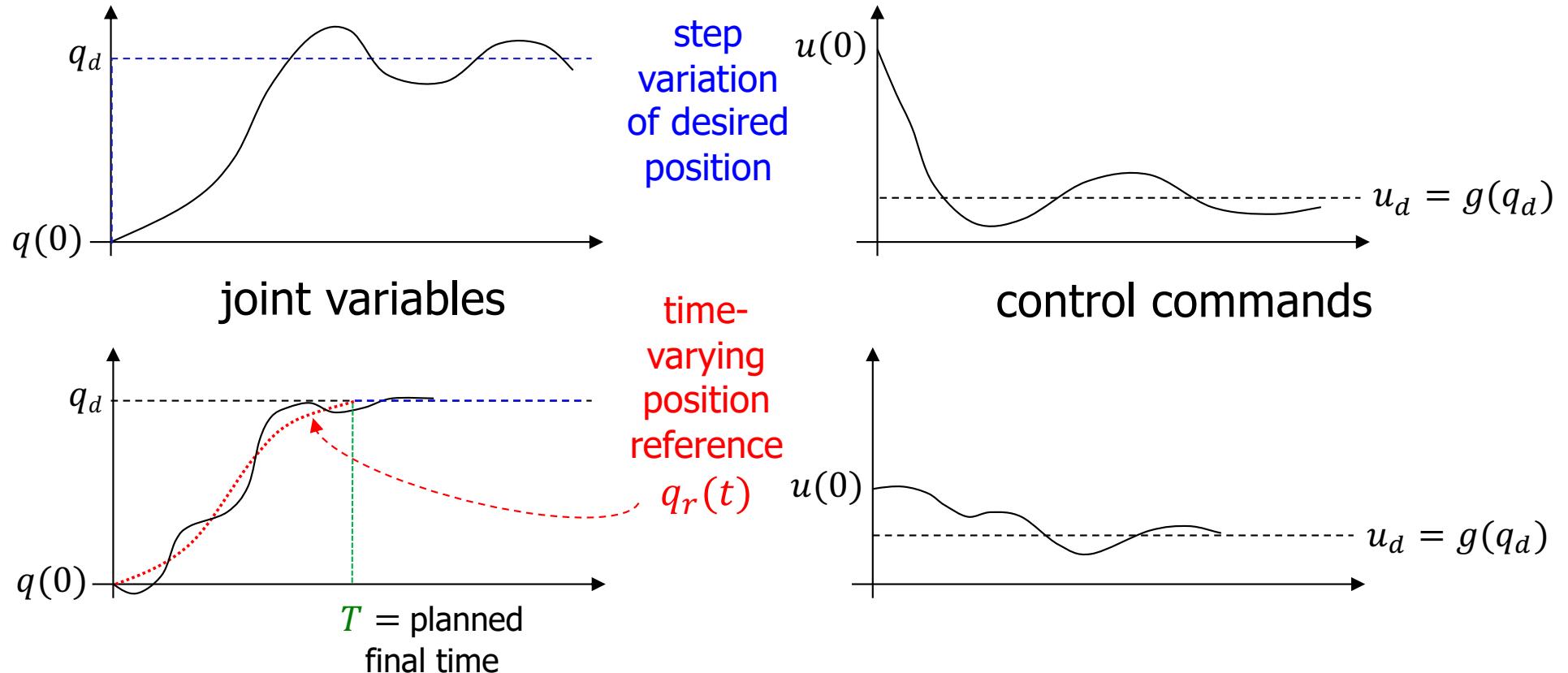
↑
often neglected

e.g., PD with
gravity
cancellation

- in this way, the position error is **initially zero**
- robot motion stays only “in the vicinity” of the reference trajectory until $t = T$, typically with small position errors (gains can be **larger!**)
- final regulation** is only a “local” problem ($e(T) = q_d - q(T)$ is small)



Qualitative comparison



- **no saturation** of commands: in principle, much larger gains can be used
- better **prediction of settling times**: local exponential convergence (designed on the linear approximation of the robot dynamics around $(q_d, 0)$)
- “fine tuning” of control gains made easier, but still a **tedious** and **delicate task**



Quantitative comparison

planar 2R arm

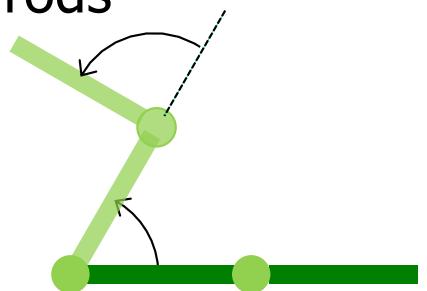
m_1	10 [kg]
m_2	5 [kg]
l_1	0.5 [m]
l_2	0.5 [m]
d_1	0.25 [m]
d_2	0.25 [m]
I_1	5/24 [kg m ²]
I_2	5/48 [kg m ²]

robot data: links are uniform thin rods

no gravity (horizontal plane)

rest-to-rest motion task:

$$q(0) = (0, 0) \text{ (straight)} \rightarrow q_d = (\pi/3, \pi/2)$$



interpolating trajectory: cubic polynomials

three case studies

a) low gains (overdamped) $K_P = \text{diag}\{80, 40\}, K_D = \text{diag}\{60, 30\}$

vs interpolating trajectory in $T = 2$ s

b) medium gains (very overdamped) $K_P = \text{diag}\{200, 100\}, K_D = \text{diag}\{200, 100\}$

vs interpolating trajectory in $T = 2$ s

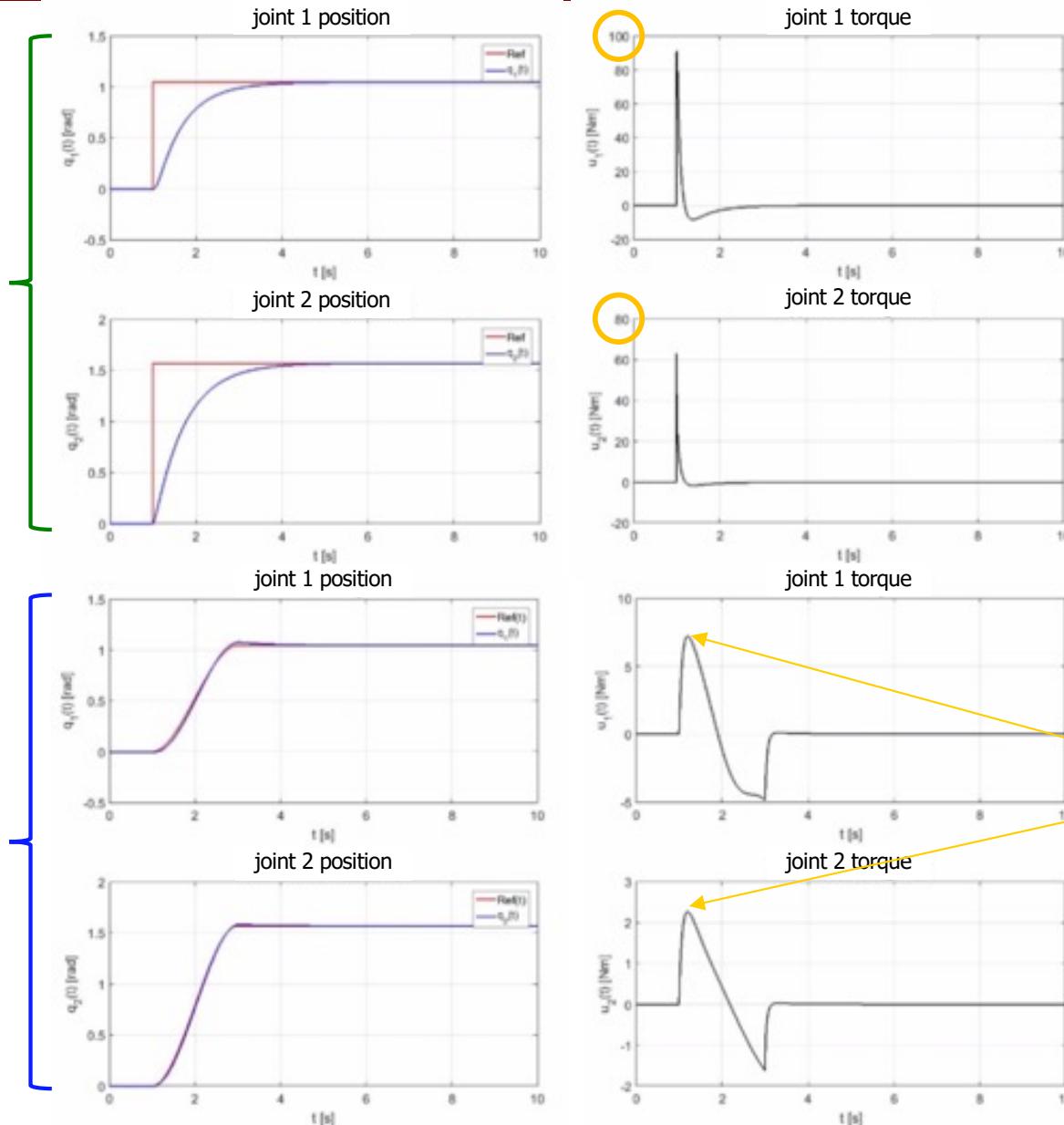
c) high gains $K_P = \text{diag}\{1250, 180\}, K_D = \text{diag}\{200, 70\}$

vs interpolating trajectory in $T = 1$ s, with torque saturation $u_{1,\max} = 30$ Nm,
 $u_{2,\max} = 10$ Nm



Comparison on a planar 2R arm – case a

PD with low gains
 $K_P = \text{diag}\{80, 40\}$
 $K_D = \text{diag}\{60, 30\}$
 (overdamped)



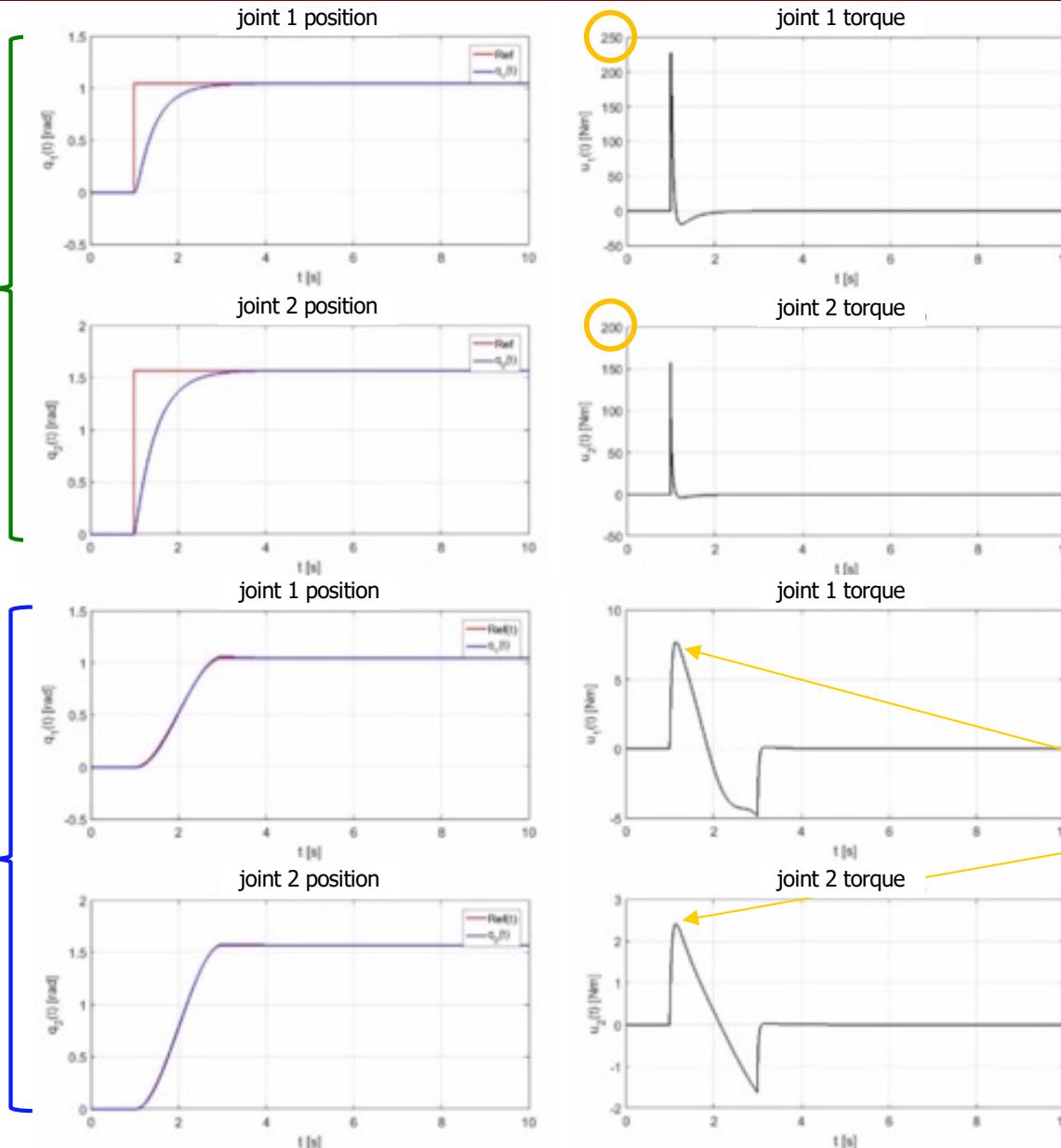
PD with same gains
 on interpolating
 trajectory of $T = 2$ s

a reduction of the
 peak control effort
 by a factor of 100
 on joint 1 &
 by a factor of 30
 on joint 2

max torques
 of 7 and 2.3 Nm

Comparison on a planar 2R arm – case b

PD with medium gains
 $K_P = \text{diag}\{200, 100\}$
 $K_D = \text{diag}\{200, 100\}$
 (very overdamped)



PD with same gains
 on interpolating
 trajectory of $T = 2$ s

even stronger
 peak reduction,
 with similar total
 control effort,
 plus improved
 tracking of
 reference trajectory
 on both joints

max torques
 of 7.5 and 2.4 Nm

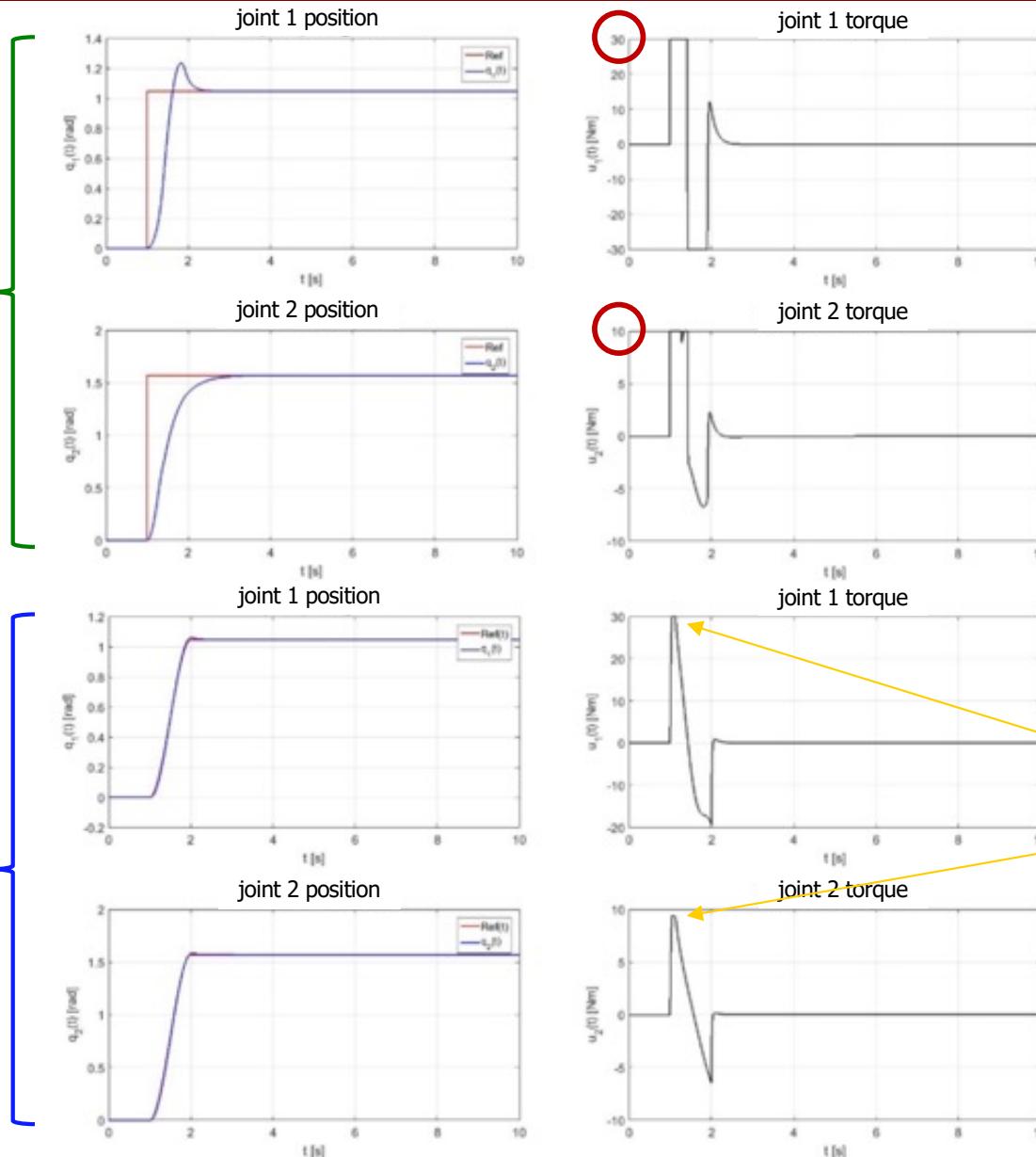


Comparison on a planar 2R arm – case c

PD with high gains
 $K_P = \text{diag}\{1250, 180\}$
 $K_D = \text{diag}\{200, 70\}$

torque saturation
 $u_{1,\max} = 30 \text{ Nm}$
 $u_{2,\max} = 10 \text{ Nm}$

PD with same gains
 on interpolating
 trajectory of $T = 1 \text{ s}$



position overshoot
 and long saturations
 are avoided,
 with **very good**
tracking of the
 faster reference
 trajectory

max torques
 of 30 and 9.5 Nm



Robotics 2

Iterative Learning for Gravity Compensation

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Control goal

- regulation of arbitrary equilibrium configurations in the presence of gravity
 - without explicit knowledge of robot dynamic coefficients (nor of the structure of the gravity term)
 - without the need of “high” position gain
 - without complex conditions on the control gains
- based on an iterative control scheme that uses
 1. PD control on joint position error + constant feedforward term
 2. iterative update of the feedforward term at successive steady-state conditions
- derive sufficient conditions for the global convergence of the iterative scheme with zero final error



Preliminaries

- robot dynamic model

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u$$

- available bound on the gradient of the gravity term

$$\left\| \frac{\partial g(q)}{\partial q} \right\| \leq \alpha$$

- regulation attempted with a joint-based PD law
(without gravity cancellation nor compensation)

$$u = K_P(q_d - q) - K_D \dot{q} \quad K_P > 0, K_D > 0$$

- at steady state, there is a **non-zero error** left

$$q = \bar{q}, \dot{q} = 0 \quad g(\bar{q}) = K_P(q_d - \bar{q}) \quad \bar{e} = q_d - \bar{q} \neq 0$$



Iterative control scheme

- **control law** at the i -th iteration (for $i = 1, 2, \dots$)

$$u = \gamma K_P(q_d - q) - K_D \dot{q} + u_{i-1} \quad \gamma > 0$$

with a constant compensation term u_{i-1} (**feedforward**)

- $K_P > 0, K_D > 0$ are chosen **diagonal** for simplicity
- q_0 is the initial robot configuration
- $u_0 = 0$ is the 'easiest' initialization of the feedforward term
- at the **steady state** of the i -th iteration ($q = q_i, \dot{q} = 0$), one has

$$g(q_i) = \gamma K_P(q_d - q_i) + u_{i-1}$$

- **update law** of the compensation term (for next iteration)

$$u_i = \gamma K_P(q_d - q_i) + u_{i-1} \quad [= g(q_i)]$$

← for implementation → [for analysis]



Convergence analysis

Theorem

- (a) $\lambda_{\min}(K_P) > \alpha$
- (b) $\gamma \geq 2$

guarantee that the sequence $\{q_0, q_1, q_2, \dots\}$ converges to q_d (and $\dot{q} = 0$) from **any** initial value q_0 (and \dot{q}_0), i.e., **globally**

- condition (a) is sufficient for the global asymptotic stability of the desired equilibrium state when using

$$u = K_P(q_d - q) - K_D \dot{q} + g(q_d)$$

with a **known** gravity term and diagonal gain matrices

- the additional **sufficient** condition (b) guarantees the convergence of the iterative scheme, yielding

$$\lim_{i \rightarrow \infty} u_i = g(q_d)$$



Proof

- let $e_i = q_d - q_i$ be the error at the end of the i -th iteration; based on the update law, it is $u_i = g(q_i)$ and thus

$$\begin{aligned}\|u_i - u_{i-1}\| &= \|g(q_i) - g(q_{i-1})\| \leq \alpha \|q_i - q_{i-1}\| \\ &\leq \alpha (\|e_i\| + \|e_{i-1}\|)\end{aligned}$$

adding and subtracting q_d

- on the other hand, from the update law it is

$$\|u_i - u_{i-1}\| = \gamma \|K_P e_i\|$$

- combining the two above relations under (a), we have

$$\gamma \alpha \|e_i\| < \gamma \lambda_{\min}(K_P) \|e_i\| \leq \gamma \|K_P e_i\| \leq \alpha (\|e_i\| + \|e_{i-1}\|)$$

$$\text{or } \|e_i\| < \frac{1}{\gamma} (\|e_i\| + \|e_{i-1}\|)$$



Proof (cont)

- condition (b) guarantees that the error sequence $\{e_0, e_1, e_2, \dots\}$

$$\|e_i\| < \frac{\frac{1}{\gamma}}{1 - \frac{1}{\gamma}} \|e_{i-1}\| = \frac{1}{\gamma - 1} \|e_{i-1}\|$$

is a **contraction mapping**, so that

$$\lim_{i \rightarrow \infty} \|e_i\| = 0$$

with asymptotic convergence from any initial state



⇒ the robot progressively approaches the desired configuration through **successive steady-state conditions**

- K_P and K_D affect each transient phase
- coefficient γ drives the convergence rate of intermediate steady states to the final one



Remarks

- combining (a) and (b), the sufficient condition only requires the **doubling** of the proportional gain w.r.t. the known gravity case

$$\widehat{K}_P = \gamma K_P \quad \rightarrow \quad \lambda_{\min}(\widehat{K}_P) > 2\alpha$$

- for a diagonal \widehat{K}_P , this condition implies a (positive) lower bound on the single diagonal elements of the matrix
- again, it is only a **sufficient** condition
 - the scheme may converge even if this condition is violated ...
- the scheme can be interpreted as using an **integral term**
 - updated only in correspondence of a **discrete sequence of time instants**
 - with a guaranteed **global** convergence (and implicit stability)



Numerical results

- 3R robot with uniform links, moving in the vertical plane

$$l_1 = l_2 = l_3 = 0.5 \text{ [m]}$$

$$m_1 = 30, m_2 = 20, m_3 = 10 \text{ [kg]} \quad \rightarrow \quad \boxed{\alpha \cong 400}$$

- with saturations of the actuating torques

$$U_{1,\max} = 800, U_{2,\max} = 400, U_{3,\max} = 200 \text{ [Nm]}$$

- three cases, from the downward position $q_0 = (0, 0, 0)$

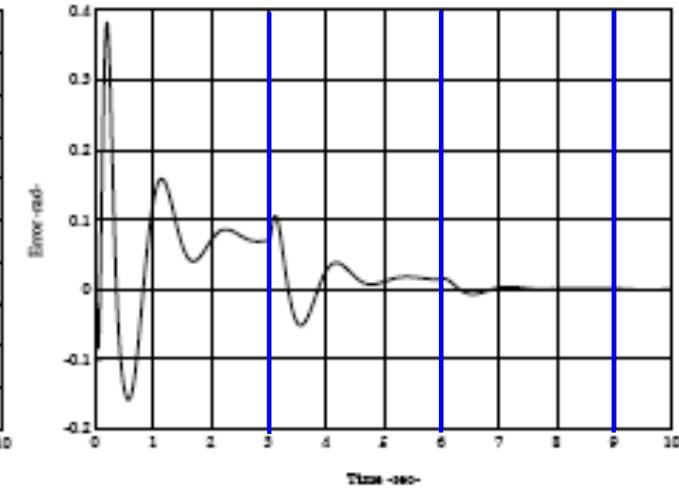
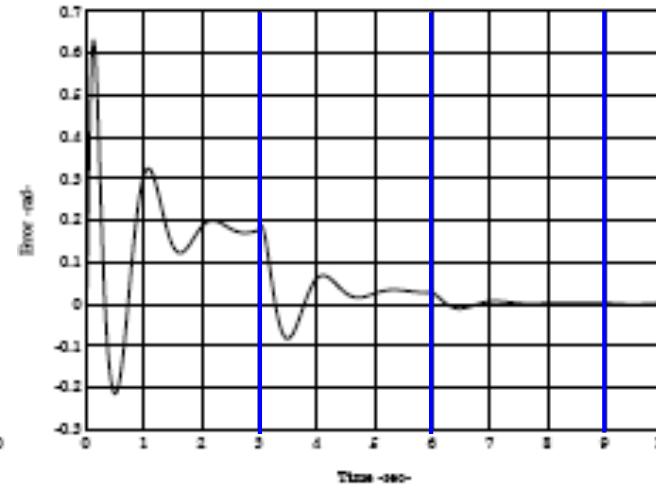
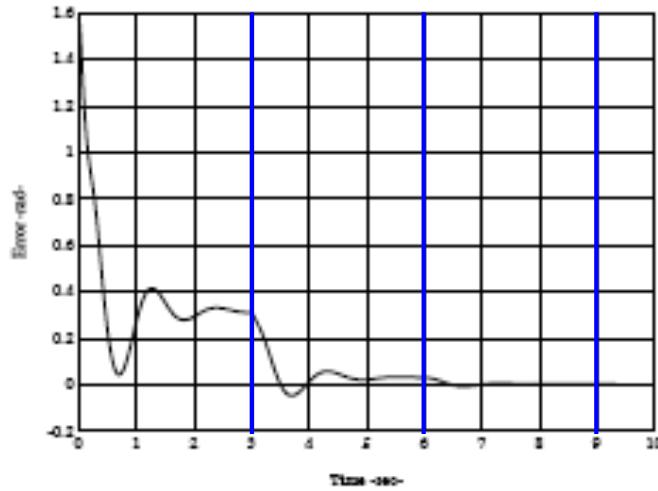
$$\text{I: } q_d = (\pi/2, 0, 0) \quad \left. \right\} \quad \left\{ \begin{array}{l} \hat{K}_P = \text{diag}\{1000, 600, 280\} \\ K_D = \text{diag}\{200, 100, 20\} \end{array} \right.$$

$$\text{II: } q_d = (3\pi/4, 0, 0) \quad \left. \right\} \quad \left\{ \begin{array}{l} \hat{K}_P = \text{diag}\{1000, 600, 280\} \\ K_D = \text{diag}\{200, 100, 20\} \end{array} \right.$$

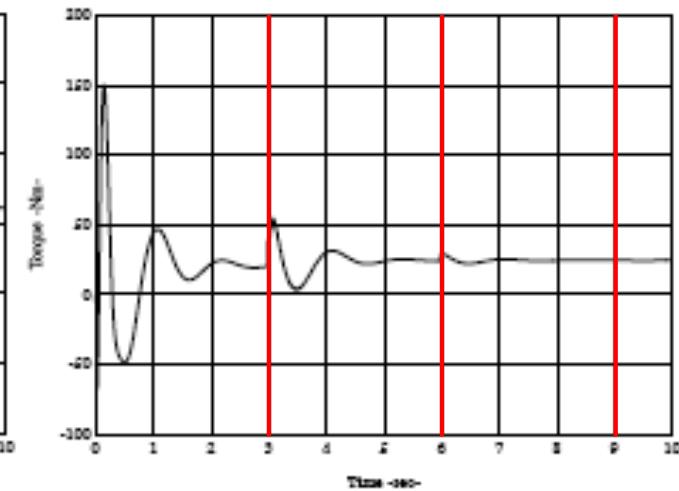
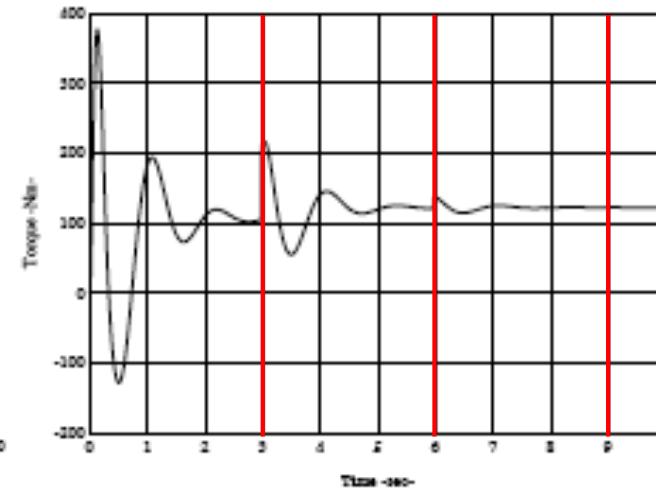
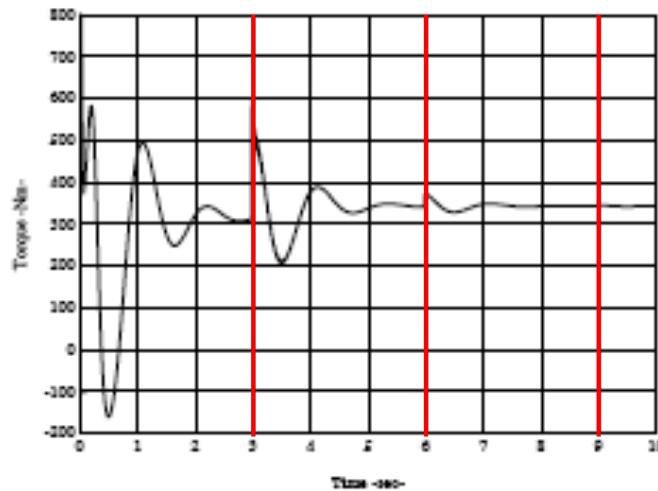
$$\text{III: } q_d = (3\pi/4, 0, 0) \quad \left. \right\} \quad \left\{ \begin{array}{l} \hat{K}_P = \text{diag}\{500, 500, 500\} \\ K_D = \text{as before} \end{array} \right.$$



Case I: $q_d = (\pi/2, 0, 0)$



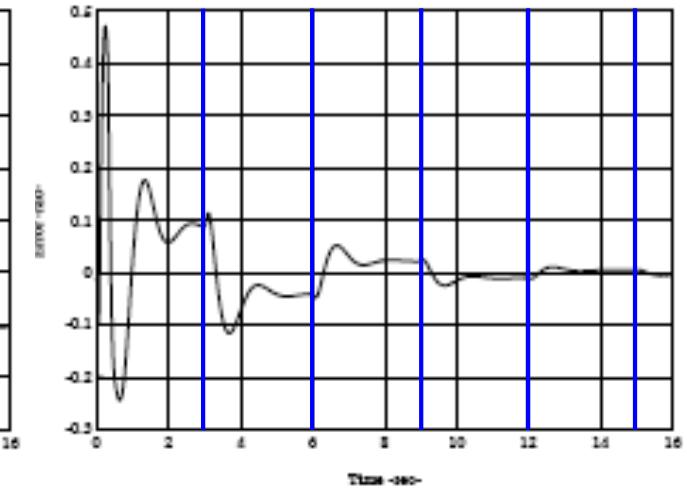
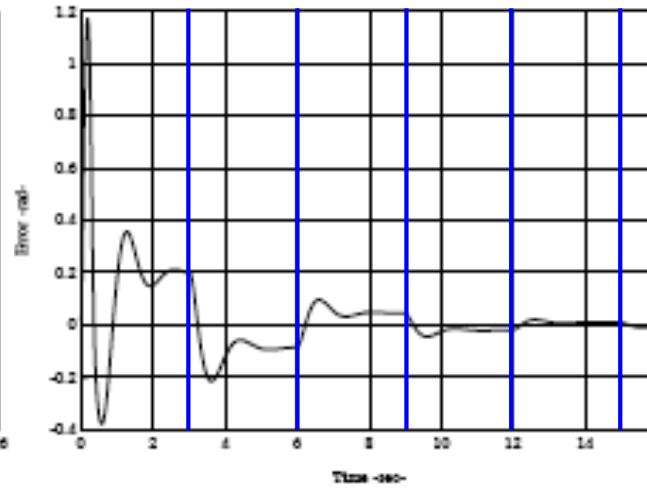
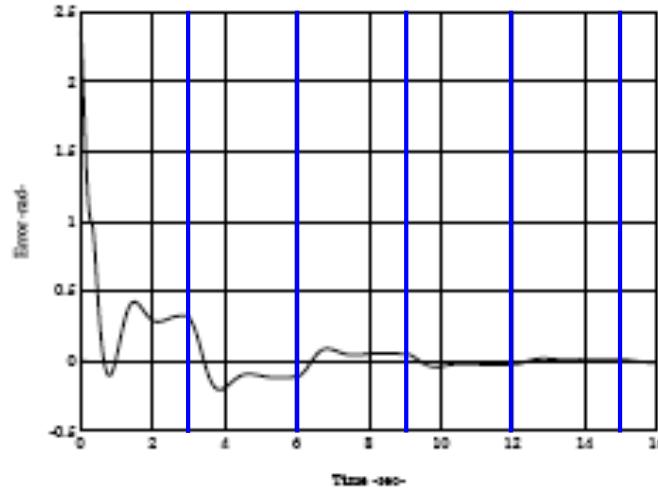
joint position errors (zero after 3 updates)



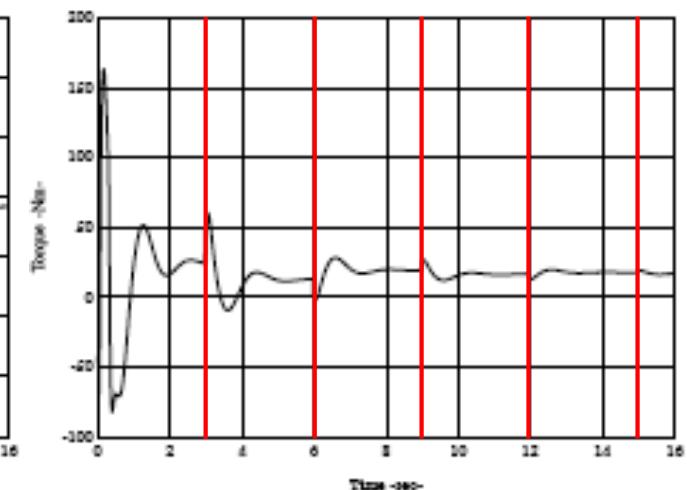
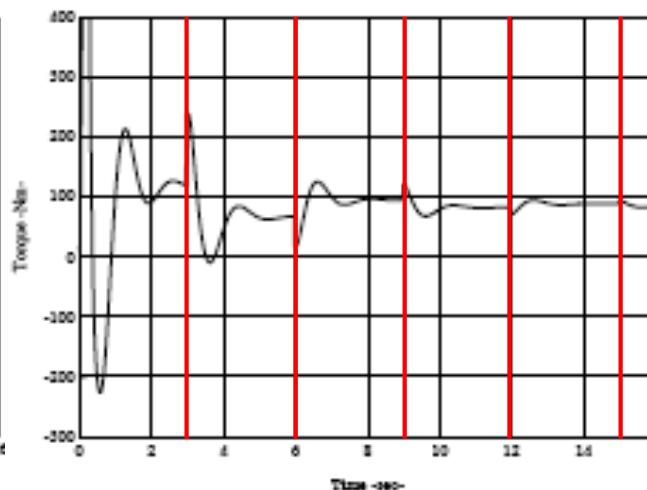
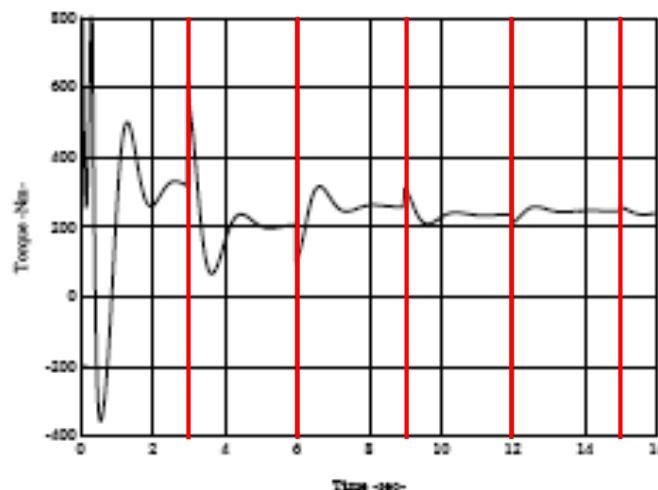
control torques



Case II: $q_d = (3\pi/4, 0, 0)$



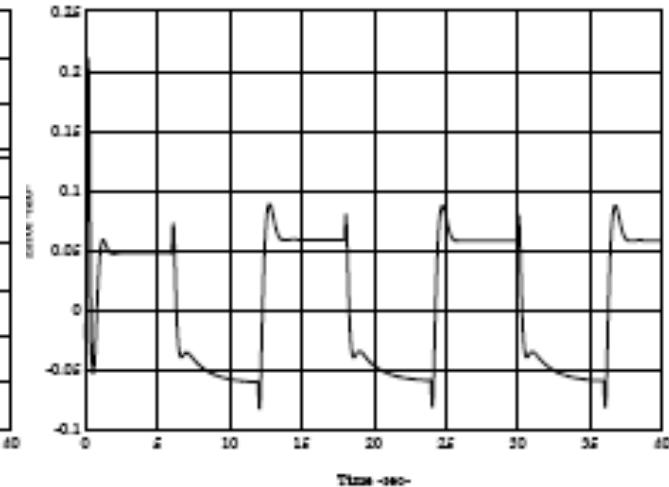
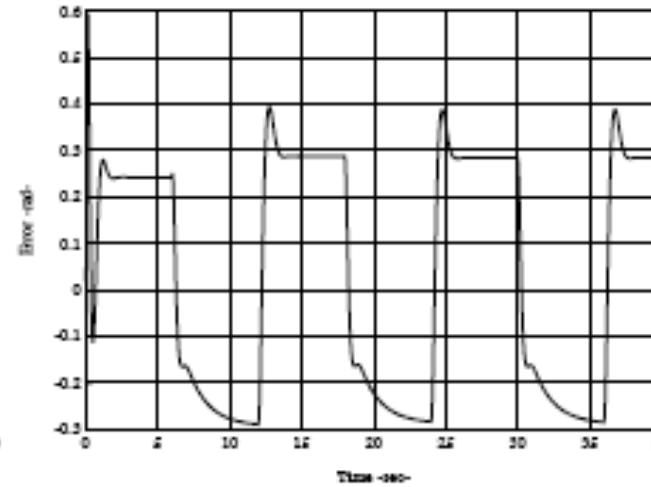
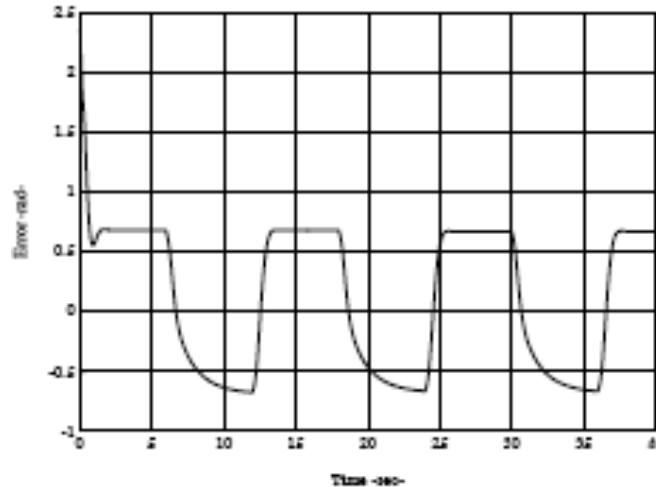
joint position errors (zero after 5 updates)



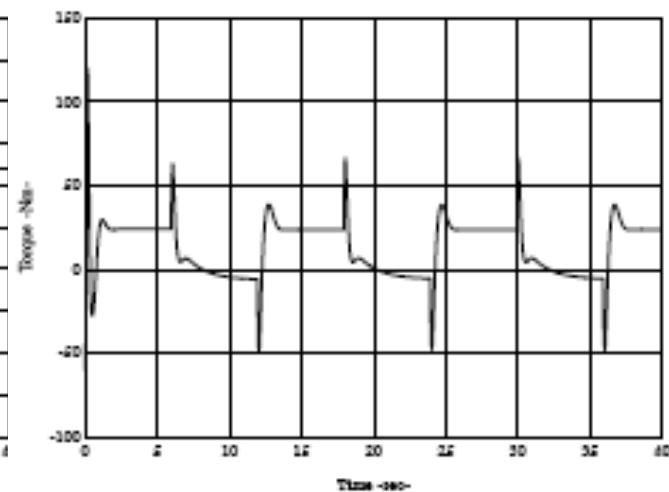
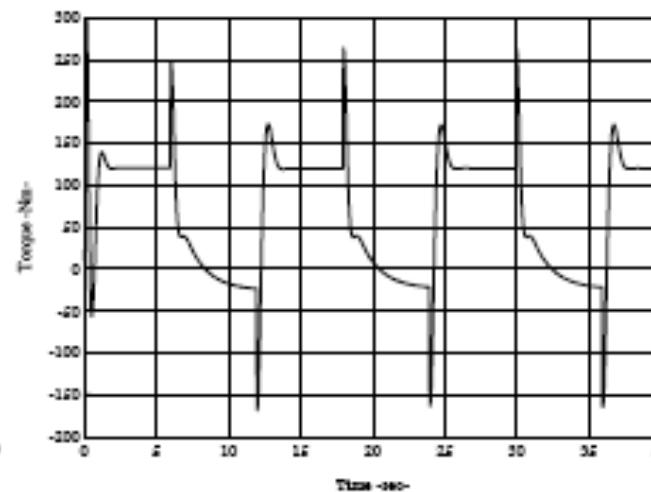
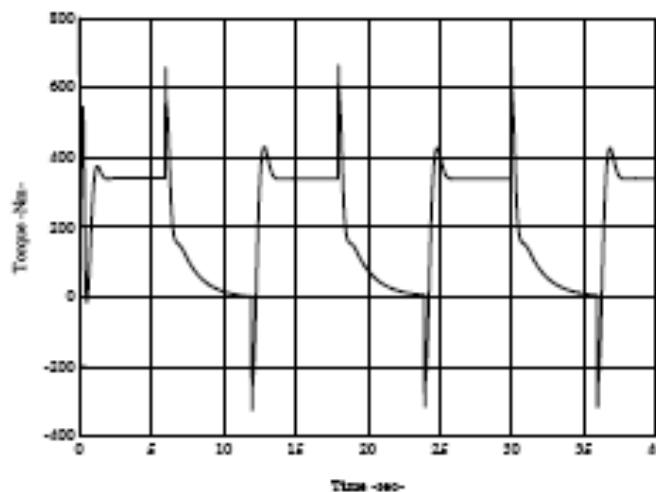
control torques



Case III: $q_d = (3\pi/4, 0, 0)$, reduced gains



joint position errors (limit cycles, no convergence!)



control torques



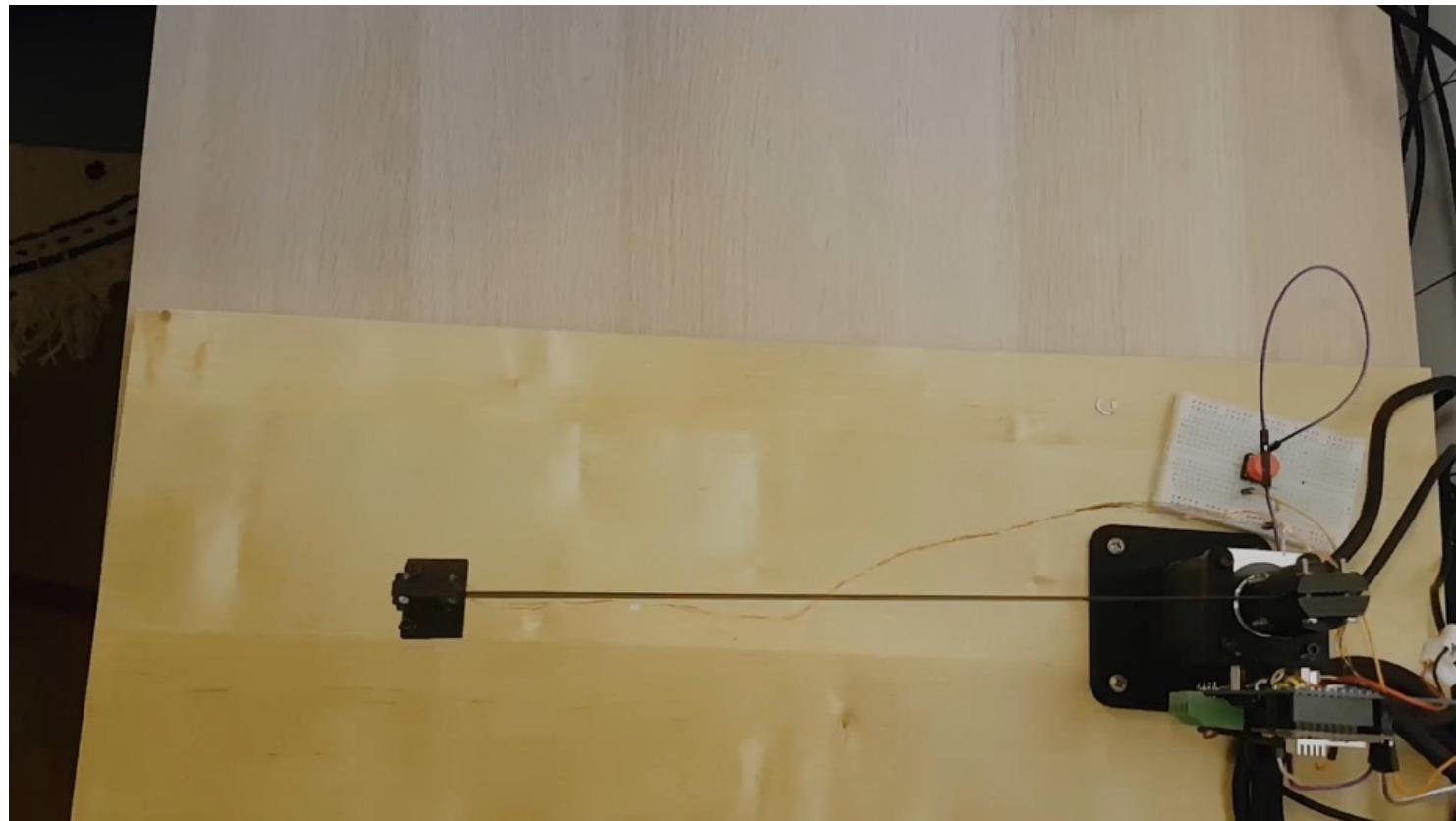
Final comments

- only **few iterations** are needed for obtaining convergence, learning the **correct gravity compensation** at the desired q_d
- **Sufficiency** of the condition on the P gain
 - even if violated, convergence can **still be** obtained (first two cases); otherwise, a limit motion cycle takes place between two equilibrium configurations that are **both incorrect** (as in the third case)
 - this shows 'how far' is sufficiency from **necessity**
- analysis can be refined to get lower bounds on the K_{P_i} (diagonal case) that are smaller, but still sufficient for convergence
 - intuitively, lower values for K_{P_i} should still work for distal joints
- in practice, update of the feedforward term occurs when the robot is **close enough to a steady state** (joint velocities and position variations are below **suitable thresholds**)



Control experiments with flexible robots without gravity

even for just a **single** but **flexible link** in the absence of gravity, a **rest-to-rest** maneuver **without residual oscillations** is difficult to be performed by a **pure PD joint control** action



[video](#)

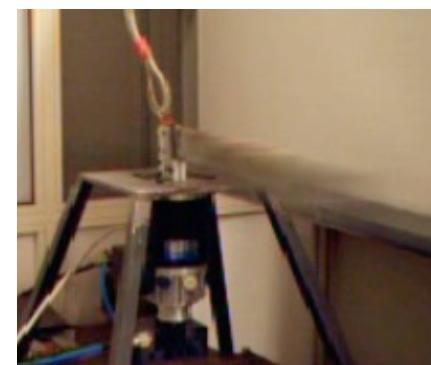
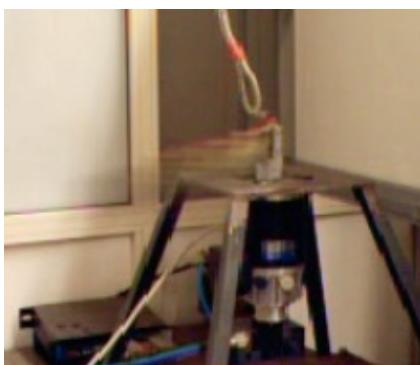
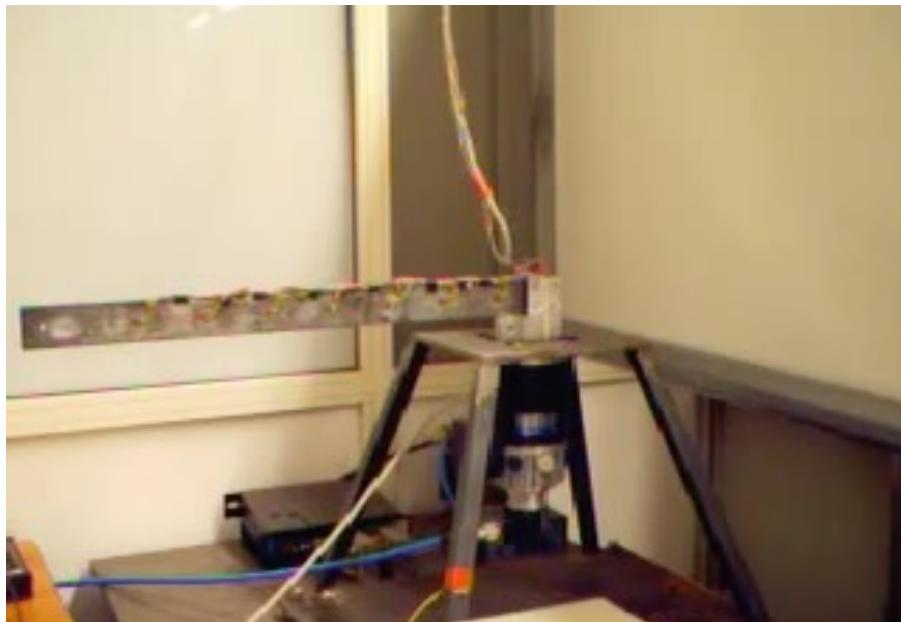
ICRA 2023

S. Drost. P. Pustina. F. Angelini, A. De Luca, G. Smit, C. Della Santina,
"Experimental validation of functional iterative learning control on a one-link flexible arm"



Control experiments with flexible robots without gravity

video



rest-to-rest maneuver in given motion time
for a single flexible link (PD + feedforward)

video

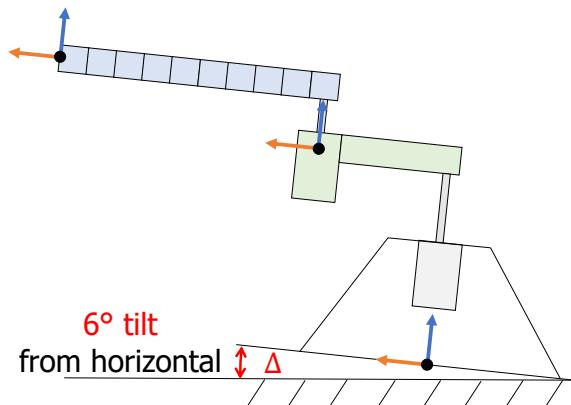


end-effector trajectory tracking for FlexArm
—a planar 2R robot with flexible forearm

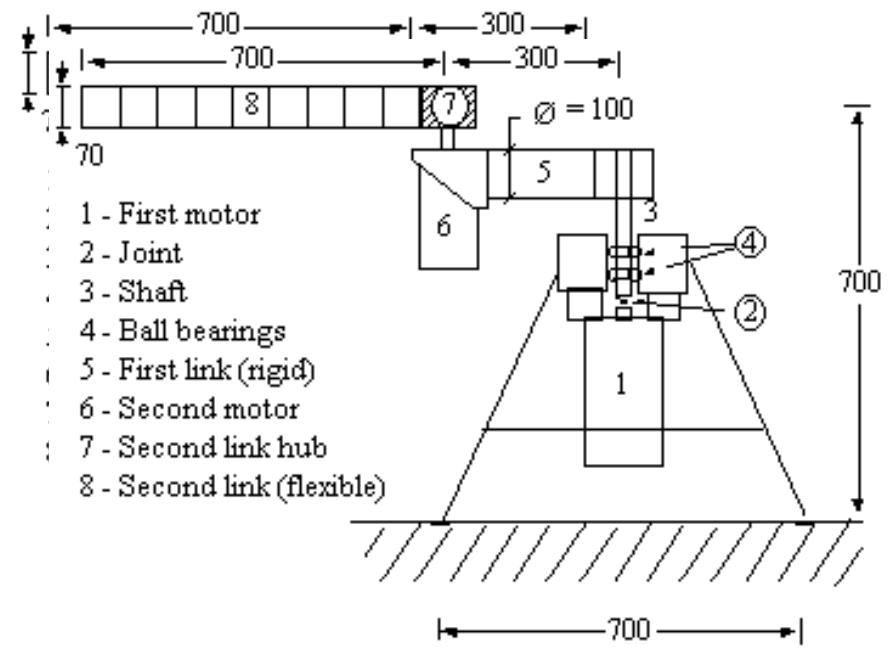
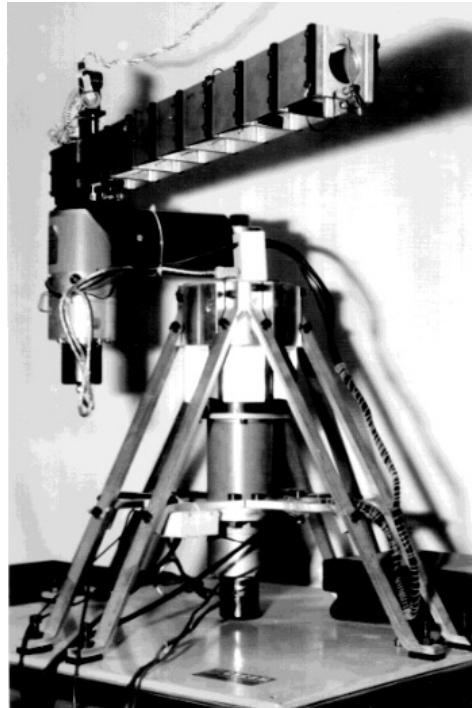


Extension to flexible robots

- the same iterative learning control approach has been extended to position regulation in robots with **flexible joints and/or links** under gravity
 - at the motor/joint level
 - at the Cartesian level (end-effector tip position, **beyond flexibility**), using a **double iterative scheme**
- experimentally validated on the **two-link FlexArm @ DIS** (now DIAG!)

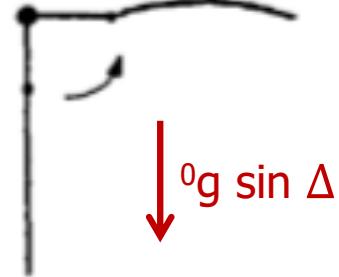


with supporting base
tilted by $\Delta \approx 6^\circ$
(inclusion of gravity)

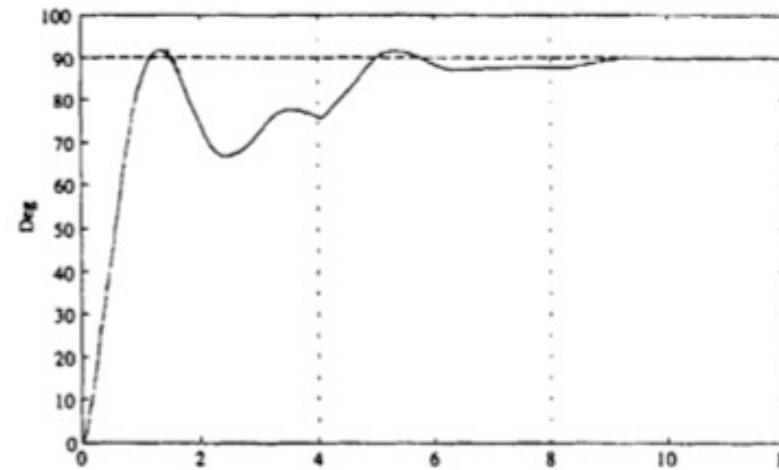




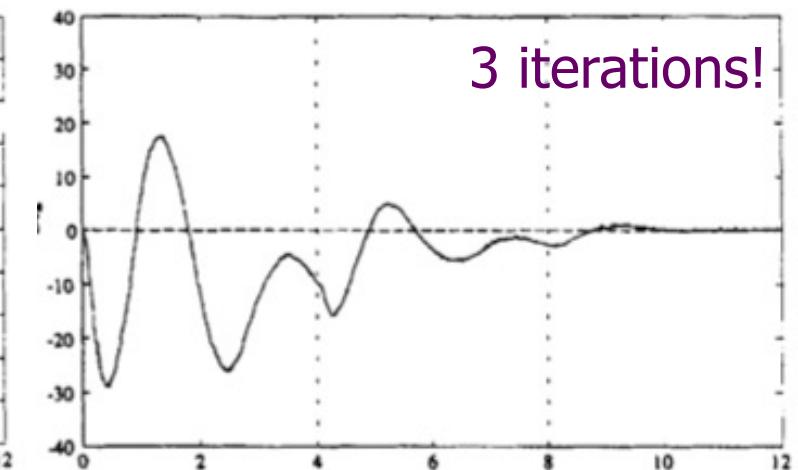
Experimental results for tip regulation



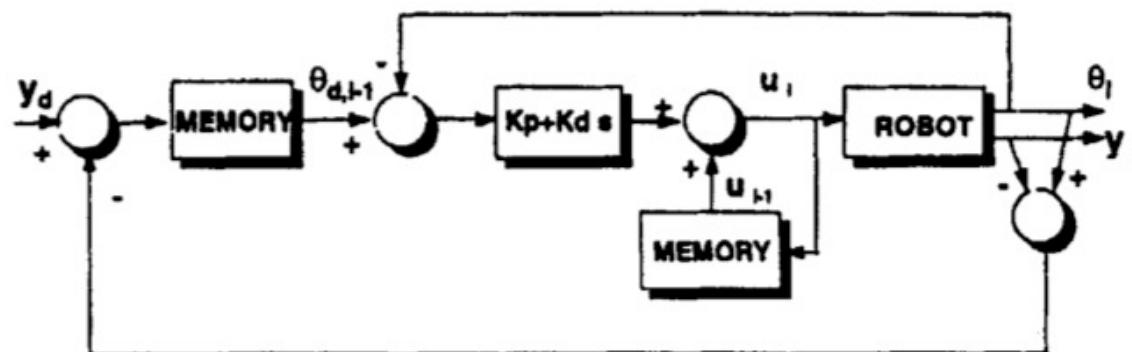
motion task:
 $(0^\circ, 0^\circ) \Rightarrow (90^\circ, 0^\circ)$



first link position

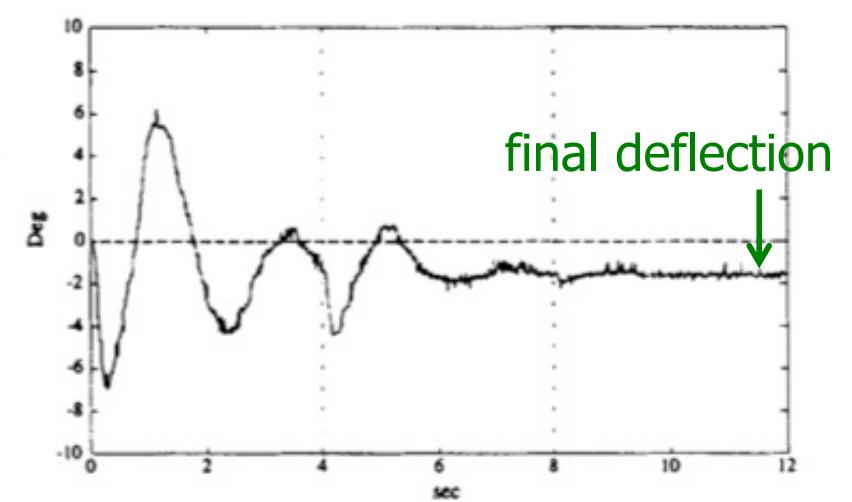


tip angle w.r.t. first link



double iterative scheme

De Luca, Panzieri: Int J Adapt Cont & Sign Proc, 1996
 (factor $\gamma \rightarrow 1/\beta$ in the original paper)



second link deflection



Robotics 2

Trajectory Tracking Control

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Inverse dynamics control

given the robot **dynamic model** with N joints

$$M(q)\ddot{q} + n(q, \dot{q}) = u$$

$c(q, \dot{q}) + g(q) + \text{friction model}$

and a twice-differentiable **desired trajectory** for $t \in [0, T]$

$$q_d(t) \rightarrow \dot{q}_d(t), \ddot{q}_d(t)$$

applying the **feedforward** torque in **nominal conditions**

$$u_d = M(q_d)\ddot{q}_d + n(q_d, \dot{q}_d)$$

yields exact reproduction of the desired motion, provided that $q(0) = q_d(0), \dot{q}(0) = \dot{q}_d(0)$ (initial **matched state**)



In practice ...

a number of differences from the **nominal condition**

- initial state is “**not matched**” to the desired trajectory $q_d(t)$
- **disturbances** on the actuators, from unexpected collisions, truncation errors on data, ...
- **inaccurate knowledge** of robot dynamic parameters $\pi \rightarrow \hat{\pi}$ (link masses, inertias, center of mass positions)
- **unknown** value of the carried payload
- presence of **unmodeled** dynamics (complex friction phenomena, transmission elasticity, ...)



require the use of **feedback information**



Introducing feedback

$$\hat{u}_d = \hat{M}(q_d)\ddot{q}_d + \hat{n}(q_d, \dot{q}_d)$$

with \hat{M} , \hat{n} estimates of terms
(or coefficients) in the dynamic model

note: \hat{u}_d can be computed off line [e.g., by $\hat{NE}_\alpha(q_d, \dot{q}_d, \ddot{q}_d)$]

feedback is introduced to make the control scheme more robust

different possible implementations depending on
amount of computational load share

- OFF LINE (↔ open loop)
- ON LINE (↔ closed loop)

two-step control design:

1. compensation (feedforward) or cancellation (feedback) of nonlinearities
2. synthesis of a linear control law stabilizing the trajectory error to zero



A series of trajectory controllers

(assuming the nominal case: $\hat{M} = M, \hat{n} = n$)

1. inverse dynamics compensation (FFW) + PD

$$u = \hat{u}_d + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})$$

local stabilization
of trajectory error
 $e(t) = q_d(t) - q(t)$
global if additional
conditions on
 K_P and K_D

2. inverse dynamics compensation (FFW) + variable PD

$$u = \hat{u}_d + \hat{M}(q_d)[K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})]$$

3. feedback linearization (FBL) + [PD+FFW] = "COMPUTED TORQUE"

$$u = \hat{M}(q)[\ddot{q}_d + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})] + \hat{n}(q, \dot{q})$$

4. feedback linearization (FBL) + [PID+FFW]

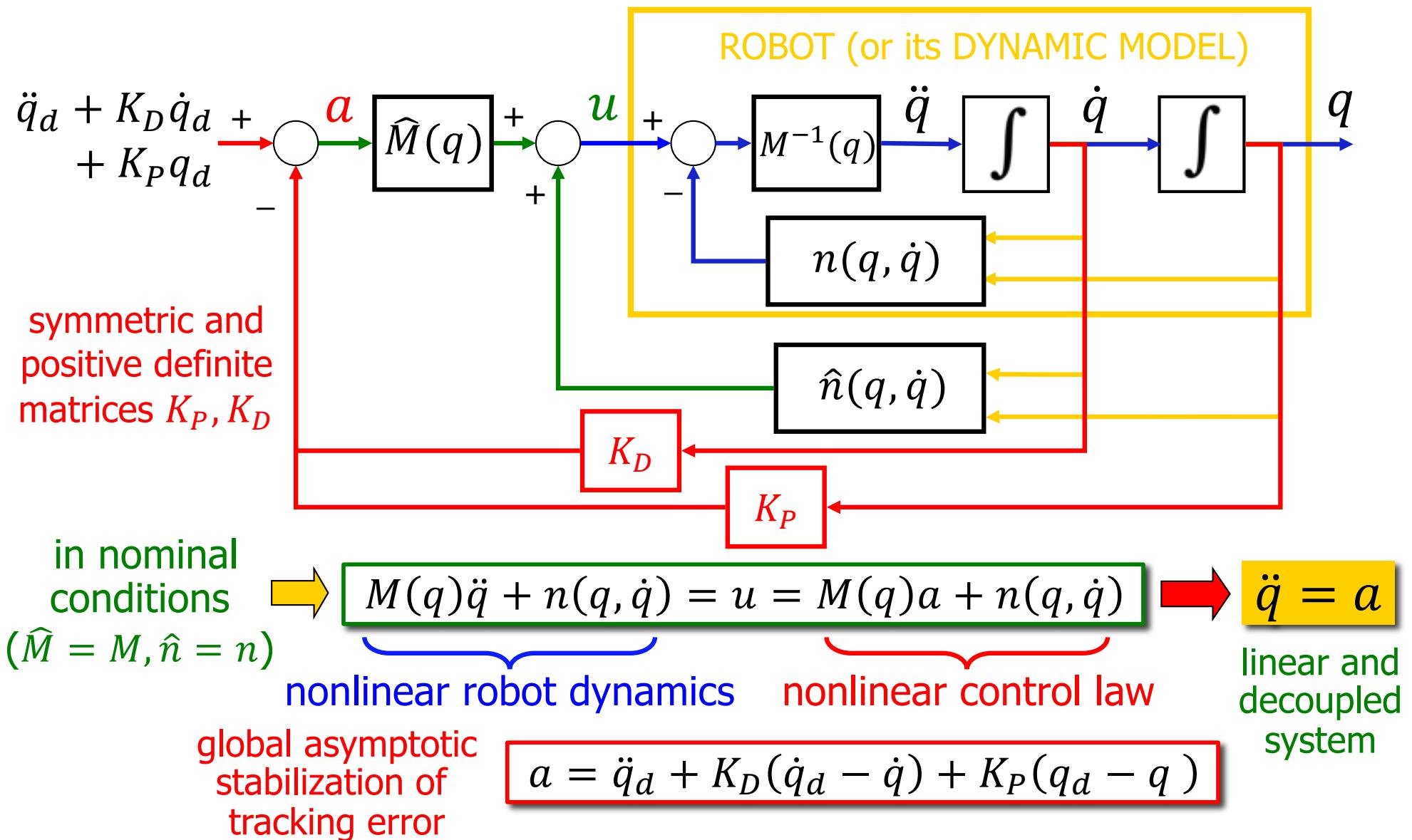
$$u = \hat{M}(q) \left[\ddot{q}_d + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q}) + K_I \int (q_d - q) dt \right] + \hat{n}(q, \dot{q})$$

global stabilization for any $K_P > 0, K_D > 0$ (and not too large $K_I > 0$)

more robust to small uncertainties/disturbances, even if more complex to implement in real time

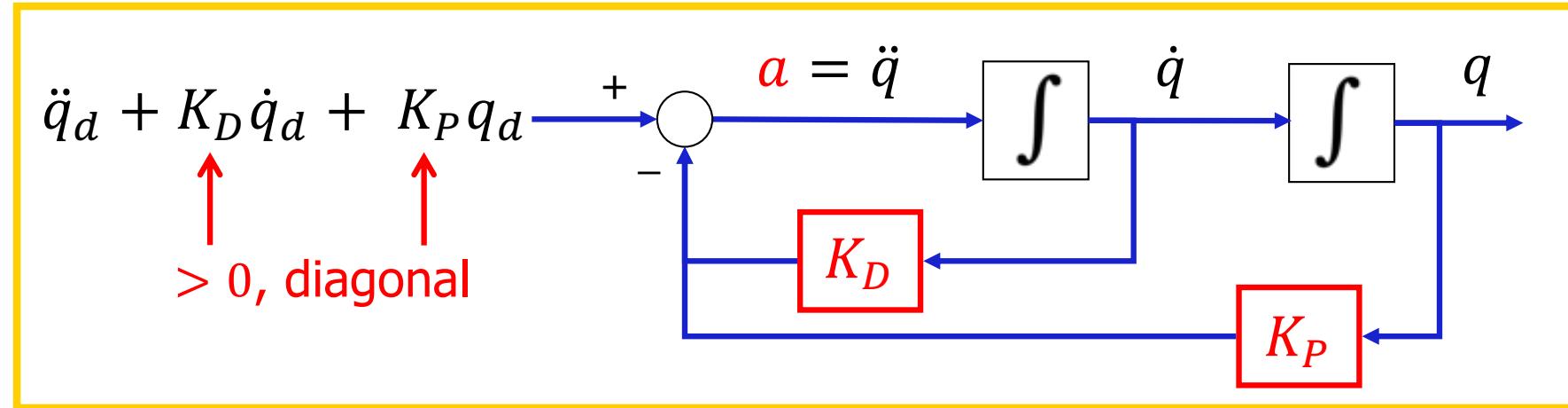


Feedback linearization control





Interpretation in the linear domain



under feedback linearization control, the robot has a dynamic behavior that is **invariant**, **linear** and **decoupled** in its whole state space ($\forall(q, \dot{q})$)

linearity

a unitary mass ($m = 1$) in the joint space !!

error transients $e_i = q_{di} - q_i \rightarrow 0$ exponentially, prescribed by K_{Pi}, K_{Di} choice

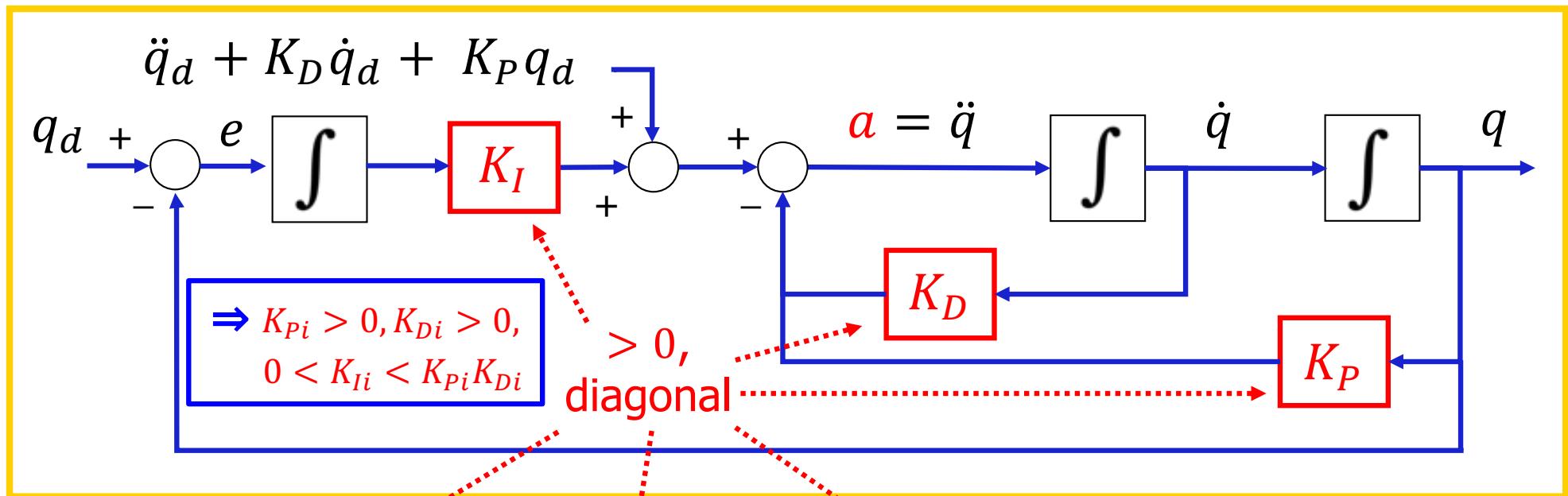
decoupling

each joint coordinate q_i evolves **independently** from the others, forced by a_i

$$\ddot{e} + K_D \dot{e} + K_P e = 0 \Leftrightarrow \ddot{e}_i + K_{Di} \dot{e}_i + K_{Pi} e_i = 0$$



Addition of an integral term: PID whiteboard...



$$\ddot{q} = \mathbf{a} = \ddot{q}_d + K_D(\dot{q}_d - \dot{q}) + K_P(q_d - q) + K_I \int (q_d - q) d\tau \quad e = q_d - q$$

$$\Rightarrow (1) \quad e_i = q_{di} - q_i \quad (i = 1, \dots, N) \quad \Rightarrow (2) \quad \ddot{e}_i + K_{Di} \dot{e}_i + K_{Pi} e_i + K_{Ii} \int e_i d\tau = 0$$

$$\mathcal{L}[e_i(t)] \Rightarrow (3) \quad \left(s^2 + K_{Di}s + K_{Pi} + K_{Ii} \frac{1}{s} \right) e_i(s) = 0$$

$$s \times \Rightarrow (4) \quad (s^3 + K_{Di}s^2 + K_{Pi}s + K_{Ii})e_i(s) = 0 \quad \Rightarrow (5)$$

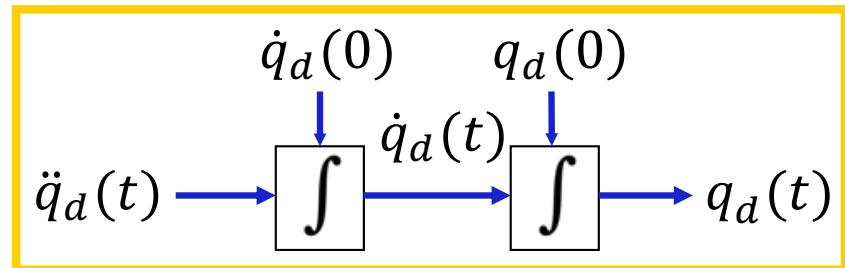
	3	1	K_{Pi}
2	K_{Di}		K_{Ii}
1	$(K_{Di}K_{Pi} - K_{Ii})/K_{Di}$		
0	K_{Ii}		

$\Rightarrow (6)$
exponential
stability
conditions by
Routh criterion



Remarks

- desired joint trajectory can be generated from **Cartesian** data



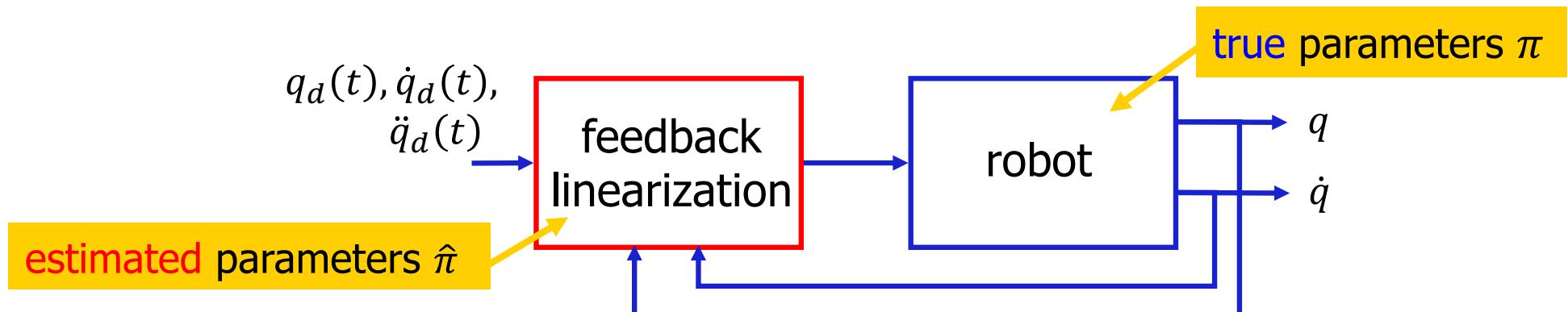
$$\ddot{p}_d(t), \dot{p}_d(0), p_d(0)$$

$$q_d(0) = f^{-1}(p_d(0))$$

$$\dot{q}_d(0) = J^{-1}(q_d(0))\dot{p}_d(0)$$

$$\ddot{q}_d(t) = J^{-1}(q_d)[\ddot{p}_d(t) - J(q_d)\dot{q}_d]$$

- real-time computation by **Newton-Euler** algo: $u_{FBL} = \widehat{NE}_\alpha(q, \dot{q}, a)$
- simulation** of feedback linearization control

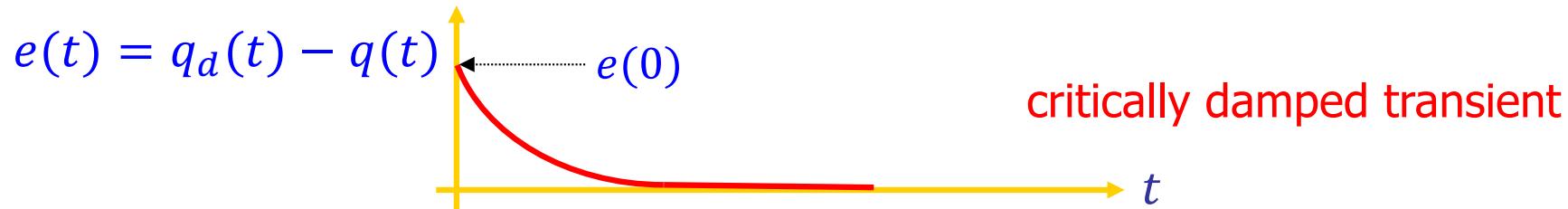


Hint:** there is no use in simulating this control law in the ideal case ($\hat{\pi} = \pi$); robot behavior will be **identical** to the linear and decoupled case of **stabilized double integrators!!



Further comments

- choice of the diagonal elements of K_P, K_D (and K_I)
 - shaping the error transients, with an eye also to motor saturations...



- parametric identification
 - to be done in advance, using the property of linearity in the dynamic coefficients of the robot dynamic model
- choice of the sampling time of a digital implementation
 - compromise between computational time and tracking accuracy, typically $T_c = 0.5 \div 10$ ms
- exact linearization by (state) feedback is a general technique of nonlinear control theory
 - can be used for robots with elastic joints, wheeled mobile robots, ...
 - non-robotics applications: satellites, induction motors, helicopters, ...



Another example of feedback linearization design

- dynamic model of robots with elastic joints

- q = link position
- θ = motor position (after reduction gears)
- B_m = diagonal matrix (> 0) of inertia of the (balanced) motors
- K = diagonal matrix (> 0) of (finite) stiffness of the joints

$4N$ state variables
 $x = (q, \theta, \dot{q}, \dot{\theta})$

$$\left\{ \begin{array}{l} M(q)\ddot{q} + c(q, \dot{q}) + g(q) + K(q - \theta) = 0 \\ B_m\ddot{\theta} + K(\theta - q) = u \end{array} \right. \quad \begin{array}{l} (1) \\ (2) \end{array}$$

- is there a control law that achieves exact linearization via feedback?

$$u = \alpha(q, \theta, \dot{q}, \dot{\theta}) + \beta(q, \theta, \dot{q}, \dot{\theta}) a$$

YES and it yields $\frac{d^4 q_i}{dt^4} = a_i, \quad i = 1, \dots, N$

linear and decoupled system:
 N chains of 4 integrators
(to be stabilized by linear control design)

Hint: differentiate (1) w.r.t. time until motor acceleration $\ddot{\theta}$ appears;
substitute this from (2); choose u so as to cancel all nonlinearities ...



Alternative global trajectory controller

$$u = M(q)\ddot{q}_d + S(q, \dot{q})\dot{q}_d + g(q) + F_V\dot{q}_d + K_P e + K_D \dot{e}$$



SPECIAL factorization such that
 $\dot{M} - 2S$ is skew-symmetric



symmetric and positive definite matrices

- **global asymptotic stability** of $(e, \dot{e}) = (0,0)$ (trajectory tracking)
- proven by **Lyapunov +Barbalat (time-varying system) +LaSalle**
- does **not** produce a complete **cancellation** of nonlinearities
 - the variables \dot{q} and \ddot{q} that appear **linearly** in the model are evaluated on the **desired** trajectory
- does **not** induce a **linear** and **decoupled** behavior of the trajectory error $e(t) = q_d(t) - q(t)$ in the closed-loop system
- however, it lends itself more easily to an **adaptive version**
- **computation:** by **4×** standard or **1×** modified NE algorithm



Analysis of asymptotic stability of the trajectory error - 1

$M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) + F_V\dot{q} = u$ robot dynamics (including friction)

control law $u = M(q)\ddot{q}_d + S(q, \dot{q})\dot{q}_d + g(q) + F_V\dot{q}_d + K_P e + K_D \dot{e}$

- Lyapunov candidate and its time derivative (with $e = q_d - q$)

$$V = \frac{1}{2}\dot{e}^T M(q)\dot{e} + \frac{1}{2}e^T K_P e \geq 0 \Rightarrow \dot{V} = \frac{1}{2}\dot{e}^T \dot{M}(q)\dot{e} + \dot{e}^T M(q)\ddot{e} + e^T K_P \dot{e}$$

- the closed-loop system equations yield

$$M(q)\ddot{e} = -S(q, \dot{q})\dot{e} - (K_D + F_V)\dot{e} - K_P e$$

- substituting and using the skew-symmetric property of $\dot{M} - 2S$

$$\dot{V} = -\dot{e}^T (K_D + F_V)\dot{e} \leq 0 \quad \dot{V} = 0 \Leftrightarrow \dot{e} = 0$$

- since the system is time-varying (due to $q_d(t)$), direct application of LaSalle theorem is NOT allowed \Rightarrow use Barbalat lemma...

$$q = q_d(t) - e, \dot{q} = \dot{q}_d(t) - \dot{e} \Rightarrow V = V(e, \dot{e}, t) = V(x, t)$$

error state x

⇒ go to
slide 10 in
block 8



Analysis of asymptotic stability of the trajectory error - 2

- since i) V is lower bounded and ii) $\dot{V} \leq 0$, we have to check only condition iii) in order to apply Barbalat lemma

$$\ddot{V} = -2\dot{e}^T(K_D + F_V)\ddot{e} \quad \dots \text{is this bounded?}$$

- from i) + ii), V is bounded $\Rightarrow e$ and \dot{e} are bounded
- assume that the desired trajectory has bounded velocity \dot{q}_d
- using the following two properties of dynamic model terms

$$0 < \alpha_m \leq \|M^{-1}(q)\| \leq \alpha_M < \infty \quad \|S(q, \dot{q})\| \leq \alpha_S \|\dot{q}\|$$

then also \ddot{e} will be bounded (in norm) since

$$\ddot{e} = -M^{-1}(q)[S(q, \dot{q})\dot{e} + K_P e + (K_D + F_V)\dot{e}]$$

$$\begin{array}{c} \uparrow \\ \text{bounded} \\ \text{in norm by } \alpha_M \end{array} \quad \begin{array}{c} \uparrow \\ \text{bounded} \\ \text{in norm by } \alpha_S \|\dot{q}\| \end{array} \quad \begin{array}{c} \uparrow \\ \text{bounded} \end{array} \quad \begin{array}{c} \uparrow \\ \text{bounded} \end{array}$$

$\left. \begin{array}{l} \text{from i) + ii), } V \text{ is bounded} \\ \text{assume that the desired trajectory has bounded velocity } \dot{q}_d \end{array} \right\} \Rightarrow \begin{array}{l} \dot{q} \text{ is} \\ \text{bounded} \end{array}$

$$\left. \begin{array}{l} \text{using the following two properties of dynamic model terms} \\ \text{then also } \ddot{e} \text{ will be bounded (in norm) since} \\ \ddot{e} = -M^{-1}(q)[S(q, \dot{q})\dot{e} + K_P e + (K_D + F_V)\dot{e}] \\ \uparrow \\ \text{bounded} \\ \text{in norm by } \alpha_M \\ \uparrow \\ \text{bounded} \\ \text{in norm by } \alpha_S \|\dot{q}\| \\ \uparrow \\ \text{bounded} \end{array} \right\} \Rightarrow \lim_{t \rightarrow \infty} \dot{V}(t) = 0$$



Analysis of asymptotic stability of the trajectory error – end of proof

- we can conclude by proceeding as in LaSalle theorem

$$\dot{V} = 0 \Leftrightarrow \dot{e} = 0$$

- the closed-loop dynamics in this situation is

$$M(q)\ddot{e} = -K_P e$$

$$\Rightarrow \ddot{e} = 0 \Leftrightarrow e = 0 \quad \rightarrow \quad (e, \dot{e}) = (0, 0)$$

is the largest
invariant set in $\dot{V} = 0$

\rightarrow (global) asymptotic tracking
will be achieved





Regulation as a special case

- what happens to the control laws designed for trajectory tracking when q_d is **constant**? are there simplifications?
- **feedback linearization**

$$u = M(q)[K_P(q_d - q) - K_D \dot{q}] + c(q, \dot{q}) + g(q)$$

- no special simplifications
- however, this is a solution to the regulation problem with **exponential stability** (and decoupled transients at each joint!)
- **alternative global controller**

$$u = K_P(q_d - q) - K_D \dot{q} + g(q)$$

- we recover the simpler PD + gravity cancellation control law!!



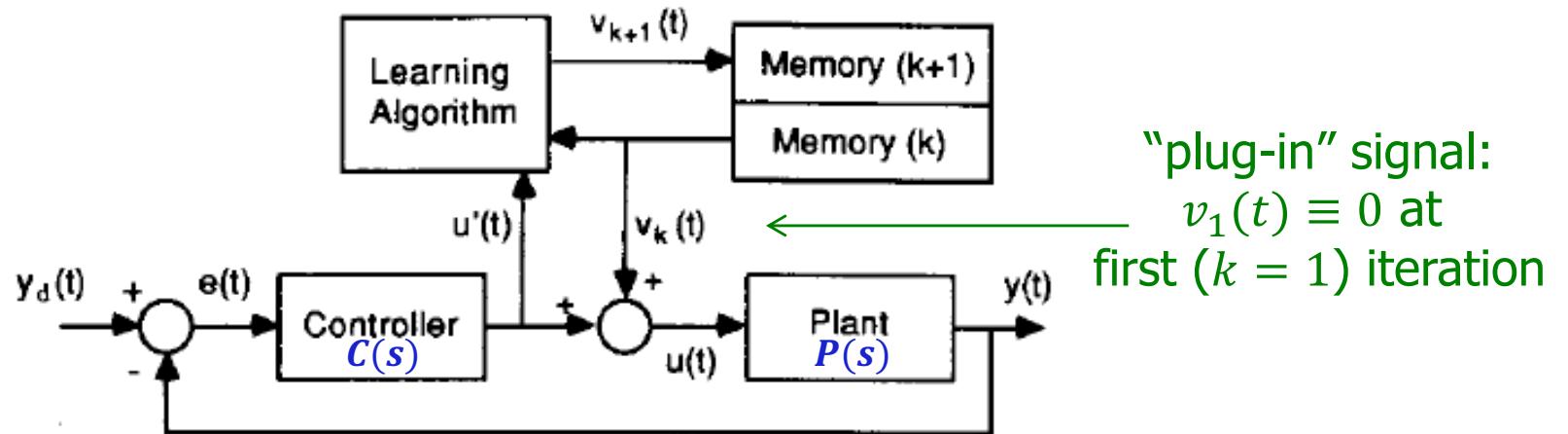
Trajectory execution without a model

- is it possible to accurately reproduce a desired smooth joint-space reference trajectory with **reduced or no information** on the robot dynamic model?
- this is feasible (and possibly simple) in case of **repetitive motion tasks** over a finite interval of time
 - trials are performed iteratively, storing the **trajectory error** information of the current execution [k -th iteration] and processing it off line before the next trial [$(k + 1)$ -iteration] starts
 - the robot should be **reinitialized** in the same initial state at the beginning of each trial (typically, with $\dot{q} = 0$)
 - the control law is made of a **non-model based part** (often, a decentralized PD law) + a **time-varying feedforward** which is updated before every trial
- this scheme is called **iterative trajectory learning**



Scheme of iterative trajectory learning

- control design can be illustrated on a **SISO linear system** in the Laplace domain



$$W(s) = \frac{y(s)}{y_d(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)}$$

closed-loop system **without** learning
 $(C(s)$ is, e.g., a PD control law)

$$u_k(s) = u'_k(s) + v_k(s) = C(s)e_k(s) + v_k(s) \quad \text{control law at iteration } k$$

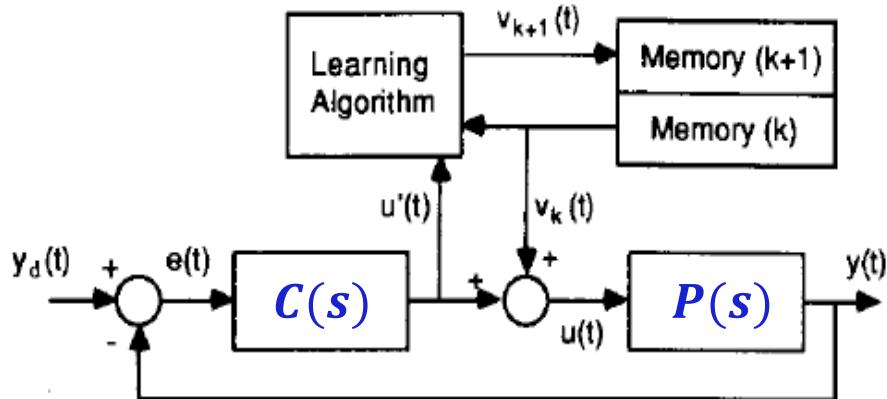
$$y_k(s) = W(s)y_d(s) + \frac{P(s)}{1 + P(s)C(s)} v_k(s) \quad \text{system output at iteration } k$$



Background math on feedback loops

whiteboard...

- algebraic manipulations on block diagram signals in the Laplace domain:
 $x(s) = \mathcal{L}[x(t)]$, $x = \{y_d, y, u', v, e\} \Rightarrow \{y_d, y_k, u'_k, v_k, e_k\}$, with transfer functions



$$\begin{aligned} y(s) &= P(s)u(s) = P(s)(v(s) + u'(s)) \\ &= P(s)v(s) + P(s)C(s)e(s) \\ &= P(s)v(s) + P(s)C(s)(y_d(s) - y(s)) \end{aligned}$$

$$\Rightarrow (1 + P(s)C(s)) y(s) = \\ = P(s)v(s) + P(s)C(s)y_d(s)$$

$$\Rightarrow y(s) = \frac{P(s)C(s)}{1 + P(s)C(s)} y_d(s) + \frac{P(s)}{1 + P(s)C(s)} v(s) = W(s)y_d(s) + W_v(s)v(s)$$

- feedback control law at iteration k

$$u'_k(s) = C(s)(y_d(s) - y_k(s)) = C(s)y_d(s) - P(s)C(s)(v_k(s) + u'_k(s))$$

$$\Rightarrow u'_k(s) = \frac{C(s)}{1 + P(s)C(s)} y_d(s) - \frac{P(s)C(s)}{1 + P(s)C(s)} v_k(s) = W_c(s)y_d(s) - W(s)v_k(s)$$

- error at iteration k

$$e_k(s) = y_d(s) - y_k(s) = y_d(s) - (W(s)y_d(s) + W_v(s)v_k(s)) = (1 - W(s))y_d(s) - W_v(s)v_k(s)$$

$$W_e(s) = 1/(1 + P(s)C(s))$$



Learning update law

- the **update** of the feedforward term is designed as

$$v_{k+1}(s) = \alpha(s)u'_k(s) + \beta(s)v_k(s)$$

with α and β **suitable filters**
(also non-causal, of the FIR type)

recursive expression
of feedforward term

$$v_{k+1}(s) = \frac{\alpha(s)C(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s))v_k(s)$$

recursive expression
of error $e = y_d - y$

$$e_{k+1}(s) = \frac{1 - \beta(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s))e_k(s)$$

- if a **contraction condition** can be enforced

$$|\beta(s) - \alpha(s)W(s)| < 1$$

(for all $s = j\omega$ **frequencies** such that ...)

then **convergence** is obtained for $k \rightarrow \infty$

$$v_\infty(s) = \frac{y_d(s)}{P(s)} \frac{\alpha(s)W(s)}{1 - \beta(s) + \alpha(s)W(s)} \quad e_\infty(s) = \frac{y_d(s)}{1 + P(s)C(s)} \frac{1 - \beta(s)}{1 - \beta(s) + \alpha(s)W(s)}$$



Proof of recursive updates

whiteboard...

- recursive expression for the **feedforward** v_k

$$\begin{aligned}
 v_{k+1}(s) &= \alpha(s)u'_k(s) + \beta(s)v_k(s) = \alpha(s)C(s)e_k(s) + \beta(s)v_k(s) \\
 &= \alpha(s)C(s)[W_e(s)y_d(s) - W_v(s)v_k(s)] + \beta(s)v_k(s) \\
 &= \frac{\alpha(s)C(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s)) v_k(s)
 \end{aligned}$$

- recursive expression for the **error** e_k

$$e_k(s) = y_d(s) - y_k(s) = y_d(s) - P(s)(v_k(s) + u'_k(s))$$

$$\Rightarrow v_k(s) = \frac{1}{P(s)}(y_d(s) - e_k(s)) - u'_k(s)$$

$$\begin{aligned}
 y_{k+1}(s) &= P(s)(v_{k+1}(s) + u'_{k+1}(s)) = P(s)(\alpha(s)u'_k(s) + \beta(s)v_k(s) + u'_{k+1}(s)) \\
 &= P(s)\left(\alpha(s)C(s)e_k(s) + \beta(s)\frac{1}{P(s)}(y_d(s) - e_k(s)) - \beta(s)C(s)e_k(s) + C(s)e_{k+1}(s)\right)
 \end{aligned}$$

$$\begin{aligned}
 e_{k+1}(s) &= y_d(s) - y_{k+1}(s) \\
 &= (1 - \beta(s))y_d(s) - [(\alpha(s) - \beta(s))P(s)C(s) - \beta(s)]e_k(s) - P(s)C(s)e_{k+1}(s) \\
 \Rightarrow e_{k+1}(s) &= \frac{1 - \beta(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s)) e_k(s)
 \end{aligned}$$



Proof of convergence

whiteboard...

from recursive expressions

$$v_{k+1}(s) = \frac{\alpha(s)C(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s)) v_k(s)$$

$$e_{k+1}(s) = \frac{1 - \beta(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s)) e_k(s)$$

compute **variations** from k to $k + 1$ (repetitive term in trajectory y_d vanishes!)

$$\Delta v_{k+1}(s) = v_{k+1}(s) - v_k(s) = (\beta(s) - \alpha(s)W(s)) \Delta v_k(s)$$

$$\Delta e_{k+1}(s) = e_{k+1}(s) - e_k(s) = (\beta(s) - \alpha(s)W(s)) \Delta e_k(s)$$

by **contraction mapping** condition $|\beta(s) - \alpha(s)W(s)| < 1 \Rightarrow \{v_k\} \rightarrow v_\infty, \{e_k\} \rightarrow e_\infty$

$$v_\infty(s) = \frac{\alpha(s)C(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s)) v_\infty(s)$$

$$e_\infty(s) = \frac{1 - \beta(s)}{1 + P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s)) e_\infty(s)$$

$$\Rightarrow v_\infty(s) = \frac{y_d(s)}{P(s)} \frac{\alpha(s)W(s)}{1 - \beta(s) + \alpha(s)W(s)} \quad e_\infty(s) = \frac{y_d(s)}{1 + P(s)C(s)} \frac{1 - \beta(s)}{1 - \beta(s) + \alpha(s)W(s)}$$



Comments on convergence

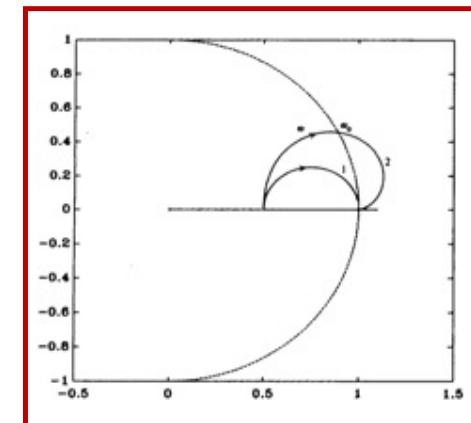
- if the choice $\beta = 1$ allows to satisfy the contraction condition, then convergence to **zero tracking error** is obtained

$$e_\infty(s) = 0$$

and the **inverse dynamics** command has been **learned**

$$\nu_\infty(s) = \frac{y_d(s)}{P(s)}$$

- in particular, for $\alpha(s) = 1/W(s)$ convergence would be in **1 iteration** only!
- the use of filter $\beta(s) \neq 1$ allows to obtain convergence (but with residual tracking error) even in presence of unmodeled high-frequency dynamics
- the **two filters** can be designed from very poor information on system dynamics, using classic tools (e.g., **Nyquist** plots)





Application to robots

- for N -dof robots modeled as

$$[B_m + M(q)]\ddot{q} + [F_V + S(q, \dot{q})]\dot{q} + g(q) = u$$

we choose as (initial = pre-learning) **control law**

$$u = u' = K_P(q_d - q) + K_D(\dot{q}_d - \dot{q}) + \hat{g}(q)$$

and design the **learning filters** (at each joint) using the **linear approximation**

$$W_i(s) = \frac{q_i(s)}{q_{di}(s)} = \frac{K_{Di}s + K_{Pi}}{\hat{B}_{mi}s^2 + (\hat{F}_{Vi} + K_{Di})s + K_{Pi}} \quad i = 1, \dots, N$$

- initialization** of feedforward uses the best estimates

$$\nu_1 = [\hat{B}_m + \hat{M}(q_d)]\ddot{q}_d + [\hat{F}_V + \hat{S}(q_d, \dot{q}_d)]\dot{q}_d + \hat{g}(q_d)$$

or **simply** $\nu_1 = 0$ (in the worst case) at first trial $k = 1$



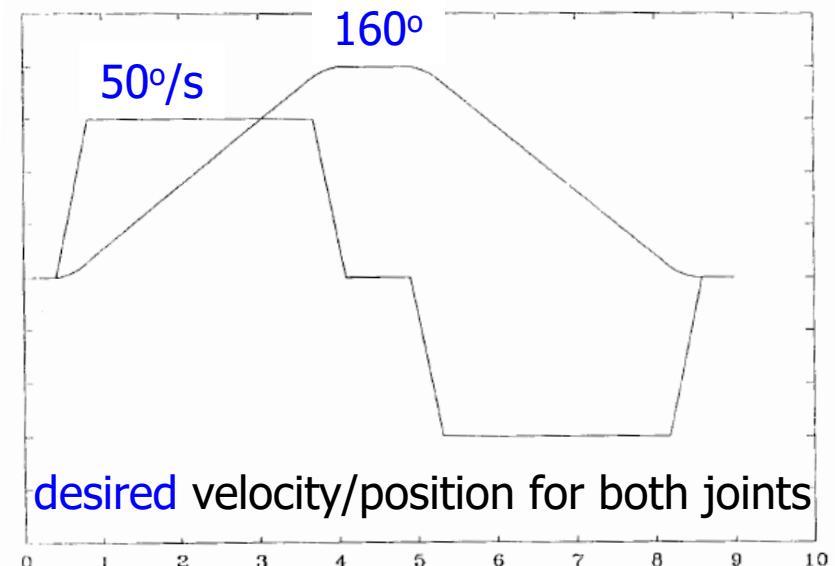
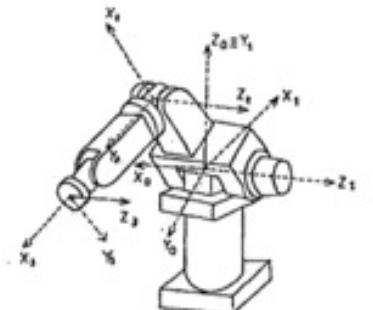
Experimental set-up

- joints 2 and 3 of 6R MIMO CRF robot prototype @DIS

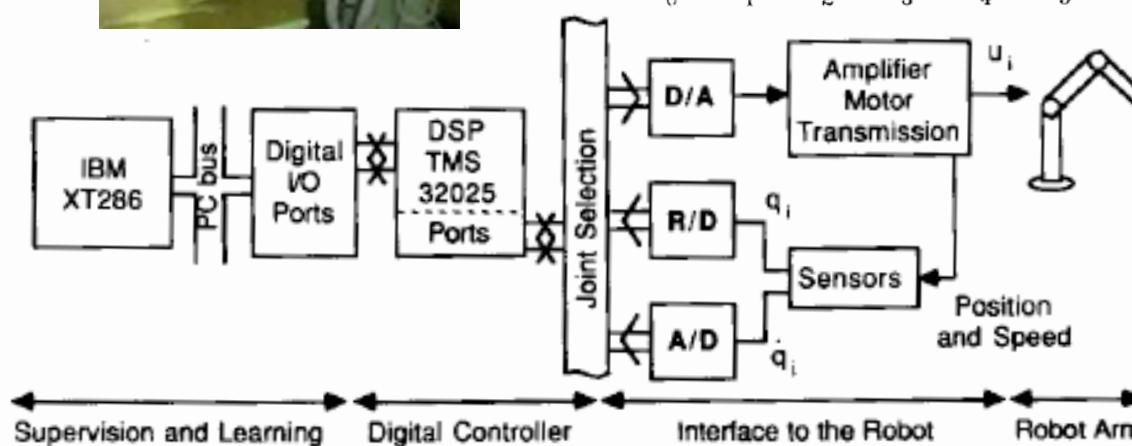
$\approx 90\%$ gravity balanced through springs

high level of dry friction

Harmonic Drives transmissions with ratio 160:1



DSP $T_c = 400\mu s$
D/A = 12 bit
R/D = 16 bit/ 2π
A/D = 11 bit/(rad/s)

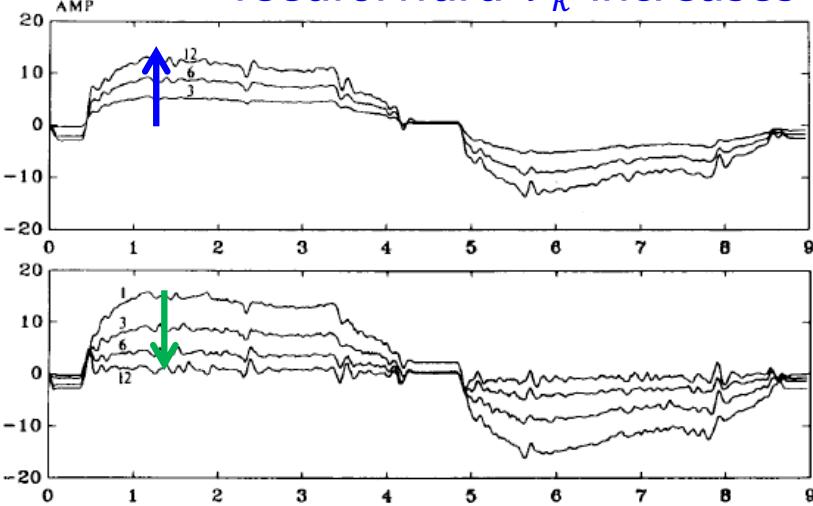
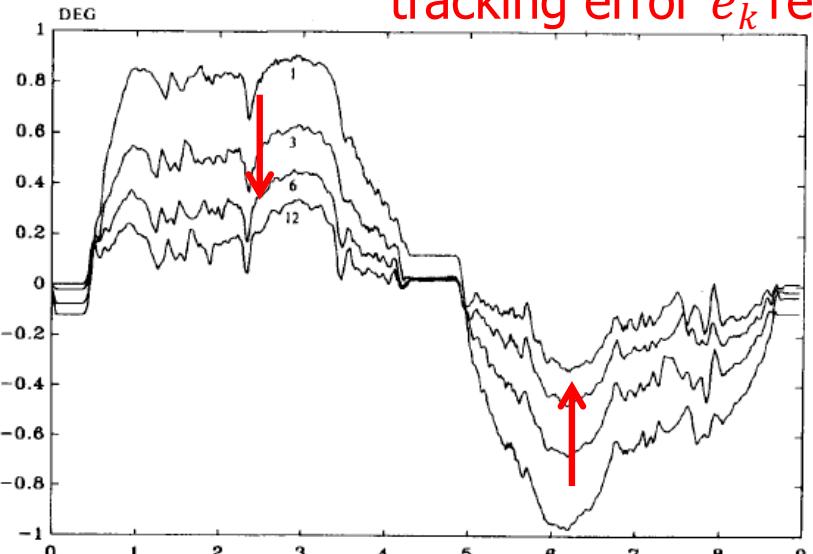


DC motors with current amplifiers
resolvers and tachometers



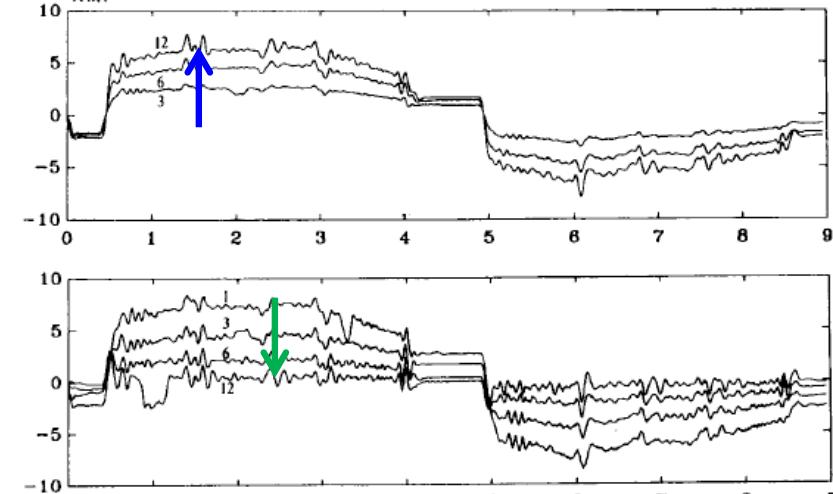
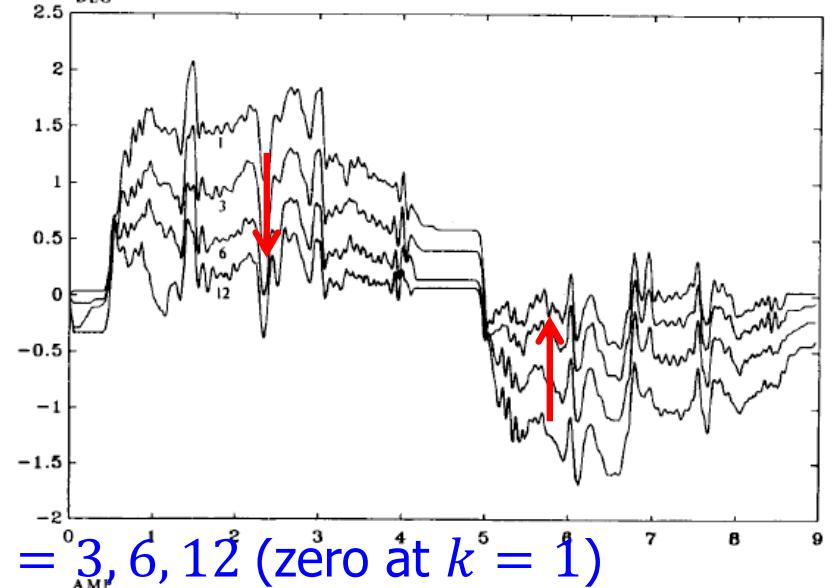
Experimental results

joint 2



feedback u'_k decreases for $k = 1, 3, 6, 12$

joint 3





On-line learning control

- re-visitation of the learning idea so as to acquire the missing dynamic information in model-based trajectory control
- **on-line learning** approach
 - the robot improves tracking performance already while executing the task in feedback mode
- uses only position measurements from encoders
 - no need of joint torque sensors
- machine learning techniques used for
 - data collection and organization
 - regressor construction for estimating model perturbations
- **fast convergence**
 - starting with a reasonably good robot model
- extensions to underactuated robots or with flexible components



Control with approximate FBL

- dynamic model, its nominal part and (unstructured) uncertainty

$$M(q)\ddot{q} + n(q, \dot{q}) = \tau \quad M = \hat{M} + \Delta M \quad n = \hat{n} + \Delta n$$

- model-based (approximate) feedback linearization

$$\tau_{FBL} = \hat{M}(q)a + \hat{n}(q, \dot{q})$$

- resulting closed-loop dynamics with **perturbation**

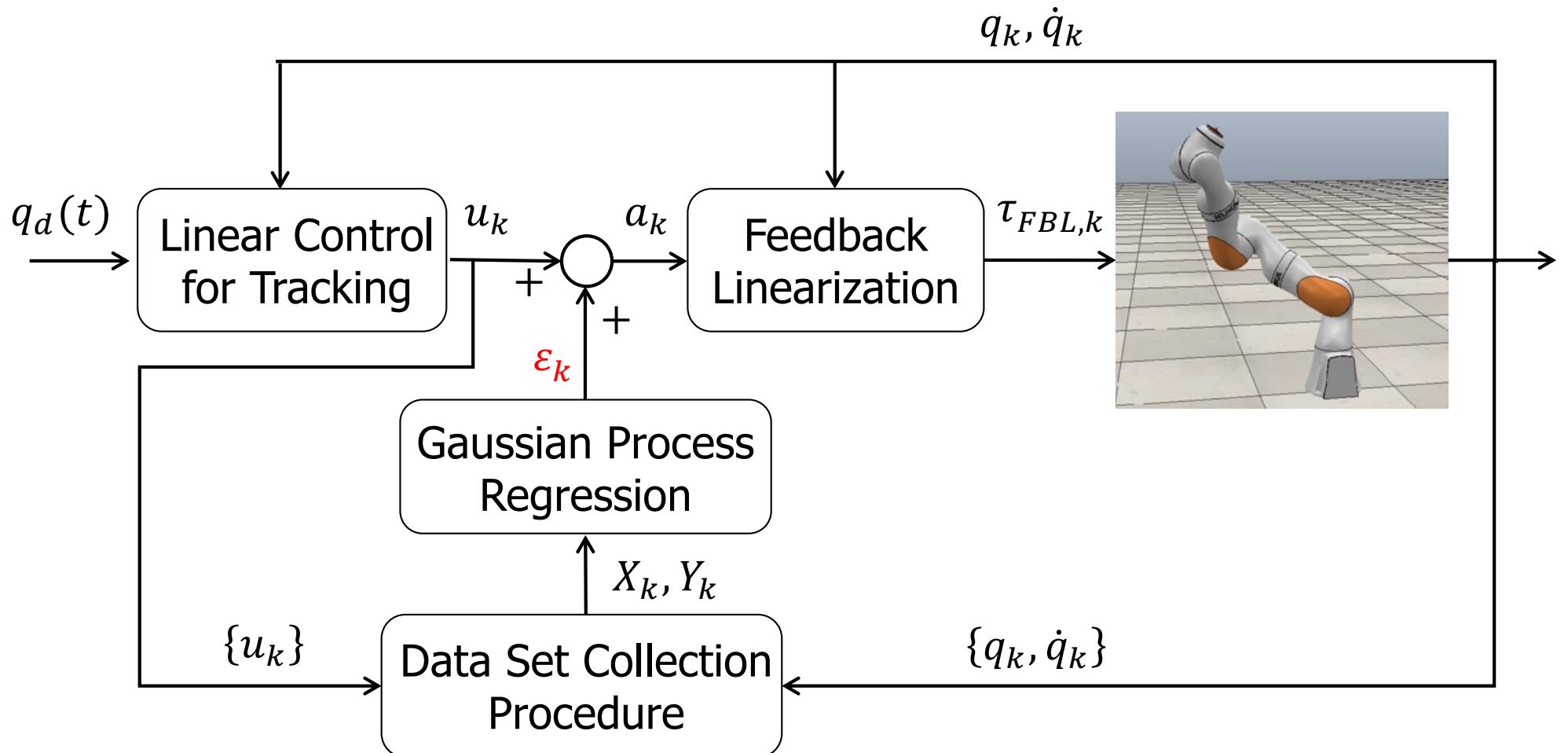
$$\ddot{q} = a + \delta(q, \dot{q}, a) \quad \leftarrow \quad \delta = (M^{-1}\hat{M} - I)a + M^{-1}(\hat{n} - n)$$

- control law for tracking $q_d(t)$ is completed by using (at $t = t_k$) a linear design (PD with feedforward) and a learning **regressor** ε_k

$$\begin{aligned} a &= a_k = u_k + \varepsilon_k \\ &= \ddot{q}_{d,k} + K_P(q_{d,k} - q_k) + K_D(\dot{q}_{d,k} - \dot{q}_k) + \varepsilon_k \end{aligned}$$



On-line learning scheme





On-line regressor

- Gaussian Process (GP) regression to estimate the perturbation δ

- from input-output observations that are noisy, with $\omega \sim \mathcal{N}(0, \Sigma_\omega)$, the generated data points at the k -th control step are

$$X_k = (q_k, \dot{q}_k, u_k) \quad Y_k = \ddot{q}_k - u_k$$

- assuming the ensemble of n_d observations with a joint Gaussian distribution

$$\begin{pmatrix} Y_{1:n_d-1} \\ Y_{n_d} \end{pmatrix} \sim \mathcal{N} \left(0, \begin{pmatrix} K & k \\ k^T & \kappa(X_{n_d}, X_{n_d}) \end{pmatrix} \right)$$

a Kernel
to be chosen

- the predictive distribution that approximates $\delta(\hat{X})$ for a generic query \hat{X} is

$$\varepsilon(\hat{X}) \sim \mathcal{N}(\mu(\hat{X}), \sigma^2(\hat{X}))$$

with

$$\mu(\hat{X}) = k^T(\hat{X})(K + \Sigma_\omega)^{-1}Y$$

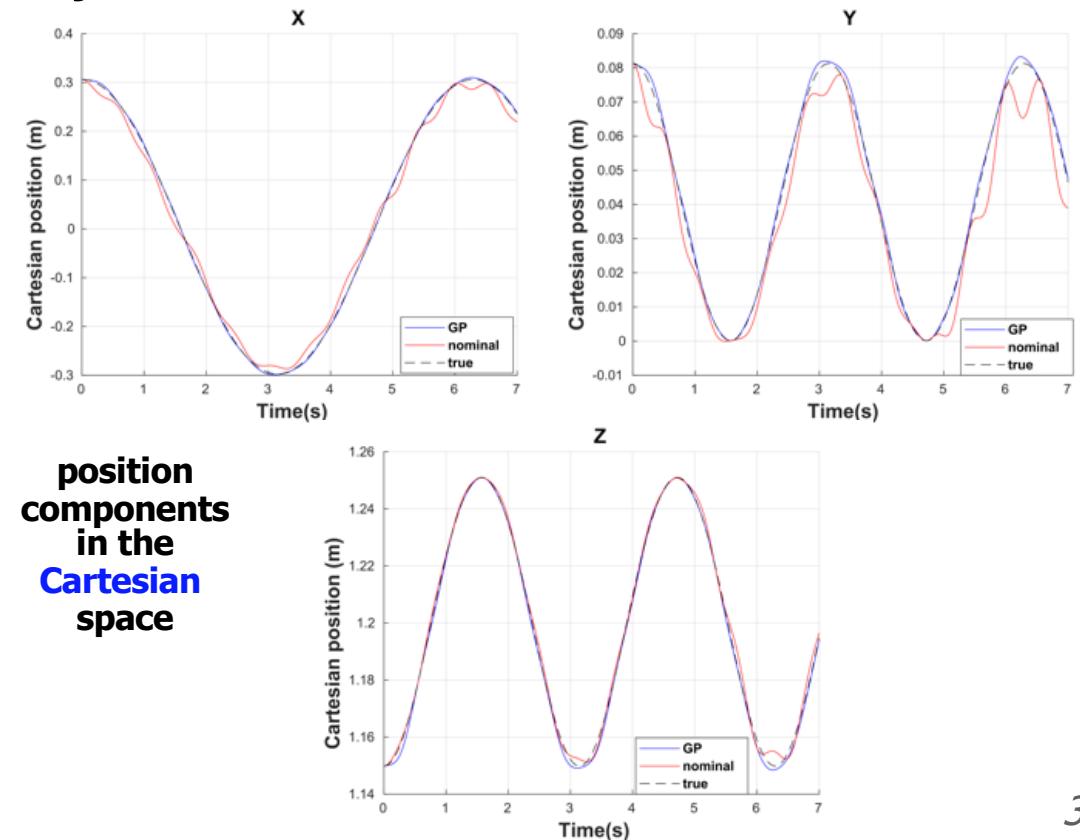
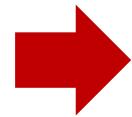
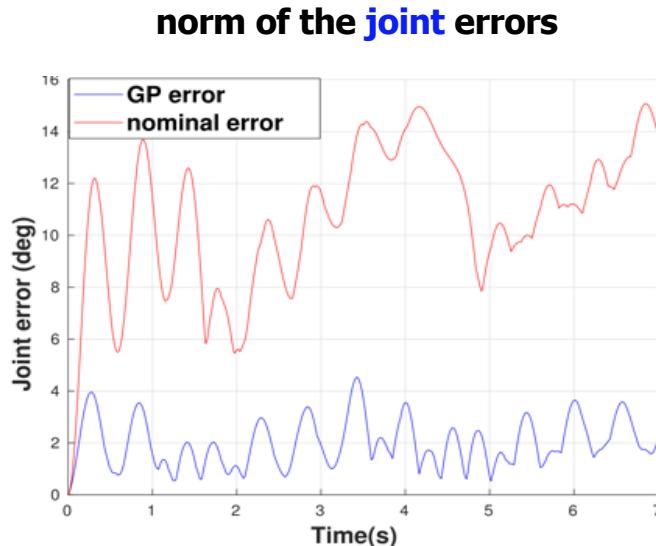
$$\sigma^2(\hat{X}) = \kappa(\hat{X}, \hat{X}) - k^T(\hat{X})(K + \Sigma_\omega)^{-1}k(\hat{X})$$

$$\left. \begin{array}{c} \varepsilon(\hat{X}) \sim \mathcal{N}(\mu(\hat{X}), \sigma^2(\hat{X})) \\ \mu(\hat{X}) = k^T(\hat{X})(K + \Sigma_\omega)^{-1}Y \\ \sigma^2(\hat{X}) = \kappa(\hat{X}, \hat{X}) - k^T(\hat{X})(K + \Sigma_\omega)^{-1}k(\hat{X}) \end{array} \right\} \Rightarrow \varepsilon_k = \varepsilon(X_k)$$



Simulation results

- Kuka LWR iiwa, 7-dof robot
- model perturbations: dynamic parameters with $\pm 20\%$ variation, uncompensated joint friction
- 7 separate GPs (one for each joint), each with 21 inputs at every $t = t_k$
- sinusoidal trajectories for each joint



... at the first and only iteration!



Simulation results

video
(slowed
down)



An Online Learning Procedure for Feedback Linearization Control without Torque Measurements

M. Capotondi, G. Turrisi, C. Gaz, V. Modugno, G. Oriolo, A. De Luca

Robotics Lab, DIAG
Sapienza Università di Roma

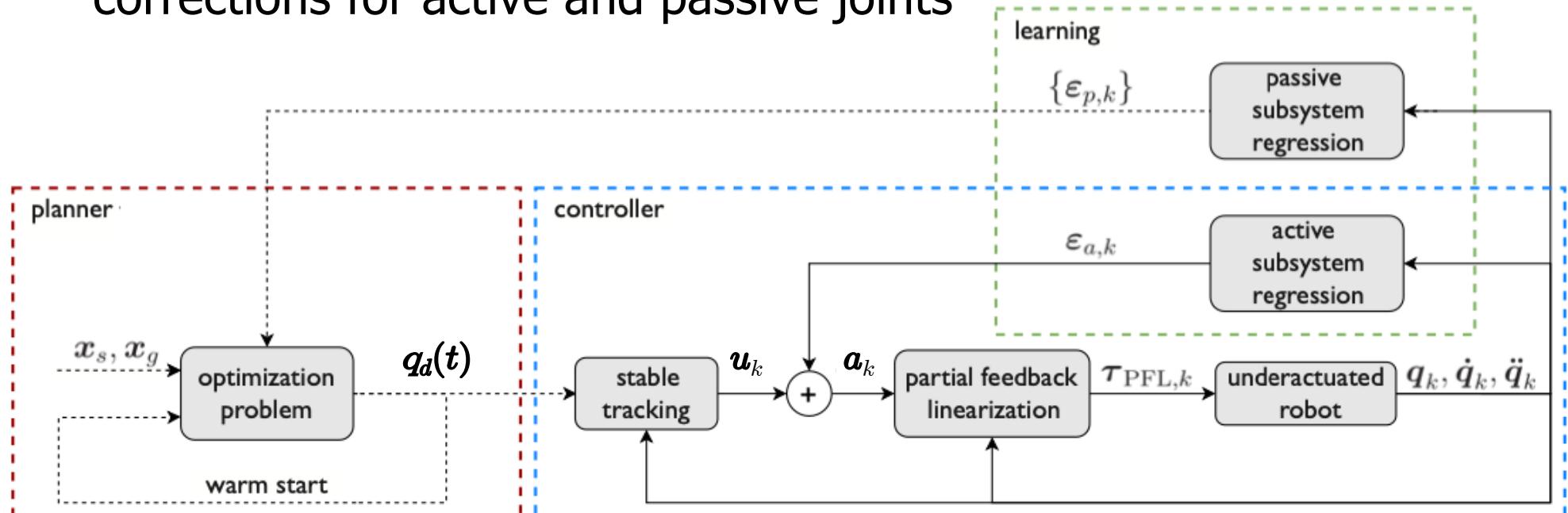
October 2019



Extension to underactuated robots

$$\begin{pmatrix} M_{aa}(q) & M_{ap}(q) \\ M_{ap}^T(q) & M_{pp}(q) \end{pmatrix} \begin{pmatrix} \ddot{q}_a \\ \ddot{q}_p \end{pmatrix} + \begin{pmatrix} n_a(q, \dot{q}) \\ n_p(q, \dot{q}) \end{pmatrix} = \begin{pmatrix} \tau \\ 0 \end{pmatrix}$$

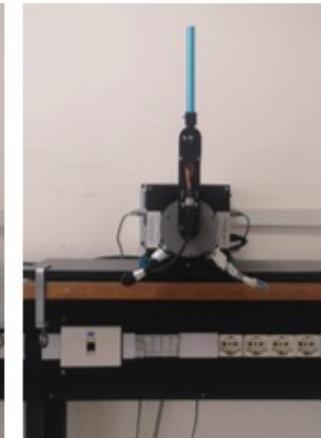
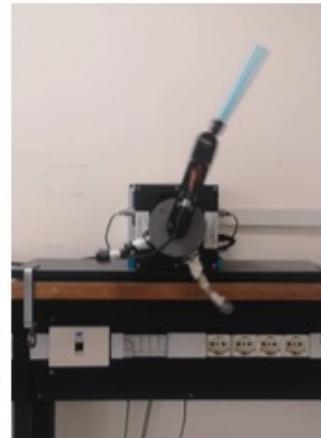
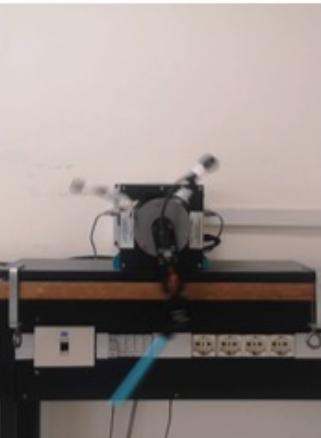
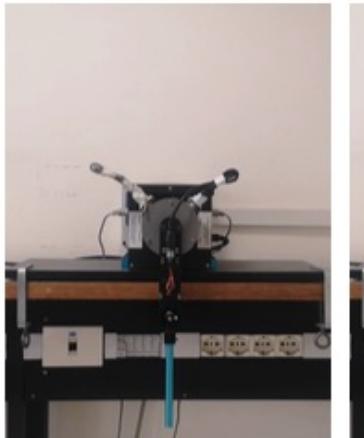
- planner optimizes motion of **passive** joints (at every iteration)
- controller for **active** joints with **partial** feedback linearization
- two **regressors** (on/off-line) for learning the required acceleration corrections for active and passive joints



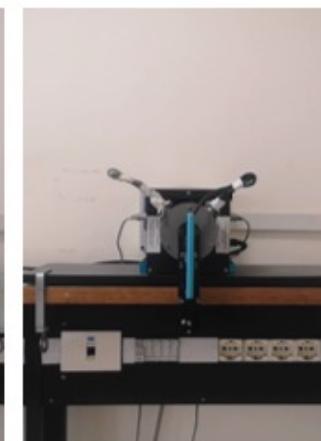


Experiments on the Pendubot

- Pendubot, 2-dof robot with passive second joint
- **swing-up** maneuvers from down-down to a new equilibrium state



⇒ up-up



⇒ down-up



Experimental results

IEEE Robotics and Automation Letters, vol. 7(1), 2022

video



Iterative Learning Control for Underactuated Robots

Giulio Turrisi, Marco Capotondi, Claudio Gaz,
Valerio Modugno, Giuseppe Oriolo, Alessandro De Luca

Robotics Lab, DIAG,
Sapienza Università di Roma

March 2021

convergence in 2 iterations!

latest video with more simulations & experiments
on YouTube https://youtu.be/1aKG_8gfvk



Robotics 2

Adaptive Trajectory Control

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Motivation and approach

- need of adaptation in robot motion control laws
 - large uncertainty on the robot dynamic parameters
 - poor knowledge of the inertial payload
- characteristics of **direct** adaptive control
 - direct aim is to bring to zero the state trajectory error, i.e., position and velocity errors
 - no need to estimate on-line the true values of the dynamic coefficients of the robot (as opposed to **indirect** adaptive control)
- main tool and methodology
 - **linear parametrization** of robot dynamics
 - **nonlinear** control law of the **dynamic** type (the controller has its own 'states')



Summary of robot parameters

- parameters assumed to be **known**
 - kinematic description based, e.g., on Denavit-Hartenberg parameters ($\{\alpha_i, d_i, a_i, i = 1, \dots, N\}$ in case of all revolute joints), including link lengths (**kinematic calibration**)
- **uncertain** parameters that can be **identified** off-line
 - masses m_i , positions r_{ci} of CoMs, and inertia matrices I_i of each link, appearing in combinations (**dynamic coefficients**) $\Rightarrow p \ll 10 \times N$
- parameters that are **(slowly) varying** during operation
 - viscous F_{Vi} , dry F_{Di} , and stiction F_{Si} friction at each joint $\Rightarrow 1 \div 3 \times N$
- **unknown** and abruptly changing parameters
 - mass, CoM, inertia matrix of the **payload** (w.r.t. the tool center point)



when a payload is firmly **attached** to the robot E-E, only the 10 parameters of the last link are modified, influencing however most part of the robot dynamics



Goal of adaptive control

- given a twice-differentiable desired joint trajectory $q_d(t)$
 - with known desired velocity $\dot{q}_d(t)$ and acceleration $\ddot{q}_d(t)$
 - possibly obtained by kinematic inversion + joint interpolation
- execute this trajectory under large dynamic uncertainties
 - with a trajectory tracking error vanishing **asymptotically**
$$e = q_d - q \rightarrow 0 \quad \dot{e} = \dot{q}_d - \dot{q} \rightarrow 0$$
 - guaranteeing **global stability**, no matter how far are the initial estimates of the unknown/uncertain parameters from their true values and how large is the initial trajectory error
- identification is **not** of particular concern: in general, the estimates of dynamic coefficients will not converge to the true ones!
- if this convergence is a specific extra requirement, then one should use (more complex) **indirect adaptive schemes**



Linear parameterization

$$M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) + F_V\dot{q} = u$$

- there exists always a (p -dimensional) **vector a** of **dynamic coefficients**, so that the robot model takes the **linear** form

$$Y(q, \dot{q}, \ddot{q}) a = u$$

- vector **a** contains only unknown or uncertain coefficients
- each component of **a** is in general a **combination** of the robot physical parameters (not necessarily all of them)
- the model **regression matrix Y** depends linearly on \ddot{q} , quadratically on \dot{q} (for the terms related to kinetic energy), and nonlinearly (trigonometrically) on q



Trajectory controllers based on model estimates

- inverse dynamics feedforward (**FFW**) + PD (**linear**) control

$$u = \underbrace{\hat{M}(q_d)\ddot{q}_d + \hat{S}(q_d, \dot{q}_d)\dot{q}_d + \hat{g}(q_d) + \hat{F}_V\dot{q}_d}_{\hat{u}_d} + K_P e + K_D \dot{e}$$

- (**nonlinear**) control based on feedback linearization (**FBL**)

$$u = \hat{M}(q)(\ddot{q}_d + K_P e + K_D \dot{e}) + \hat{S}(q, \dot{q})\dot{q} + \hat{g}(q) + \hat{F}_V\dot{q}$$

$$\boxed{\hat{M}, \hat{S}, \hat{g}, \hat{F}_V \quad \Leftrightarrow \quad \text{estimate } \hat{a}}$$

- approximate estimates of dynamic coefficients may lead to **instability** with **FBL** due to temporary 'non-positive' PD gains (e.g., $\hat{M}(q)K_P < 0!$)
- not easy** to turn these laws in **adaptive** schemes: inertia inversion/use of acceleration (FBL); bounds on PD gains (FFW)



A control law more easily made 'adaptive'

- nonlinear trajectory tracking control (without cancellations) having global asymptotic stabilization properties

$$u = \hat{M}(q)\ddot{q}_d + \hat{S}(q, \dot{q})\dot{q}_d + \hat{g}(q) + \hat{F}_V\dot{q}_d + K_P e + K_D \dot{e}$$

- a natural **adaptive version** would require ...

$\dot{\hat{a}} =$ designing a suitable **update law**
(in continuous time)

- without extra assumptions, it can be shown that joint velocities become eventually "clamped" to those of the **desired** trajectory (zero **velocity** error), but a residual **position** error may be left
- idea: **on-line modification** with a **reference velocity**

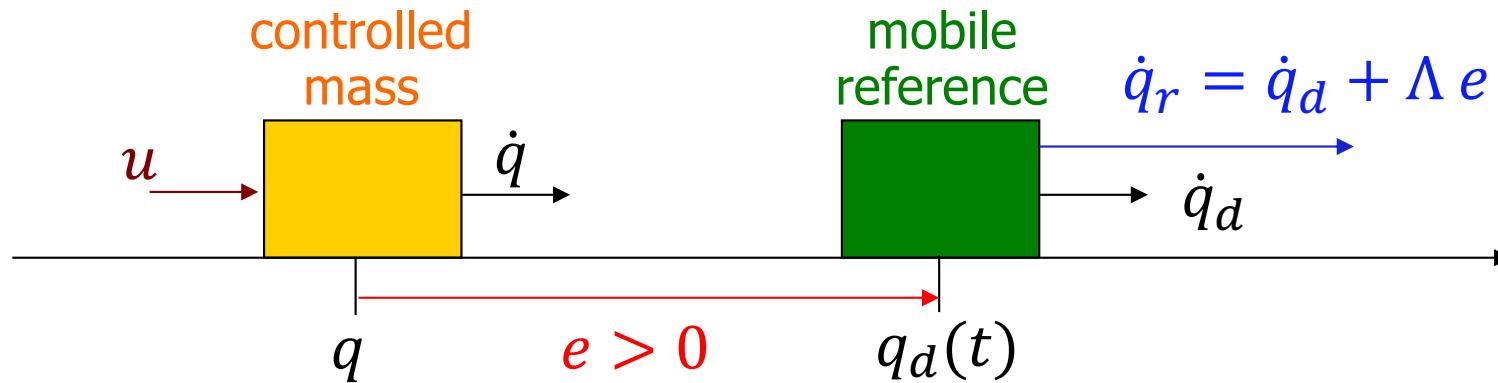
$$\dot{q}_d \quad \rightarrow \quad \boxed{\dot{q}_r = \dot{q}_d + \Lambda(q_d - q)} \quad \Lambda > 0$$

typically, $\Lambda = K_D^{-1}K_P$ (all matrices will be chosen **diagonal**)



Intuitive interpretation of \dot{q}_r

- elementary case
 - a mass 'lagging behind' a mobile reference ($e > 0$) at constant speed



→ 'enhanced' velocity error $s = \dot{q}_r - \dot{q} > \dot{q}_d - \dot{q} = \dot{e}$

$$u = K_D s = K_D(\dot{q}_r - \dot{q}) = K_D(\dot{q}_d + \Lambda e - \dot{q}) = K_D \underbrace{\dot{e}}_{K_P} + K_D \Lambda e$$

- a mass 'leading in front' of its mobile reference ($e < 0$)
- in a symmetric way, a 'reduced' velocity error will appear ($s < \dot{e}$)



Adaptive control law design

- substituting $\dot{q}_r = \dot{q}_d + \Lambda e$, $\ddot{q}_r = \ddot{q}_d + \Lambda \dot{e}$ in the previous nonlinear controller for trajectory tracking

$$u = \hat{M}(q)\ddot{q}_r + \hat{S}(q, \dot{q})\dot{q}_r + \hat{g}(q) + \hat{F}_V\dot{q}_r + K_P e + K_D \dot{e}$$

$$= Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{a} + K_P e + K_D \dot{e}$$

dynamic parameterization of
the control law using current estimates PD stabilization
(diagonal matrices, >0)
(note here the 4 arguments in $Y(\cdot)$!)

- update law for the estimates of the dynamic coefficients (\hat{a} becomes the p -dimensional state of the dynamic controller)

$$\dot{\hat{a}} = \Gamma Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)(\dot{q}_r - \dot{q})$$

$\underbrace{}_S$

$\Gamma > 0$ (diagonal)
estimation gains
(variation rate of estimates)

'modified' velocity error



Asymptotic stability of trajectory error

Theorem

The introduced adaptive controller makes the **tracking error** along the desired trajectory **globally asymptotically stable**

$$e = q_d - q \rightarrow 0, \dot{e} = \dot{q}_d - \dot{q} \rightarrow 0$$

Proof

- a **Lyapunov candidate** for the closed-loop system (robot + dynamic controller) is given by

$$V = \frac{1}{2} s^T M(q) s + \frac{1}{2} e^T R e + \frac{1}{2} \tilde{a}^T \Gamma^{-1} \tilde{a} \geq 0$$

$$s = \dot{q}_r - \dot{q} (= \dot{e} + \Lambda e)$$

modified velocity error

$$R > 0$$

constant matrix
(to be specified later)

$$\tilde{a} = a - \hat{a}$$

error in parametric
estimation

$$V = 0 \iff \hat{a} = a, \quad q = q_d, \quad s = 0 \quad (\Rightarrow \dot{q} = \dot{q}_d)$$



Proof (cont)

- the time derivative of V is

$$\dot{V} = \frac{1}{2} s^T \dot{M}(q) s + s^T M(q) \dot{s} + e^T R \dot{e} - \tilde{a}^T \Gamma^{-1} \dot{\tilde{a}}$$

since $\dot{\tilde{a}} = -\dot{\hat{a}}$ ($\dot{a} = 0$)

- the closed-loop dynamics is given by

$$\begin{aligned} M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) + F_V \dot{q} &= \\ &= \widehat{M}(q)\ddot{q}_r + \widehat{S}(q, \dot{q})\dot{q}_r + \widehat{g}(q) + \widehat{F}_V \dot{q}_r + K_P e + K_D \dot{e} \end{aligned}$$

subtracting the two sides from $M(q)\ddot{q}_r + S(q, \dot{q})\dot{q}_r + g(q) + F_V \dot{q}_r$ leads to

$$\begin{aligned} M(q) \dot{s} + (S(q, \dot{q}) + F_V) s &= \\ &= \widetilde{M}(q)\ddot{q}_r + \widetilde{S}(q, \dot{q})\dot{q}_r + \widetilde{g}(q) + \widetilde{F}_V \dot{q}_r - K_P e - K_D \dot{e} \end{aligned}$$

with $\widetilde{M} = M - \widehat{M}$, $\widetilde{S} = S - \widehat{S}$, $\widetilde{g} = g - \widehat{g}$, $\widetilde{F}_V = F_V - \widehat{F}_V$



Proof (cont)

- from the property of **linearity in the dynamic coefficients**, it follows

$$\textcircled{M(q)\dot{s}} + (S(q, \dot{q}) + F_V)s = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{a} - K_P e - K_D \dot{e}$$

- substituting in \dot{V} , together with $\dot{\tilde{a}} = \Gamma Y^T s$, and using the skew-symmetry of matrix $\dot{M} - 2S$ we obtain

$$\begin{aligned}\dot{V} &= \frac{1}{2} s^T [\dot{M}(q) - 2S(q, \dot{q})]s - s^T F_V s + s^T Y \tilde{a} \\ &\quad - s^T (K_P e + K_D \dot{e}) + e^T R \dot{e} - \tilde{a}^T Y^T s \\ &= -s^T F_V s - s^T (K_P e + K_D \dot{e}) + e^T R \dot{e}\end{aligned}$$

- replacing $s = \dot{e} + \Lambda e$ and being $F_V = F_V^T$ (diagonal)

$$\dot{V} = -e^T (\Lambda^T F_V \Lambda + \Lambda^T K_P) e$$

a complete quadratic form in e, \dot{e} !

$$\xrightarrow{\hspace{1cm}} -e^T (2\Lambda^T F_V + \Lambda^T K_D + K_P - \textcircled{R}) \dot{e} - \dot{e}^T (F_V + K_D) \dot{e}$$



Proof (end)

- defining now (all matrices are **diagonal!**)

$$\Lambda = K_D^{-1}K_P > 0 \quad R = 2K_P(I + K_D^{-1}F_V) > 0$$

cancels the cross-term in $e^T(\dots)\dot{e}$ and leads to

$$\begin{aligned}\dot{V} &= -e^T \Lambda^T (F_V + K_D) \Lambda e - \dot{e}^T (F_V + K_D) \dot{e} \\ &= -e^T K_P K_D^{-1} (F_V + K_D) K_D^{-1} K_P e - \dot{e}^T (F_V + K_D) \dot{e} \leq 0\end{aligned}$$

and thus

$$\boxed{\dot{V} = 0 \iff e = \dot{e} = 0}$$

the thesis follows from Barbalat lemma + LaSalle theorem



the maximal invariant set of states $\subseteq \{\dot{V} = 0\}$ has **zero trajectory error** ($e = \dot{e} = 0$) and **a constant value** for \hat{a} , not necessarily the true one ($\tilde{a} \neq 0$)

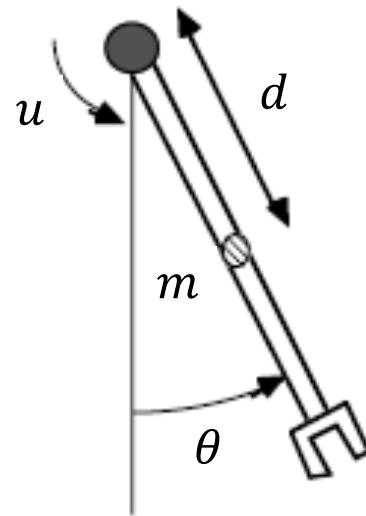


Remarks

- if the desired trajectory $q_d(t)$ is **persistently exciting**, then also the estimates of the dynamic coefficients converge to their true values
- **condition** of persistent excitation
 - for **linear** systems: # of frequency components in the desired trajectory should be at least **twice as large** as # of unknown coefficients
 - for **nonlinear** systems: the condition can be checked only **a posteriori** (a squared motion integral should always be positive bounded from below)
- in case of known absence of (viscous) friction ($F_V \equiv 0$), the same proof applies (a bit easier in the final part)
- the adaptive controller **does not require** the inverse of the inertia matrix (true or estimated), nor the actual robot acceleration (only the desired acceleration), nor further lower bounds on $K_P > 0, K_D > 0$
- adaptation can also be used **only for a subset** of dynamic coefficients, with the others being known ($Y_a = Y_{adapt} \hat{a}_{adapt} + Y_{known} a_{known}$)
- the **non-adaptive version** (using accurate estimates) is a static tracking controller based on the **passivity** property of robot dynamics



Case study: Single-link under gravity



model $I\ddot{\theta} + mg_0d \sin \theta + f_V \dot{\theta} = u$ (with friction)

linear parameterization

$$Y(\theta, \dot{\theta}, \ddot{\theta})a = [\ddot{\theta} \quad \sin \theta \quad \dot{\theta}] \begin{bmatrix} I \\ mg_0d \\ f_V \end{bmatrix} = u$$

adaptive controller

$$e = \theta_d - \theta \quad \Lambda > 0$$

$$\dot{\theta}_r = \dot{\theta}_d + \frac{k_P}{k_D} e$$

$$\gamma_i > 0, i = 1, 2, 3$$

$$u = \widehat{I} \ddot{\theta}_r + \widehat{mg_0d} \sin \theta + \widehat{f_V} \dot{\theta}_r + k_P e + k_D \dot{e}$$

$$\dot{\hat{a}} = \begin{pmatrix} \dot{\widehat{I}} \\ \dot{\widehat{mg_0d}} \\ \dot{\widehat{f_V}} \end{pmatrix} = \begin{pmatrix} \gamma_1 \ddot{\theta}_r \\ \gamma_2 \sin \theta \\ \gamma_3 \dot{\theta}_r \end{pmatrix} (\dot{\theta}_r - \dot{\theta})$$

$$\Gamma \cdot \gamma^T (\ddot{\theta}_r - \ddot{\theta})$$



Simulation data

- **real** dynamic coefficients

$$I = 7.5, \quad mg_0d = 6, \quad f_V = 1$$

- **initial** estimates

$$\widehat{I} = 5, \quad \widehat{mg_0d} = 5, \quad \widehat{f}_V = 2$$

- control parameters

$$k_P = 25, \quad k_D = 10, \quad \gamma_i = 5, \quad i = 1,2,3$$

- **test trajectories** (starting with $\theta(0) = 0, \dot{\theta}(0) = 0$)

- **first**

$$\theta_d(t) = -\sin t$$

Note: same test trajectories
used also for robust control

- **second**

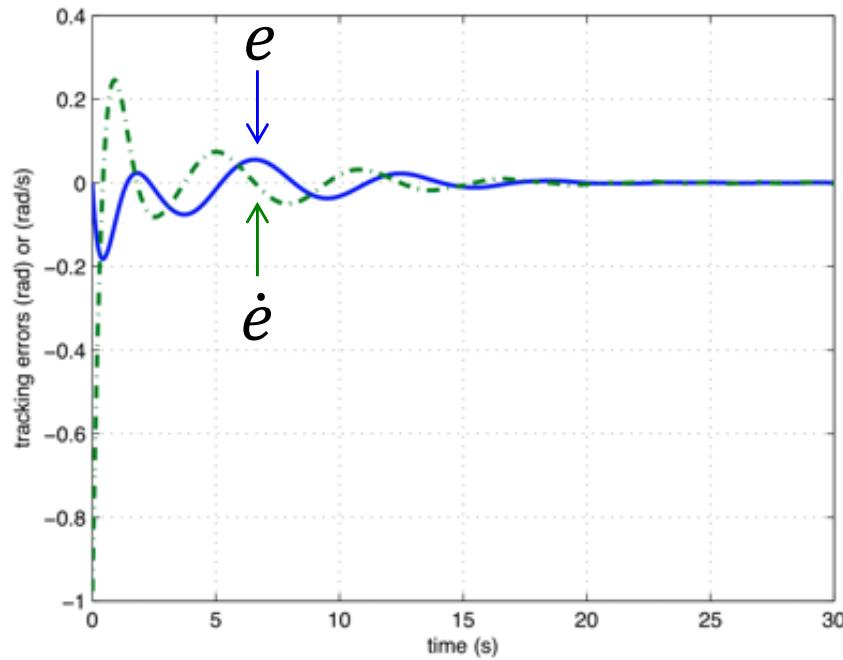
$\ddot{\theta}_d(t)$ = (periodic) bang-bang acceleration profile with
 $A = 1 \text{ rad/s}^2, \omega = 1 \text{ rad/s}$



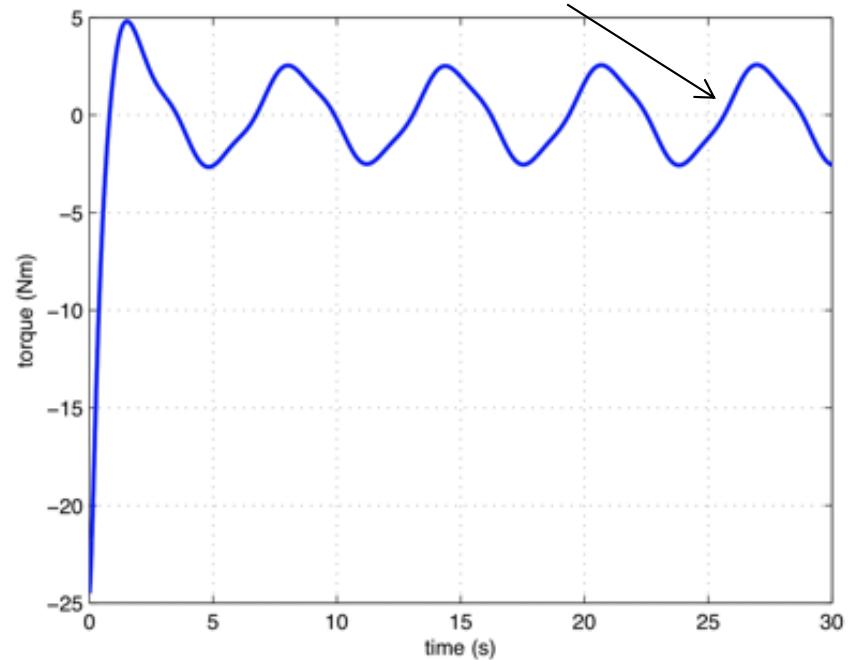
Results

first trajectory

note the nonlinear system dynamics
(no sinusoidal regime at steady state!)



position and velocity errors

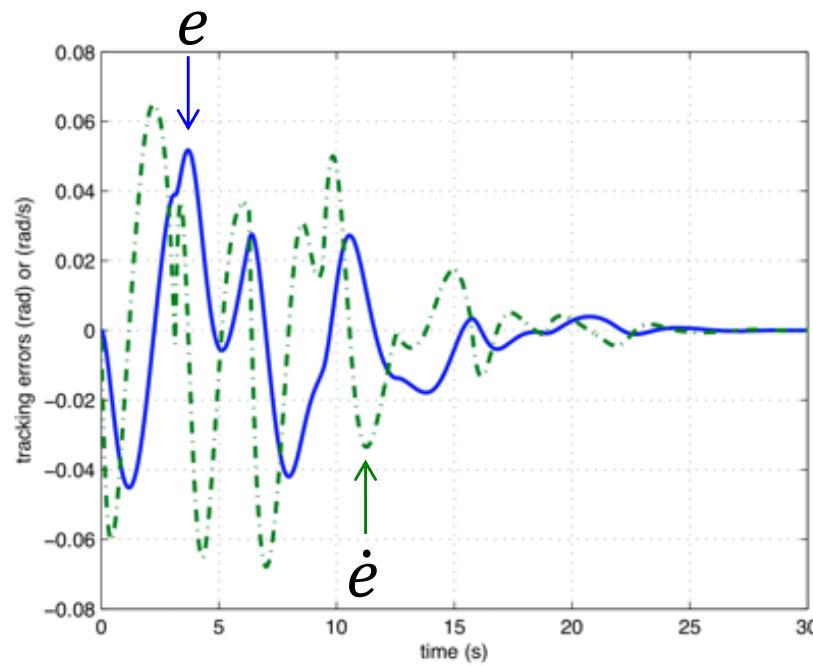


control torque

$$\theta_d(t) = -\sin t$$

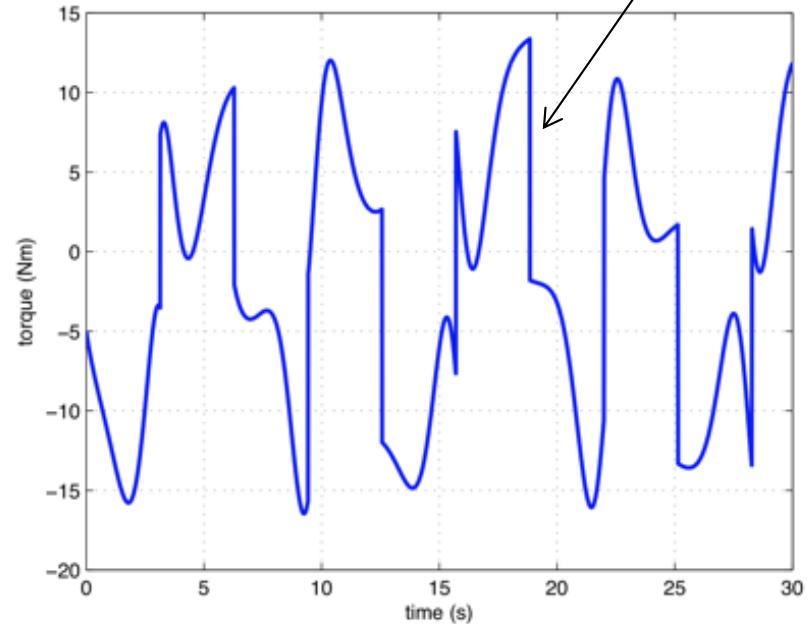


Results second trajectory



position and velocity errors

note the torque discontinuities
(due to those of the desired acceleration)

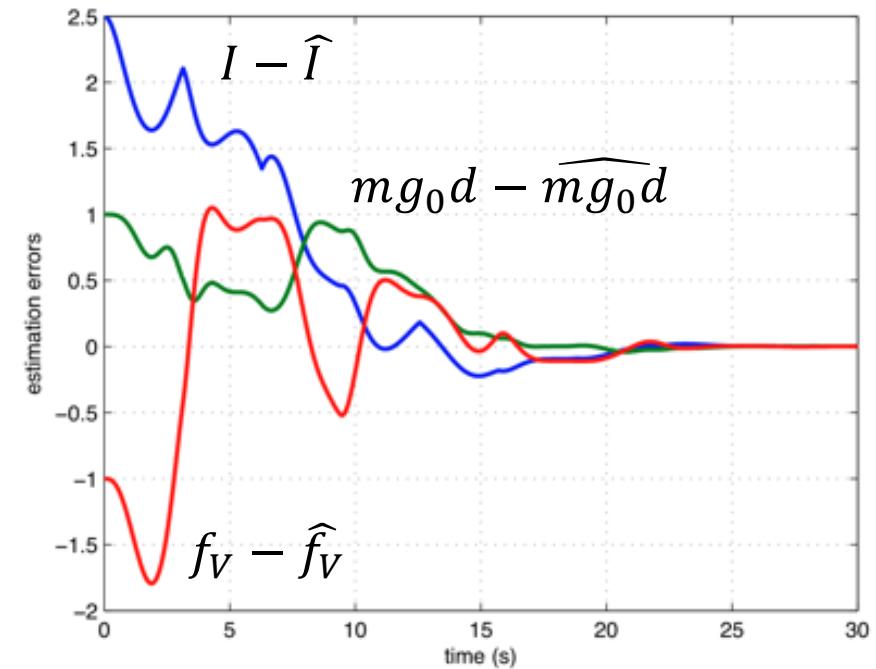
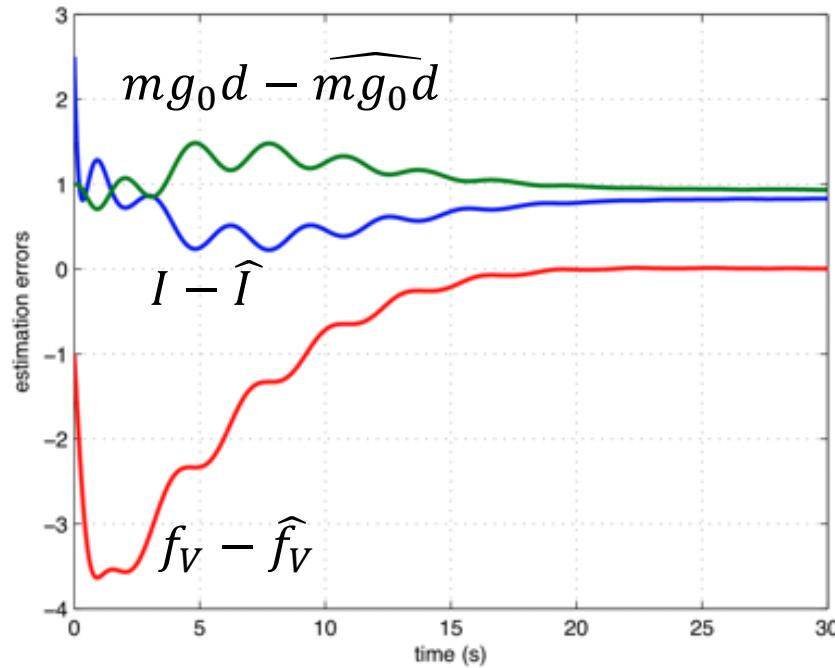


control torque

$$\ddot{\theta}_d(t) = \text{(periodic) bang-bang acceleration profile}$$



Estimates of dynamic coefficients



$$\text{errors } \tilde{a} = a - \hat{a}$$

first trajectory

only the estimate of the viscous
friction coefficient converges
to the true value

second trajectory

all three estimates of
dynamic coefficients converge
to their true values



A special case: Adaptive regulation

- adaptation in case q_d is **constant**
- **no special simplifications** for the presented adaptive control law (designed for the general tracking case...)

$$u = \hat{M}(q)\ddot{q}_r + \hat{S}(q, \dot{q})\dot{q}_r + \hat{g}(q) + \hat{F}_v\dot{q}_r + K_P e + K_D \dot{e}$$

$$\dot{\hat{a}} = \Gamma Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)(\dot{q}_r - \dot{q})$$

since $\dot{q}_r = \Lambda(q_d - q)$ and $\ddot{q}_r = -\Lambda\dot{q}$ **do not** vanish!

- a **different** case would be the availability of an adaptive version of the trajectory tracking controller

$$u = \hat{M}(q)\ddot{q}_d + \hat{S}(q, \dot{q})\dot{q}_d + \hat{g}(q) + \hat{F}_v\dot{q}_d + K_P e + K_D \dot{e}$$

since, when q_d collapses to a constant, **only the adaptation of the gravity term** would be left over (which is what one would naturally expect...)



An efficient adaptive regulator

- use a linear parameterization of the **gravity term** only

$$g(q) = G(q)a_g$$

with a p_g -dimensional vector a_g

- an adaptive regulator yielding **global asymptotic stability** of the equilibrium state $(q_d, 0)$ is provided by

$$u = G(q)\hat{a}_g + K_P(q_d - q) - K_D\dot{q}$$

$$\dot{\hat{a}}_g = \gamma G^T(q) \left(\frac{2e}{1 + 2\|e\|^2} - \beta\dot{q} \right), \quad \gamma > 0$$

where $e = q_d - q$, $K_P > 0$, $K_D > 0$ (symmetric), and $\beta > 0$ is chosen sufficiently **large**

(see paper by P. Tomei, IEEE TRA, 1991; available as extra material on the course web)



An adaptive regulator

Sketch of asymptotic stability analysis

- use the function

$$V = \frac{\beta}{2} (\dot{q}^T M(q) \dot{q} + e^T K_P e) - \frac{2\dot{q}^T M(q)e}{1 + 2\|e\|^2} + \frac{1}{2} (\hat{a}_g - a_g)^T (\hat{a}_g - a_g)$$

- a sufficient condition for V to be a **Lyapunov candidate** is that

$$\beta > \frac{2M_M}{\sqrt{M_m K_{P,m}}}$$

- a sufficient condition which guarantees **also** that

$$\dot{V} = \dots \leq -a\|e\|^2 - b\|\dot{q}\|^2 \leq 0, \quad a > 0, b > 0$$

is

$$\beta > \max \left\{ \frac{2M_M}{\sqrt{M_m K_{P,m}}}, \frac{1}{K_{D,m}} \left(\frac{K_{D,m}^2}{2K_{P,m}} + 4M_M + \frac{\alpha_S}{\sqrt{2}} \right) \right\}$$

$\|S(q, \dot{q})\| \leq \alpha_S \dot{q}$

Note: for any **symmetric, positive definite** matrix A

$$A_M = \lambda_{\max}(A) = \sqrt{\lambda_{\max}(A^T A)} = \|A\| \quad \text{and thus, e.g., } \frac{1}{2} \dot{q}^T M(q) \dot{q} \geq \frac{1}{2} M_m \|\dot{q}\|^2$$

$$A_m = \lambda_{\min}(A)$$



Robotics 2

Control in the Cartesian Space

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Regulation of robot Cartesian pose

- “PD +” type control for regulation problems
 - proportional to the **Cartesian pose error**, with a derivative term (on **velocity**) + cancellation/compensation of gravity **in joint space**
- robot
 - dynamics $M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) = u$ dimension of spaces
joint = n
 - kinematics $p = f(q) \rightarrow \dot{p} = J(q)\dot{q}$ Cartesian = m
- **goal:** asymptotic stabilization of the end-effector pose

$$p = p_d, \dot{q} = \dot{q}_d = 0 \rightarrow \dot{p}_d = 0$$

Note: if $m = n$, then $\dot{q} = 0 \Leftrightarrow \dot{p} = 0$ up to singularities

if $m < n$, then the goal is **not** uniquely associated to a complete robot state: $n - m$ joint coordinates are missing ...



A Cartesian regulation law

(*)

$$u = J^T(q)K_P(p_d - p) - K_D\dot{q} + g(q)$$

$K_P, K_D > 0$
(symmetric)

Theorem

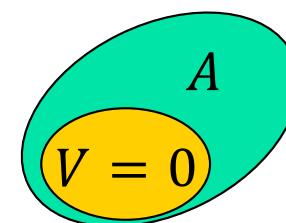
under the control law (*), the robot state will converge asymptotically
to the set
$$\begin{aligned} A &= \{\dot{q} = 0, q: K_P(p_d - f(q)) \in N(J^T(q))\} \\ &\supseteq \{\dot{q} = 0, q: f(q) = p_d\} \end{aligned}$$

Proof

define $e_p = p_d - p$ (Cartesian error) and the associated
Lyapunov-like candidate function

$$V = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}e_p^T K_P e_p$$

with $V = 0 \Leftrightarrow (\dot{q}, q) \in \{\dot{q} = 0, q: f(q) = p_d\} \subseteq A$





Proof (cont)

differentiating $V = \frac{1}{2} \dot{q}^T M(q) \dot{q} + \frac{1}{2} e_p^T K_P e_P \geq 0$

$$\begin{aligned}\dot{V} &= \dot{q}^T (M\ddot{q} + \frac{1}{2} \dot{M}\dot{q}) - e_p^T K_P \dot{p} \\ &= \dot{q}^T (u - S\dot{q} - g + \frac{1}{2} \dot{M}\dot{q}) - e_p^T K_P \dot{p} \\ &= \dot{q}^T (J^T K_P e_P - K_D \dot{q} + g - g) - e_p^T K_P J \dot{q} \\ &= -\dot{q}^T K_D \dot{q} \leq 0\end{aligned}$$

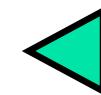
with $\dot{V} = 0 \Leftrightarrow \dot{q} = 0$

in this situation, the **closed-loop equations** become

$$M(q)\ddot{q} + g(q) = J^T(q)K_P e_P + g(q) \rightarrow \ddot{q} = M^{-1}(q)J^T(q)K_P e_P$$

$\rightarrow \ddot{q} = 0 \Leftrightarrow K_P e_P \in N(J^T(q))$

by applying LaSalle theorem, the thesis follows





Corollary

for a given initial state $(q(0), \dot{q}(0))$, if the robot **does not encounter any singularity** of $J^T(q)$ (configurations where $\rho(J^T) < m \leq n$) during its motion, then there is **asymptotic stabilization** to one single state (when $m = n$) or to a set of states (when $m < n$) such that

$$e_P = 0, \dot{q} = 0$$

Note: singular configurations q of $J^T(q)$ coincide with those of $J(q)$

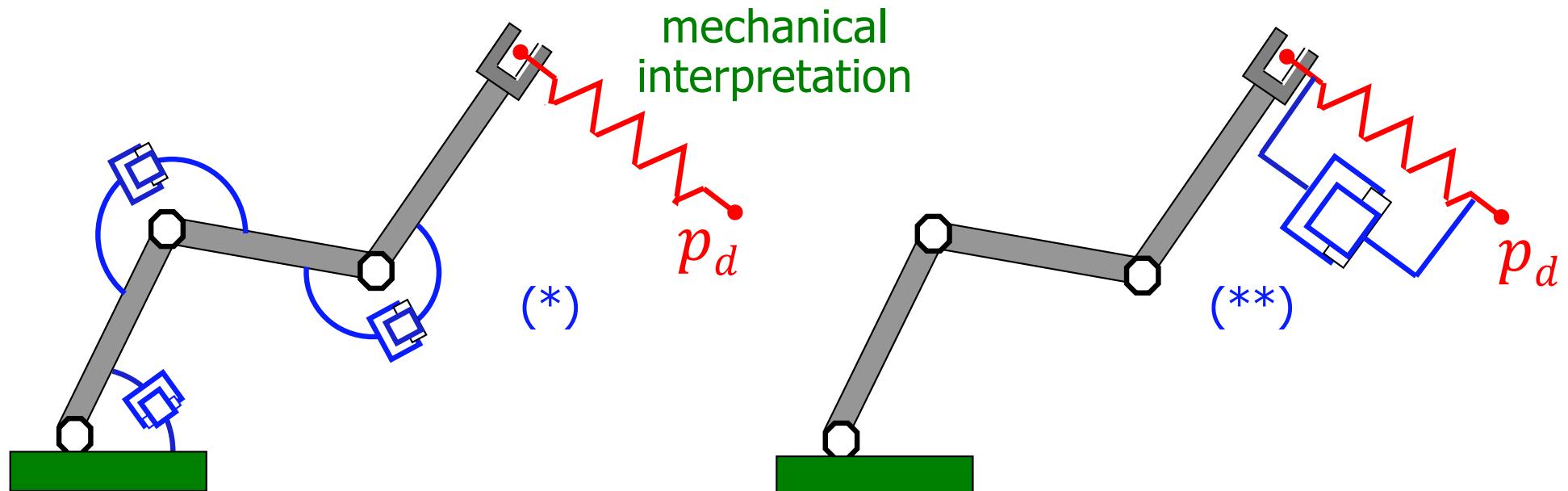


A possible variant for regulation

“all Cartesian” PD control + gravity cancellation in joint space

$$(**) \quad u = J^T(q)[K_P(p_d - p) - K_D\dot{p}] + g(q)$$

$K_P, K_D > 0$
(symmetric)



J^T transforms the “virtual” **elastic**, for (*), or **visco-elastic**, for (**), force/torque acting on the end-effector into control torques at the joints



Feedback linearization in Cartesian space

robot

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u$$

output

$$y = p, \quad p = f(q)$$

Cartesian
position/orientation

assume: $m = n$

algorithm

differentiate the output(s) as many times as needed up to the appearance of (at least one of) the input torque(s), then verify if it is possible to solve for the input = "inversion"

uniform
"relative degree"
 $\rho = 2$
for all outputs

$$y = f(q)$$

$$\dot{y} = J(q)\dot{q}$$

$$\ddot{y} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$$

$$= J(q)M^{-1}(q)[u - c(q, \dot{q}) - g(q)] + \dot{J}(q)\dot{q}$$

from the dynamic model

Theorem

for a non-redundant robot, it is possible to exactly linearize and decouple the dynamic behavior at the **Cartesian** level **if and only if**

$$\det J(q) \neq 0$$



Feedback linearization in Cartesian space (in the right coordinates!)

control law

$$u = M(q)J^{-1}(q)a + c(q, \dot{q}) + g(q) - M(q)J^{-1}(q)\dot{J}(q)\dot{q}$$

$$= \beta(q)a + \alpha(q, \dot{q})$$

→ $\ddot{\tilde{y}} = \ddot{p} = J(q)M^{-1}(q)[u - c(q, \dot{q}) - g(q)] + \dot{J}(q)\dot{q} = a$

p, \dot{p} are the so-called "**linearizing**" coordinates

closed-loop equations (in the **joint space**)

$$M^{-1} * M\ddot{q} + c + g = MJ^{-1}[a - \dot{J}\dot{q}] + c + g$$

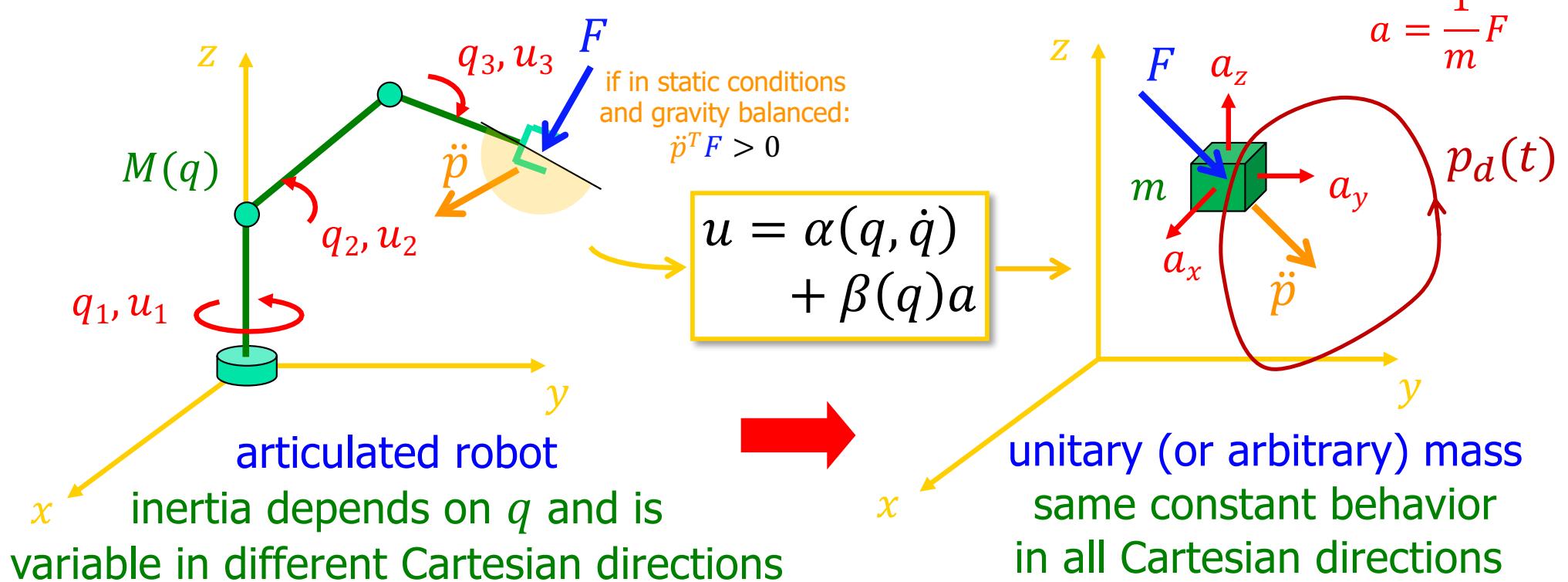
→ $\ddot{q} = J^{-1}(q)a - J^{-1}(q)\dot{J}(q)\dot{q}$

purely
kinematic
equations

(but still **nonlinear** and **coupled!!**)



Physical interpretation



when a force F is applied at the end-effector

- the uncontrolled robot will accelerate with \ddot{p} in a different direction
on the other hand
- a mass m accelerates always in the **same** direction of the applied force F



Alternative derivation in purely Cartesian terms

the previous exact linearizing and decoupling law can be rewritten in **Cartesian terms** using a **control force/torque F**

$$u = M(q)J^{-1}(q)a + c(q, \dot{q}) - M(q)J^{-1}(q)\dot{J}(q)\dot{q} + g(q)$$

joint torque u is moved to the **Cartesian space** as $F = J^{-T}(q)u$ (for $m = n$)

$$\begin{aligned} F &= [J^{-T} MJ^{-1}]a \xrightarrow{\text{Cartesian inertia}} [JM^{-1}J^T]^{-1} = M_p(p) \\ &\quad + [J^{-T} c - J^{-T} MJ^{-1}\dot{J}\dot{q}] \xrightarrow{\text{Cartesian Coriolis/centrifugal terms}} \\ &\quad + [J^{-T} g] \xrightarrow{\text{Cartesian gravity}} \\ &= M_p a + c_p + g_p \end{aligned}$$

 this is the feedback linearization law applied to the **Cartesian dynamic model** of the robot

$$M_p(p)\ddot{p} + c_p(p, \dot{p}) + g_p(p) = F$$

$$\ddot{p} = a$$



Remarks - 1

- the design of a **Cartesian trajectory tracking control** is completed by **stabilizing** the tracking error in the **m independent** chains of double integrators, i.e., by setting

$$a_i = \ddot{p}_{di} + K_{Di}(\dot{p}_{di} - \dot{p}_i) + K_{Pi}(p_{di} - p_i) \quad \begin{matrix} \text{scalars} \\ K_{Pi} > 0, K_{Di} > 0 \\ i = 1, \dots, m \end{matrix}$$

- the transient behavior of the Cartesian error along a desired trajectory is **exponentially stable** (with arbitrary eigenvalues assigned by choosing the diagonal gains of K_P, K_D)
- for $p_d = \text{constant}$ (regulation task), the control law becomes

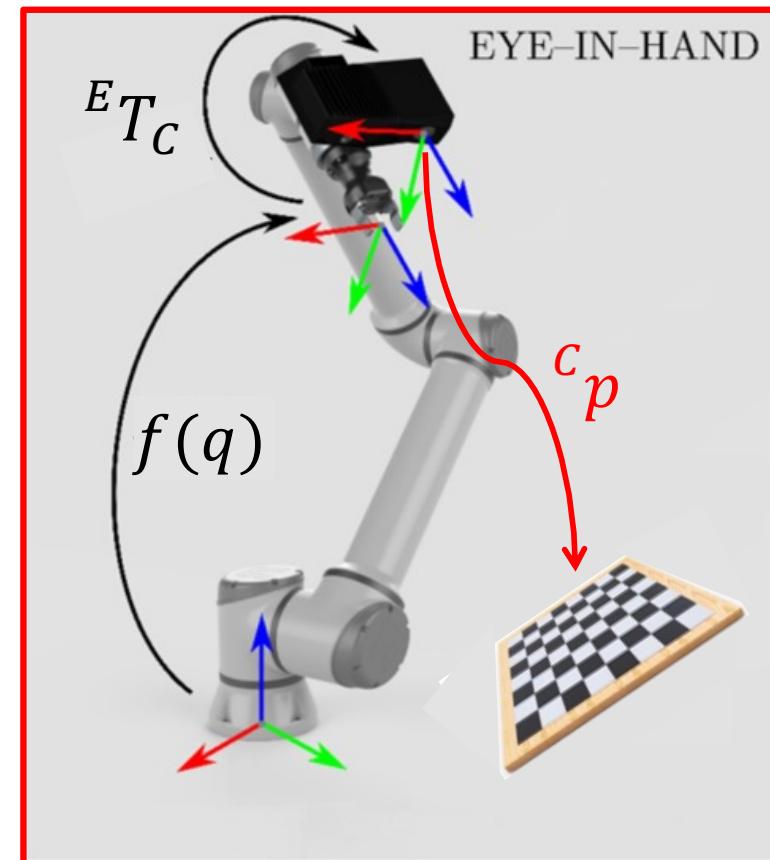
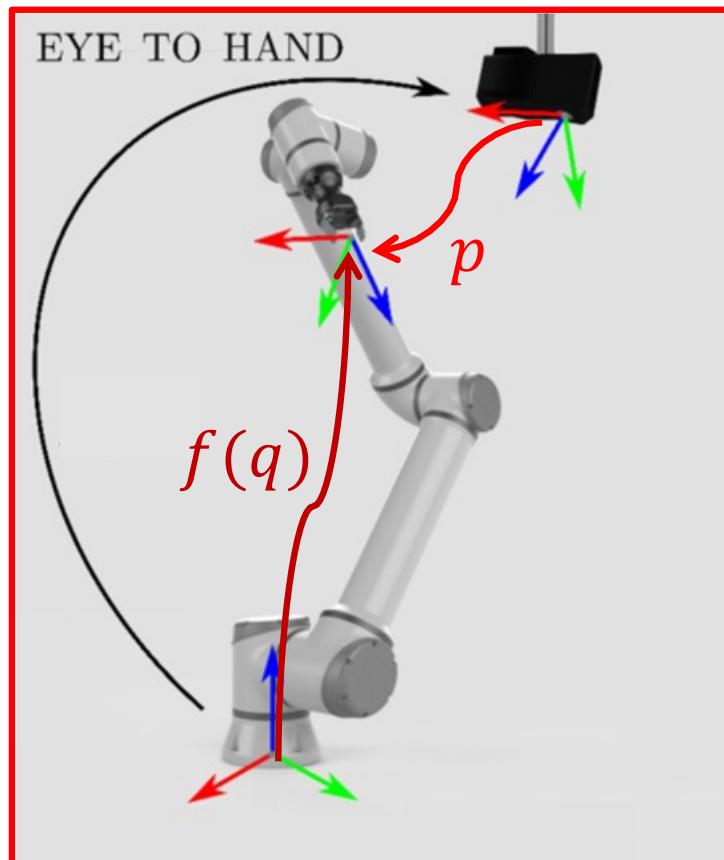
$$u = M(q)J^{-1}(q)[K_P e_P - K_D J(q)\dot{q}] + c(q, \dot{q}) + g(q) - M(q)J^{-1}(q)\dot{J}(q)\dot{q}$$

which is computationally heavier than a control law designed directly for regulation, such as previous laws **(*)** or **(**)**, but it keeps the property of an exponentially stable transient error



Remarks - 2

- the Cartesian pose/velocity can either be directly **measured** by external sensors (cameras: eye-to-hand/eye-in-hand) or **computed** through direct and differential kinematics of the robot





Remarks - 3

- in **redundant** robots ($m < n$), by replacing $MJ^{-1} = (JM^{-1})^{-1}$ in the control law with some (weighted) pseudoinverse $(JM^{-1})_W^\#$, one still obtains **input-output** decoupling and linearization, but not exact linearization of the whole **state** dynamics
 - there is an additional internal dynamics left of dimension $n - m$



More on the redundant case ...

- suppose $m < n$, but with a Jacobian J of full rank m
- let the control law (with null-space torque term u_0) be defined as

$$u = (J(q)M^{-1}(q))_W^\# \left(a - j(q)\dot{q} + J(q)M^{-1}(q)(c(q, \dot{q}) + g(q)) \right) \\ + \left(I - (J(q)M^{-1}(q))_W^\# J(q)M^{-1}(q) \right) u_0$$

where $(JM^{-1})_W^\# = W^{-1}M^{-1}J^T(JM^{-1}W^{-1}M^{-1}J^T)^{-1}$

- three standard choices for $W > 0$

$$W = I \implies (JM^{-1})^\# = M^{-1}J^T(JM^{-2}J^T)^{-1}$$

$$W = M^{-1} \implies (JM^{-1})_{M^{-1}}^\# = J^T(JM^{-1}J^T)^{-1}$$

$$W = M^{-2} \implies (JM^{-1})_{M^{-2}}^\# = M J^T(JJ^T)^{-1} = MJ^\#$$

each associated control torque optimizes a different criterion (see the slides on redundant robots)

- all give the same $\ddot{p} = a$, with u_0 available for null-space control



Conclusions

- most of the control laws presented in the joint space (i.e., driven by a joint error) can be **translated** with relative ease to the Cartesian space, e.g.
 - regulation with constant gravity compensation
 - adaptive regulation
 - robust control for trajectory tracking
 - adaptive control for trajectory tracking
- the **main issues** are related to
 - kinematic singularities, both for the Jacobian transpose and the Jacobian inverse control laws: suitable modifications are needed to obtain **singularity robustness**
 - kinematic redundancy ($m < n$): use of a **stabilizing null-space torque** control is needed for the extra $n - m$ generalized coordinates (locally, $n - m$ joint variables)



Robotics 2

Robot Interaction with the Environment

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



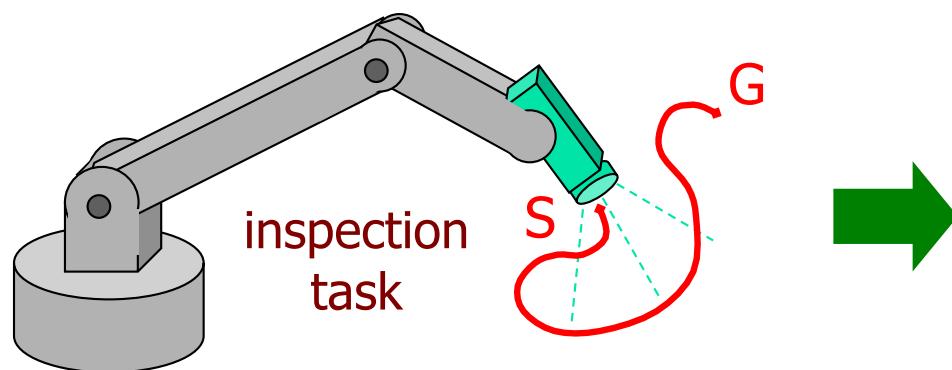


Robot-environment interaction

a robot (end-effector) may interact with the **environment**

- **modifying the state** of the environment (e.g., pick-and-place operations)
- **exchanging forces** (e.g., assembly or surface finishing tasks)

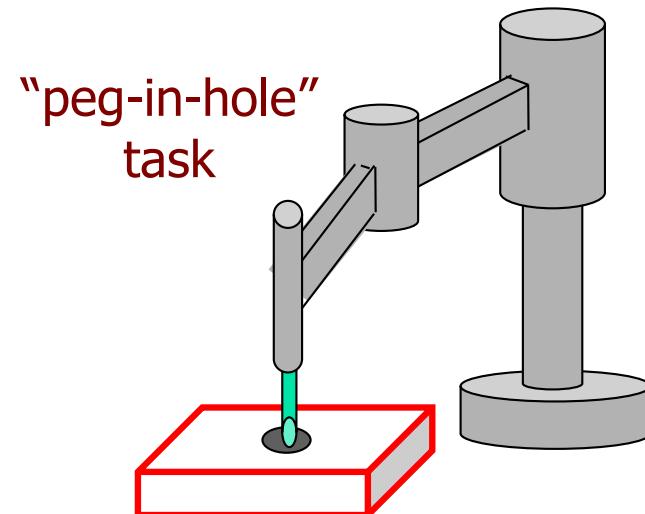
control of free motion



sensors: position (encoders)
at the joints* or
vision at the Cartesian level

*and velocity (by numerical differentiation
or, more rarely, with tachos)

control of compliant motion



sensors: as before +
6D force/torque
(at the robot wrist)



Robot compliance

PASSIVE

robot end-effector equipped with **mechatronic devices** that “comply” with the **generalized forces** applied at the TCP = Tool Center Point

RCC = Remote Center of Compliance device



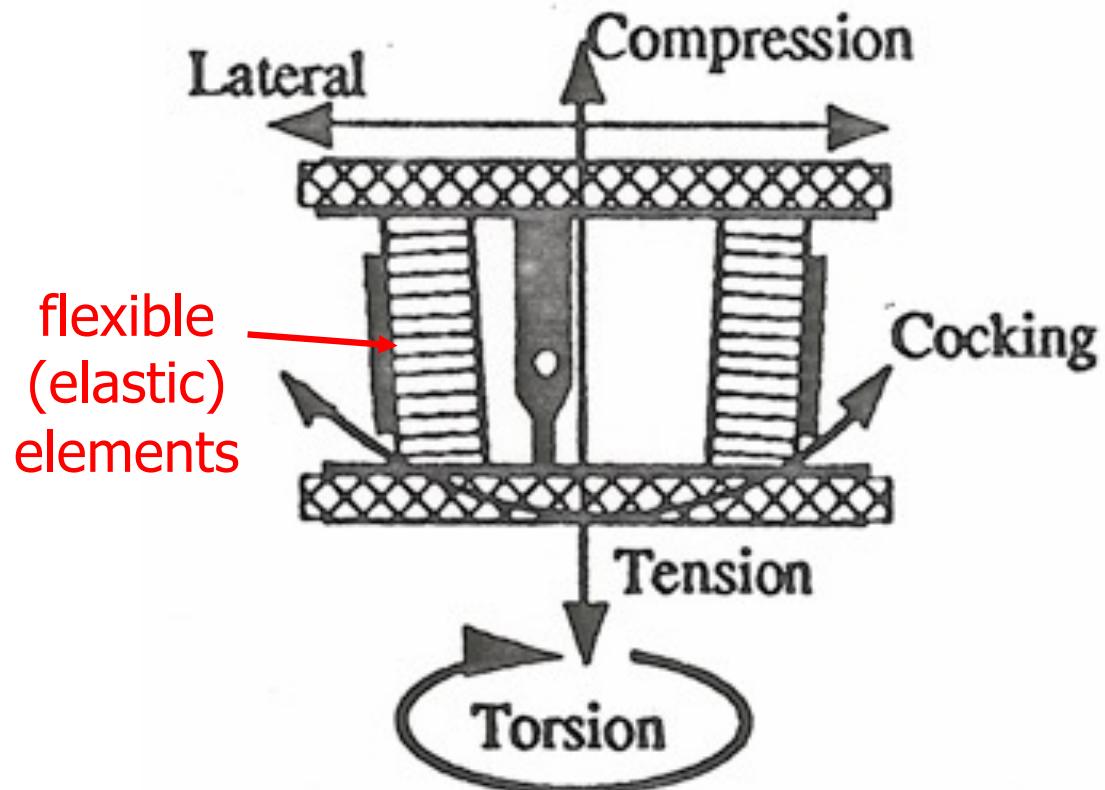
ACTIVE

robot is moved by a **control law** so as to react in a desired way to **generalized forces** applied at the TCP (typically measured by a F/T sensor)

- **admittance** control
contact forces \Rightarrow velocity commands
- **stiffness/compliance** control
contact displacements \Rightarrow force commands
- **impedance** control
contact displacements \Leftrightarrow contact forces



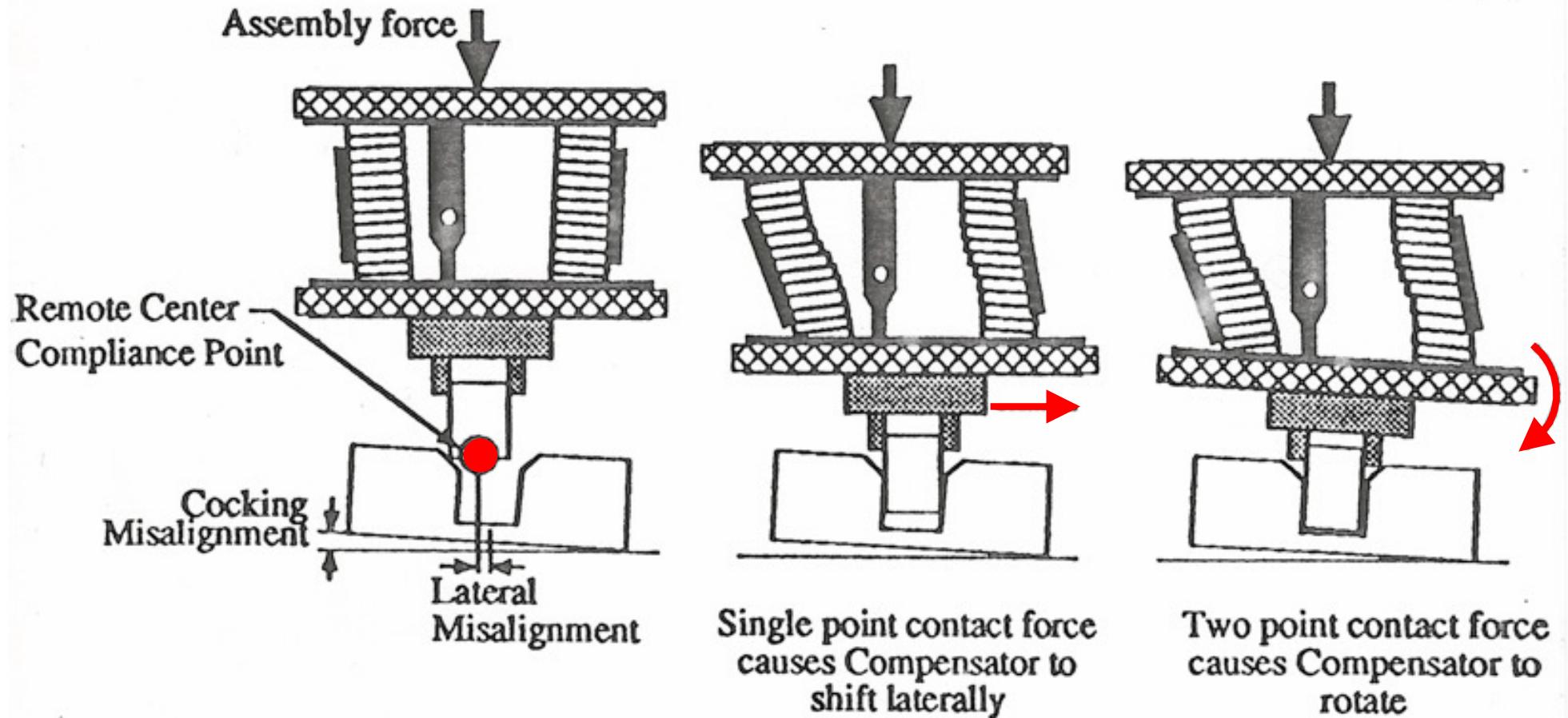
RCC device



RCC models of
different size
by ATI

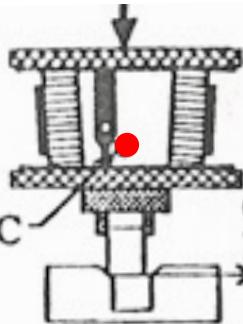


RCC behavior in case of misalignment errors in assembly tasks



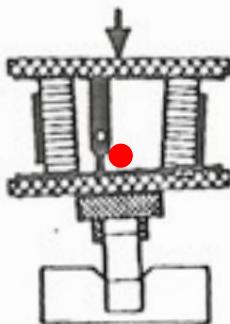


Effects of RCC positioning



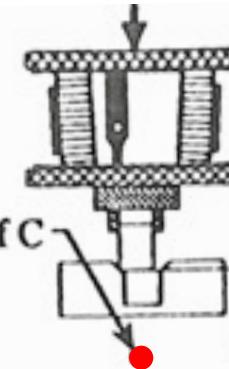
Contact
Force

C of C



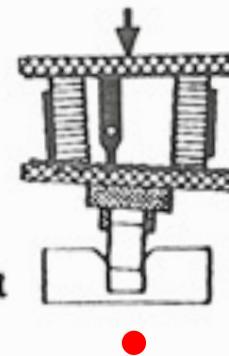
With the C of C far above the point of contact a lateral contact force causes the part to enter at an angle, causing a two point contact.

too high...



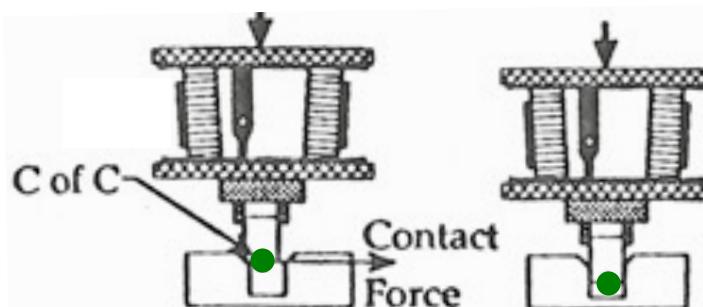
Contact
Force

C of C



With the C of C far below the point of contact the part enters at an angle causing two point contact

too low...



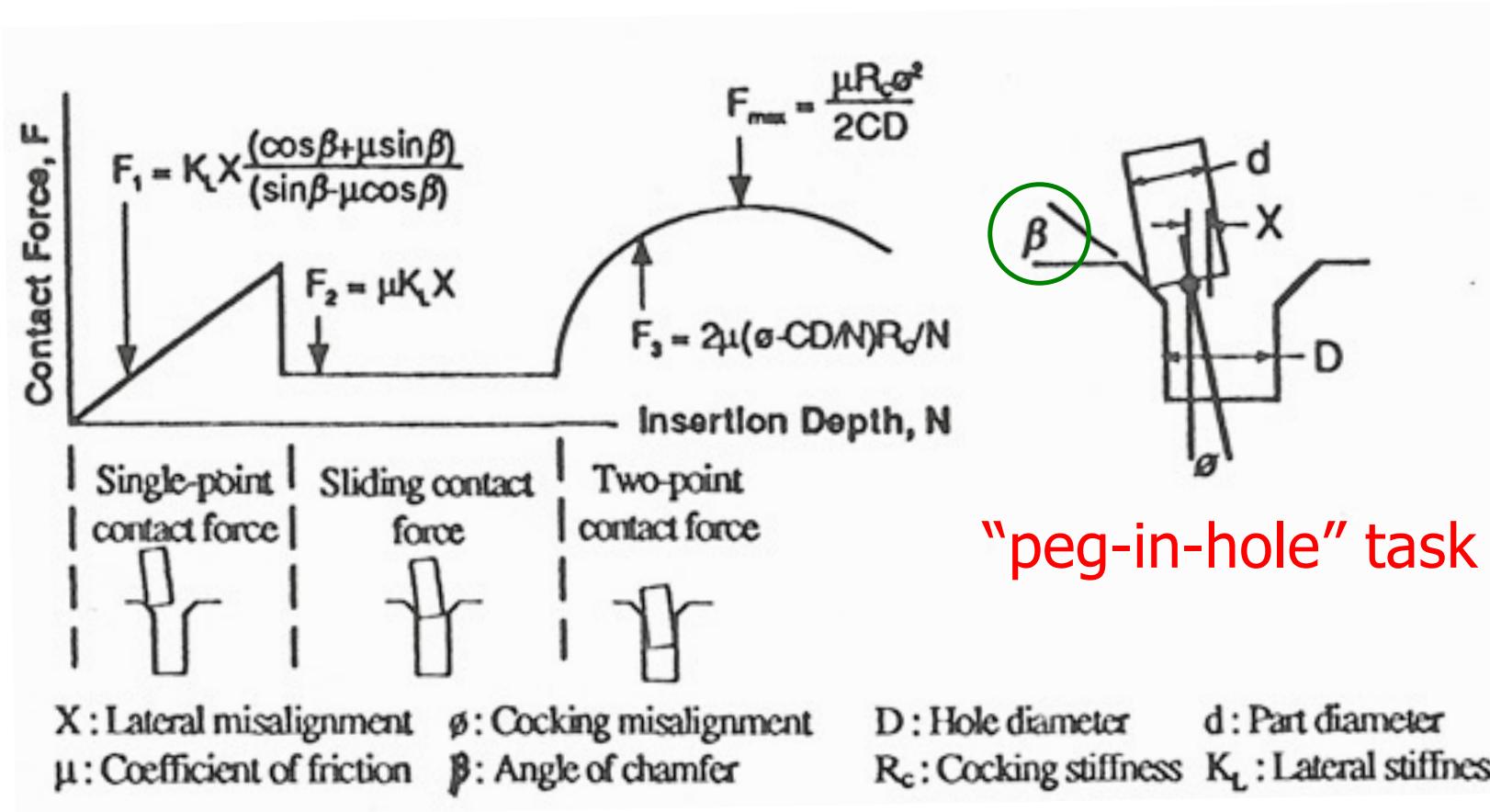
Contact
Force

When the C of C is near the contact point the part enters correctly

correct!
(TCP = RCC)



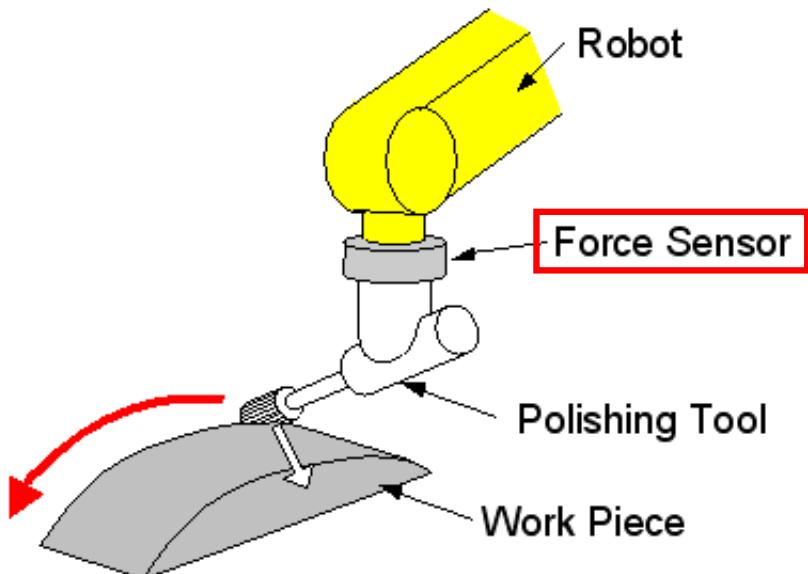
Typical evolution of assembly forces



chamfer angle β = to ease the insertion,
related also to the tolerances of the hole



Active compliance for contour following



Following with constant pushing force



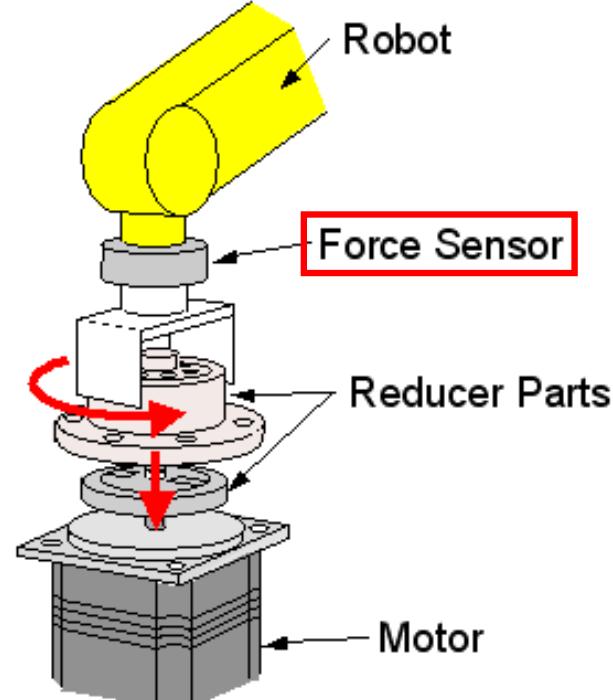
Washstand



Metal Cabinet



Active compliance “matching” of mechanical parts



Phase matching by force sensing



Gear Parts

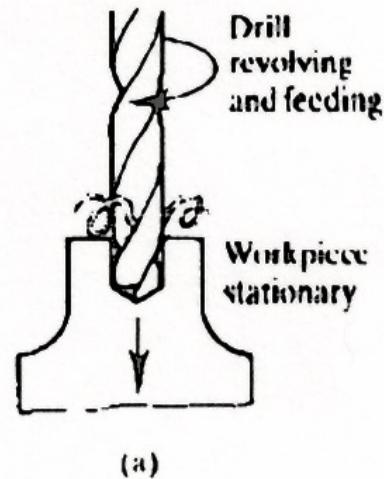


Tasks with environment interaction

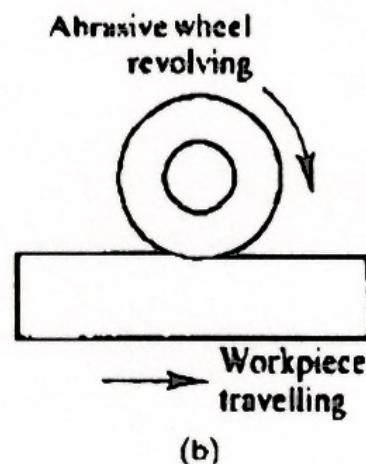
- mechanical machining
 - deburring, surface finishing, polishing, assembly,...
- tele-manipulation
 - force feedback improves performance of human operators in master-slave (leader-follower) systems
- contact exploration for shape identification
 - force and velocity/vision sensor fusion allow 2D/3D geometric identification of unknown objects and their contour following
- dexterous robot hands
 - power grasp and fine in-hand manipulation require force/motion cooperation and coordinated control of the multiple fingers
- cooperation of multi-manipulator systems
 - the environment includes one or more other robots with their own dynamic behaviors
- physical human-robot interaction
 - humans as active, dynamic environments that need to be handled under full safety premises ...



Examples of mechanical machining



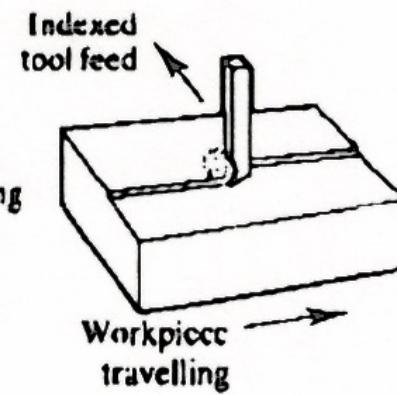
(a)



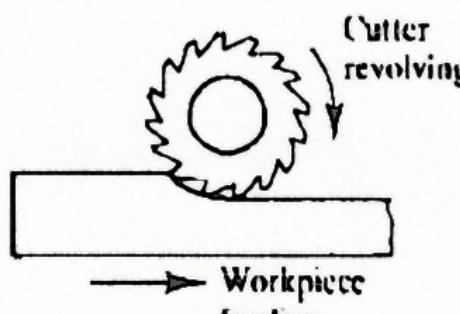
(b)



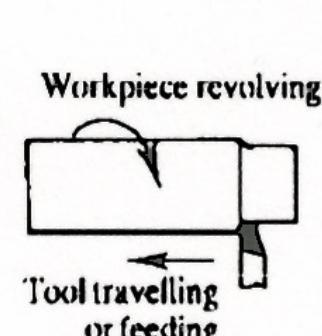
(c)



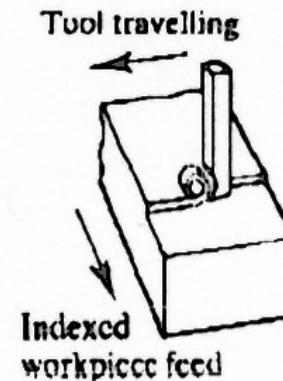
(d)



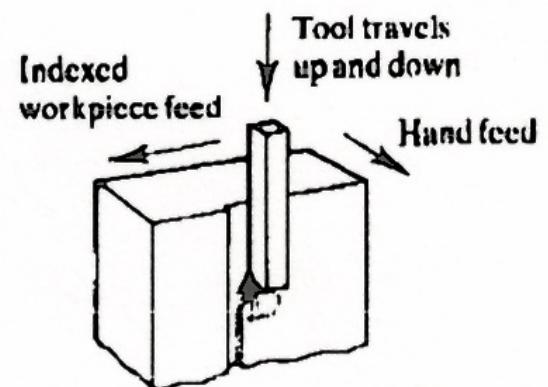
(e)



(f)



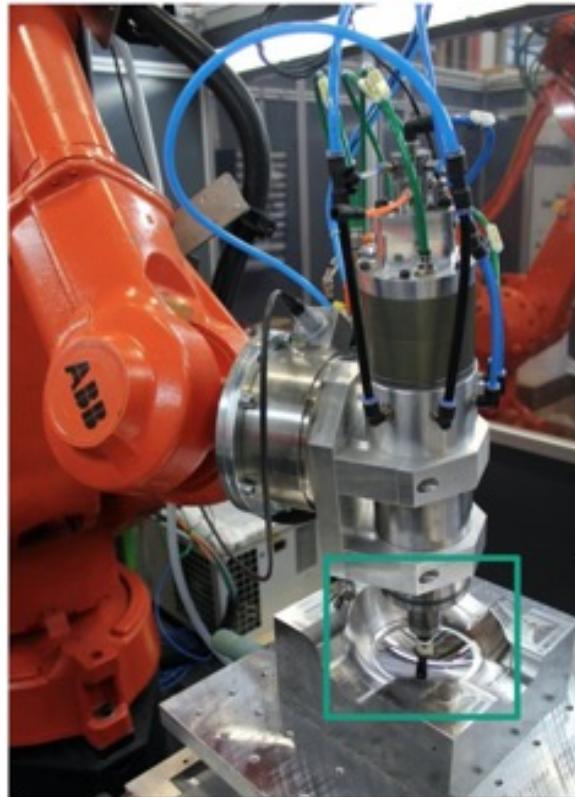
(g)



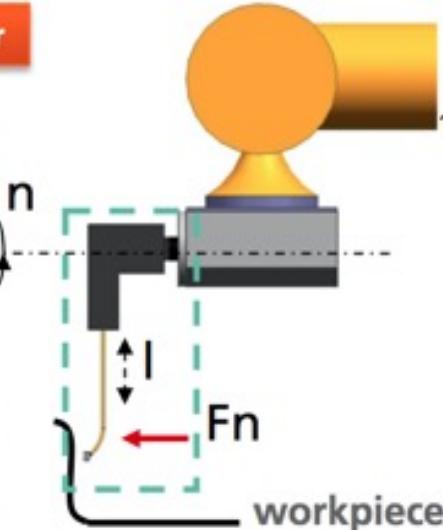
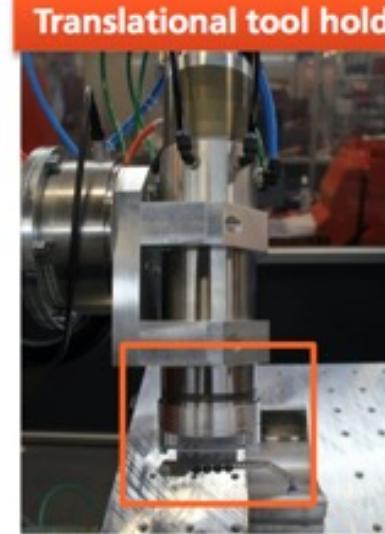
(h)



Abrasive finishing of surfaces



Translational tool holder

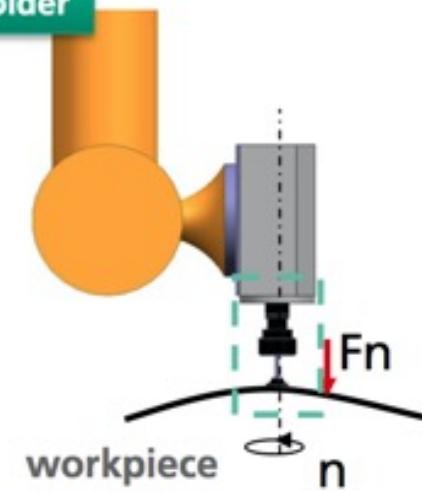


Rotational tool holder



Main properties:

- synchronous motor
- rotation : 100 - 36.000 rpm
- power : 6 kW
- mass : 16 kg
- automated tool exchanger
- pneumatic canals for the force control (x3)





Abrasive finishing of surfaces

video

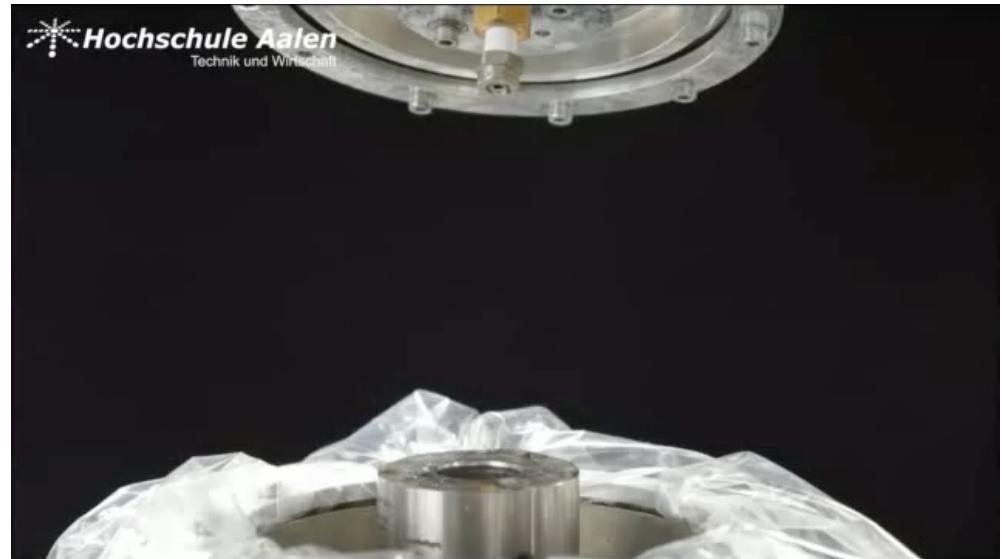


technological processes: cold forging of surfaces
and hammer peening by pneumatic machine



Non-contact surface finishing

video



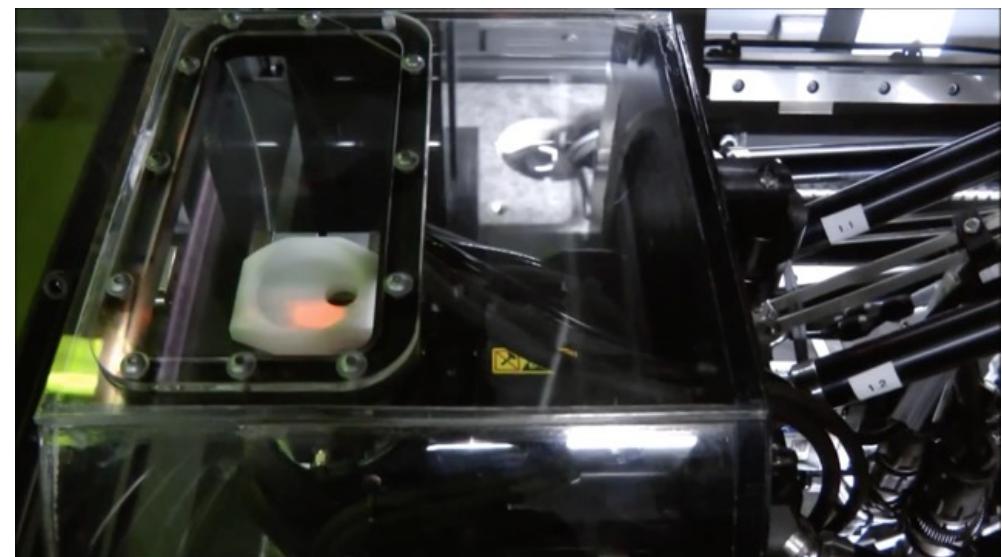
Fluid Jet technology



H2020 EU project for the
Factory of the Future (FoF)

Pulsed Laser technology

video





In all cases ...

- for physical interaction tasks, the **desired motion** specification and execution should be integrated with complementary data for the **desired force**
 → **hybrid force/motion** planning and control objectives
- the exchanged forces/torques at the contact(s) with the environment can be explicitly **set under control** or simply **kept limited** in an indirect way



Evolution of control approaches

a bit of history from the late 70's-mid '80s ...

- explicit control of forces/torques only [Whitney]
 - used in quasi-static operations (assembly) in order to avoid deadlocks during part insertion
- active admittance and compliance control [Paul, Shimano, Salisbury]
 - contact forces handled through position (**stiffness**) or velocity (**damping**) control of the robot end-effector
 - robot reacts as a compressed **spring** (with **damper**) in selected/all directions
- impedance control [Hogan]
 - a desired dynamic behavior is imposed to the robot-environment interaction, e.g., a “model” with forces acting on a **mass-spring-damper**
 - mimics the human arm behavior moving in an unknown environment
- hybrid force-motion control [Mason]
 - decomposes the **task space** in complementary sets of directions where **either** force **or** motion is controlled, based on
 - a **purely kinematic** robot model [Raibert, Craig]
 - the actual **dynamic model** of the robot [Khatib]



appropriate for fast and accurate motion in dynamic interaction...

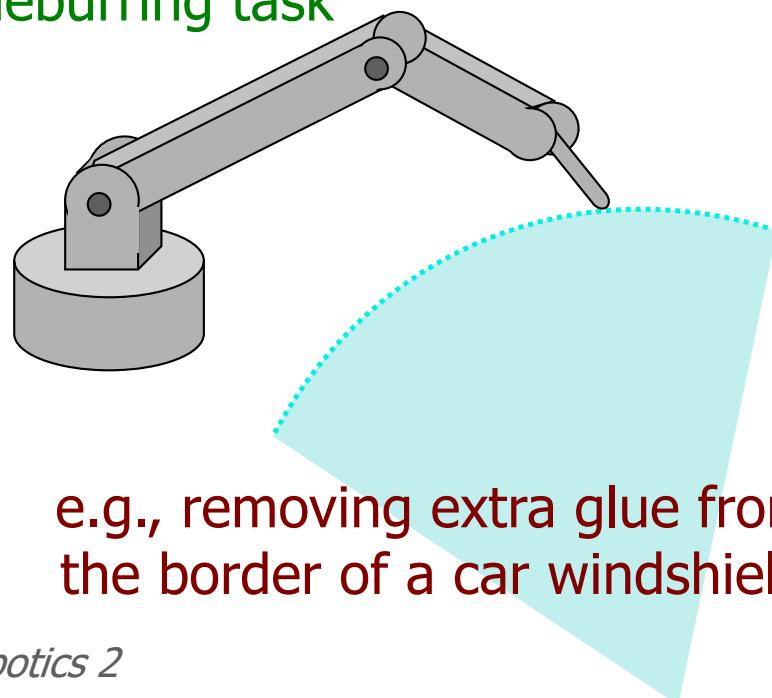


Interaction tasks of interest

interaction tasks with the environment that require

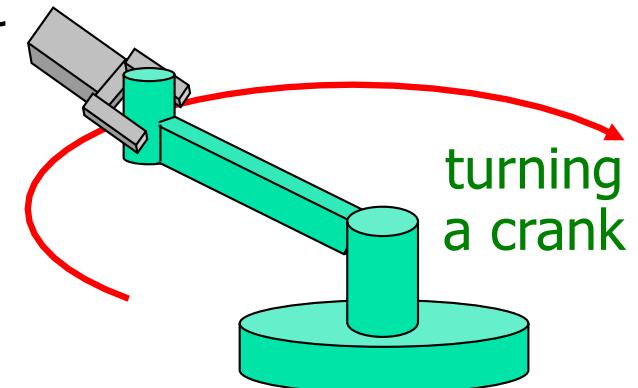
- accurate following/reproduction by the robot end-effector of desired trajectories (even at high speed) defined on the surface of objects
- control of forces/torques applied at the contact with environments having low (soft) or high (rigid) stiffness

deburring task



e.g., removing extra glue from
the border of a car windshield

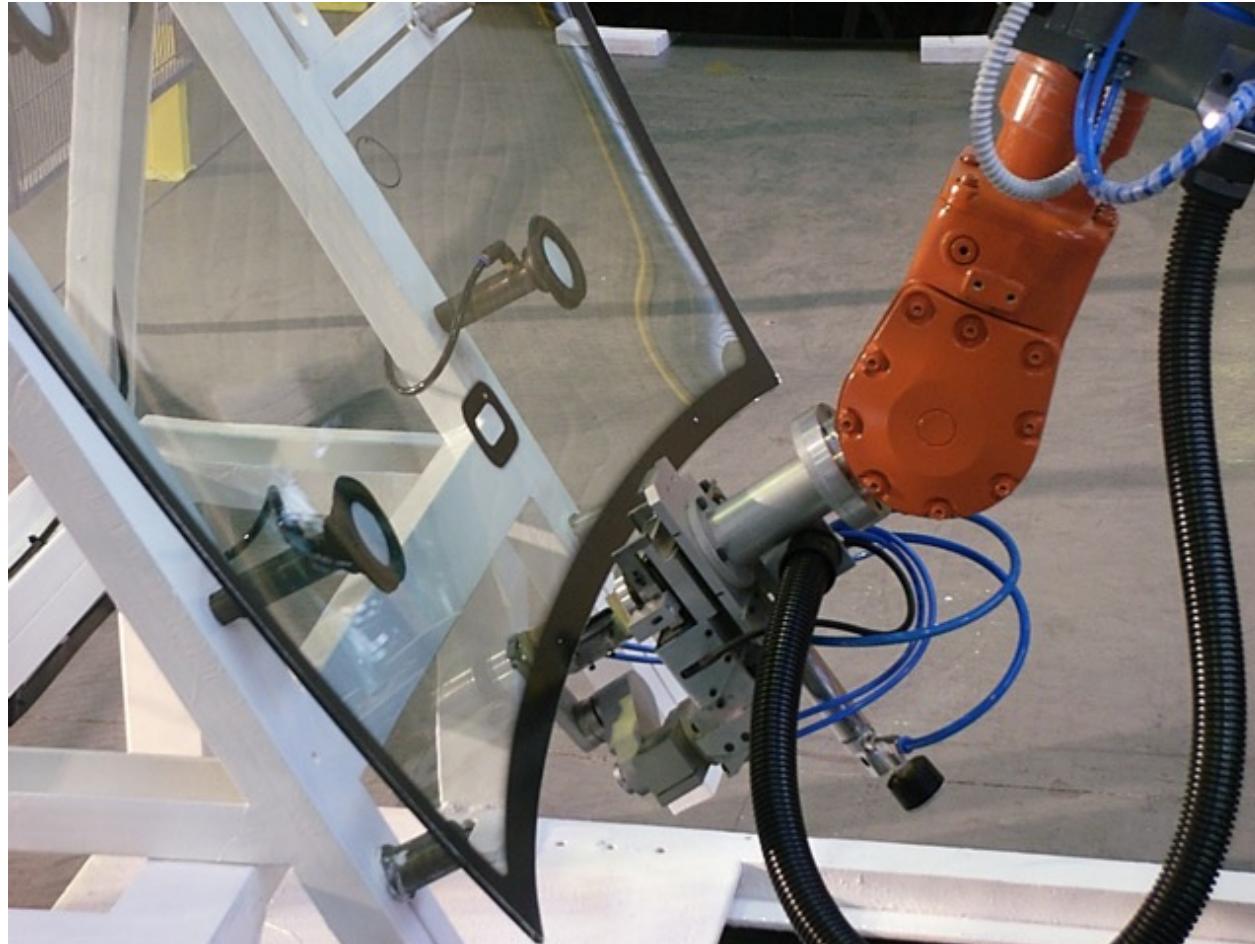
robot



e.g., opening a door



Robotized deburring of windshields



c/o ABB Excellence Center in Cecchina (Roma), 2002



Impedance vs. Hybrid control

environment model (\leftrightarrow domain of control application)

impedance control

- environment = mechanical system undergoing **small but finite deformations**
- contact forces arise as the result of a balance of two **coupled dynamic systems** (robot+environment)
→ desired dynamic characteristics are assigned to the force/motion interaction

hybrid force/motion control

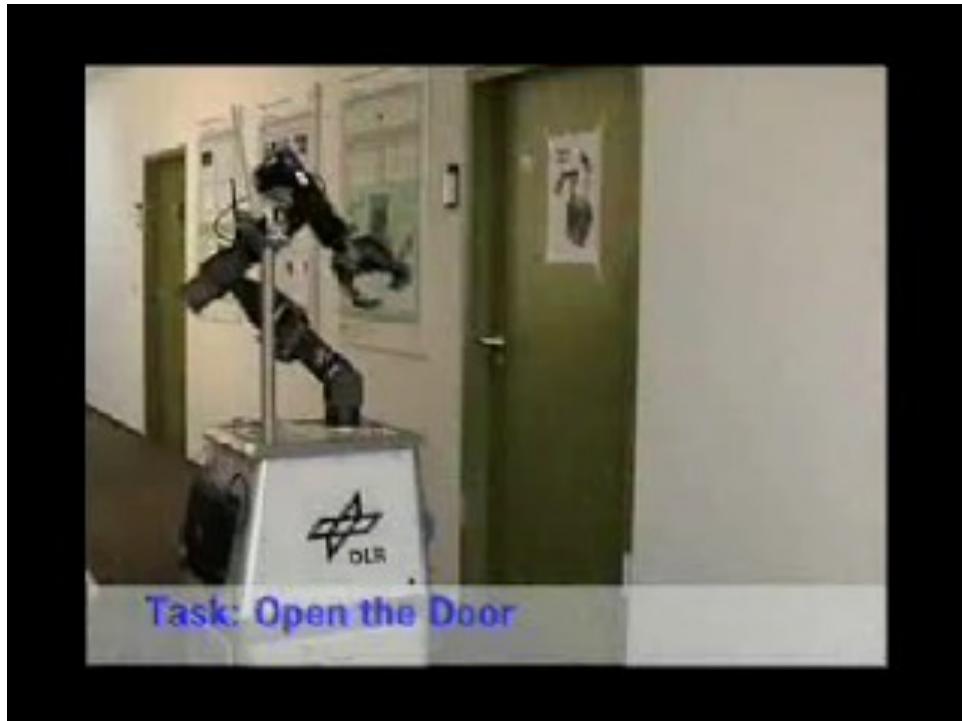
- a **rigid environment** reduces the degrees of freedom of the robot when in (bi-/uni-lateral) contact
- contact forces result from attempts to violate **geometric constraints** imposed by the environment
→ task space is decomposed in sets of directions where **only motion** or **only reaction forces** are feasible

- the required **level of knowledge** about the environment geometry is only **apparently** different between the two control approaches
- however, **measuring contact forces** may not be needed in impedance control, while it always necessary in hybrid force/motion control

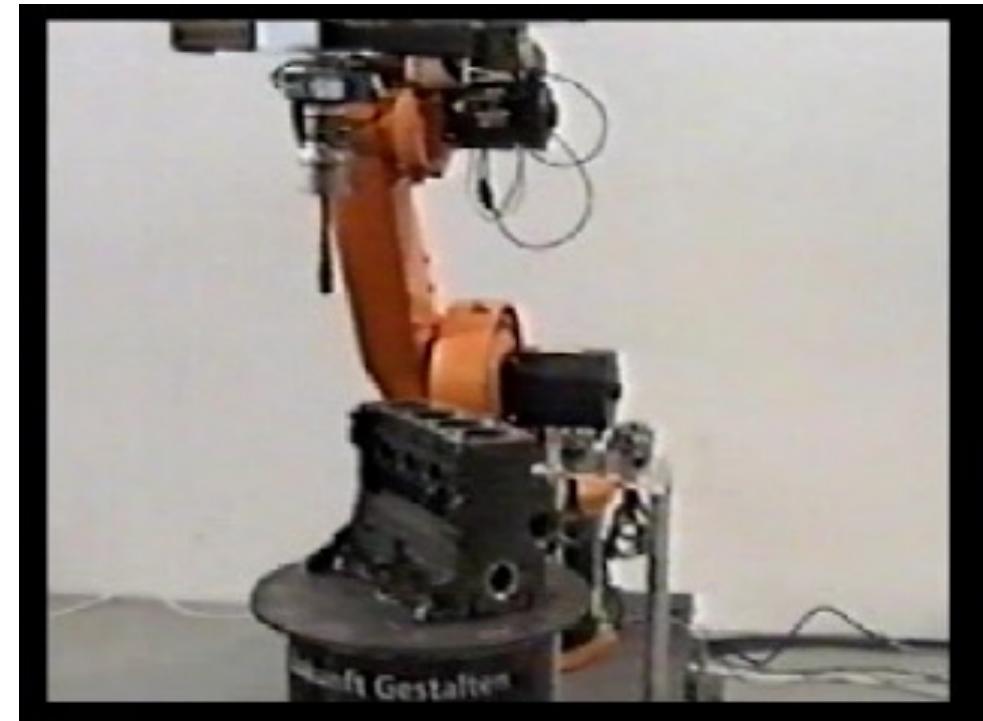


Impedance vs. Hybrid control

- opening a door with a mobile manipulator under **impedance control**
- piston insertion in a motor based on **hybrid control** of force-position (visual)



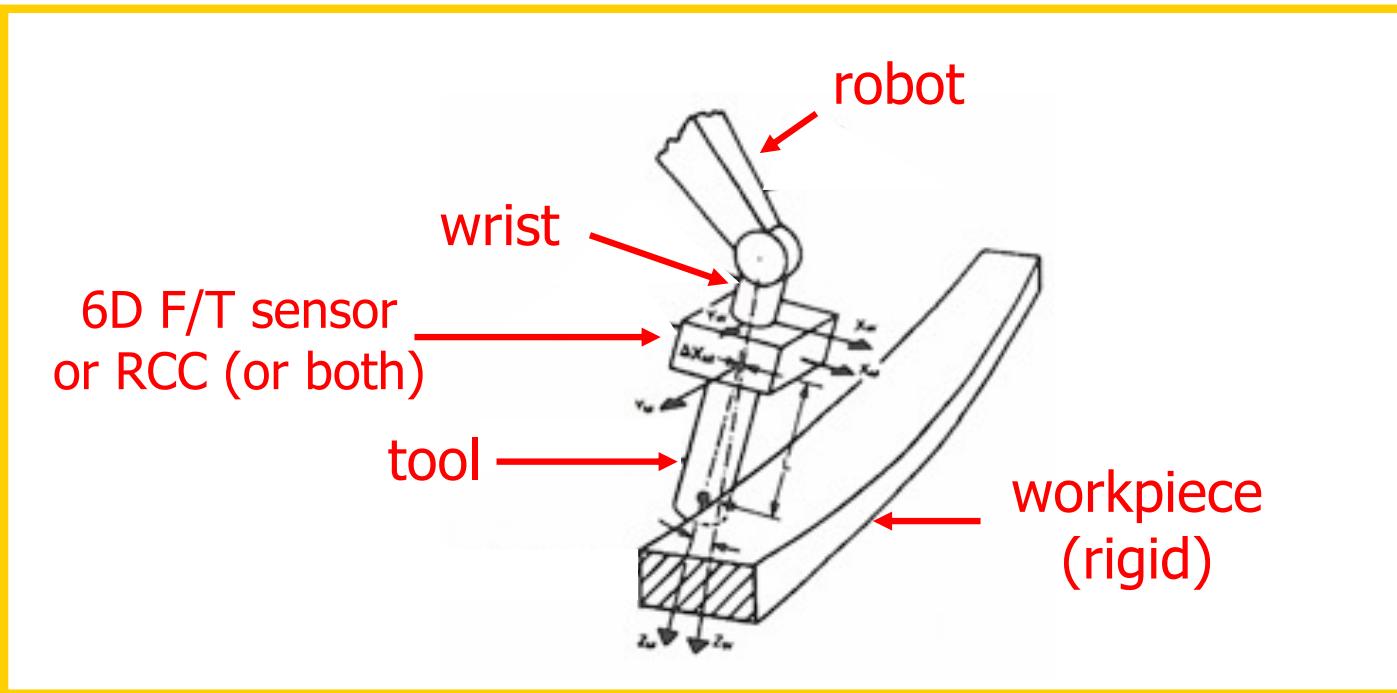
video



video



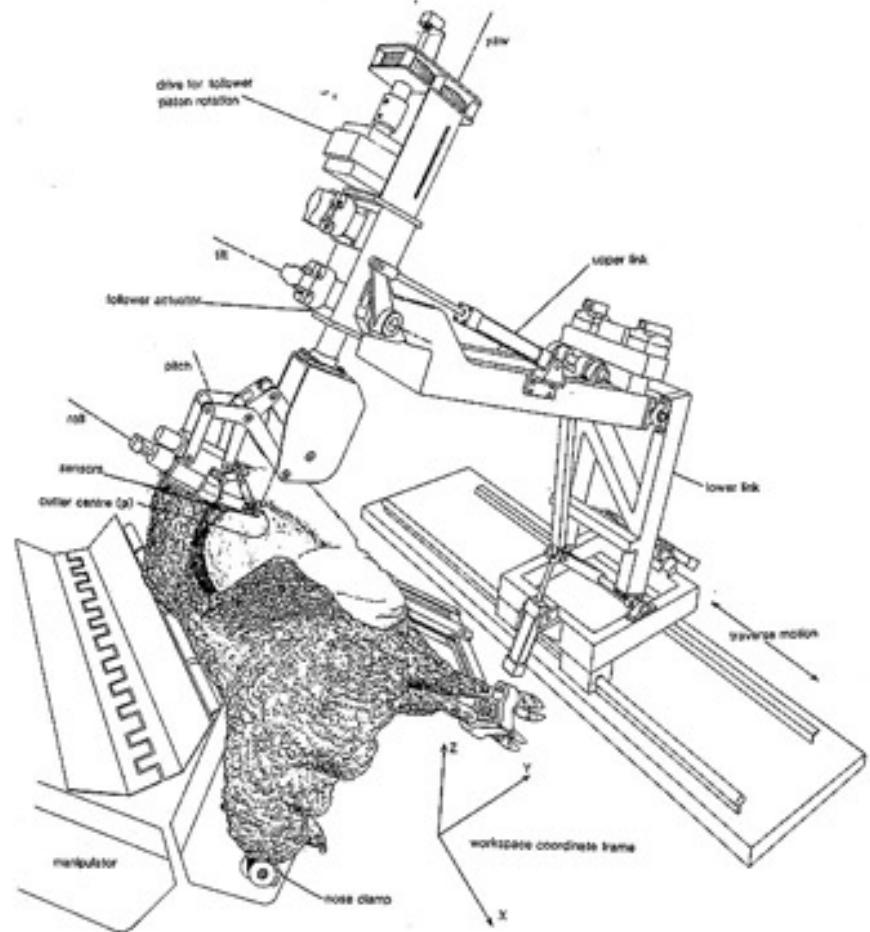
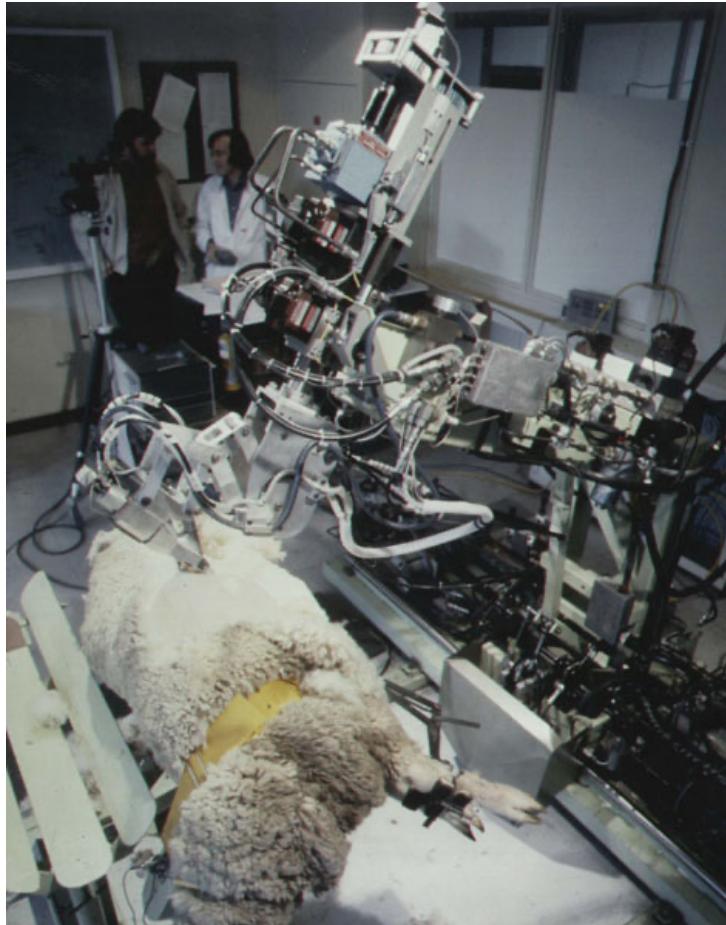
A typical constrained situation ...



the robot end-effector follows in a stable and accurate way the geometric profile of a **very stiff** workpiece, while applying a desired contact force



An unusual compliant situation ...

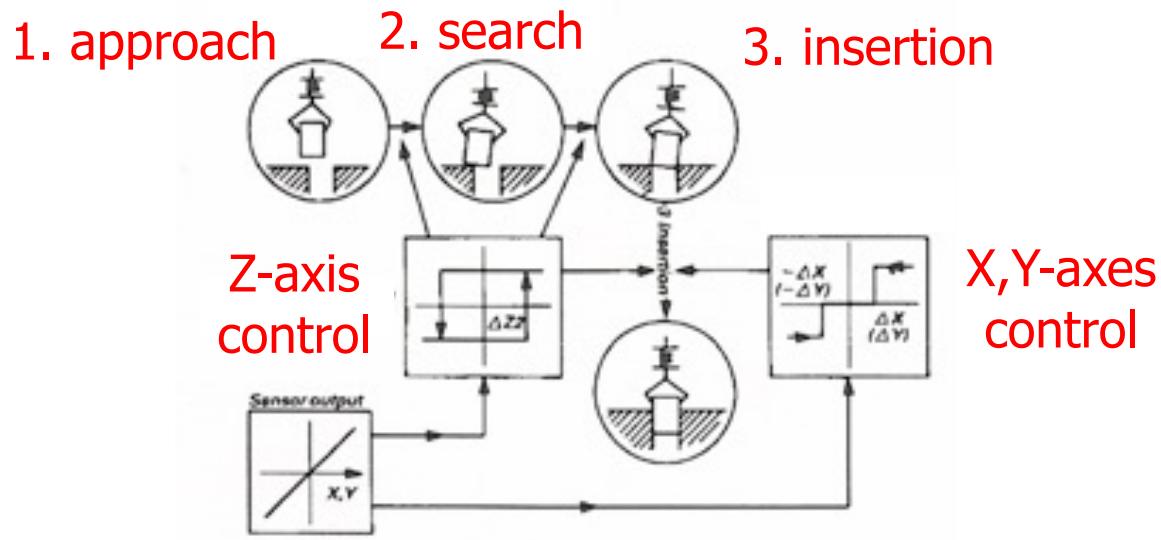


Trevelyan (AUS): Oracle robotic system in a test dated 1981

...is the sheep happy?



A mixed interaction situation

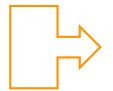


processing/reasoning on force measurements
leads to a sequence of **fine motions**
⇒ correct completion of insertion task with
help of (sufficiently large) passive compliance

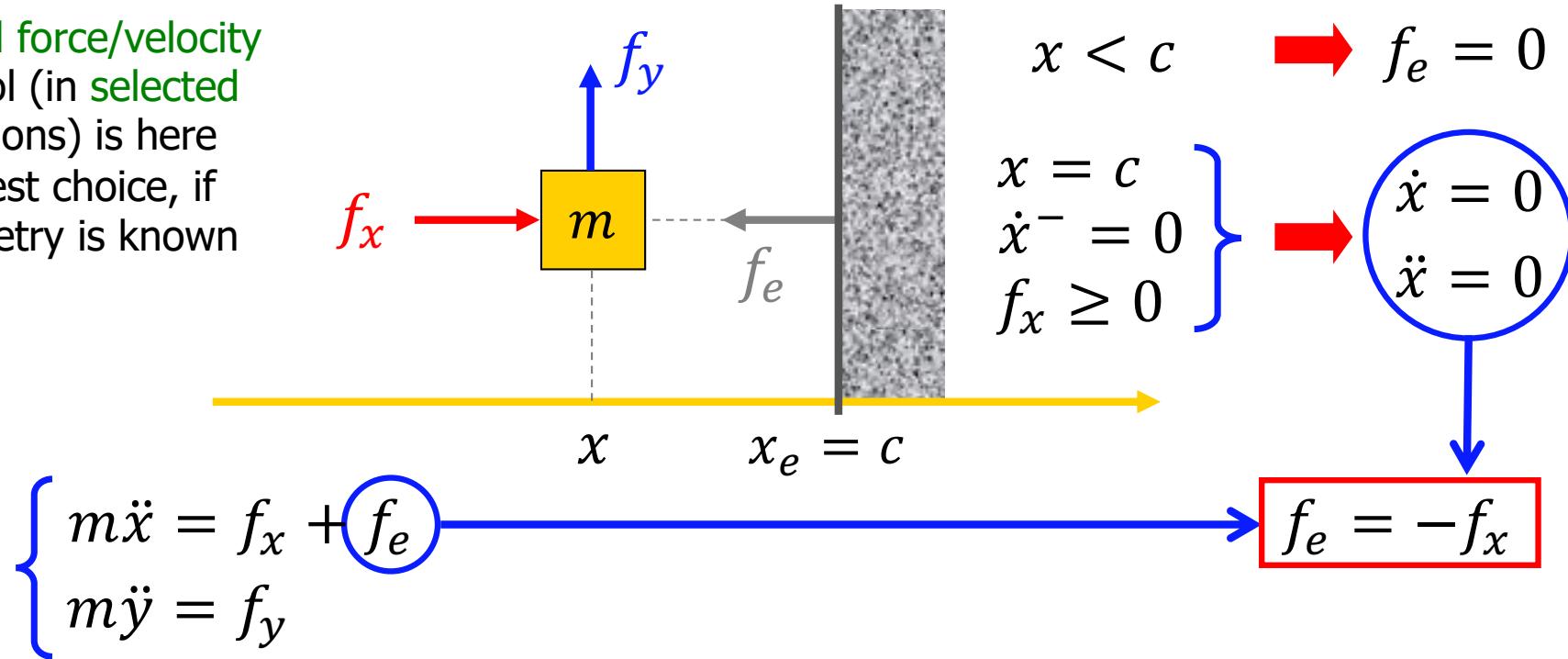


Ideally constrained contact situation

a first possible modeling choice for very stiff environments



hybrid force/velocity control (in selected directions) is here the best choice, if geometry is known



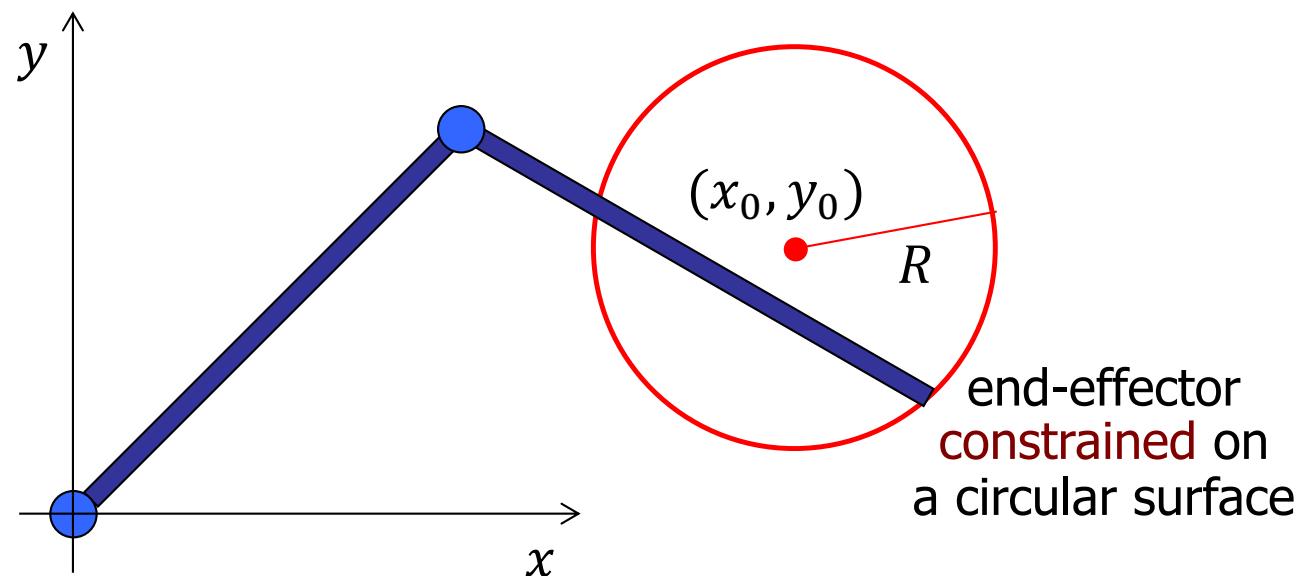
"ideal" = robot (here, a Cartesian mass) + environment are both **infinitely STIFF** (and **no friction** at the contact)

if a possible **impact** ($x = c, \dot{x}^- > 0$) is purely "elastic" (i.e., with conservation of total momentum and total kinetic energy) $\Rightarrow \dot{x}^+ = -\dot{x}^-$ (f_e is an impulse!)



In more complex situations

- how can we describe **more complex contact situations**, where the **end-effector** of an articulated robot (not yet reduced to a Cartesian mass via feedback linearization control) is **constrained** to move **on an environment surface** with nonlinear geometry?
- example: a planar 2R robot with end-effector moving on a circle





Constrained robot dynamics - 1

- consider a robot in free space described by its Lagrange **dynamic model** and a **task output function** (e.g., the end-effector pose)

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u$$

$$r = f(q) \quad q \in \mathbb{R}^n$$

- suppose that the task variables are subject to $m < n$ (bilateral) **geometric constraints** in the general form $k(r) = 0$ and define

$$h(q) = k(f(q)) = 0$$

- the **constrained robot dynamics** can be derived using again the Lagrange formalism, by defining an **augmented Lagrangian** as

$$L_a(q, \dot{q}, \lambda) = L(q, \dot{q}) + \lambda^T h(q) = T(q, \dot{q}) - U(q) + \lambda^T h(q)$$

where the **Lagrange multipliers** λ (a m -dimensional vector) can be interpreted as the **generalized forces** that arise at the contact when attempting to violate the constraints



Constrained robot dynamics - 2

- applying the **Euler-Lagrange equations** in the extended space of generalized coordinates q and multipliers λ yields

$$\frac{d}{dt} \left(\frac{\partial L_a}{\partial \dot{q}} \right)^T - \left(\frac{\partial L_a}{\partial q} \right)^T = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right)^T - \left(\frac{\partial L}{\partial q} \right)^T - \left(\frac{\partial}{\partial q} (\lambda^T h(q)) \right)^T = u$$

$$\left(\frac{\partial L_a}{\partial \lambda} \right)^T = h(q) = 0 \quad \xleftarrow{\text{contact forces do NOT produce work}}$$

→
$$\begin{cases} M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u + A^T(q)\lambda & (\star) \\ \text{subject to} \quad h(q) = 0 \end{cases}$$

where we defined the **Jacobian of the constraints** as the matrix

$$A(q) = \frac{\partial h(q)}{\partial q}$$

that will be assumed of **full row rank** ($= m$)



Constrained robot dynamics - 3

- we can **eliminate the appearance of the multipliers** as follows
 - differentiate the constraints twice w.r.t. time

$$h(q) = 0 \Rightarrow \dot{h} = \frac{\partial h(q)}{\partial q} \dot{q} = A(q)\dot{q} = 0 \Rightarrow \ddot{h} = A(q)\ddot{q} + \dot{A}(q)\dot{q} = 0$$

- substitute the joint accelerations from the dynamic model **(★)** (dropping dependencies)

$$AM^{-1}(u + A^T\lambda - c - g) + \dot{A}\dot{q} = 0$$

- solve for the multipliers

invertible $m \times m$ matrix,
when A is full rank

the inertia-weighted
pseudoinverse of the
constraint Jacobian A

$$\begin{aligned} \lambda &= (AM^{-1}A^T)^{-1}(AM^{-1}(c + g - u) - \dot{A}\dot{q}) \\ &= (A_M^\#)^T(c + g - u) - (AM^{-1}A^T)^{-1}\dot{A}\dot{q} \end{aligned}$$

to be replaced in the dynamic model...

constraint
forces λ are
uniquely
determined
by the robot
state (q, \dot{q})
and **input u !!**



Constrained robot dynamics - 4

- the final **constrained dynamic model** can be rewritten as

$$M(q)\ddot{q} = \left[I - A^T(q)(A_M^\#(q))^T \right] (u - c(q, \dot{q}) - g(q)) - M(q)A_M^\#(q)\dot{A}(q)\dot{q}$$

 **dynamically consistent** projection matrix

where $A_M^\#(q) = M^{-1}(q)A^T(q)(A(q)M^{-1}(q)A^T(q))^{-1}$ and with

$$\lambda = (A_M^\#(q))^T(c(q, \dot{q}) + g(q) - u) - (A(q)M^{-1}(q)A^T(q))^{-1}\dot{A}(q)\dot{q}$$

- if the robot state $(q(0), \dot{q}(0))$ at time $t = 0$ satisfies the constraints, i.e.,

$$h(q(0)) = 0, \quad A(q(0))\dot{q}(0) = 0$$

then the robot evolution described by the above dynamics will be consistent with the constraints **for all $t \geq 0$** and **for any $u(t)$**

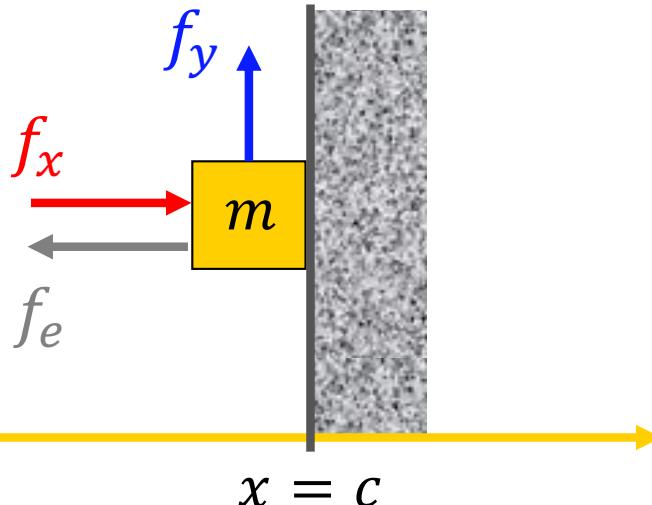
- this is a useful **simulation model** (constrained **direct** dynamics)



Example – ideal mass constrained robot dynamics

$$q = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$u = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$



$$M \ddot{q} = u$$

robot dynamics
in free motion

$$M = \begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix}$$

$$h(q) = x - c = 0 \Rightarrow A(q) = \begin{pmatrix} 1 & 0 \end{pmatrix} \Rightarrow A_M^\#(q) = \dots = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

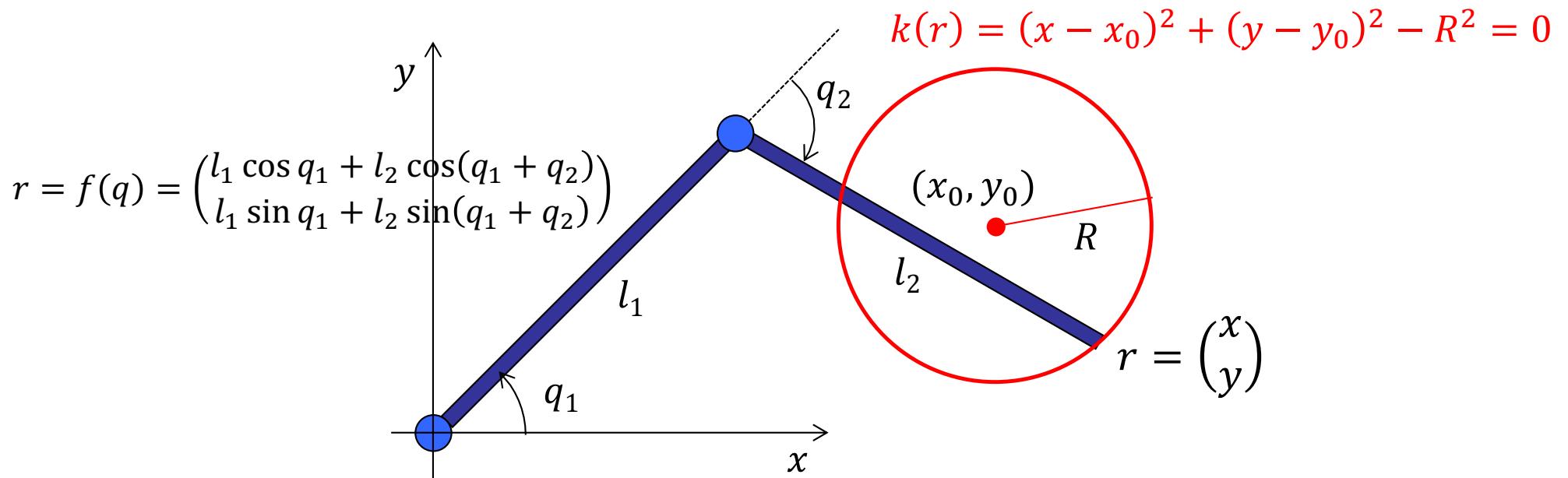
$$\left(I - A^T(q)(A_M^\#(q))^T \right) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{dynamically consistent projection matrix}$$

$$\lambda = -(A_M^\#(q))^T u = -(1 \ 0) u = -f_x \quad \text{multiplier (contact force } f_e \text{)}$$

$$M \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = M \ddot{q} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} u = \begin{pmatrix} 0 \\ f_y \end{pmatrix} \quad \text{constrained robot dynamics}$$



Example – planar 2R robot constrained robot dynamics



$$h(q) = k(f(q)) = (l_1 \cos q_1 + l_2 \cos(q_1 + q_2) - x_0)^2 + (l_1 \sin q_1 + l_2 \sin(q_1 + q_2) - y_0)^2 - R^2 = 0$$

$$\begin{aligned} \dot{h} &= \frac{\partial k}{\partial r} \frac{\partial r}{\partial q} \dot{q} = [2(x - x_0) \quad 2(y - y_0)] J_r(q) \dot{q} \\ &= [2(l_1 c_1 + l_2 c_{12} - x_0) \quad 2(l_1 s_1 + l_2 s_{12} - y_0)] J_r(q) \dot{q} = A(q) \dot{q} \end{aligned}$$



Reduced robot dynamics - 1

- by imposing m constraints $h(q) = 0$ on the n generalized coordinates q , it is also possible to **reduce** the description of the constrained robot dynamics to a **$n - m$ dimensional** configuration space

- start from constraint matrix $A(q)$ and **select** a matrix $D(q)$ such that

$$\begin{pmatrix} A(q) \\ D(q) \end{pmatrix} \text{ is a } \begin{matrix} \text{nonsingular} \\ n \times n \end{matrix} \text{ matrix} \quad \rightarrow \quad \begin{pmatrix} A(q) \\ D(q) \end{pmatrix}^{-1} = (E(q) \quad F(q))$$

- define the $(n - m)$ -dimensional vector of **pseudo-velocities** ν as the linear combination (at a given q) of the robot generalized velocities

$$\nu = D(q)\dot{q} \quad \rightarrow \quad \dot{\nu} = D(q)\ddot{q} + \dot{D}(q)\dot{q}$$

- inverse relationships (from “pseudo” to “generalized” velocities and accelerations) are given by

$$\dot{q} = F(q)\nu \quad \ddot{q} = F(q)\dot{\nu} - (E(q)\dot{A}(q) + F(q)\dot{D}(q))F(q)\nu$$

↔ properties of **block products** in inverse matrices have been used for eliminating the appearance of \dot{F} (often F is only known **numerically**)



Reduced robot dynamics – 2

whiteboard ...

$$\begin{pmatrix} A(q) \\ D(q) \end{pmatrix}^{-1} = (E(q) \quad F(q)) \quad \text{a number of properties from this definition...}$$

two matrix inverse products

$$\begin{pmatrix} A(q) \\ D(q) \end{pmatrix} (E(q) \quad F(q)) = \begin{pmatrix} A(q)E(q) & A(q)F(q) \\ D(q)E(q) & D(q)F(q) \end{pmatrix} = \begin{pmatrix} I_{m \times m} & 0 \\ 0 & I_{(n-m) \times (n-m)} \end{pmatrix}$$

$$(E(q) \quad F(q)) \begin{pmatrix} A(q) \\ D(q) \end{pmatrix} = E(q)A(q) + F(q)D(q) = I_{n \times n}$$

→ differentiating w.r.t. time $\dot{E}A + E\dot{A} + \dot{F}D + F\dot{D} = 0$ ◀

from pseudo-velocity $v = D(q)\dot{q}$

since F is a right inverse of the full row rank matrix D ($DF = I$)

three useful identities!

0

$I_{(n-m) \times (n-m)}$

(in fact,
 $D\dot{q} = DFv$
 $= v$)

→ differentiating w.r.t. time $\dot{q} = F(q)v$

$$\begin{aligned} \ddot{q} &= F\dot{v} + \dot{F}v = F\dot{v} + (\dot{F}D)\dot{q} \stackrel{(\triangleleft)}{=} F\dot{v} - (\dot{E}A + E\dot{A} + F\dot{D})Fv \\ &= F(q)\dot{v} - (E(q)\dot{A}(q) + F(q)\dot{D}(q))F(q)v \end{aligned}$$



Reduced robot dynamics - 3

- consider again the dynamic model (★), dropping dependencies

$$M\ddot{q} + c + g = u + A^T \lambda$$

- since $AE = I$, multiplying on the left by E^T isolates the multipliers

$$E^T(M\ddot{q} + c + g - u) = \lambda$$

- since $AF = 0$, multiplying on the left by F^T eliminates the multipliers

$$F^T M \ddot{q} = F^T(u - c - g)$$

- substituting in the latter the generalized accelerations and velocities with the pseudo-accelerations and pseudo-velocities leads finally to

invertible
 $(n-m) \times (n-m)$ positive definite matrix \rightarrow $(F^T M F) \dot{v} = F^T(u - c - g + M(E\dot{A} + F\dot{D})Fv)$

which is the reduced $(n - m)$ -dimensional dynamic model

- similarly, the expression of the multipliers becomes

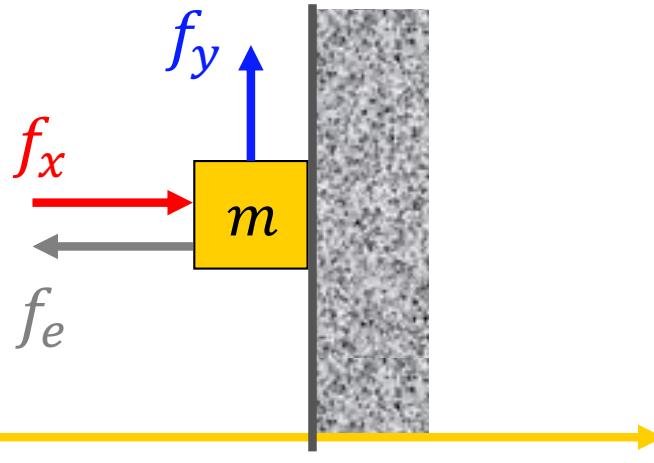
$$\lambda = E^T(MF\dot{v} - M(E\dot{A} + F\dot{D})Fv + c + g - u) \quad (\S)$$



Example – ideal mass reduced robot dynamics

$$q = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$u = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$



$$M \ddot{q} = u$$

robot dynamics
in free motion

$$M = \begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix}$$

$$h(q) = x - c = 0 \Rightarrow A = \begin{pmatrix} 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} A \\ D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} E & F \end{pmatrix}$$

➡ $v = D\dot{q} = \dot{y}$ pseudo-velocity

$$\lambda = E^T(MFv - u)$$

$$= (1 \ 0) \left(\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \ddot{y} - \begin{pmatrix} f_x \\ f_y \end{pmatrix} \right) = -(1 \ 0) \begin{pmatrix} f_x \\ f_y \end{pmatrix} = -f_x$$

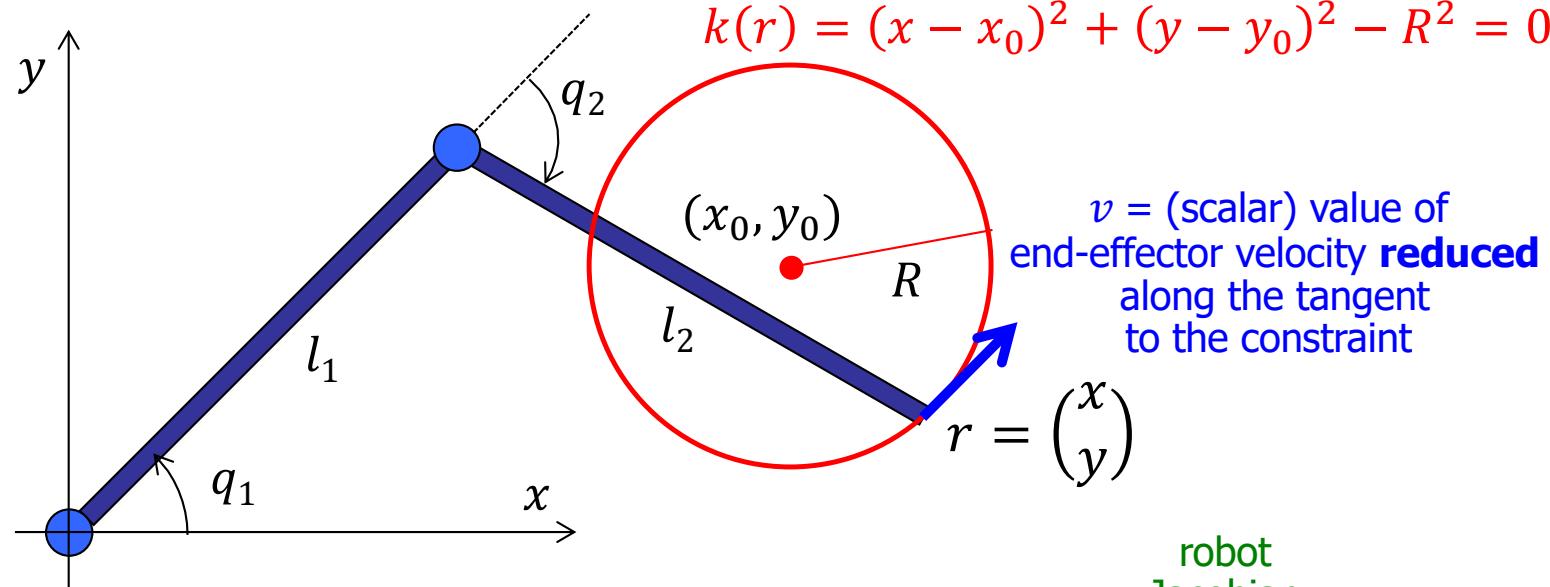
$$(F^T M F)v = (0 \ 1) \begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} v = m \ddot{y} = f_y = F^T u$$

multiplier
(contact
force f_e)

reduced
robot dynamics



Example – planar 2R robot reduced robot dynamics



$$\begin{aligned} A(q) &= [2(x - x_0) \quad 2(y - y_0)] J_r(q) \\ &= [2(l_1 c_1 + l_2 c_{12} - x_0) \quad 2(l_1 s_1 + l_2 s_{12} - y_0)] J_r(q) \end{aligned}$$

a feasible selection of matrix $D(q)$

$$D(q) = \left[-\frac{1}{2}(y - y_0) \quad \frac{1}{2}(x - x_0) \right] J_r(q) \quad \rightarrow \quad \det \begin{pmatrix} A(q) \\ D(q) \end{pmatrix} = R^2 \cdot \det J_r(q) \neq 0$$

$$\begin{pmatrix} A(q) \\ D(q) \end{pmatrix}^{-1} = (E(q) \quad F(q)) \quad \rightarrow \quad \boxed{v} = D(q)\dot{q} \quad \rightarrow \quad \dot{q} = F(q)v = J_r^{-1}(q) \begin{pmatrix} -2(y - y_0)/R^2 \\ 2(x - x_0)/R^2 \end{pmatrix} v$$

a scalar



Control based on reduced robot dynamics

- the reduced $n - m$ dynamic expressions are more compact but also more complex and less used for simulation purposes than the n -dimensional constrained dynamics
- however, they are useful for **control design** (reduced **inverse** dynamics)
- in fact, it is straightforward to verify that the **feedback linearizing** control law

$$u = (c + g - M(E\dot{A} + F\dot{D})F\nu) + MFu_1 - A^T u_2$$

applied to the **reduced robot dynamics** and to the **expression (§) of the multipliers** leads to the closed-loop system

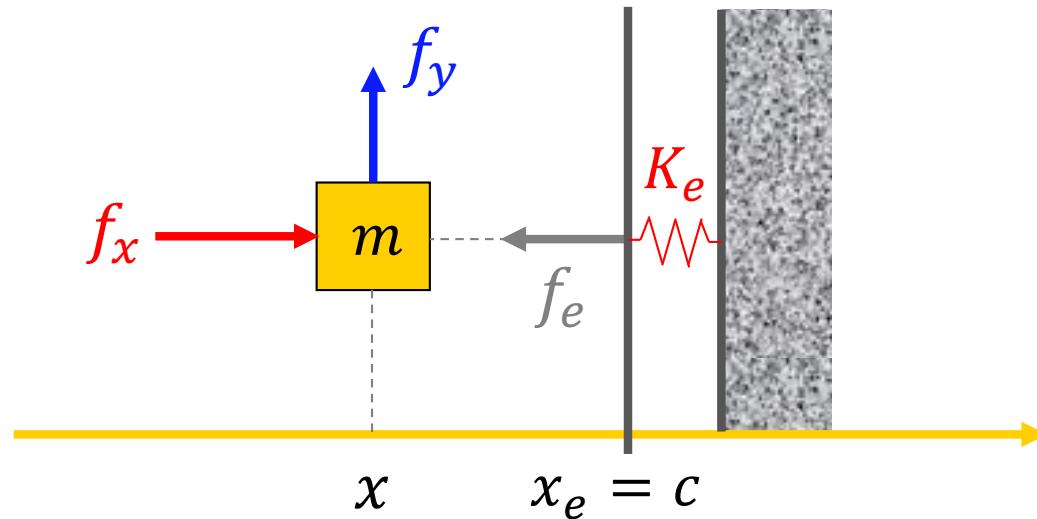
$$\dot{\nu} = u_1 \quad \lambda = u_2$$

Note: these are **exactly** in the form of the ideal mass example of slide #24, with $\nu = \dot{y}$, $u_1 = f_y/m$, $\lambda = f_e$, $u_2 = -f_x$ (being $n = 2$, $m = 1$, $n - m = 1$)



Compliant contact situation

a second possible modeling choice for softer environments



compliance/impedance control (in all directions) is here a good choice that allows to handle

- uncertain position
- uncertain orientation of the wall

$$\begin{cases} m\ddot{x} = f_x + f_e \\ m\ddot{y} = f_y \end{cases} \quad \begin{cases} x < c & \rightarrow f_e = 0 \\ x \geq c & \rightarrow f_e = K_e(x - x_e) \end{cases}$$

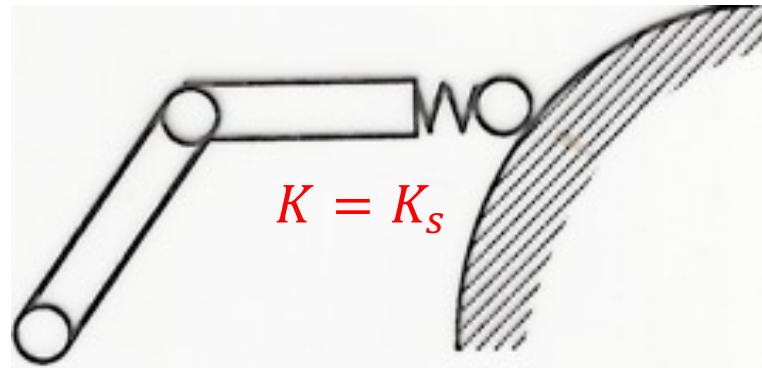
with $K_e > 0$ being the **stiffness** of the environment



Robot-environment contact types

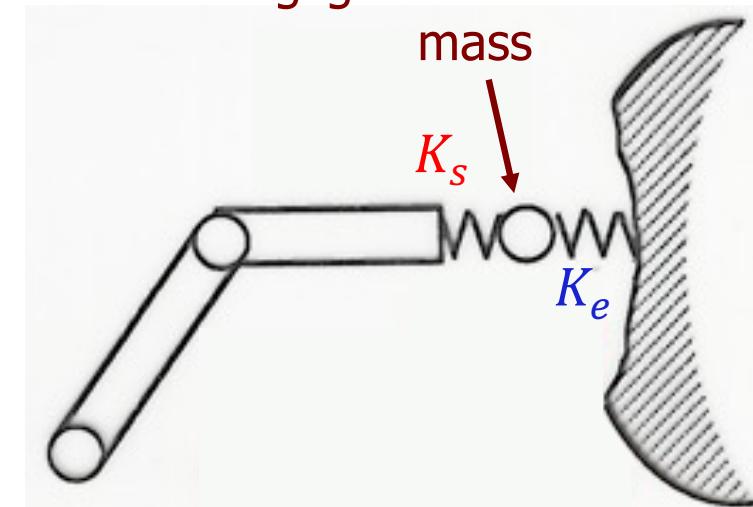
modeled by a single elastic constant

compliant
force sensor



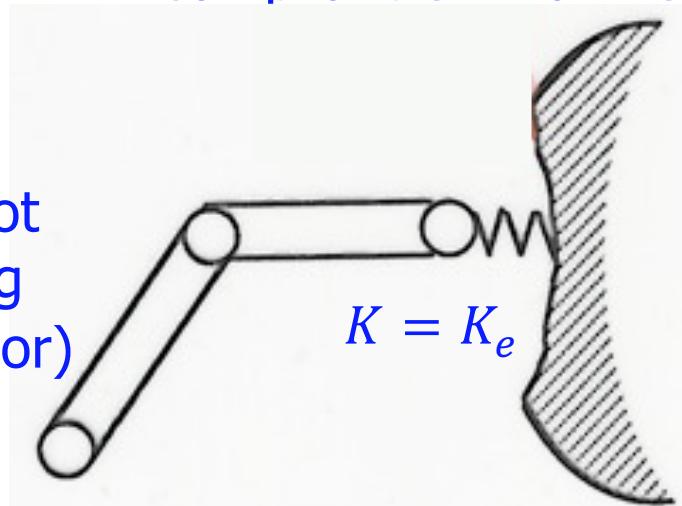
rigid environment

negligible intermediate
mass



compliant environment

rigid robot
(including
force sensor)



$$\frac{1}{K} = \frac{1}{K_s} + \frac{1}{K_e} \rightarrow K = \frac{K_s K_e}{K_s + K_e}$$

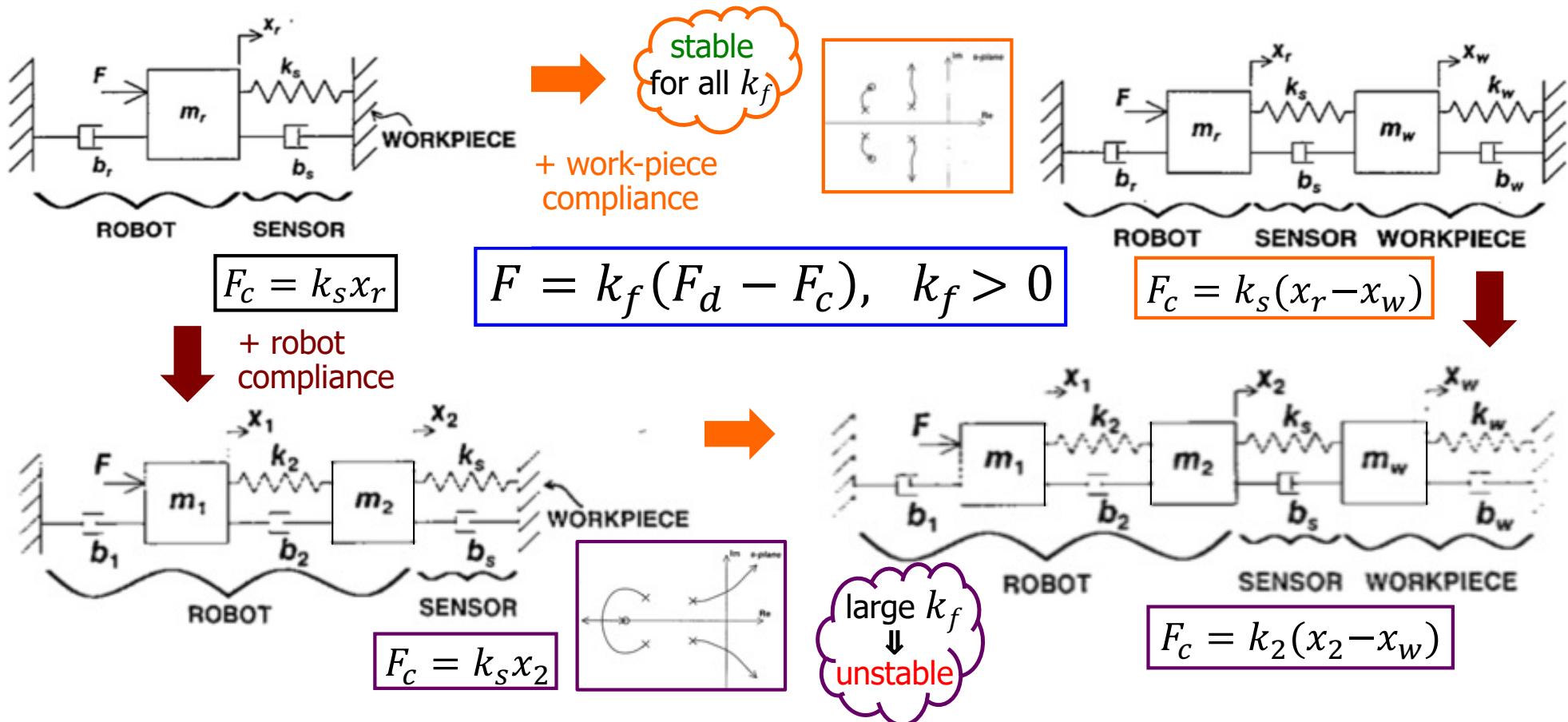
series of springs =
sum of compliances
(inverse of stiffnesses)



Force control

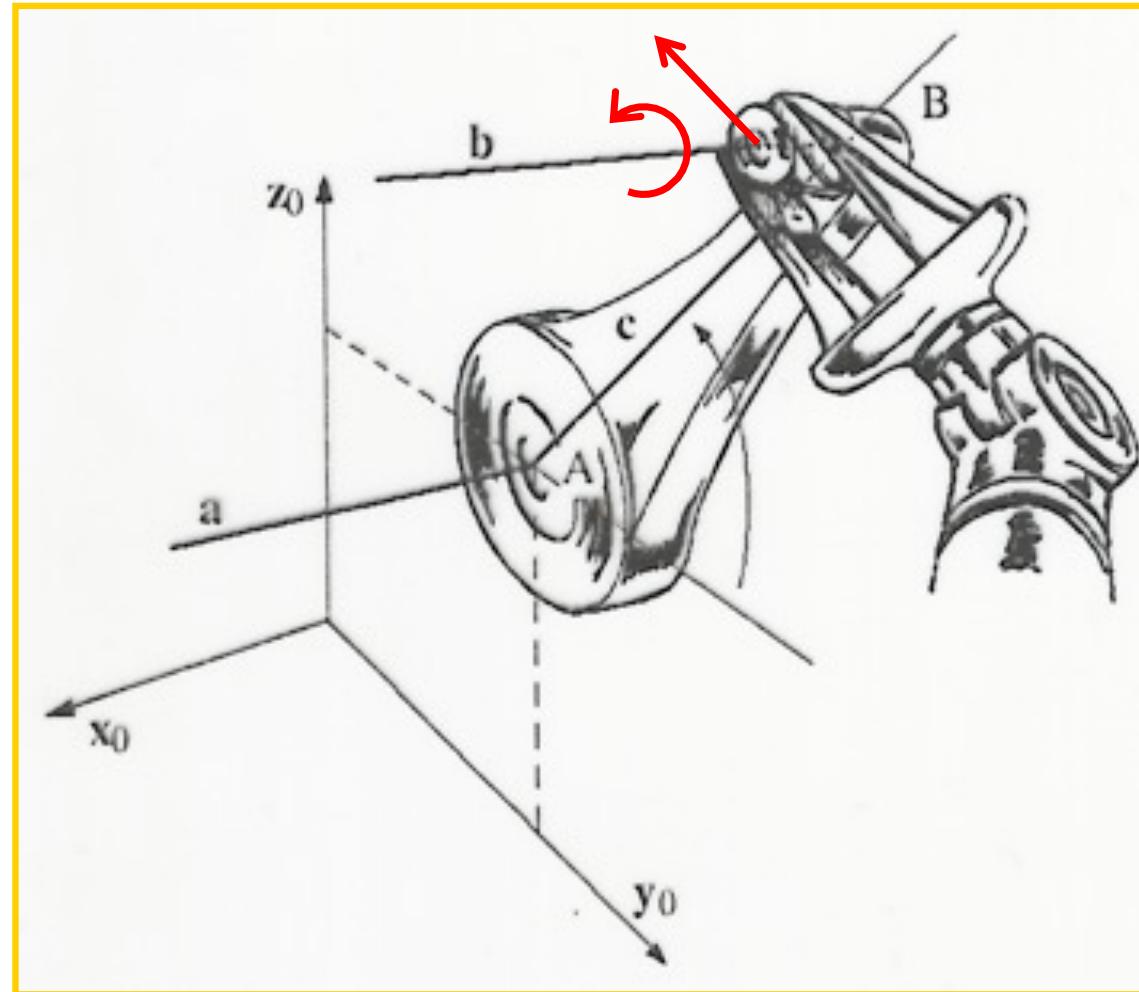
1-dof robot-environment linear dynamic models

- with a **force sensor** (having stiffness k_s and damping b_s) measuring the contact force F_c
- stability** analysis of a **proportional** control loop for regulation of the contact force (to a desired constant value F_d) can be made using the **root-locus method** (for a varying k_f)
- by including/excluding **work-piece compliance** and/or robot (transmission) compliance





Tasks requiring hybrid control



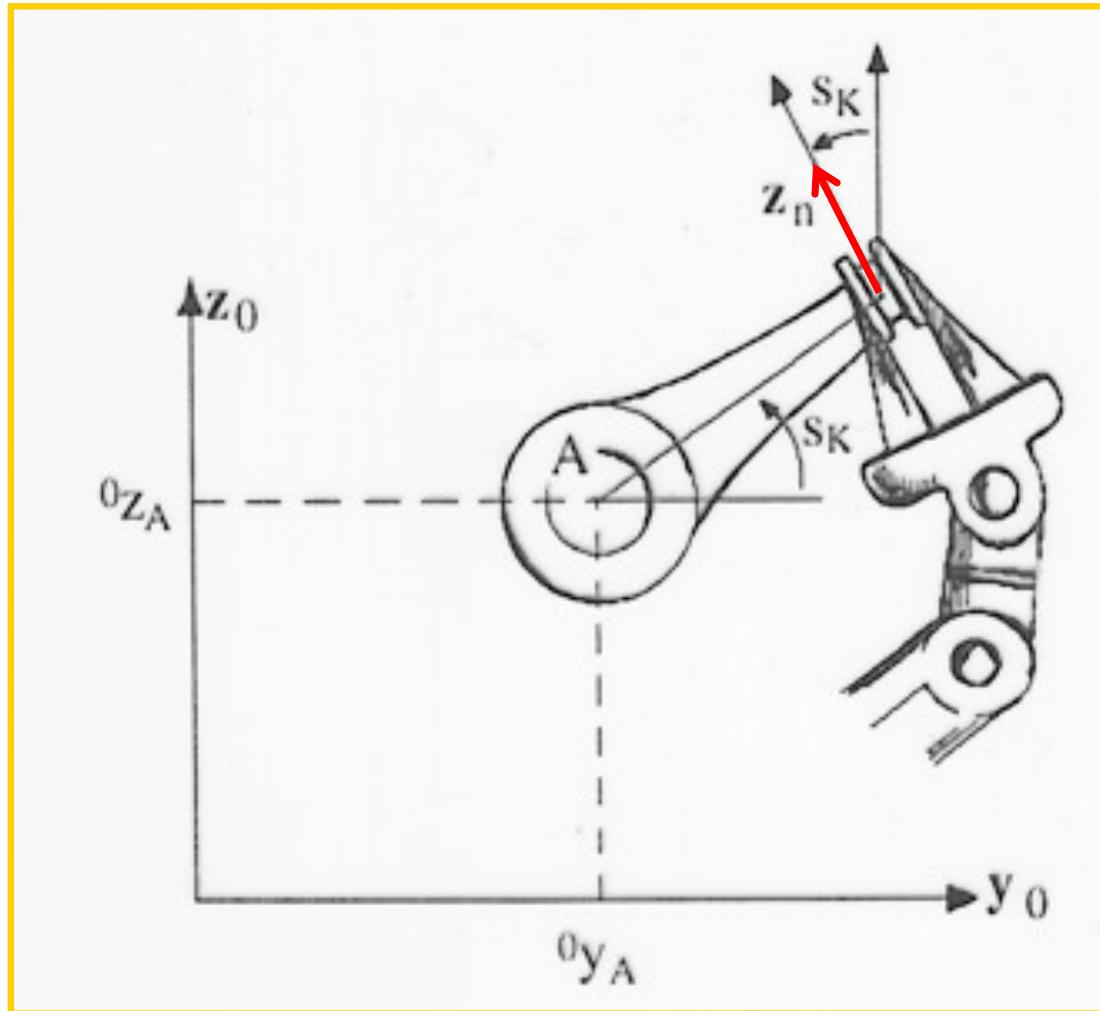
two generalized directions of instantaneous free motion at the contact:
tangential velocity & angular velocity around handle axis

↑
four directions of generalized reaction forces at the contact

the robot should turn a crank having a **free-spinning** handle



Tasks requiring hybrid control



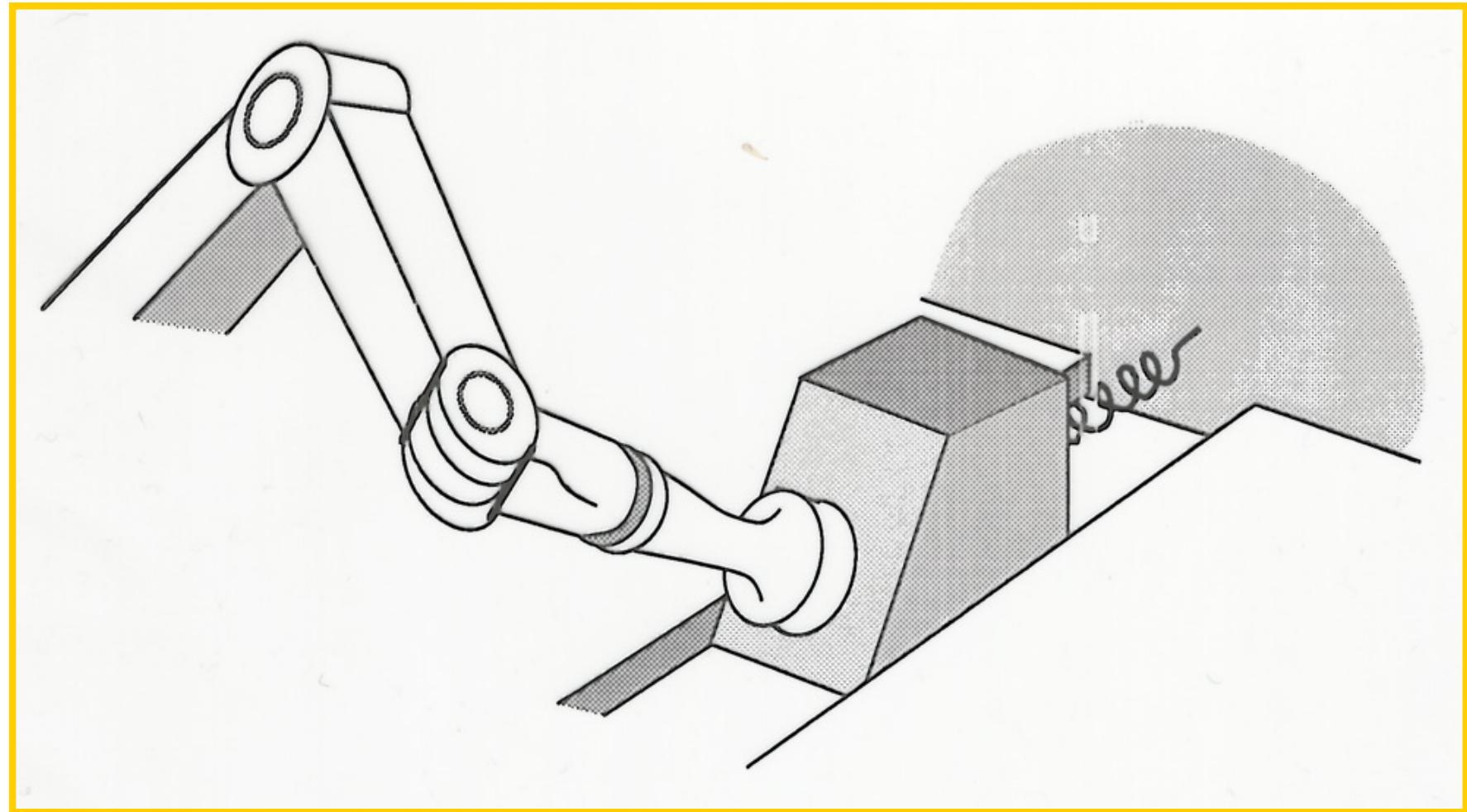
one direction **only**
of instantaneous
free motion
at the contact:
tangential velocity

↑
five directions
of generalized
reaction forces
at the contact

the robot should turn a crank
having a **fixed handle**



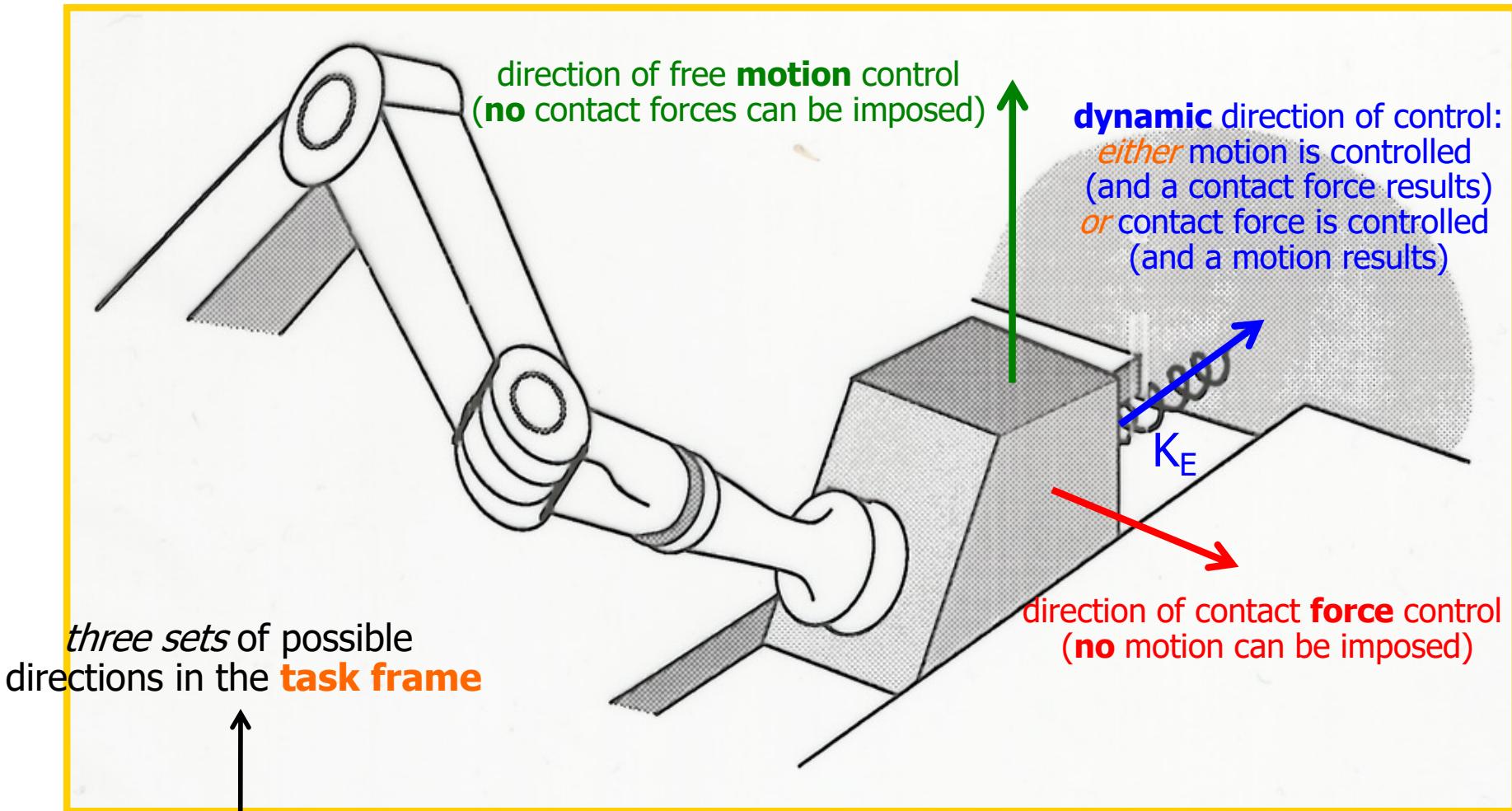
Tasks requiring hybrid control



the robot should push a mass
elastically coupled to a wall and constrained in a guide



Tasks requiring hybrid control



generalized hybrid modeling and control for dynamic environments

A. De Luca, C. Manes: IEEE Trans. Robotics and Automation, vol. 10, no. 4, 1994



Robotics 2

Impedance Control

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Impedance control

- imposes a desired **dynamic behavior** to the interaction between robot end-effector and environment
- the desired performance is specified through a **generalized dynamic impedance**, namely a complete set of **mass-spring-damper** equations (typically chosen as linear and decoupled, but also nonlinear)
- a model describing how reaction forces are generated in association with environment deformation is **not** explicitly required
- suited for tasks in which **contact forces** should be “kept small”, while their accurate regulation is not mandatory
- since a control loop based on **force error** is missing, **contact forces** are only indirectly assigned **by controlling position**
- the choice of a specific stiffness in the impedance model along a Cartesian direction results in a **trade-off** between contact forces and position accuracy in that direction

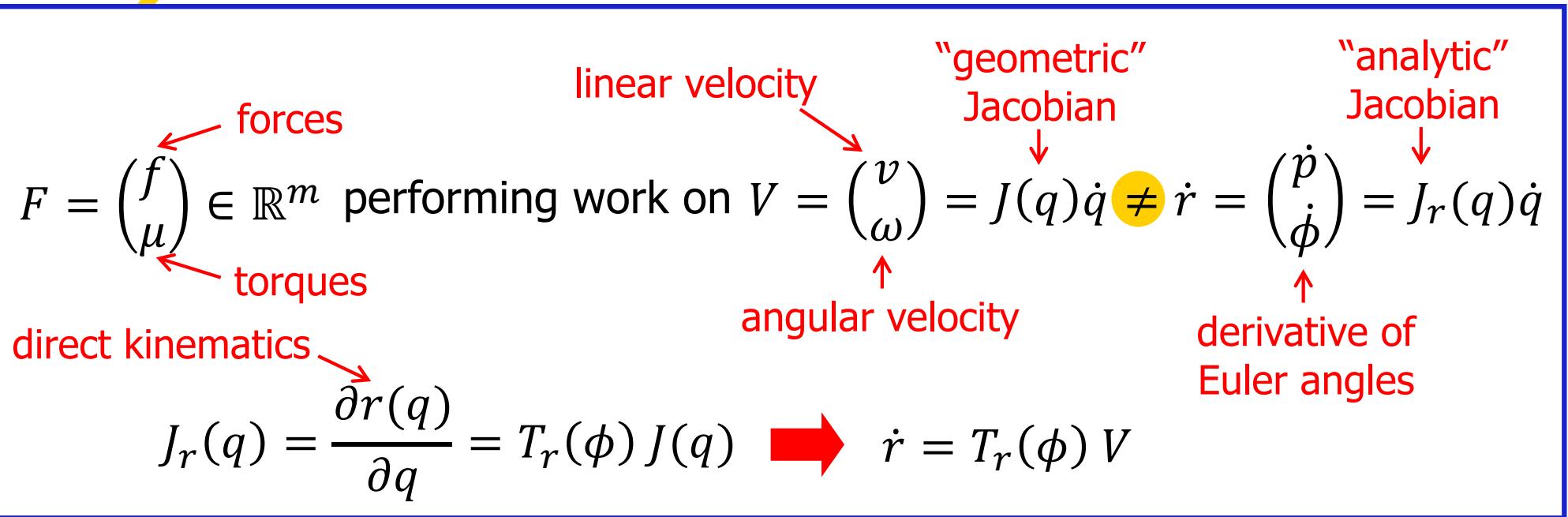


Dynamic model of a robot in contact

$$q \in \mathbb{R}^n$$

$$M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) = u + J^T(q)F$$

generalized
Cartesian force



$$M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) = u + J_r^T(q)F_r$$

with

$$F_r = T_r^{-T}(\phi) F$$

generalized forces performing work on \dot{r}

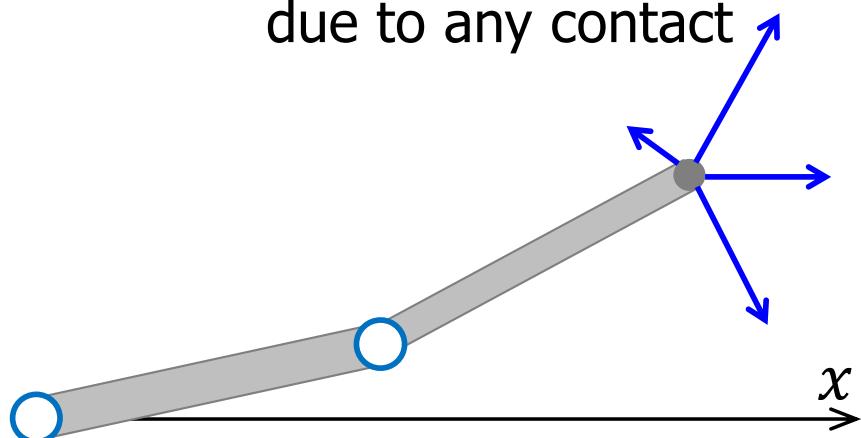


Contact forces vs. constraint forces

whiteboard...

$$M(q)\ddot{q} + \dots = u + J^T(q)F$$

every possible force F
due to any contact



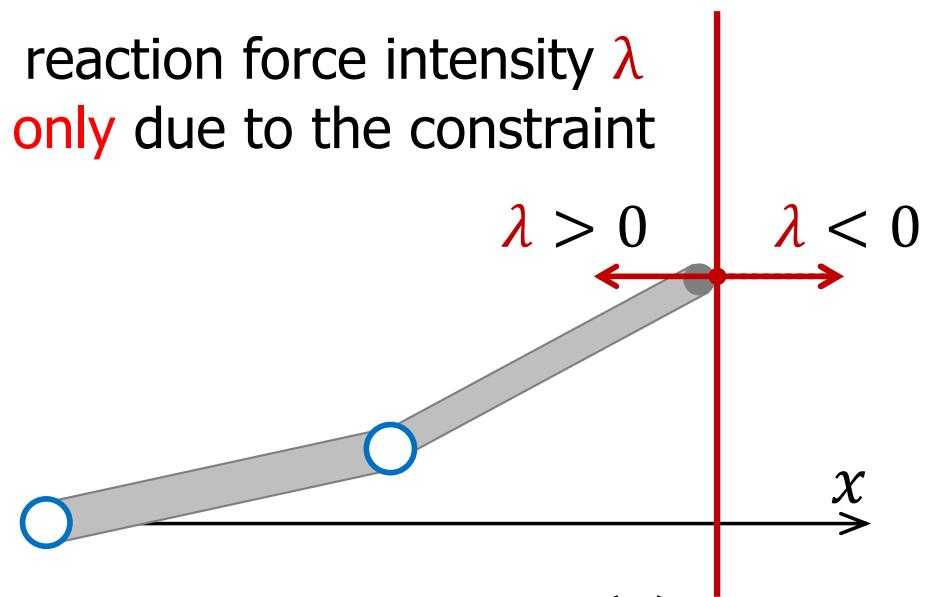
$$p = \begin{pmatrix} p_x(q) \\ p_y(q) \end{pmatrix} = r(q)$$

$$\dot{p} = \frac{\partial r(q)}{\partial q} \dot{q} = J(q) \dot{q}$$

$$\Rightarrow F = \begin{pmatrix} F_x \\ F_y \end{pmatrix}$$

$$M(q)\ddot{q} + \dots = u + A^T(q)\lambda$$

reaction force intensity λ
only due to the constraint



$$p_x(q) = k$$

$$\dot{p}_x = (1 \quad 0) J(q) \dot{q} = A(q) \dot{q} = 0$$

$$A^T(q)\lambda = J^T(q) \begin{pmatrix} 1 \\ 0 \end{pmatrix} \lambda = J^T(q) \begin{pmatrix} -F_x \\ 0 \end{pmatrix}$$

$$\Rightarrow F_x = -\lambda$$



Dynamic model in Cartesian coordinates

assuming
 $n = m$

$$M_r(q)\ddot{r} + S_r(q, \dot{q})\dot{r} + g_r(q) = J_r^{-T}(q)u + F_r$$

with

$$M_r(q) = J_r^{-T}(q)M(q)J_r^{-1}(q) = (J_r(q)M^{-1}(q)J_r^T(q))^{-1}$$

$$S_r(q, \dot{q}) = J_r^{-T}(q)S(q, \dot{q})J_r^{-1}(q) - M_r(q)\dot{J}_r(q)J_r^{-1}(q)$$

$$g_r(q) = J_r^{-T}(q)g(q)$$

... and the usual structural properties

- $M_r > 0$, if J_r is non-singular
- $\dot{M}_r - 2S_r$ is skew-symmetric, if $\dot{M} - 2S$ satisfies the same property
- the Cartesian dynamic model of the robot can be linearly parameterized in terms of a set of dynamic coefficients



Design of the control law

designed in two steps:

1. feedback linearization in the Cartesian space (with force measure)

$$u = J_r^T(q)[M_r(q)a + S_r(q, \dot{q})\dot{r} + g_r(q) - F_r]$$

$$\rightarrow \ddot{r} = a \quad \text{closed-loop system}$$

2. imposition of a dynamic impedance model

most of the times
it is "decoupled"
(diagonal matrices)

$$M_m(\ddot{r} - \ddot{r}_d) + D_m(\dot{r} - \dot{r}_d) + K_m(r - r_d) = F_r$$

↑ ↑ ↑ ↑
desired (apparent) desired desired external forces
inertia (> 0) damping (≥ 0) stiffness (> 0) from the environment

is realized by choosing

$$a = \ddot{r}_d + M_m^{-1}[D_m(\dot{r}_d - \dot{r}) + K_m(r_d - r) + F_r]$$

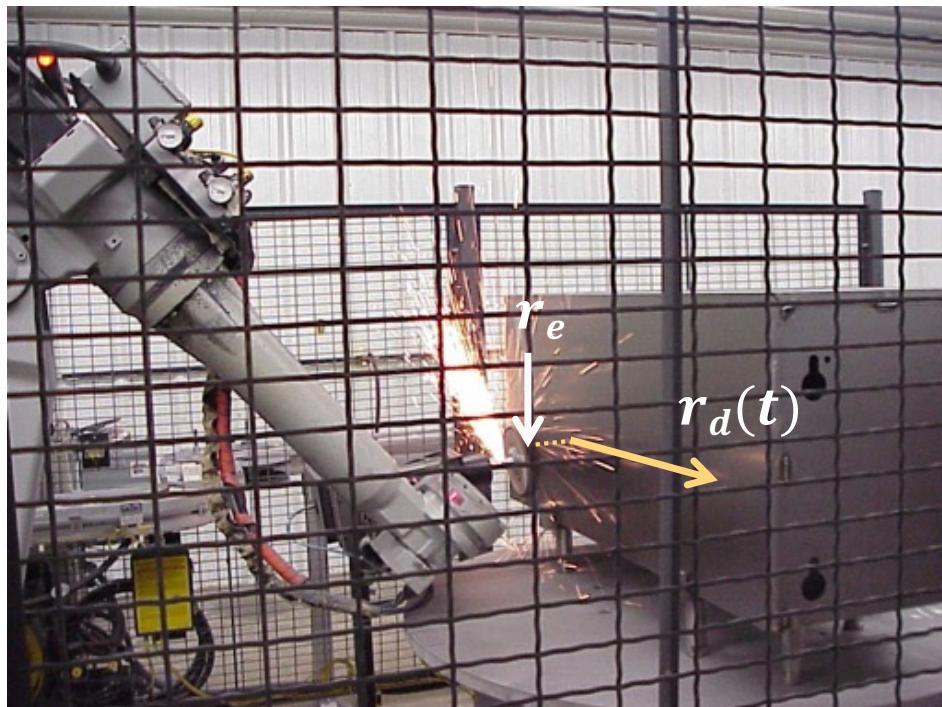
Note: $r_d(t)$ is the desired motion, which typically "slightly penetrates" inside the compliant environment (inducing contact forces)...



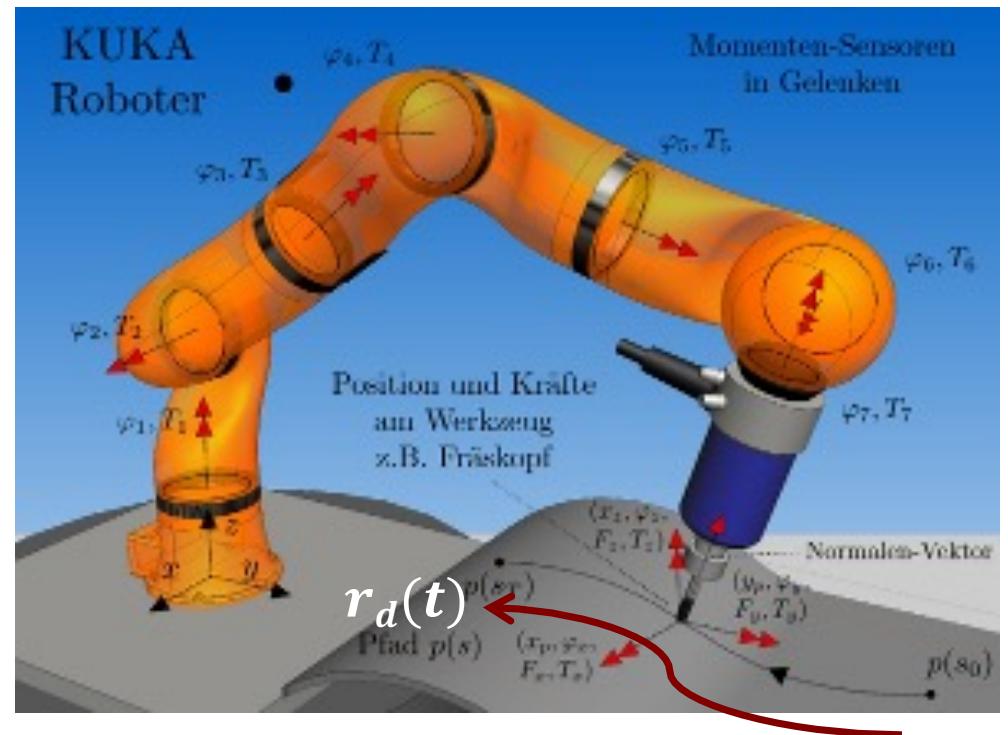
Examples of desired reference r_d in impedance/compliance control

$$M_m(\ddot{r} - \ddot{r}_d) + D_m(\dot{r} - \dot{r}_d) + K_m(r - r_d) = F_r$$

the desired motion $r_d(t)$ is slightly inside
the environment (keeping thus contact)



robot in grinding task



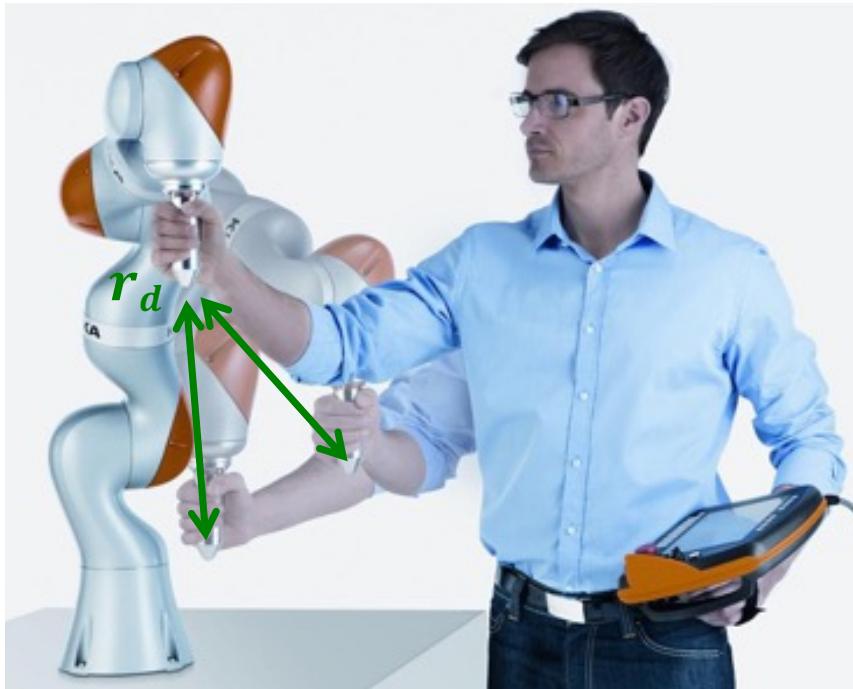
robot writing on a surface



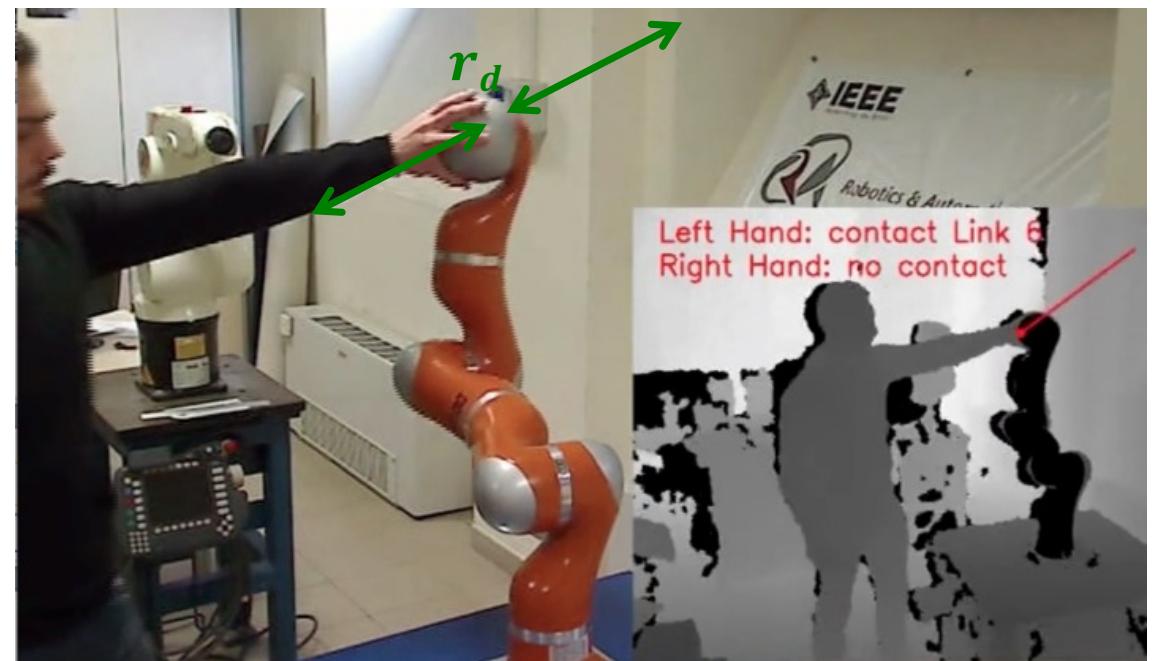
Examples of desired reference x_d in impedance/compliance control

$$M_m(\ddot{r} - \cancel{\dot{r}_d}) + D_m(\dot{r} - \cancel{\dot{r}_d}) + K_m(r - r_d) = F_r$$

constant desired pose r_d is the free Cartesian rest position in a human-robot interaction task



KUKA iiwa robot with human operator



KUKA LWR robot in pHRI (collaboration)



Control law in joint coordinates

substituting and simplifying...

$$u = M(q)J_r^{-1}(q)\{\ddot{r}_d - \dot{J}_r(q)\dot{q} + M_m^{-1}[D_m(\dot{r}_d - \dot{r}) + K_m(r_d - r)]\} \\ + S(q, \dot{q})\dot{q} + g(q) + J_r^T(q)[M_r(q)M_m^{-1} - I]F_r$$

matrix weighting the measured contact forces

- the following identity holds for the term involving contact forces

$$J_r^T(q)[M_r(q)M_m^{-1} - I]F_r = [M(q)J_r^{-1}(q)M_m^{-1} - J_r^T(q)]F_r$$

which eliminates from the control law also the appearance of the last remaining Cartesian quantity (the Cartesian inertia matrix)

- while the control design is based on dynamic analysis and desired (impedance) behavior described in the Cartesian space, the final control implementation is always at the robot joint level



Choice of the impedance model

rationale ...

- adapt/match to the dynamic characteristics of the environment (in particular, of its estimated stiffness) in a complementary way
- avoid large impact forces due to uncertain geometric characteristics (position, orientation) of the environment
- mimic the behavior of a human arm
 - ➔ fast and stiff in "free" motion, slow and compliant in "guarded" motion

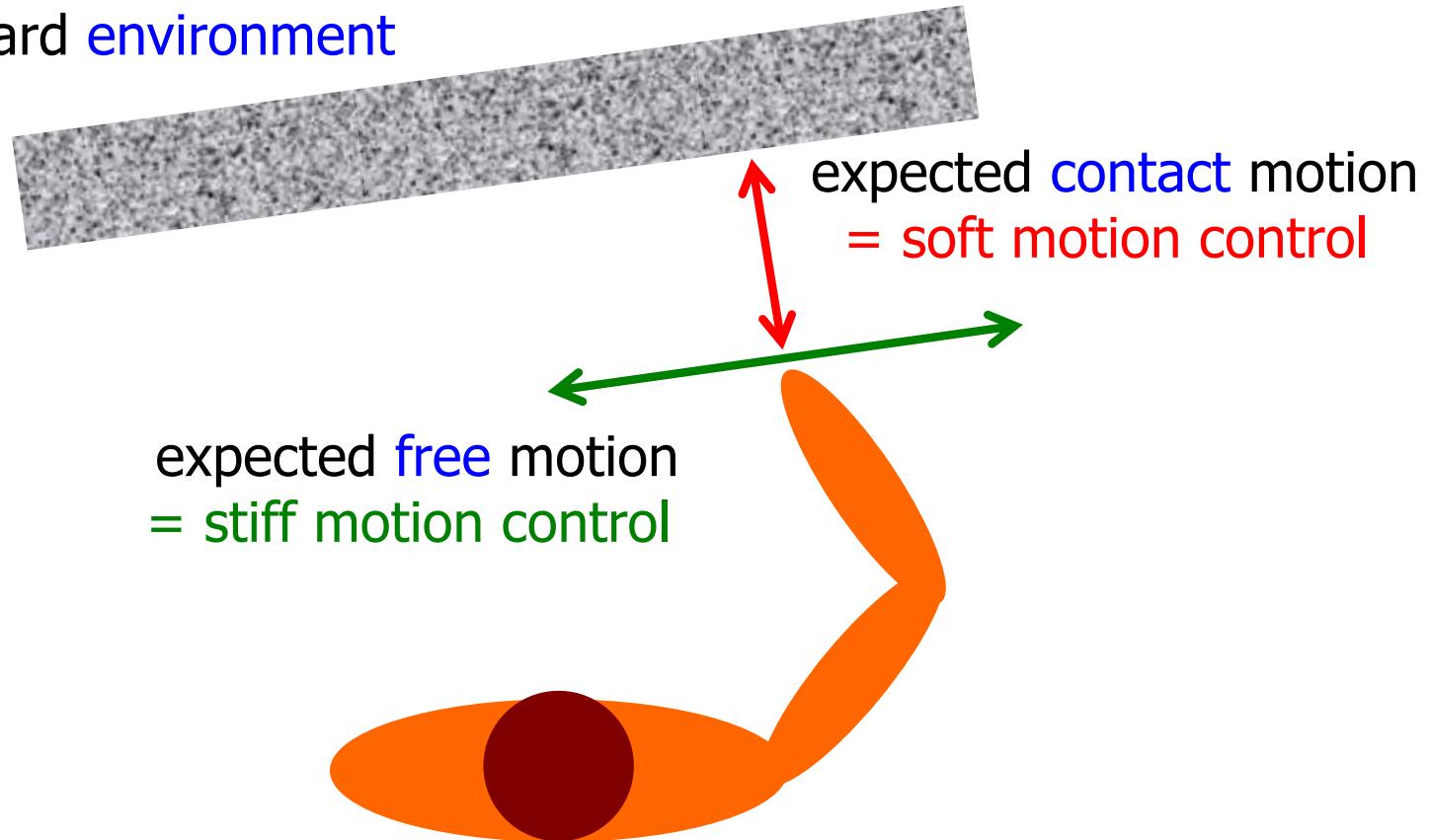


- large $M_{m,i}$ and small $K_{m,i}$ in Cartesian directions where contact is foreseen (➔ low contact forces)
- large $K_{m,i}$ and small $M_{m,i}$ in Cartesian directions that are supposed to be free (➔ good tracking of desired motion trajectory)
- damping coefficients $D_{m,i}$ are used then to shape transient behaviors



Human arm behavior

hard environment



in the selected i -th Cartesian direction:

the **stiffer** is the environment, the **softer** is the chosen model stiffness $K_{m,i}$

Experiments with impedance control human interaction with a Panda robot



7R Franka Emika Panda robot



high rotational $K_{m,\phi}$
low $K_{m,p}$
(compliant in position)



high translational $K_{m,p}$
low $K_{m,\phi}$
(compliant in orientation)

trajectory tracking with
physical interaction
(uniformly compliant)



LRS – RPTU
Kaiserslautern



A notable simplification - 1

choose the apparent inertia **equal to** the natural Cartesian inertia of the robot

$$M_m = M_r(q) = J_r^{-T}(q)M(q)J_r^{-1}(q) = (J_r(q) M^{-1}(q) J_r^T(q))^{-1}$$

then, the control law becomes

$$\begin{aligned} u = & M(q)J_r^{-1}(q)\{\ddot{r}_d - \dot{J}_r(q)\dot{q}\} + S(q, \dot{q})\dot{q} + g(q) \\ & + J_r^T(q)[D_m(\dot{r}_d - \dot{r}) + K_m(r_d - r)] \end{aligned}$$

WITHOUT contact force feedback! (a F/T sensor is no longer needed...)



this is a **pure motion control** law applied also during interaction,
but designed so as to keep **limited contact forces** at the end-effector level
(as before, K_m is chosen as a function of the **expected** environment stiffness)



A notable simplification - 2

technical issue: if the impedance model (now, nonlinear) is still supposed to represent a real mechanical system, then in correspondence to a desired non-constant inertia ($M_r(q)$) there should be Coriolis and centrifugal terms...



$$M_r(q)(\ddot{r} - \ddot{r}_d) + (S_r(q, \dot{q}) + D_m)(\dot{r} - \dot{r}_d) + K_m(r - r_d) = F_r$$

nonlinear impedance model ("only" gravity terms disappear)

redoing computations, the control law becomes

$$\begin{aligned} u = & M(q)J_r^{-1}(q)\{\ddot{r}_d - J_r(q)J_r^{-1}(q)\dot{r}_d\} + S(q, \dot{q})J_r^{-1}(q)\dot{r}_d + g(q) \\ & + J_r^T(q)[D_m(\dot{r}_d - \dot{r}) + K_m(r_d - r)] \end{aligned}$$

which is indeed slightly more complex, but has the following advantages:

- guarantee of asymptotic convergence to zero tracking error (on $r_d(t)$)
when $F_r = 0$ (no contact situation) \Rightarrow Lyapunov + skew-symmetry of $\dot{M}_r - 2S_r$
- further simplifications when r_d is constant



Cartesian regulation revisited

(without contact, $F_r = 0$)

when r_d is constant ($\dot{r}_d = 0, \ddot{r}_d = 0$), from the previous expression we get the control law

$$u = g(q) + J_r^T(q)[K_m(r_d - r) - D_m\dot{r}] \quad (\star)$$

Cartesian PD control with gravity cancellation...

when $F_r = 0$ (absence of contact), we know already that this control law ensures **asymptotic stability of r_d** , provided $J_r(q)$ has full rank

proof
(alternative)

Lyapunov candidate $V_1 = \frac{1}{2} \dot{r}^T M_r(q) \dot{r} + \frac{1}{2} (r_d - r)^T K_m (r_d - r)$

$$\dot{V}_1 = \dot{r}^T M_r(q) \ddot{r} + \frac{1}{2} \dot{r}^T \dot{M}_r(q) \dot{r} - \dot{r}^T K_m (r_d - r) = \dots = -\dot{r}^T D_m \dot{r} \leq 0$$

using skew-symmetry of $\dot{M}_r - 2S_r$ and $g_r = J_r^{-T} g$



Cartesian stiffness control (with contact, $F_r \neq 0$)

when $F_r \neq 0$, convergence to r_d is not assured
(it may not even be a closed-loop equilibrium...)

- for analysis, assume an elastic contact model for the environment
$$F_r = K_e(r_e - r) \quad \text{with stiffness } K_e \geq 0 \text{ and rest position } r_e$$
- closed-loop system behavior

Lyapunov candidate

$$V_2 = \frac{1}{2} \dot{r}^T M_r(q) \dot{r} + \frac{1}{2} (r_d - r)^T K_m (r_d - r) + \frac{1}{2} (r_e - r)^T K_e (r_e - r)$$

$$= V_1 + \frac{1}{2} (r_e - r)^T K_e (r_e - r)$$

$$\begin{aligned} \dot{V}_2 &= \dot{r}^T M_r(q) \ddot{r} + \frac{1}{2} \dot{r}^T \dot{M}_r(q) \dot{r} - \dot{r}^T K_m (r_d - r) - \dot{r}^T K_e (r_e - r) \\ &= \dots = -\dot{r}^T D_m \dot{r} + \dot{r}^T (F_r - K_e (r_e - r)) = -\dot{r}^T D_m \dot{r} \leq 0 \end{aligned}$$



Stability analysis (with $F_r \neq 0$)

when $\dot{r} = \ddot{r} = 0$, at a closed-loop system **equilibrium** it is

$$K_m(r_d - r) + K_e(r_e - r) = 0$$

which has the **unique** solution

$$r = (K_m + K_e)^{-1}(K_m r_d + K_e r_e) =: r_E$$

(check that the Lyapunov candidate V_2 has in fact its **minimum** in r_E !)

LaSalle \rightarrow r_E **asymptotically stable equilibrium**

$$r_E \approx \begin{cases} r_e & \text{for } K_e \gg K_m \text{ (rigid environment)} \\ r_d & \text{for } K_m \gg K_e \text{ (rigid controller)} \end{cases}$$



at **steady state**
the contact force is
 $F_r = K_e(r_e - r_E)$

Note: the Cartesian stiffness control law (\star) is often called **compliance control** in the literature

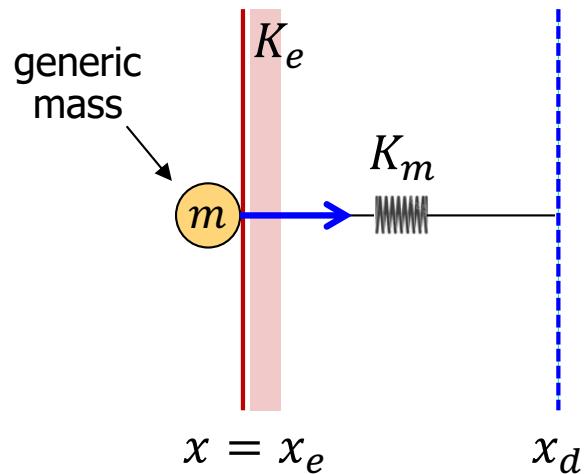


Equilibrium condition

whiteboard...

$$\mathbf{r}_E = (K_m + K_e)^{-1}(K_m r_d + K_e r_e)$$

let $r = x \in \mathbb{R}$

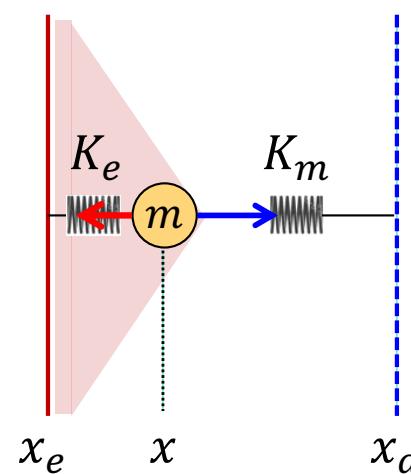


at the initial contact

$$m\ddot{x} = F_c - D_m \dot{x}$$

$$F_c = K_m(x_d - x_e) > 0$$

part of the Cartesian control force



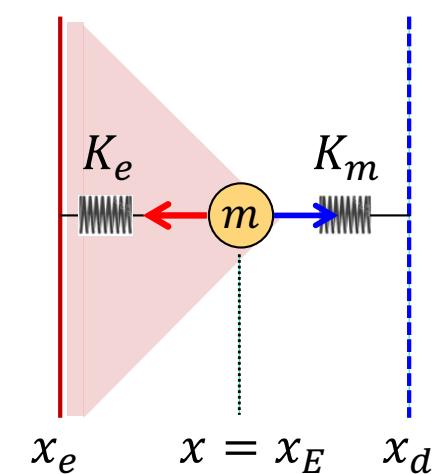
during the transient

$$m\ddot{x} = F_c + F_r - D_m \dot{x}$$

$$F_c = K_m(x_d - x) > 0$$

$$F_r = -K_e(x - x_e) < 0$$

$$|F_c| \neq |F_r|$$



at steady-state (equilibrium)

$$0 = F_c + F_r$$

$$F_c = K_m(x_d - x_E) > 0$$

$$F_r = -K_e(x_E - x_e) < 0$$

$$\rightarrow x_E = \frac{K_m x_d + K_e x_e}{K_m + K_e}$$



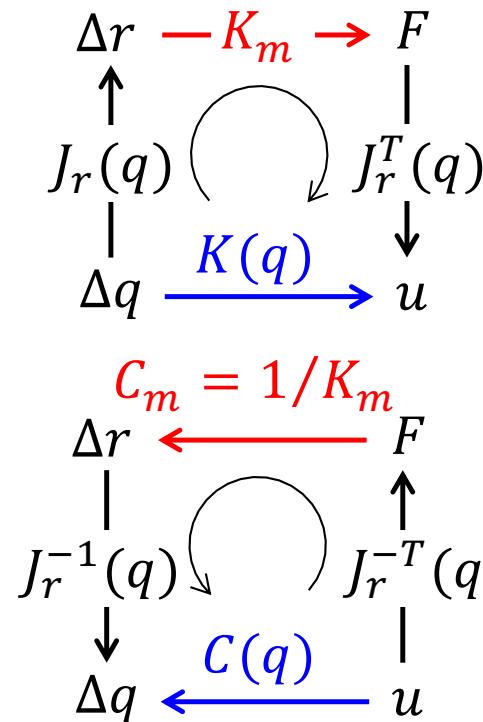
Active equivalent of RCC device

- IF**
- displacements from the desired position r_d are **small**, namely

$$(r_d - r) \approx J_r(q)(q_d - q)$$
 - $g(q) = 0$ (gravity compensated), $D_m = 0$ (or $\dot{r} \approx 0$, i.e., small enough)

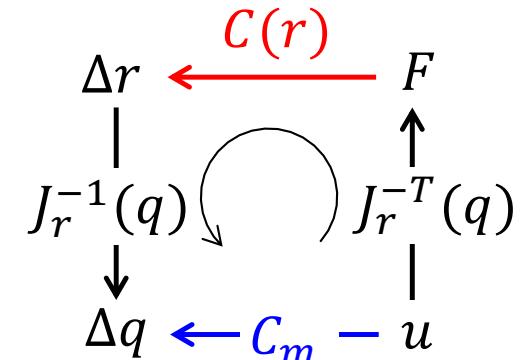
THEN

$$u = J_r^T(q)K_mJ_r(q)(q_d - q) = K(q)(q_d - q)$$



constant Cartesian-level stiffness K_m
(or compliance $C_m = 1/K_m$)
corresponds to
variable joint-level stiffness $K(q)$
(or compliance $= C(q)$)

... and vice versa



this is the “active” counterpart of a Remote Center of Compliance (RCC) device



Admittance control

- in some cases, we don't have access to low-level robot torque (or motor current) commands \Rightarrow **closed control architecture**
- for handling the interaction with the environment, one uses often **admittance control**: contact forces $F_c \Rightarrow$ velocity commands \dot{q}
- **implementation** (with compliant matrices C)
 - in **joint** space or in **Cartesian (task)** space – with **singularity** issues ...
 - at the **velocity** or **incremental position** level

$$u_c = J^T(q)F_c \rightarrow \dot{q} = C_q u_c \rightarrow \boxed{\dot{q} = C_q J^T(q)F_c} \quad C_q \geq 0$$

\updownarrow
 Δq (to be added to the current q)

$$F_c \rightarrow \dot{r} = C_r F_c \rightarrow \boxed{\dot{q} = J^{-1}(q)C_r F_c} \quad C_r \geq 0$$

\updownarrow
(in case of redundancy) $J^\#(q)$

Experiments with admittance control human interaction with a KUKA LWR robot



7R KUKA LWR4+ robot

handling of task singularities
through **performance constraints**

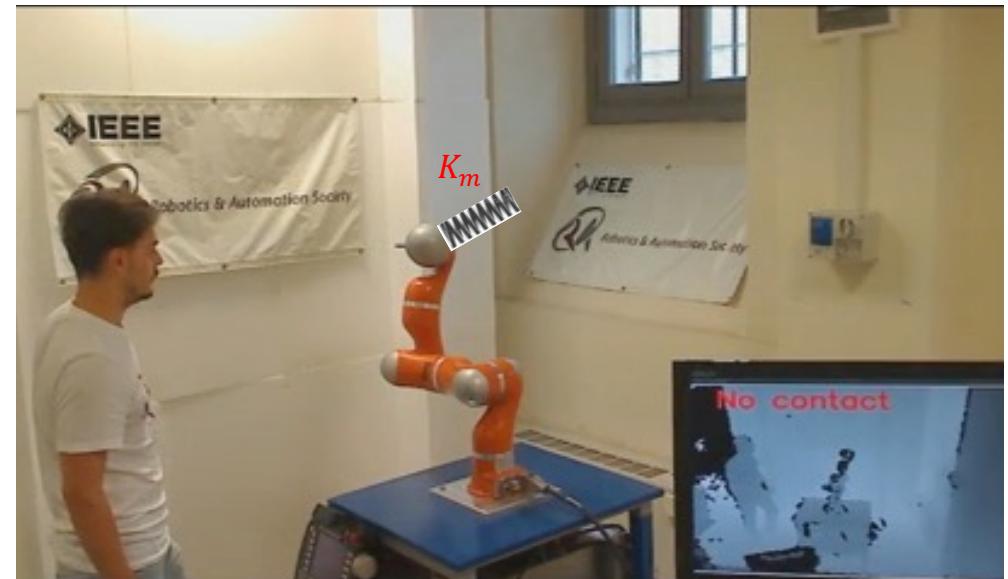
admittance control at **any contact point**
without using a force/torque sensor

video



ICRA 2016, University of Patras

video



Sep 2013, DIAG Laboratory of Robotics



Robotics 2

Hybrid Force/Motion Control

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





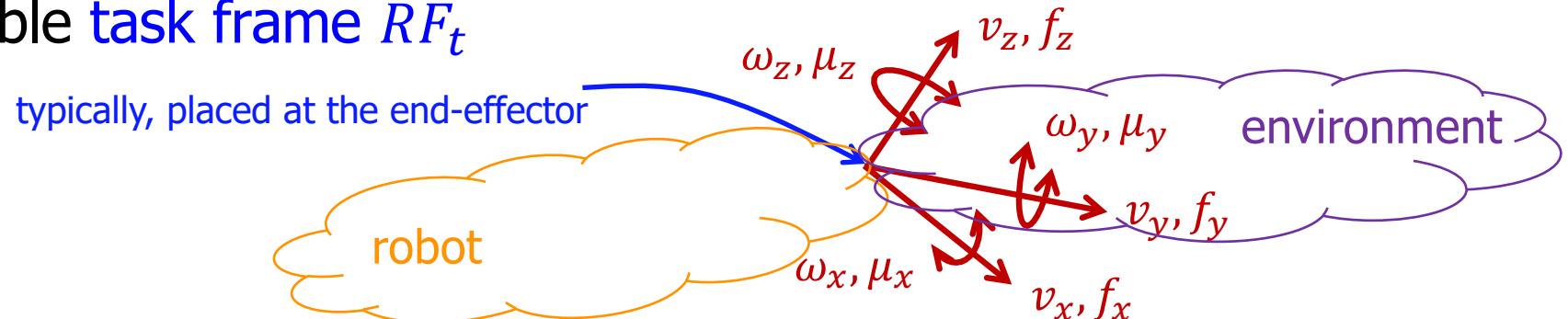
Hybrid force/motion control

- we consider **contacts/interactions** between a robot and a stiff environment that **naturally constrains** the end-effector motion
- **compared** to an approach using the constrained/reduced robot dynamics with (bilateral) **geometric constraints**, the **differences** are
 - the hybrid control law is designed in **ideal conditions**, but now unconstrained directions of motion and constrained force directions are defined in a more direct way using a **task frame formalism**
 - all **non-ideal conditions** (compliant surfaces, friction at the contact, errors in contact surface orientation) are handled explicitly in the control scheme by a **geometric filtering of the measured quantities**
 - considering only signal components that should appear in certain directions based on the nominal task model, and treating those that should not be there as **disturbances** to be rejected
- the hybrid control law avoids to introduce conflicting behaviors (force vs. motion control) in any of the task space directions!!



Natural constraints

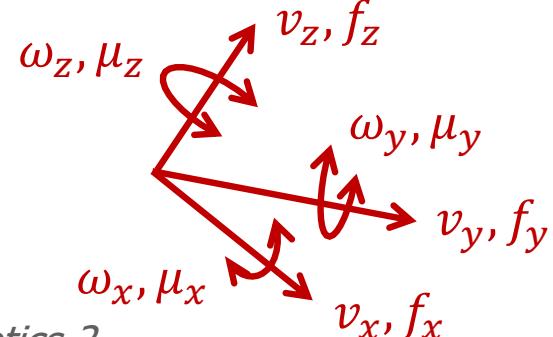
- in **ideal conditions** (robot and environment are perfectly rigid, contact is frictionless), **two sets of generalized directions** can be defined in the **task space**
 - end-effector motion (v/ω) is prohibited along/around **$6 - k$ directions** (since the environment reacts there with forces/torques)
 - reaction forces/torques (f/μ) are absent along/around **k directions** (where the environment does not prevent end-effector motions)
- these constraints have been called the **natural constraints** on motion and force associated to the task geometry
- the two sets of directions are characterized through the axes of a suitable **task frame RF_t**





Artificial constraints

- the way **task execution** should be performed can be expressed in terms of so-called **artificial constraints** that specify the desired values (to be imposed by the control law)
 - for the **end-effector velocities** (v/ω) along/around k directions where feasible motions can occur
 - for the **contact forces/torques** (f/μ) along/around $6 - k$ directions where admissible reactions of the environment can occur
- the two sets of directions are **complementary** (they cover the 6D generalized task space) and mutually **orthogonal**, while the **task frame** can be **time-varying** ("moves with task progress")
 - directions are intended as 6D **screws**: twists $V = (v^T \omega^T)^T$ and wrenches $F = (f^T \mu^T)^T$

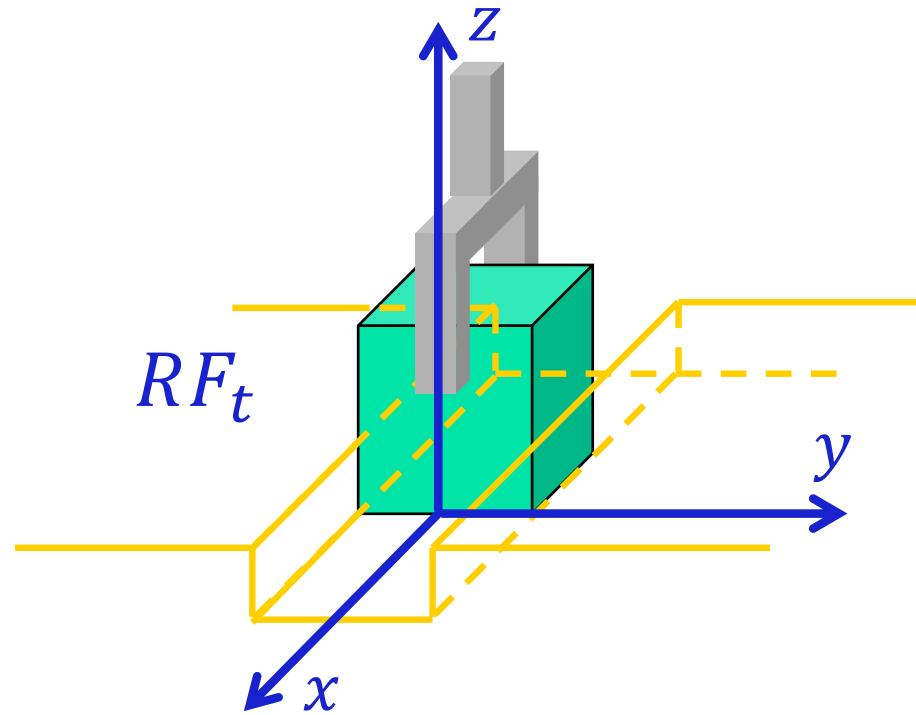


$$F^T V = 0 \Leftrightarrow \text{orthogonality}$$

but **ill-defined** (don't use it!) for $V_1^T V_2$ or $F_1^T F_2$



Task frame and constraints - example 1



v = linear velocity
 ω = angular velocity
 f = force
 μ = torque

task: slide the cube along a guide

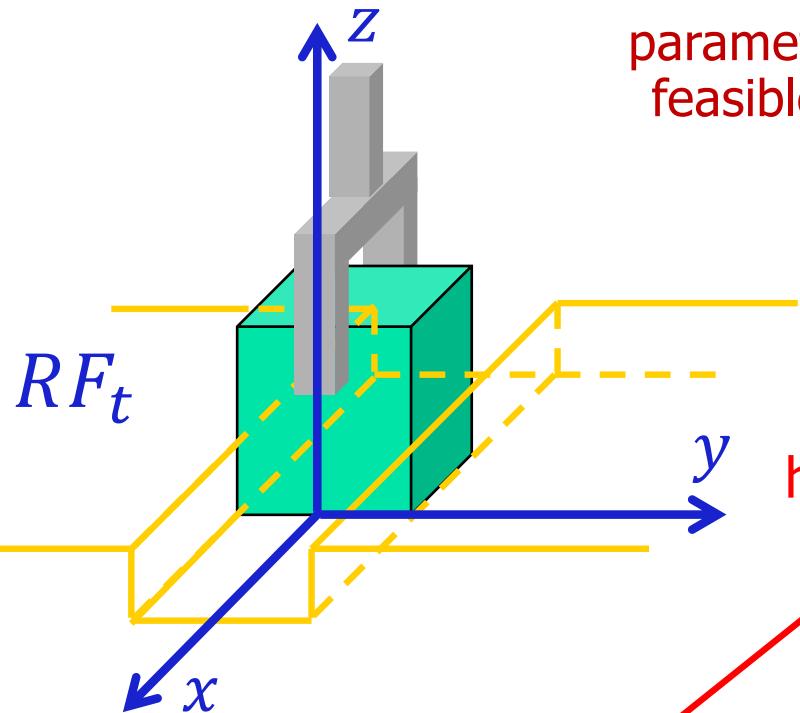
natural (geometric) constraints

$$\left. \begin{array}{l} v_y = v_z = 0 \\ \omega_x = \omega_z = 0 \\ f_x = \mu_y = 0 \end{array} \right\} \begin{array}{l} 6 - k = 4 \\ k = 2 \end{array}$$

$$6 - k = 4 \quad \left\{ \begin{array}{l} f_y = f_{y,des} (= 0) \text{ (to avoid internal stress)} \\ \mu_x = \mu_{x,des} (= 0), \mu_z = \mu_{z,des} (= 0) \\ f_z = f_{z,des} \text{ (to keep contact)} \\ \omega_y = \omega_{y,des} = 0 \text{ (to slide and not to roll !!)} \\ v_x = v_{x,des} \end{array} \right. \quad k = 2$$



Selection of directions - example 1



parametrization of feasible reactions

$$\begin{pmatrix} f \\ \mu \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_y \\ f_z \\ \mu_x \\ \mu_z \end{pmatrix} = Y \begin{pmatrix} f_y \\ f_z \\ \mu_x \\ \mu_z \end{pmatrix}$$

parametrization of feasible motions

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v_x \\ \omega_y \end{pmatrix} = T \begin{pmatrix} v_x \\ \omega_y \end{pmatrix}$$

here, constant and unitary
("selection" of columns from
the 6×6 identity matrix)

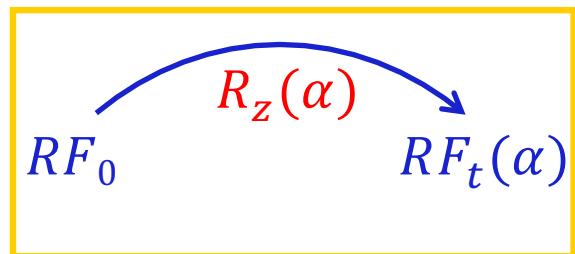
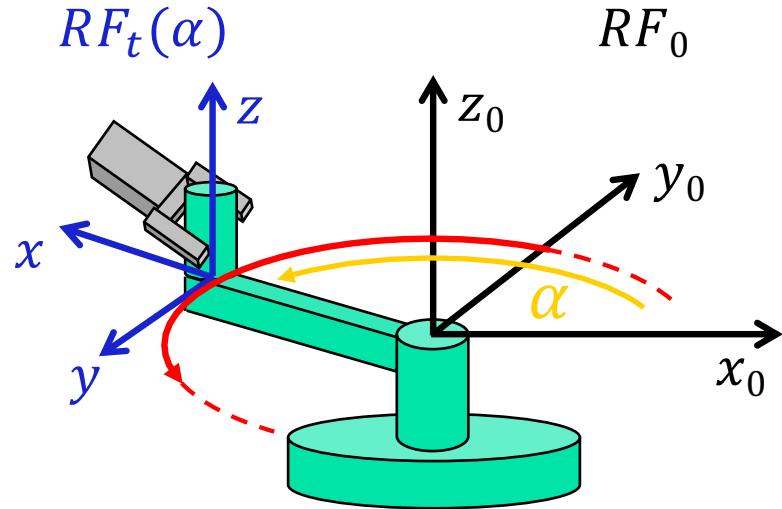
$$T^T Y = 0$$

reaction forces/torques
do **not** perform work on
feasible motions

$$(f^T \quad \mu^T) \begin{pmatrix} v \\ \omega \end{pmatrix} = 0$$



Task frame and constraints - example 2



task: turning a crank
(free handle)

natural constraints

$$v_x = v_z = 0$$

$$\omega_x = \omega_y = 0$$

$$f_y = \mu_z = 0$$

artificial constraints

$$f_x = f_{x,des} (= 0), f_z = f_{z,des} (= 0)$$

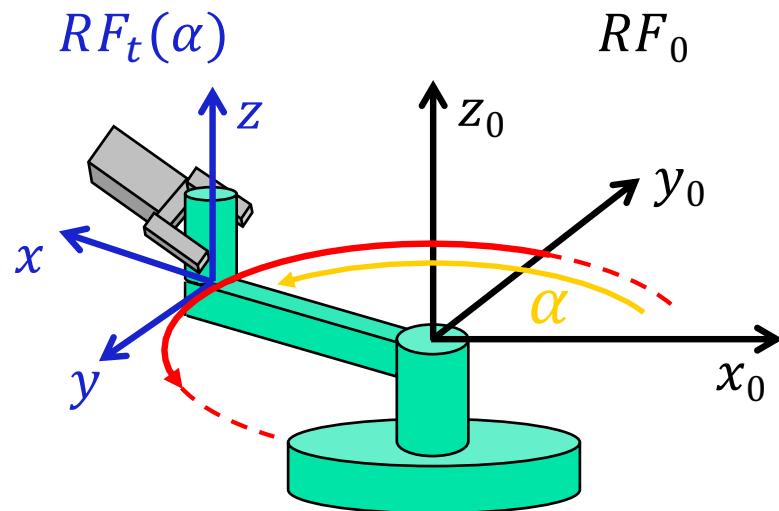
$$\mu_x = \mu_{x,des} (= 0), \mu_y = \mu_{y,des} (= 0)$$

$v_y = v_{y,des}$ (the tangent speed of rotation)

$\omega_z = \omega_{z,des}$ (= 0 if handle should not spin)



Selection of directions – example 2



parametrization of feasible motions

$$\begin{pmatrix} {}^0v \\ {}^0\omega \end{pmatrix} = \begin{pmatrix} R^T(\alpha) & 0 \\ 0 & R^T(\alpha) \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_y \\ \omega_z \end{pmatrix}$$

$$= T(\alpha) \begin{pmatrix} v_y \\ \omega_z \end{pmatrix}$$

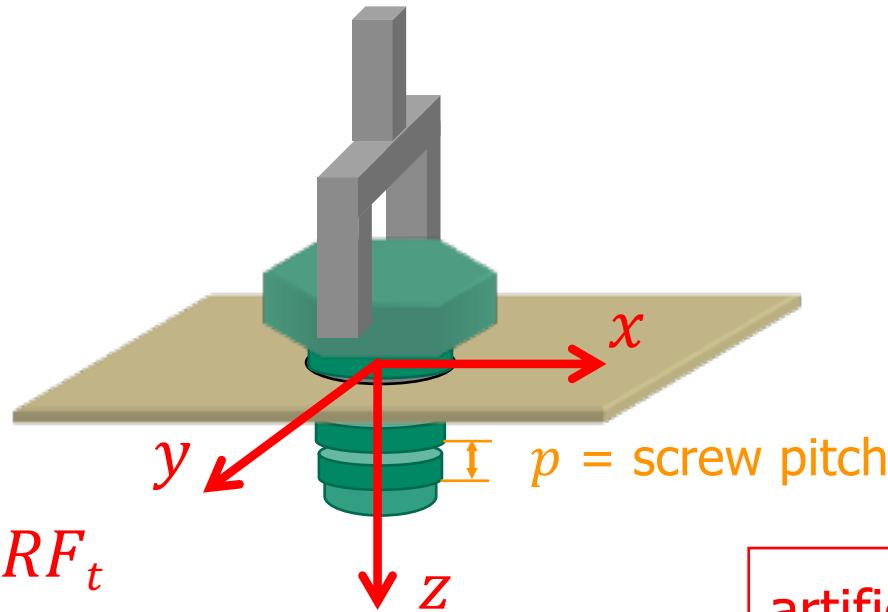
$$T^T(\alpha)Y(\alpha) = 0$$

parametrization of feasible reactions

$$\begin{pmatrix} {}^0f \\ {}^0\mu \end{pmatrix} = \begin{pmatrix} R^T(\alpha) & 0 \\ 0 & R^T(\alpha) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} f_x \\ f_z \\ \mu_x \\ \mu_y \end{pmatrix} = Y(\alpha) \begin{pmatrix} f_x \\ f_z \\ \mu_x \\ \mu_y \end{pmatrix}$$



Task frame and constraints - example 3



task: insert a screw
in a bolt

natural constraints (partial...)

$$v_x = v_y = 0$$

$$\omega_x = \omega_y = 0$$

the screw proceeds **along** and **around** the **z -axis**, but **not** in an **independent** way! (1 dof)

accordingly, f_z and μ_z **cannot** be **independent**

artificial constraints (abundant...)

$$f_x = f_{x,des} = 0, f_y = f_{y,des} = 0$$

$$\mu_x = \mu_{x,des} = 0, \mu_y = \mu_{y,des} = 0$$

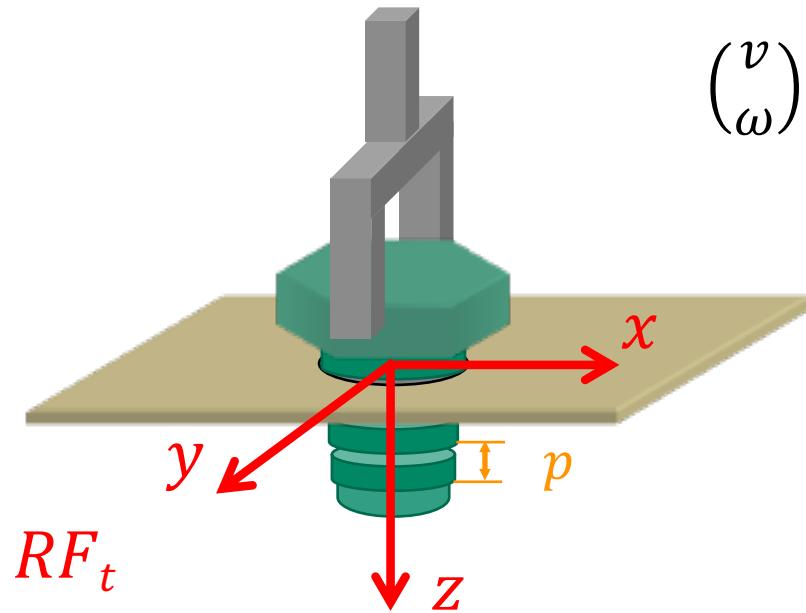
$$v_z = v_{z,des}, \omega_z = \omega_{z,des} = (2\pi/p)v_{z,des}$$

$$f_z = f_{z,des}, \mu_z = \mu_{z,des} \text{ (one function of the other!)}$$

wrench (force/torque) direction should be **orthogonal** to motion twist!



Selection of directions – example 3



the columns of \mathbf{T} and \mathbf{Y}
do not necessarily coincide
with selected columns
of the 6×6 identity matrix
 \Rightarrow generalized (screw)
directions

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & \frac{2\pi}{p} \end{pmatrix}^T v_z = T v_z \quad (k = 1)$$

or $\omega_z = 2\pi \frac{v_z}{p}$

\mathbf{Y} : such that $T^T \mathbf{Y} = 0$



$$f_z = -\frac{2\pi}{p} \mu_z$$

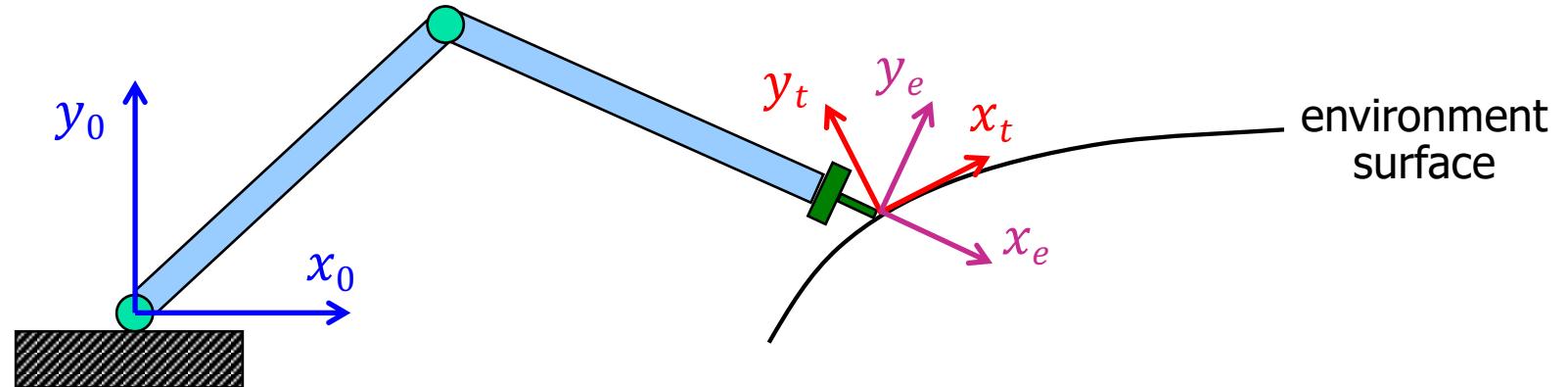
$(6 - k = 5)$

$$\begin{pmatrix} f \\ \mu \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2\pi/p \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_x \\ f_y \\ \mu_x \\ \mu_y \\ \mu_z \end{pmatrix} = \mathbf{Y} \begin{pmatrix} f_x \\ f_y \\ \mu_x \\ \mu_y \\ \mu_z \end{pmatrix}$$



Frames of interest – example 4

planar motion of a 2R robot ($n = 2$) in pointwise contact with a surface (task dimension $m = 2$)



- **task frame RF_t** used for an independent definition of the hybrid **reference values** (here: ${}^t v_{x,des}$ [$k = 1$] and ${}^t f_{y,des}$ [$m - k = 1$]) and for computing the errors that drive the **feedback control law**
- **sensor frame RF_e** (here: RF_2) where the **force** ${}^e f = ({}^e f_x, {}^e f_y)$ is measured
- **base frame RF_0** in which the end-effector **velocity** is expressed (here: ${}^0 v = ({}^0 v_x, {}^0 v_y)$ of O_2), computed using robot Jacobian and joint velocities

all quantities (and errors!) should be expressed ("rotated")
in the **same** reference frame \Rightarrow the **task frame!**



General parametrization of hybrid tasks

a “description” of robot-environment contact type:
it implicitly defines the task frame

$$\begin{cases} \begin{pmatrix} v \\ \omega \end{pmatrix} = T(s)\dot{s} & s \in \mathbb{R}^k \\ \begin{pmatrix} f \\ \mu \end{pmatrix} = Y(s)\lambda & \lambda \in \mathbb{R}^{m-k} \end{cases}$$

parametrizes robot E-E free motion
parametrizes reaction forces/torques

in general, it is $m = 6$
(as in most of the previous examples)

+

reaction forces/torques do not perform work on E-E displacements

$$T^T(s)Y(s) = 0$$

axes directions of **task frame** depend in general on s
(i.e., on robot E-E pose in the environment)

robot dynamics

$$M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) = u + J^T(q) \begin{pmatrix} f \\ \mu \end{pmatrix}$$

robot kinematics

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = J(q)\dot{q}$$



Hybrid force/velocity control

- **control objective:** to impose desired task evolutions to the parameters s of **motion** and to the parameters λ of **force**

$$s \rightarrow s_d(t) \quad \lambda \rightarrow \lambda_d(t)$$

- the control law is designed in **two steps**

1. exact **linearization and decoupling** in the **task frame** by feedback

$$\begin{array}{c} \text{closed-loop} \\ \text{model} \end{array} \rightarrow \begin{pmatrix} \ddot{s} \\ \ddot{\lambda} \end{pmatrix} = \begin{pmatrix} a_s \\ a_\lambda \end{pmatrix}$$

2. (**linear**) design of a_s and a_λ so as to impose the desired dynamic behavior to the errors $e_s = s_d - s$ and $e_\lambda = \lambda_d - \lambda$

- **assumptions:** $n = m$ (= 6 usually), $J(q)$ out of singularity

Note: in “simple” cases, \dot{s} and $\dot{\lambda}$ drive single components of v or ω and of f or μ ; accordingly, T and Y are just columns of 0/1 selection matrices



Feedback linearization in task space

$$J(q)\dot{q} = \begin{pmatrix} v \\ \omega \end{pmatrix} = T(s)\dot{s} \rightarrow J\ddot{q} + \dot{J}\dot{q} = T\ddot{s} + \dot{T}\dot{s} \rightarrow \ddot{q} = J^{-1}(T\ddot{s} + \dot{T}\dot{s} - \dot{J}\dot{q})$$

$$M(q)\ddot{q} + S(q, \dot{q})\dot{q} + g(q) = u + J^T(q) \begin{pmatrix} f \\ \mu \end{pmatrix} = u + J^T(q)Y(s)\lambda$$

(under the assumptions made)

$$(M(q)J^{-1}(q)T(s) : -J^T(q)Y(s)) \begin{pmatrix} \ddot{s} \\ \lambda \end{pmatrix} + M(q)J^{-1}(q)(\dot{T}(s)\dot{s} - \dot{J}(q)\dot{q}) + S(q, \dot{q})\dot{q} + g(q) = u$$

$$u = (MJ^{-1}T : -J^TY) \begin{pmatrix} a_s \\ a_\lambda \end{pmatrix} + MJ^{-1}(\dot{T}\dot{s} - \dot{J}\dot{q}) + S\dot{q} + g$$

linearizing and
decoupling
control law

$$\rightarrow \begin{pmatrix} \ddot{s} \\ \lambda \end{pmatrix} = \begin{pmatrix} a_s \\ a_\lambda \end{pmatrix} \left. \right\} \begin{matrix} k \\ m - k \end{matrix} \quad \begin{matrix} s \text{ has "relative degree" } = 2 \\ \lambda \text{ has "relative degree" } = 0 \end{matrix}$$



Stabilization with a_s and a_λ

as usual, it is sufficient to apply **linear** control techniques for the exponential stabilization of tracking errors (on each single, input-output decoupled channel)

$$a_s = \ddot{s}_d + K_D(\dot{s}_d - \dot{s}) + K_P(s_d - s)$$

$K_P, K_D > 0$
and diagonal

$$\ddot{e}_s + K_D \dot{e}_s + K_P e_s = 0 \quad \rightarrow \quad e_s = s_d - s \rightarrow 0$$

$K_I \geq 0$
diagonal

$$a_\lambda = \lambda_d + K_I \int (\lambda_d - \lambda) dt$$

$a_\lambda = \lambda_d$ would be enough,
but adding an integral
with the **force error**
gives more robustness
to (constant) disturbances

$$e_\lambda + K_I \int e_\lambda dt = 0 \quad \rightarrow \quad e_\lambda = \lambda_d - \lambda \rightarrow 0$$

we need “values” for s , \dot{s} and λ to be
extracted from actual **measurements** !



“Filtering” position and force measures

- s and \dot{s} are obtained from measures of q and \dot{q} , equating the descriptions of the end-effector pose and velocity “from the robot side” (direct and differential kinematics) and “from the environment side” (function of s, \dot{s})

example

$$\begin{aligned} {}^0r &= {}^0f(q) = \begin{pmatrix} L \cos s \\ L \sin s \\ 0 \end{pmatrix} \rightarrow s = \text{atan2}\{{}^0f_y(q), {}^0f_x(q)\} \\ &\quad \text{Diagram: A 3D coordinate system } (x_0, y_0, z_0) \text{ with a green cylinder at the origin. A blue cylinder is attached to a grey robotic arm. The distance from the origin to the center of the blue cylinder is } L. \text{ The angle between the vertical } z_0 \text{ axis and the line of sight to the center of the blue cylinder is } s = \alpha. \text{ A blue arrow labeled } r \text{ points from the origin to the center of the blue cylinder.} \\ J(q)\dot{q} &= T(s)\dot{s} \rightarrow \dot{s} = T^\#(s)J(q)\dot{q} \end{aligned}$$

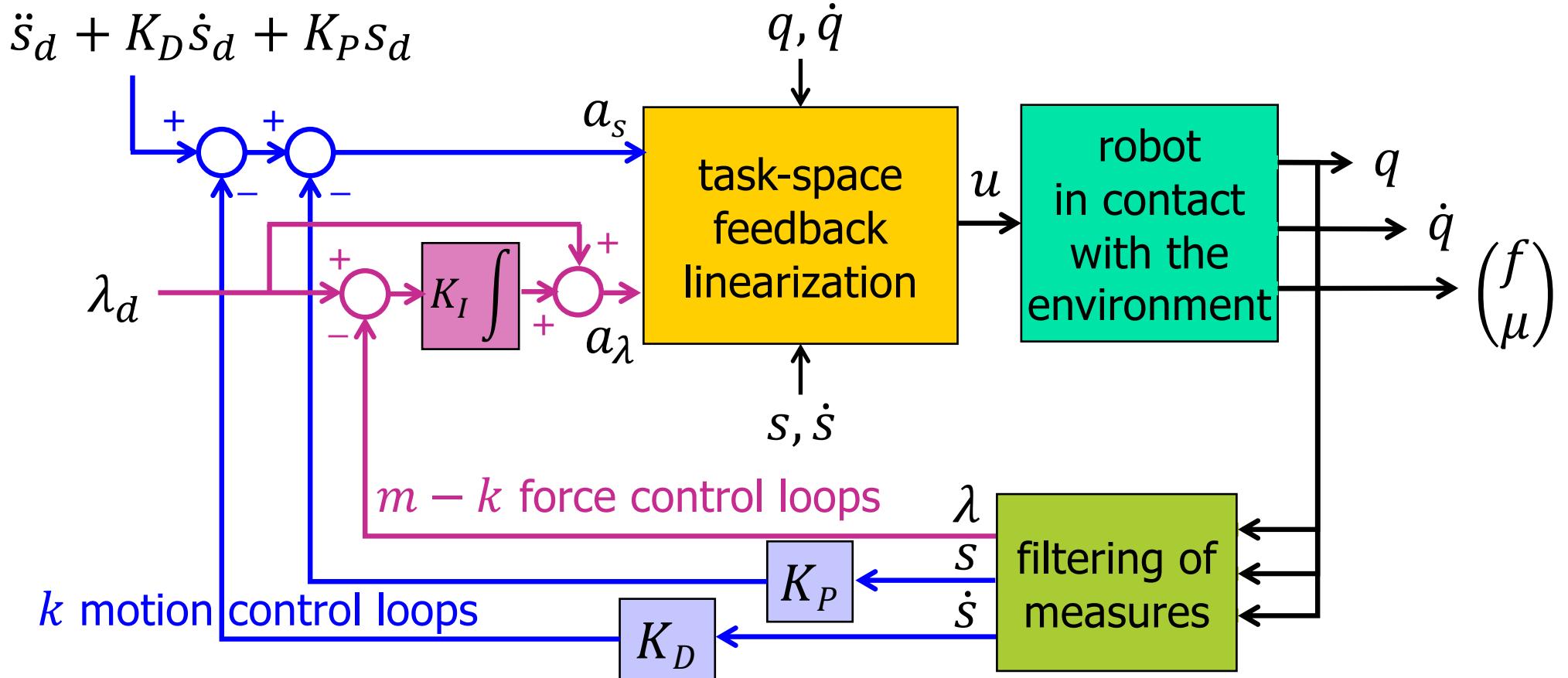
- λ is obtained from force/torque measures at end-effector

$$\begin{pmatrix} f \\ m \end{pmatrix} = Y(s)\lambda \rightarrow \lambda = Y^\#(s) \begin{pmatrix} f \\ m \end{pmatrix}$$

pseudoinverses
of “tall” matrices
having full
column rank, e.g.,
 $T^\# = (T^T T)^{-1} T^T$
(or weighted)



Block diagram of hybrid control



usually $m = 6$ (complete 3D space)

limit cases $k = m$: no force control loops, only motion (free motion)

$k = 0$: no motion control loops, only force ("frozen" robot end-effector)



Block diagram of hybrid control

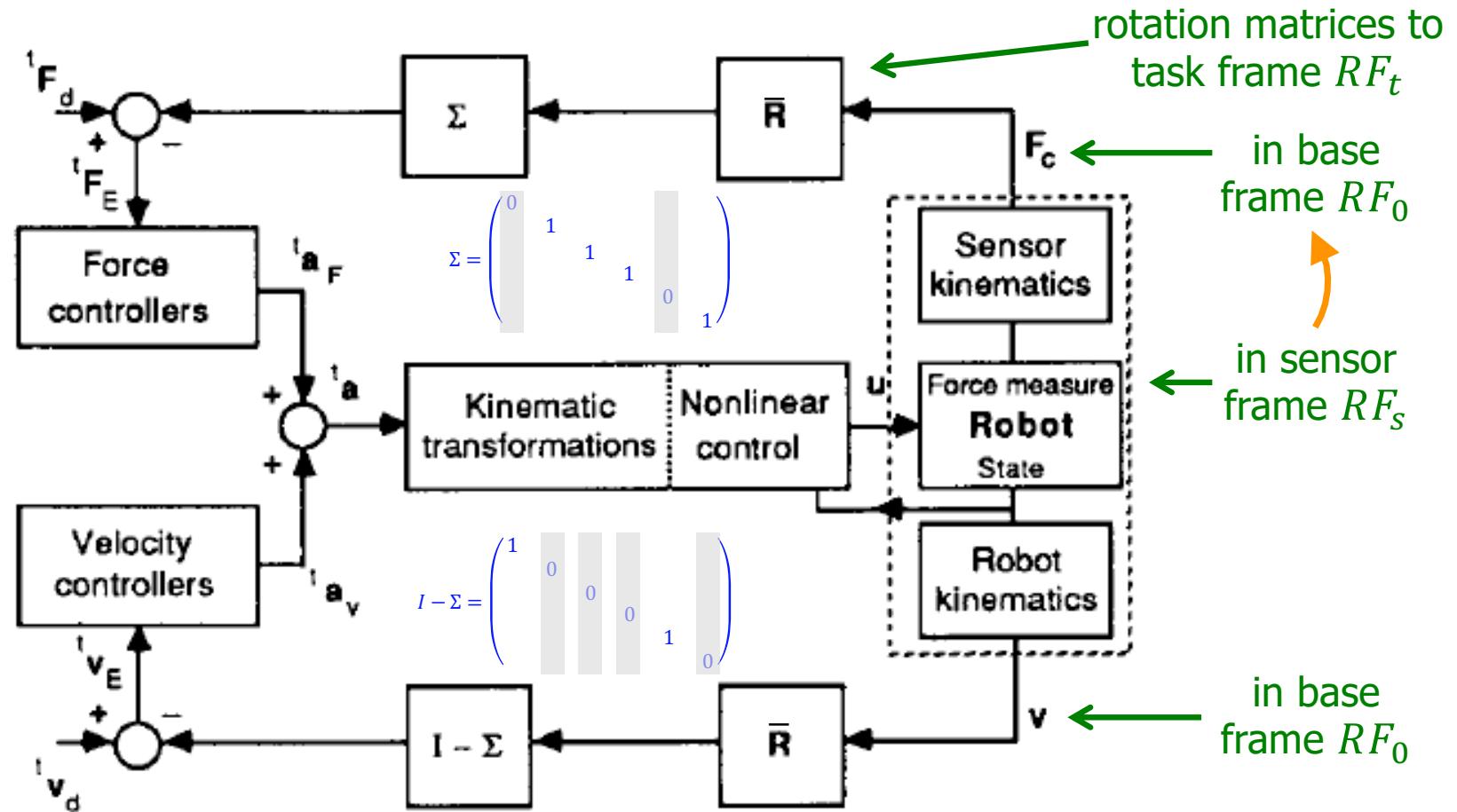
simpler case of 0/1 selection matrices

compact notation
in this slide

$$F = \begin{pmatrix} f \\ \mu \end{pmatrix}$$

$$V = \begin{pmatrix} v \\ \omega \end{pmatrix}$$

$$\bar{R} = \begin{pmatrix} R & 0 \\ 0 & R \end{pmatrix}$$



λ and \dot{s} are just single components of f (or μ) and v (or ω)

Y and T are replaced by 0/1 selection matrices: Σ and $I - \Sigma$



Force control via an impedance model

- in a force-controlled direction of the hybrid task space, when the **contact stiffness is limited** (i.e., far from infinite, as assumed in the ideal case), one may use **impedance model ideas** to explicitly **control the contact force**
 - let x be the position of the robot along such a direction, x_d the (constant) contact point, $k_s > 0$ the contact (viz., sensor) stiffness, and $f_d > 0$ the desired contact force
- the impedance model is chosen then as

$$m_m \ddot{x} + d_m \dot{x} + k_s(x - x_d) = f_d$$

where the **force sensor** measures $f_s = k_s(x - x_d)$, and only $m_m > 0$ and $d_m > 0$ are free model parameters

- after feedback linearization ($\ddot{x} = a_x$), the command a_x is designed as

$$a_x = (1/m_m)[(f_d - f_s) - d_m \dot{x}]$$

which is a **P-regulator** of the desired force, **with velocity damping**

- the **same** control law works also before the contact ($f_s = 0$), guaranteeing a steady-state speed $\dot{x}_{ss} = f_d/d_m > 0$ in the **approaching phase**



First experiments with hybrid control

First Experiments with Hybrid Force/Velocity Control

Università di Roma "La Sapienza"
DIS, LabRob
February 1991



video

First Experiments with Hybrid Force/Velocity Control

(part II)

Università di Roma "La Sapienza"
DIS, LabRob
February 1991



video

MIMO-CRF robot
(DIS, Laboratorio di Robotica, 1991)

Sources of inconsistency in force and velocity measurements



1. presence of **friction** at the contact
 - a reaction force component appears that opposes motion in a “free” motion direction (in case of Coulomb friction, the tangent force intensity depends also on the applied normal force ...)
 2. **compliance** in the robot structure and/or at the contact
 - a (small) displacement may be present also along directions that are nominally “constrained” by the environment
- NOTE:** if the environment geometry at the contact is perfectly known, the task inconsistencies due to 1. and 2. on parameters s and λ are already **filtered out** by the pseudo-inversion of matrices T and Y
3. uncertainty on **environment geometry** at the contact
 - can be reduced/eliminated by real-time **estimation processes** driven by external sensors (e.g., vision –but also force!)

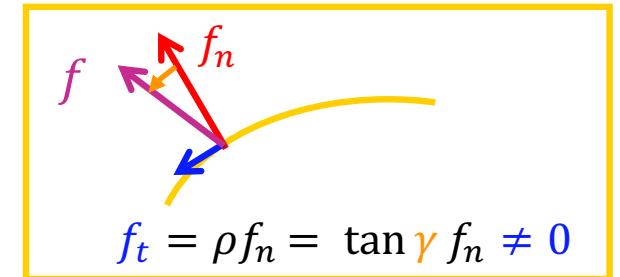


Estimation of an unknown surface

how difficult is to **estimate** the unknown profile of the environment surface, using information from velocity and force measurements at the contact?

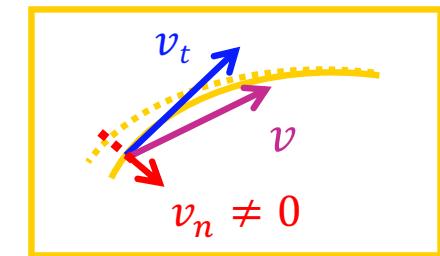
1. normal = nominal direction of measured **force**

... in the presence of contact motion with friction, the **measured** force f is slightly rotated from the actual normal by an (unknown) angle γ



2. tangent = nominal direction of measured **velocity**

... compliance in the robot structure (joints) and/or at the contact may lead to a **computed** velocity v having a small component along the actual normal to the surface



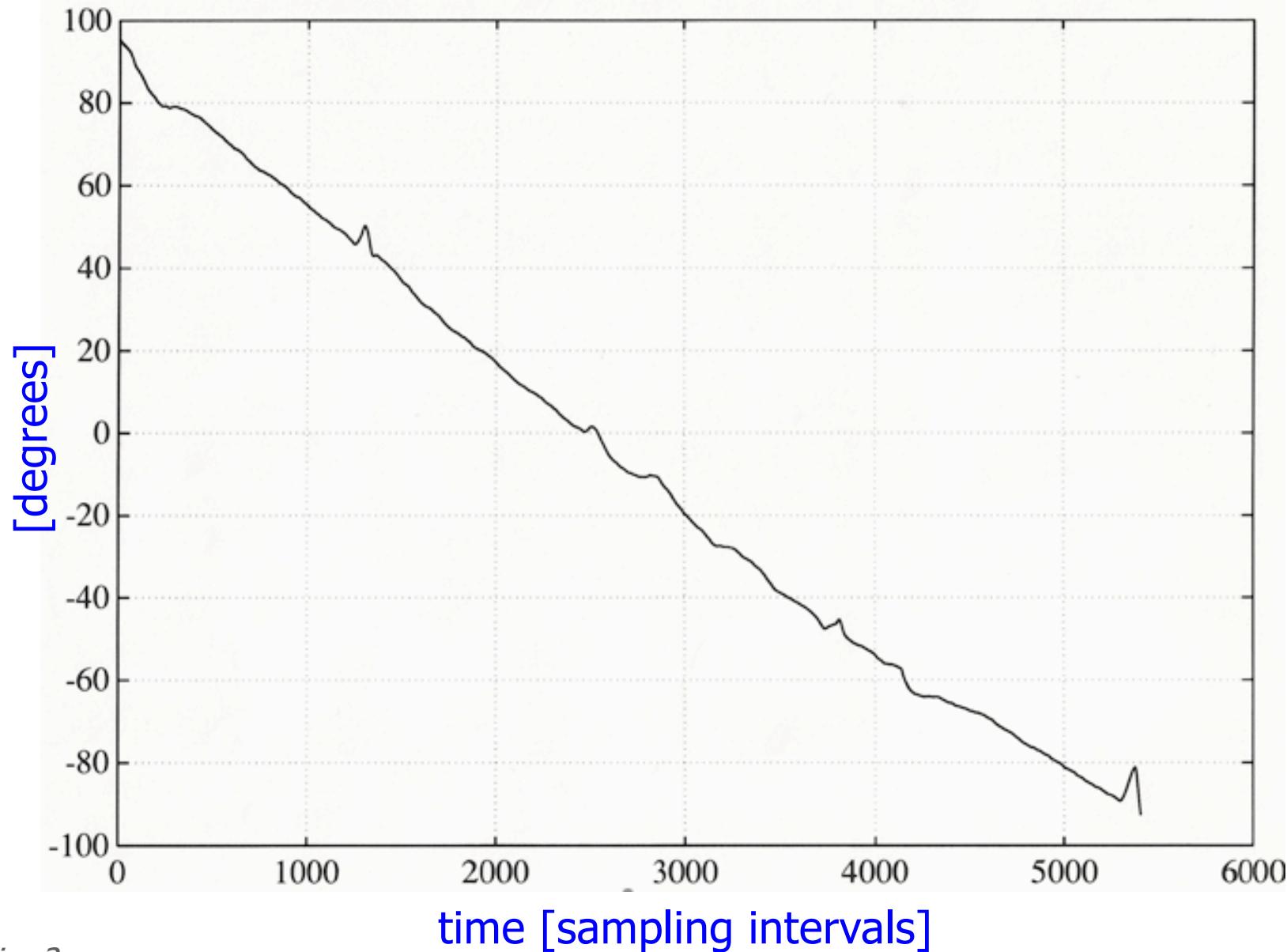
3. mixed method (sensor fusion) with RLS

- a. tangent direction is estimated by a **recursive least squares** method from position measurements
- b. friction angle is estimated by a **recursive least squares** method, using the current estimate of the tangent direction and from force measurements

to approach an unknown surface or to recover contact (in case of loss), the robot uses simple exploratory moves



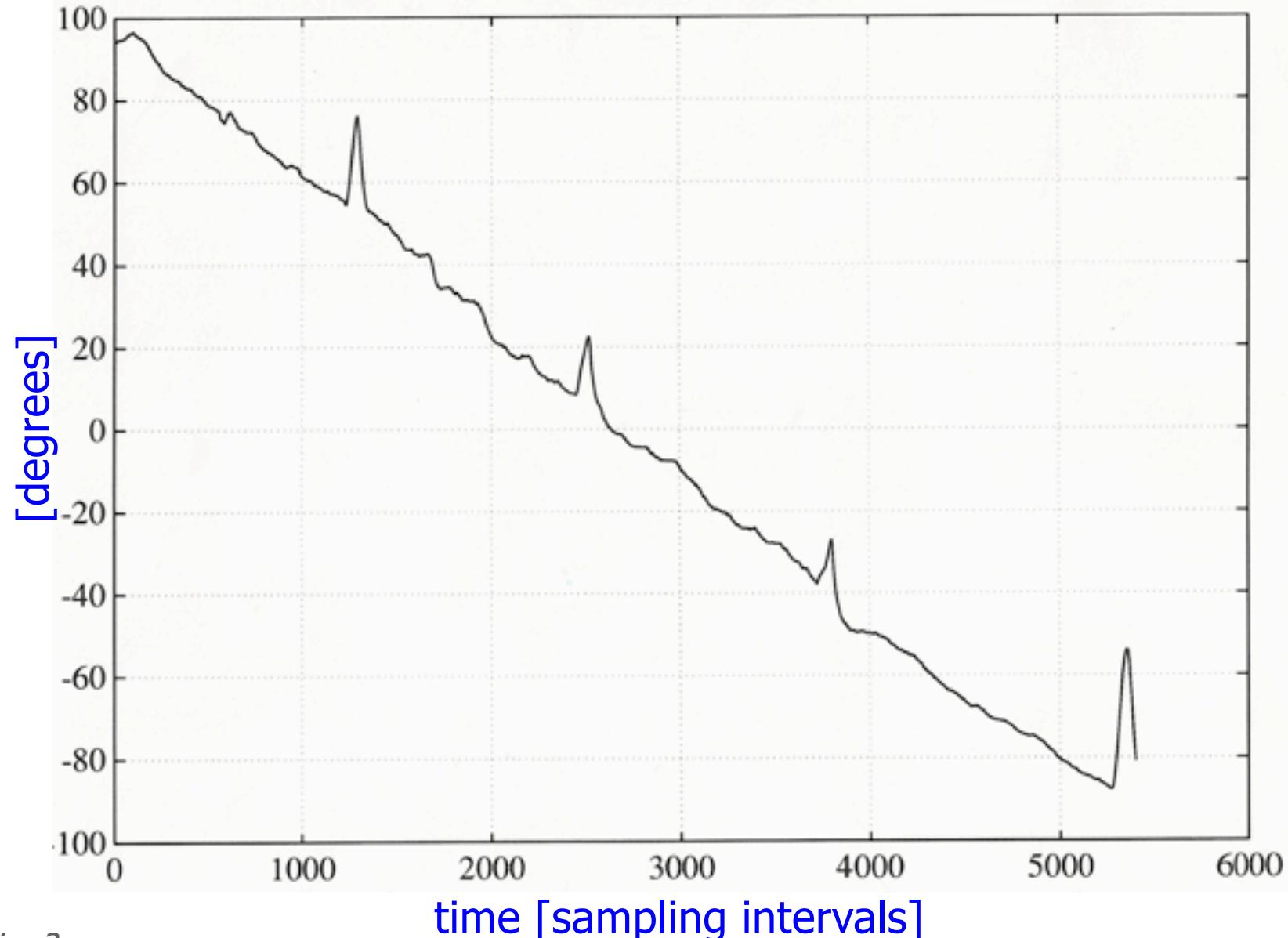
Position-based estimation of the tangent (for a circular surface traced at constant speed)





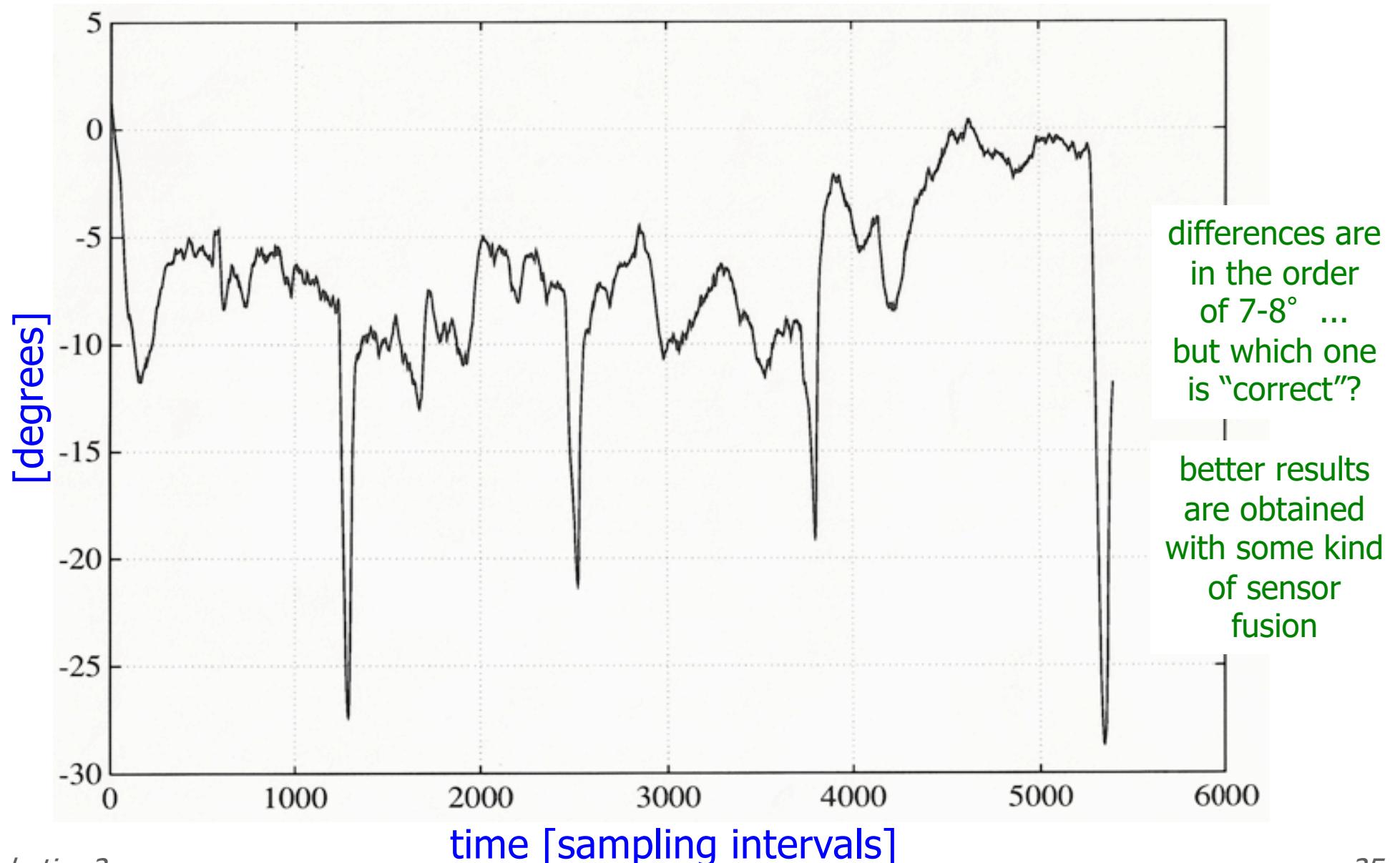
Force-based estimation of the tangent

(for the same circular surface traced at constant speed)





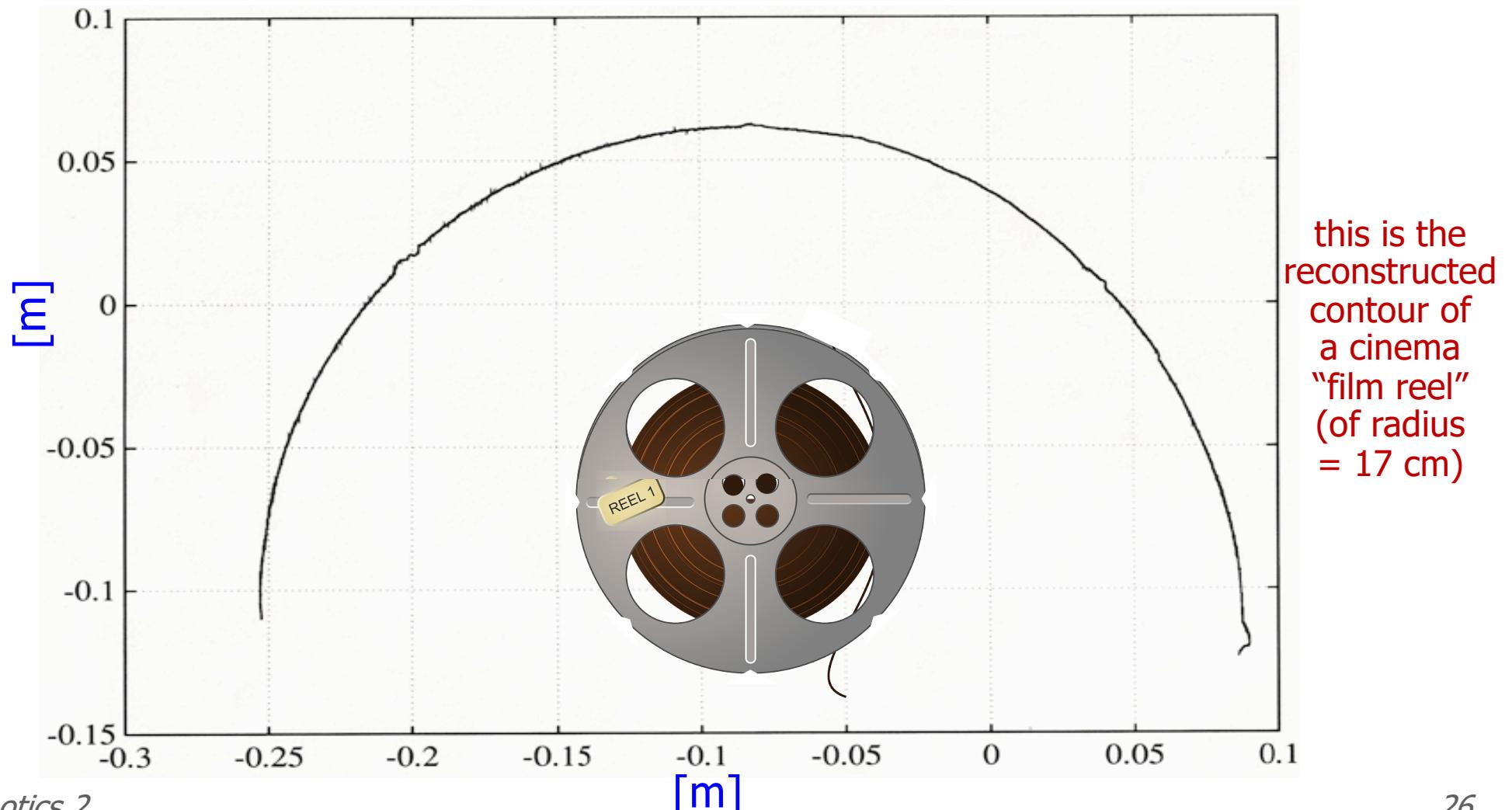
Difference between estimated tangents





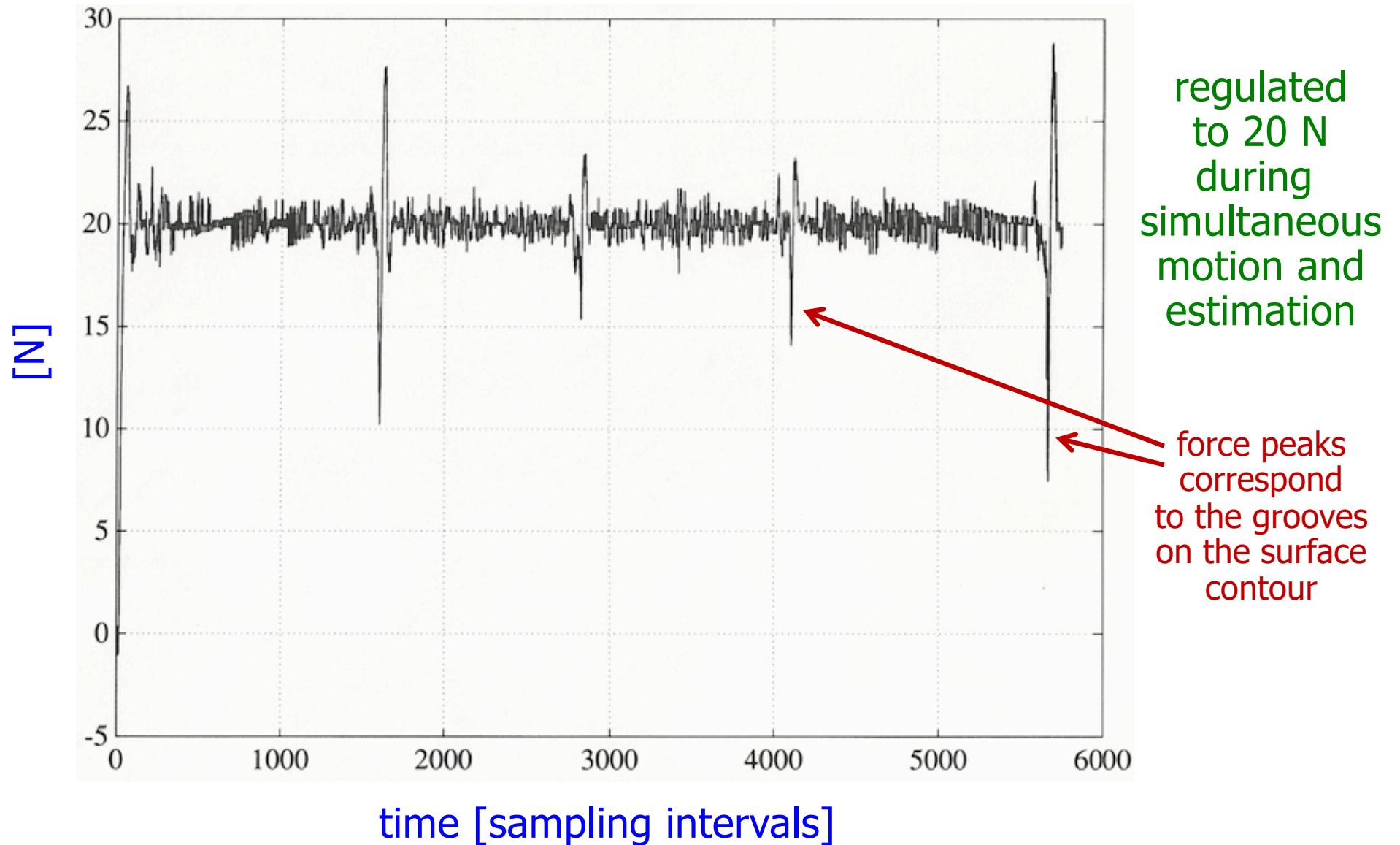
Reconstructed surface profile

estimation by a RLS (Recursive Least Squares) method: we continuously update the coefficients of two quadratic polynomials that fit locally the unknown contour, using data fusion from both force and position/velocity measurements



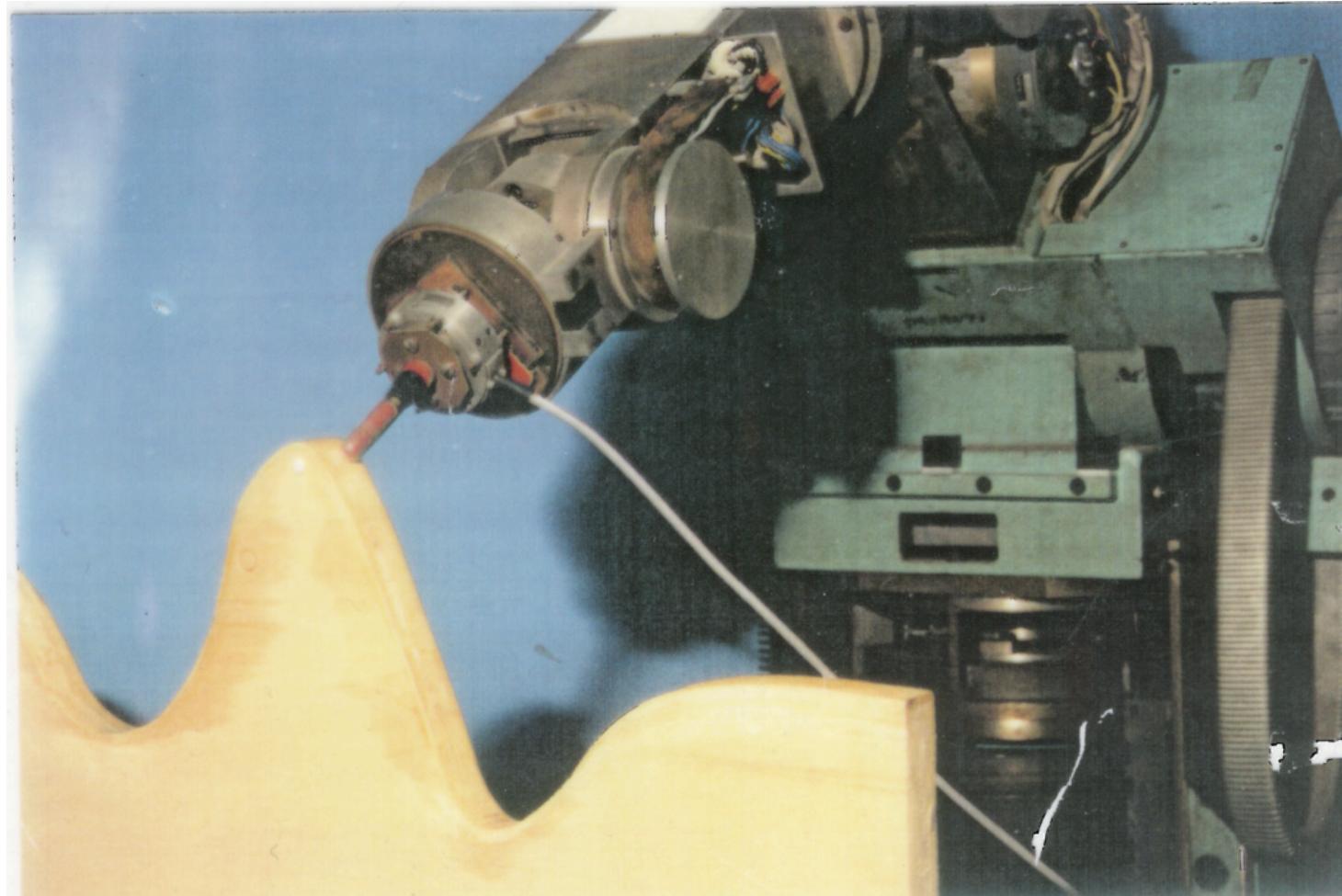


Normal force





Contour estimation and hybrid control performed simultaneously



MIMO-CRF robot (DIS, Laboratorio di Robotica, 1992)



Contour estimation and hybrid control

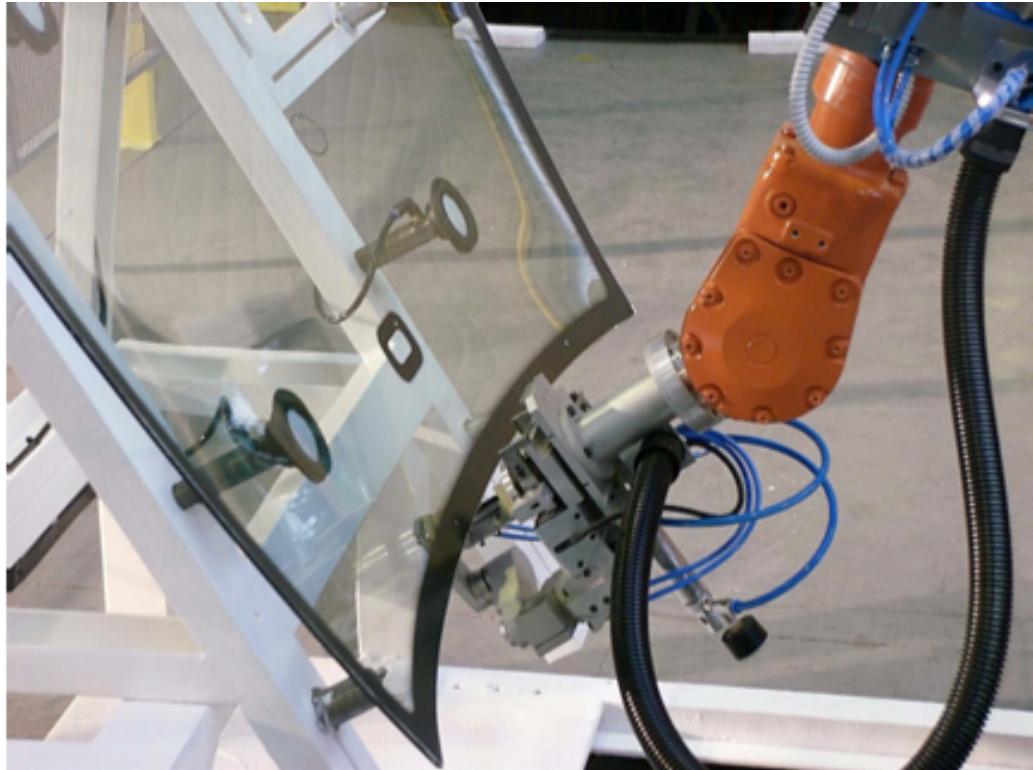
Hybrid Force/Velocity Control and Identification of Surfaces

**Università di Roma "La Sapienza"
DIS, LabRob
September 1992**



video

Robotized deburring of car windshields



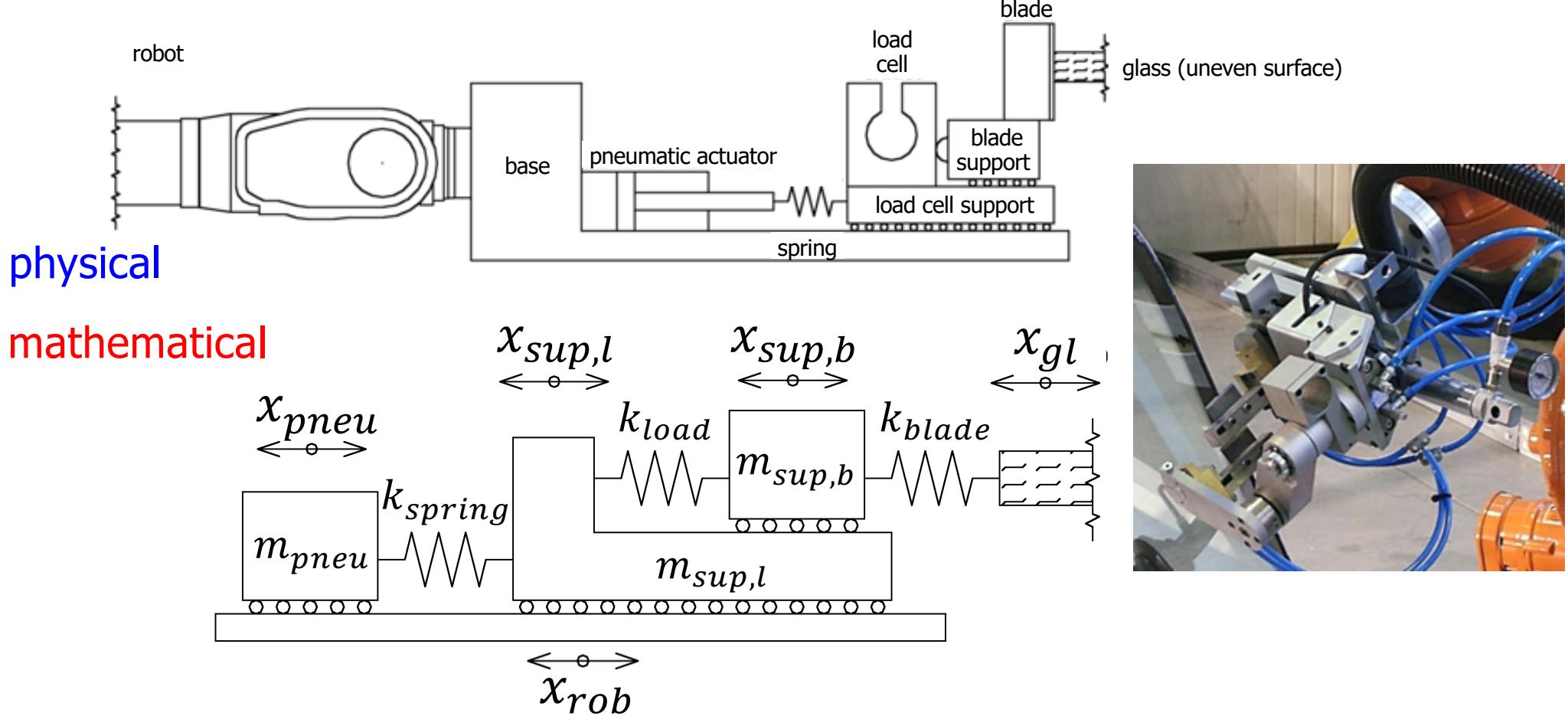
- car windshield with **sharp edges** and fabrication tolerances, with **excess of material** (PVB = Polyvinyl butyral for gluing glass layers) on the contour
- robot end-effector follows the pre-programmed path, despite the small errors w.r.t. the nominal windshield profile, thanks to the **compliance** of the deburring tool
- contact force between tool blades and workpiece can be independently controlled by a **pneumatic actuator** in the tool

the robotic deburring tool contains in particular

- **two blades** for cutting the exceeding plastic material (PVB), the first one actuated, the second passively pushed against the surface by a spring
- a **load cell** for measuring the 1D applied normal force at the contact
- on-board **control system**, exchanging data with the ABB robot controller



Model of the deburring work tool



for a stability analysis (based on linear models and root locus techniques)
of force control in a single direction and in presence of multiple masses/springs,
see again Eppinger & Seering, IEEE CSM, 1987 (material in the course web site)



Summary through video segments



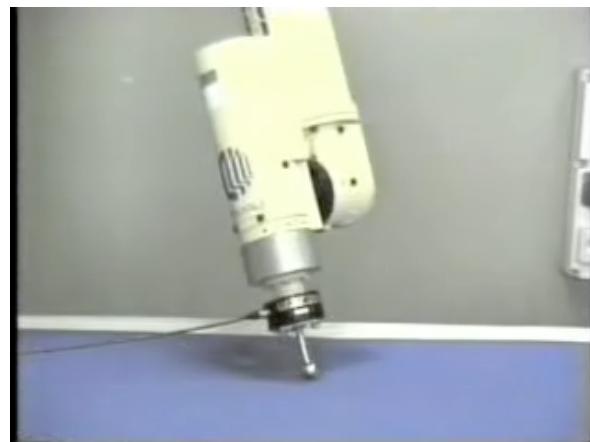
compliance control
(active Cartesian stiffness
control **without** F/T sensor)



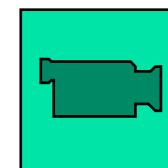
impedance control
(with F/T sensor)



force control
(realized as external loop
providing the reference to
an internal position loop
-see Appendix)



hybrid force/position control

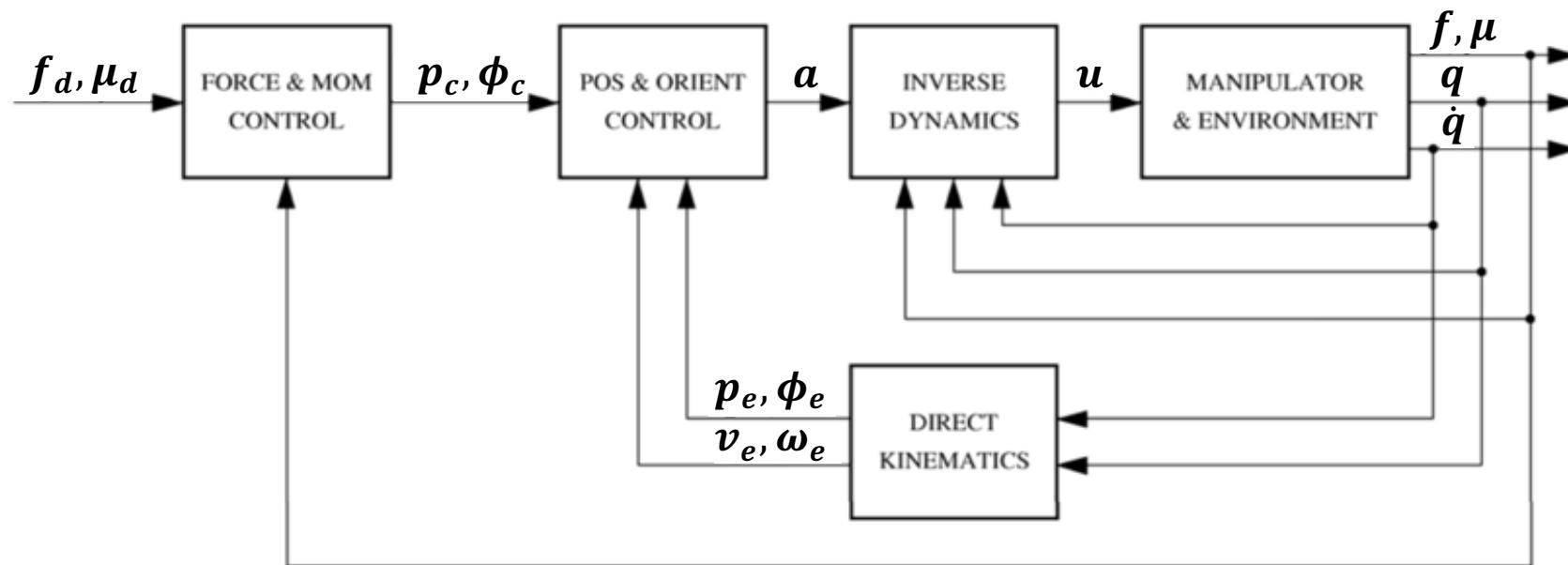


COMAU Smart robot
c/o Università di Napoli, 1994
(full video on course web site)



Appendix

- force control can also be realized as an external loop providing reference values to an internal motion loop (see video in slide #32)
- inner-outer (or cascaded) control scheme
 - angular position quantities (E-E orientation, errors, commands) can be expressed in different ways (Euler angles ϕ , rotation matrices R , ...)





Robotics 2

Visual servoing

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Visual servoing

- objective
 - use information acquired by **vision sensors** (cameras) for **feedback control** of the pose/motion of a robot (or of parts of it)
- + data acquisition ~ human eye, with very large information content in the acquired images
- difficult to extract essential data, nonlinear perspective transformations, high sensitivity to ambient conditions (lightening), noise



Some applications

automatic navigation of robotic systems (agriculture, automotive)

video



video



surveillance

video



bio-motion synchronization (surgical robotics)

video





Image processing

- **real-time** extraction of characteristic parameters useful for robot motion control
 - **features:** points, area, geometric center, higher-order moments, ...
- low-level processing
 - binarization (threshold on grey levels), equalization (histograms), edge detection, ...
- segmentation
 - based on regions (possibly in binary images)
 - based on contours
- interpretation
 - association of characteristic parameters (e.g., texture)
 - problem of **correspondence** of points/characteristics of objects in different images (stereo or on image flows)

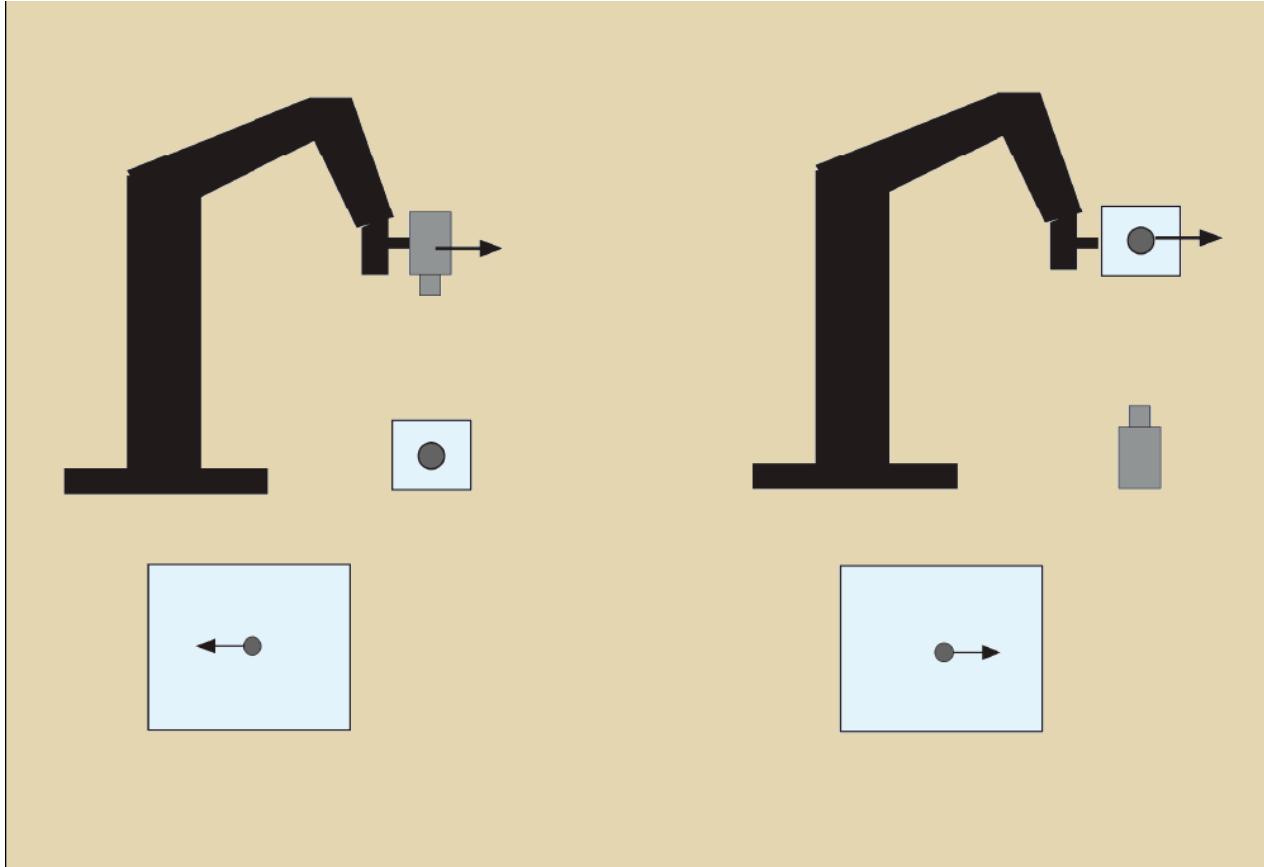


Configuration of a vision system

- one, two, or more cameras
 - grey-level or color
- 3D/stereo vision
 - obtained even with a single (moving) camera, with the object taken from different (known) points of view
- camera positioning
 - fixed (eye-to-hand)
 - mounted on the manipulator (eye-in-hand)
- robotized vision heads
 - motorized (e.g., stereo camera on humanoid head or pan-tilt camera on Magellan mobile robot)



Camera positioning



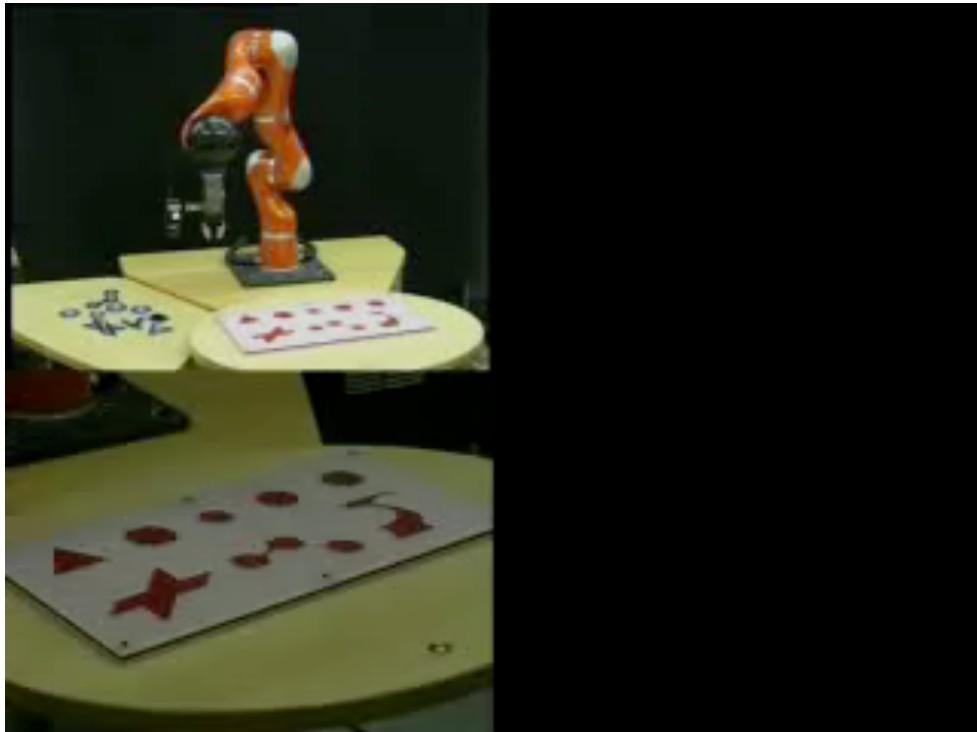
eye-in-hand

eye-to-hand



Vision for assembly

video



video

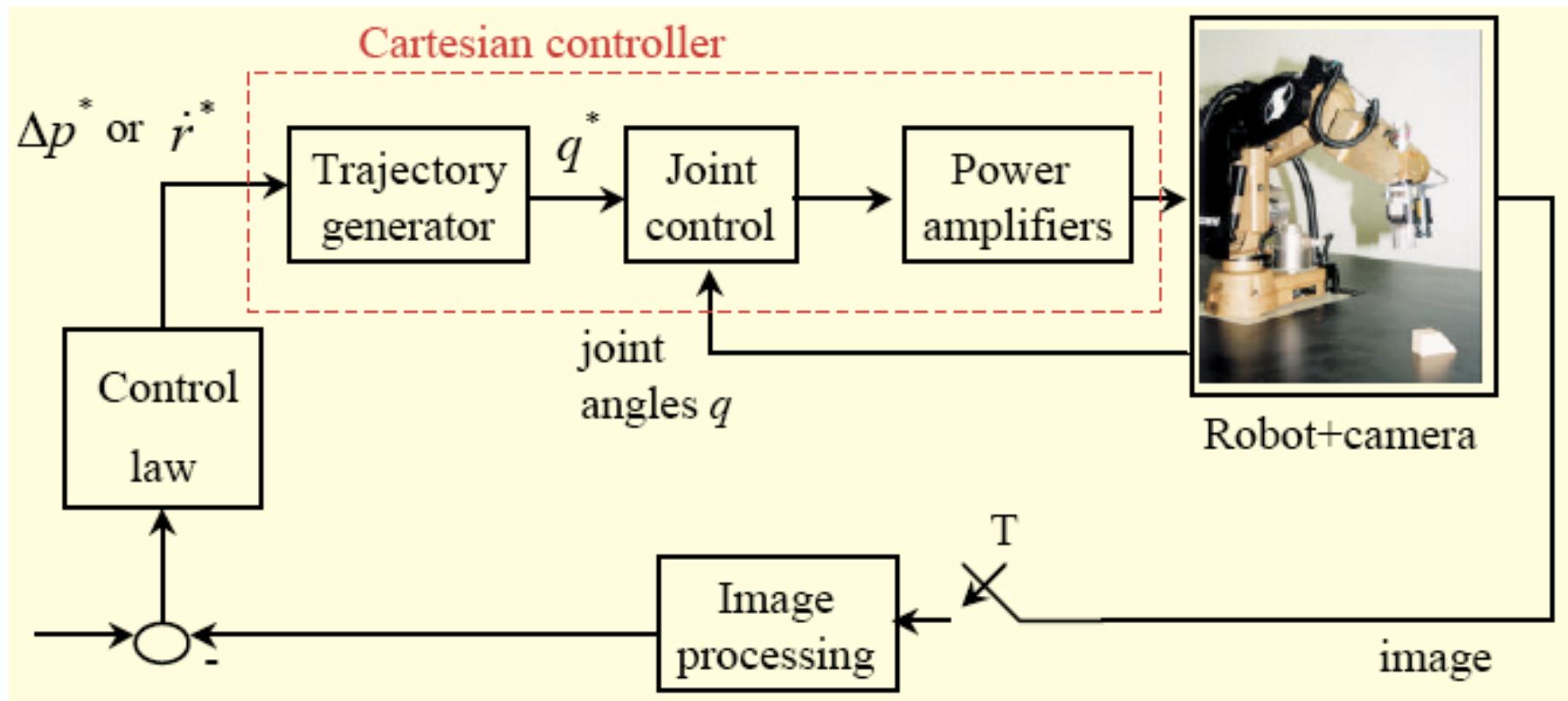


PAPAS-DLR system
(eye-in-hand, **hybrid force-vision**)

robust w.r.t. motion of
target objects



Indirect/external visual servoing

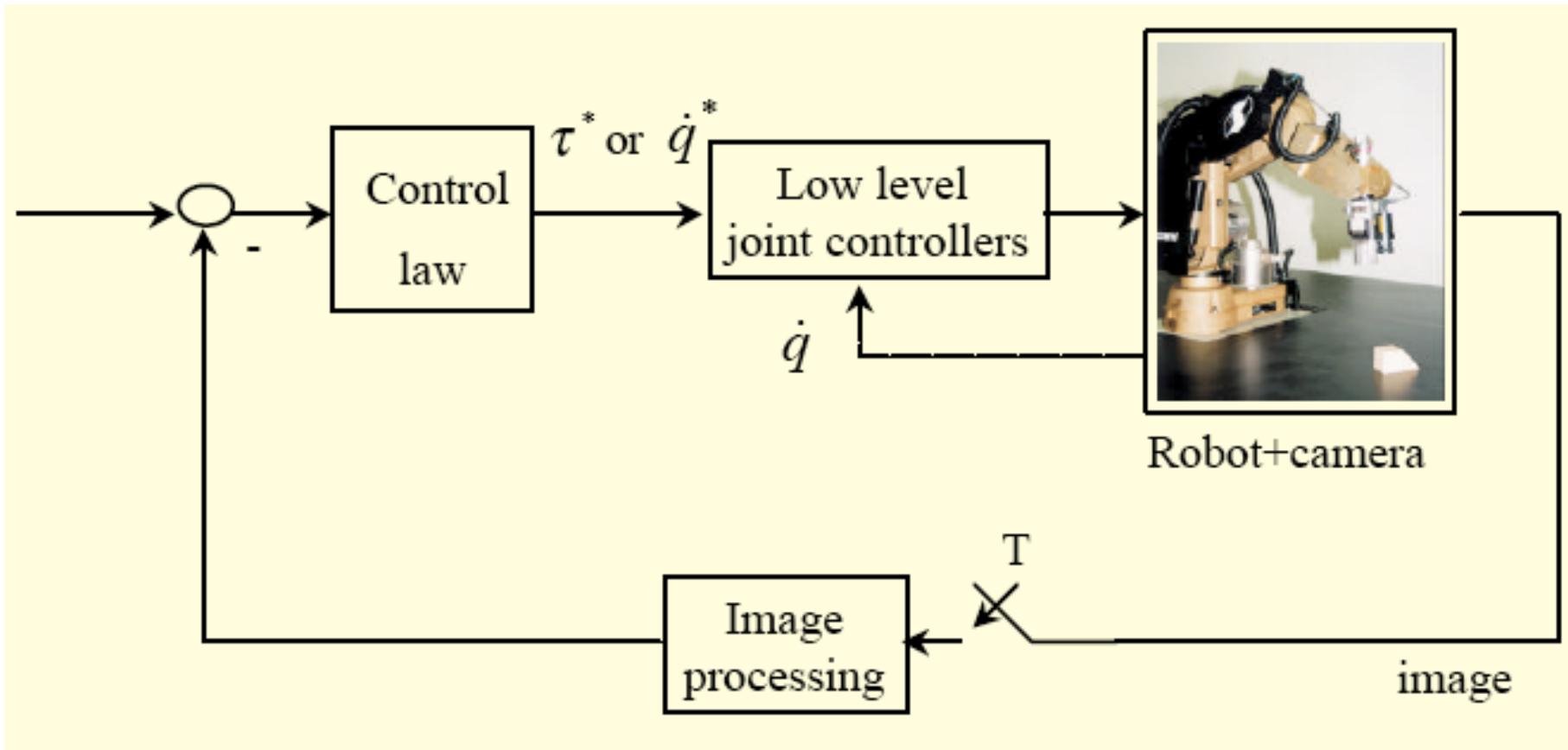


vision system **provides set-point references** to a **Cartesian motion controller**

- “easy” control law (same as without vision)
- appropriate for relatively slow situations (control sampling $f = 1/T < 50\text{Hz}$)



Direct/internal visual servoing



replace Cartesian controller with one based on vision that **directly computes joint reference commands**

- control law is more complex (involves robot kinematics/dynamics)
- preferred for fast situations (control sampling $f = 1/T > 50\text{Hz}$)



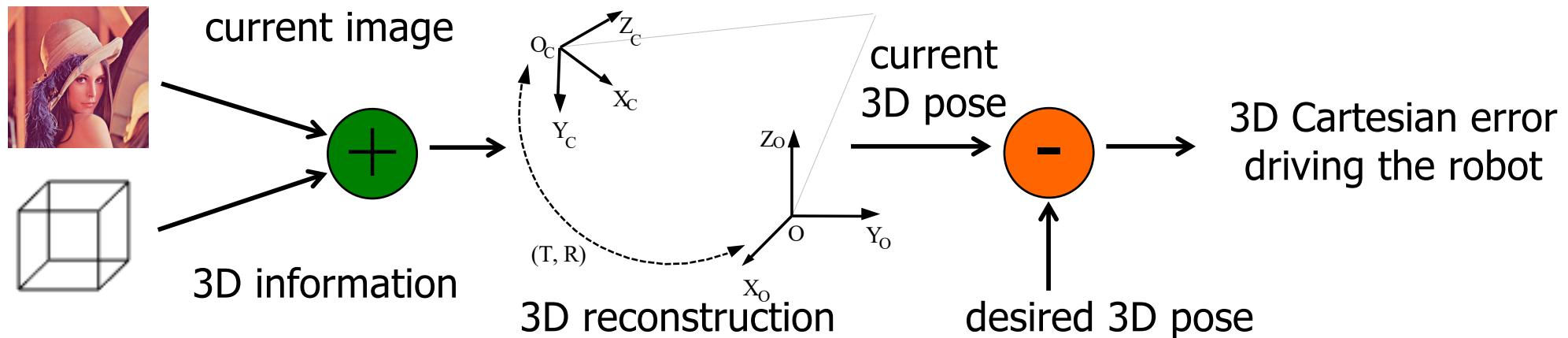
Classification of visual servoing schemes

- position-based visual servoing (**PBVS**)
 - information extracted from images (**features**) is used to reconstruct the **current 3D pose** (pose/orientation) of an object
 - combined with the knowledge of a **desired 3D pose**, we generate a **Cartesian** pose error signal that drives the robot to the goal
- image-based visual servoing (**IBVS**)
 - error is computed directly on the values of the features extracted on the **2D image plane**, **without** going through a 3D reconstruction
 - the robot should move so as to bring the current image features (what it “sees” with the camera) to their desired values
- some mixed schemes are possible (e.g., **2½D methods**)

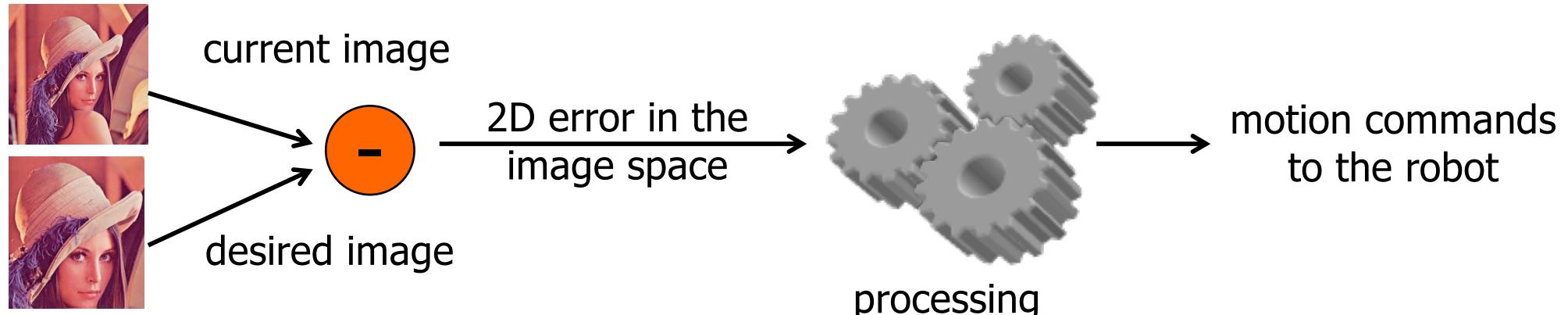


Comparison between the two schemes

- position-based visual servoing (**PBVS**)

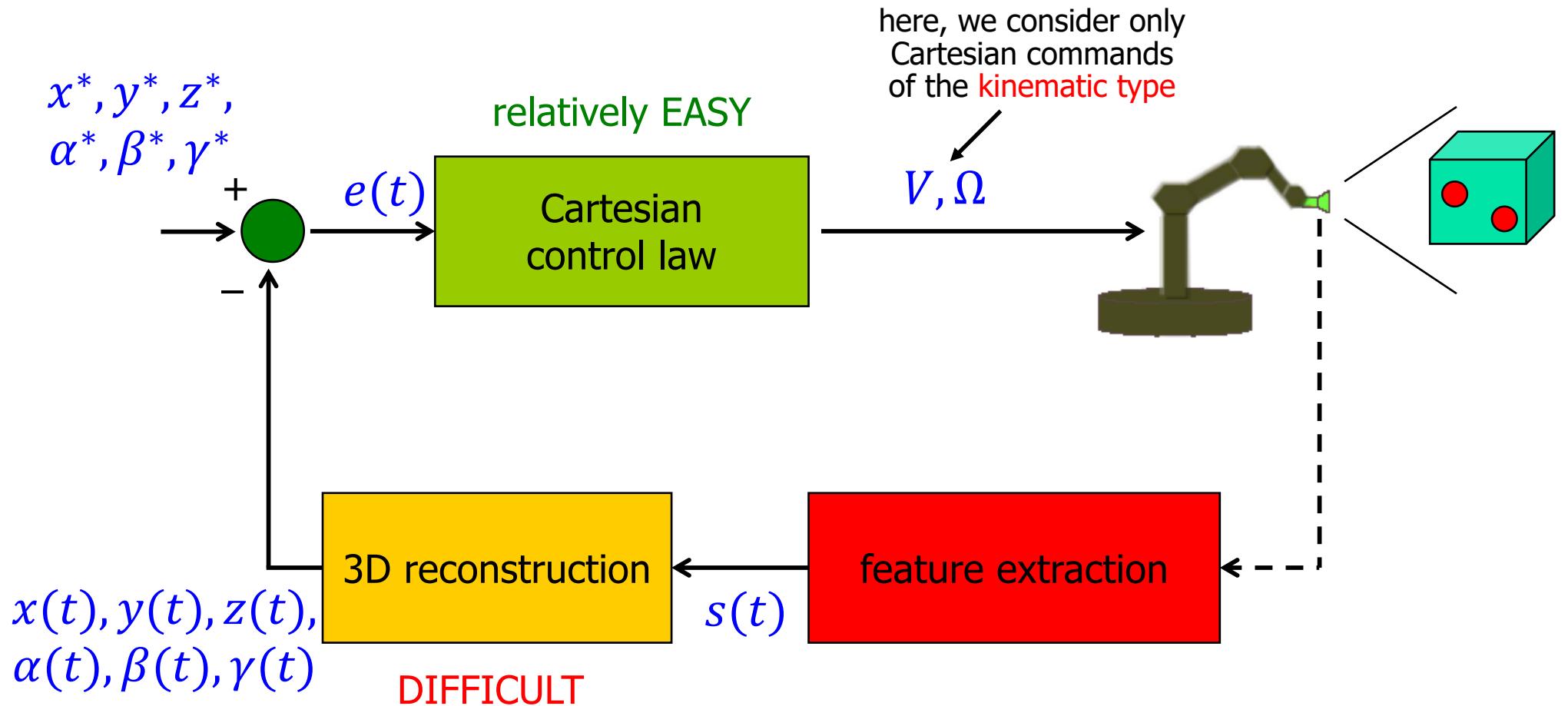


- image-based visual servoing (**IBVS**)





PBVS architecture

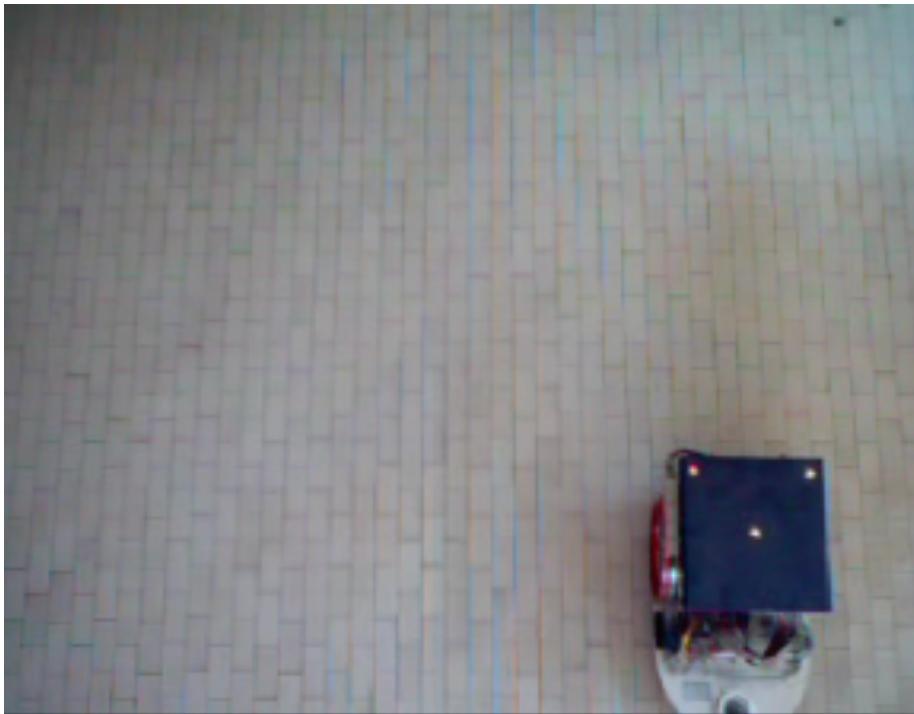


highly sensitive to camera calibration parameters



Examples of PBVS

video



eye-to-“robot” (SuperMario)

position/orientation of the camera
and scene geometry

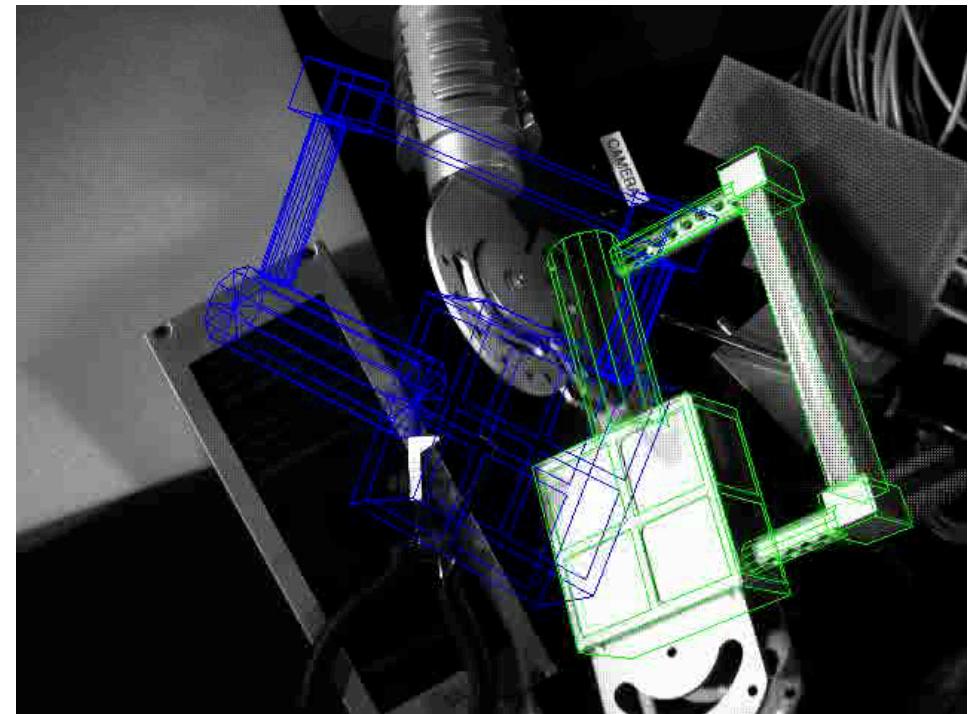


eye-in-hand (Puma robot)

position and orientation of the robot
(with mobile or fixed base)

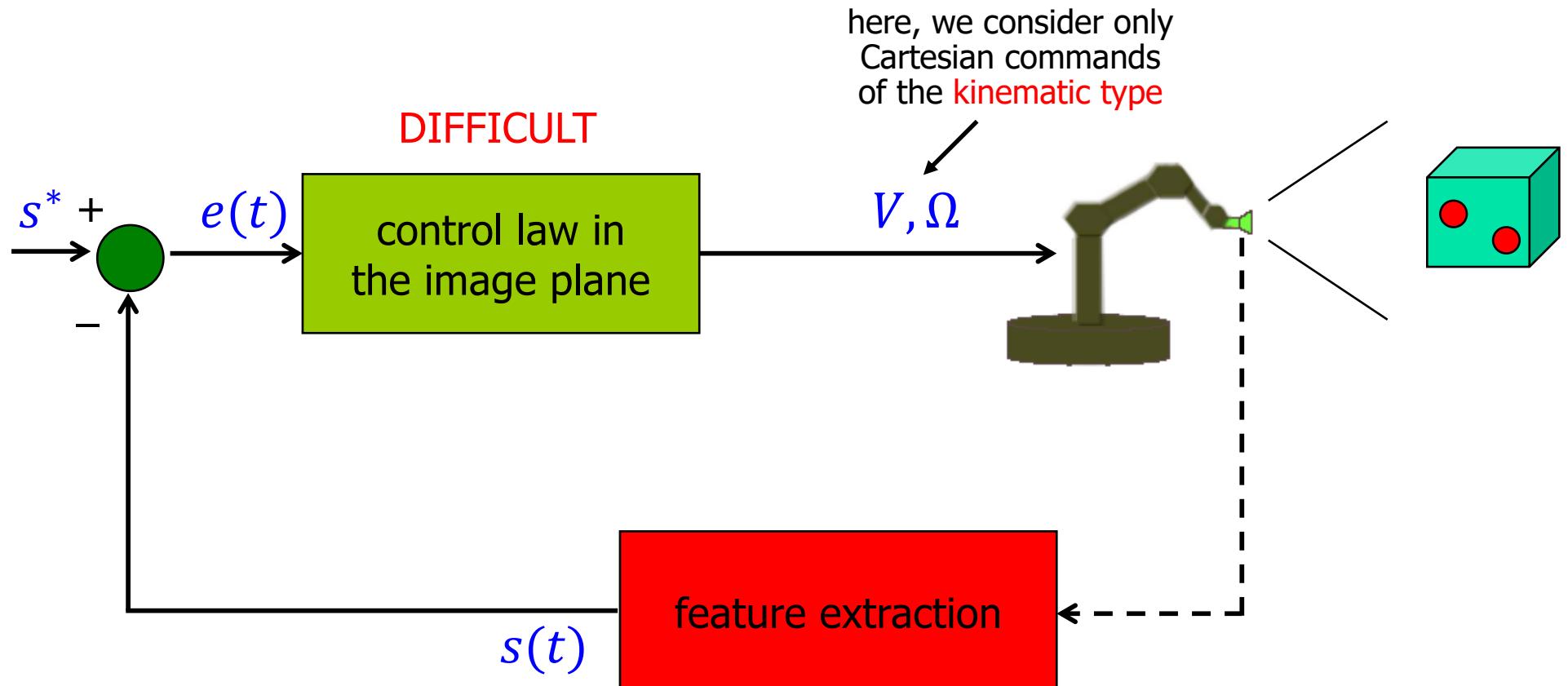
known a priori!

video





IBVS architecture

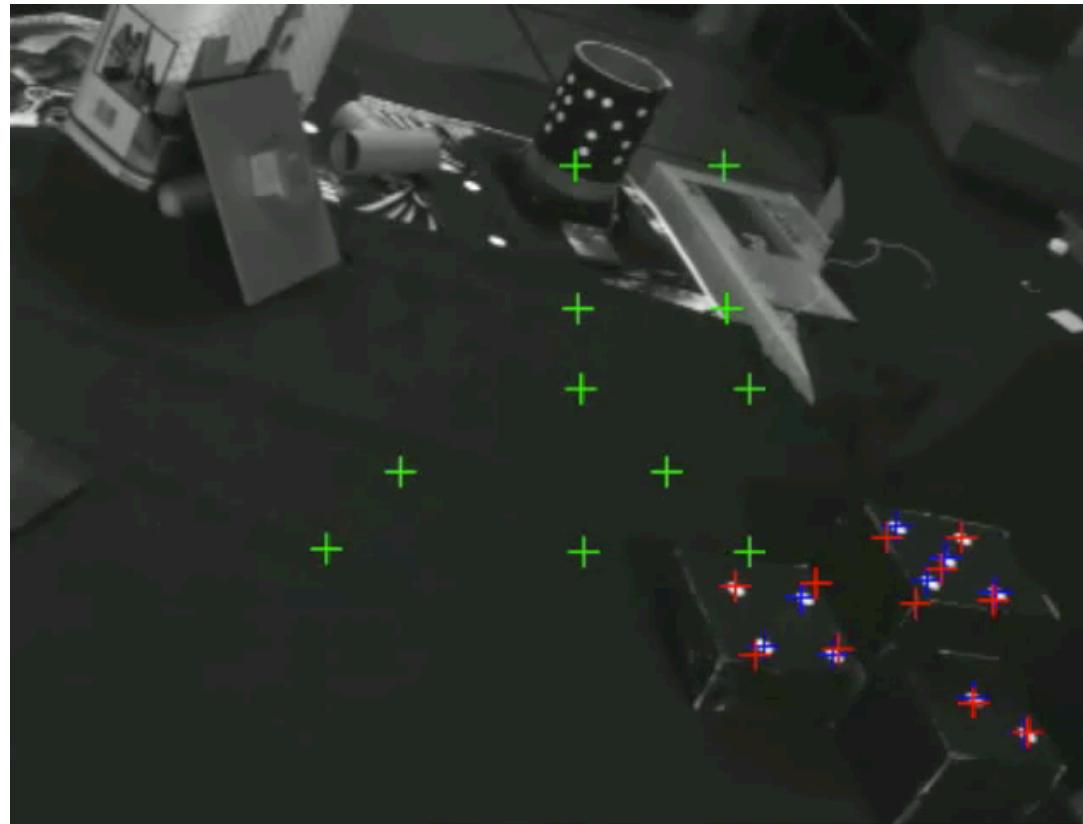


almost insensitive to intrinsic/extrinsic camera calibration parameters



An example of IBVS

here, the features are **points** (selected from the given set, or in suitable combinations)



video

desired feature positions
current feature positions



the error in the image plane (task space!) drives/controls the motion of the robot



PBVS vs IBVS

PBVS = position-based
visual servoing

video



IBVS = image-based
visual servoing

video



F. Chaumette, INRIA Rennes

reconstructing the instantaneous
(relative) 3D pose of the object

using (four) point features
extracted from the 2D image

...and intermediate 2½-D visual servoing...

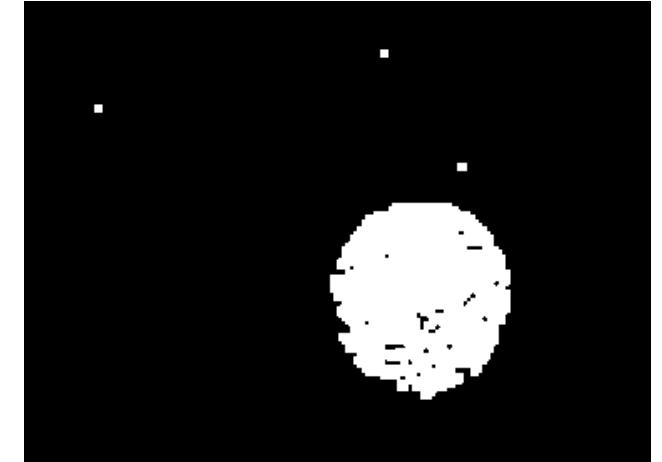
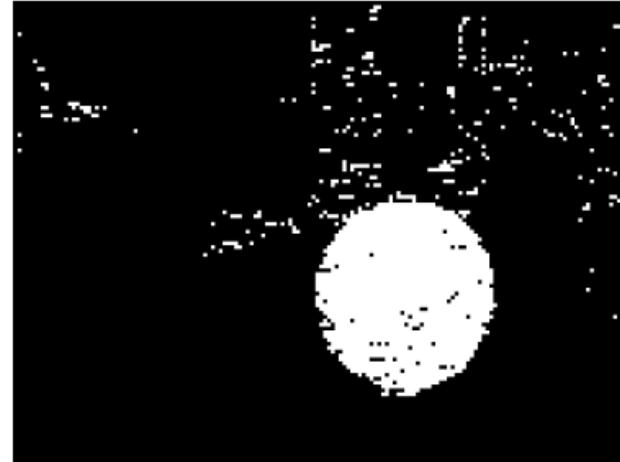
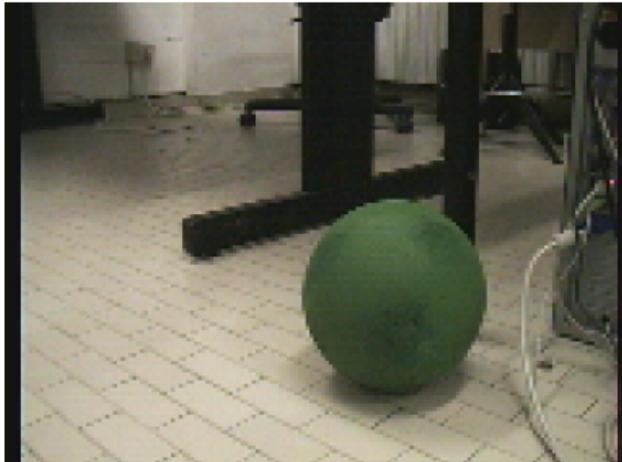


Steps in an IBVS scheme

- **image acquisition**
 - frame rate, delay, noise, ...
- **feature extraction**
 - with image processing techniques (it could be a difficult and time consuming step!)
- **comparison with “desired” feature values**
 - definition of an **error** signal in the **image plane**
- **generation of motion of the camera/robot**
 - perspective transformations
 - differential kinematics of the manipulator
 - control laws of **kinematic** (most often) or **dynamic** type (e.g., PD + gravity cancelation --- **see reference textbook**)



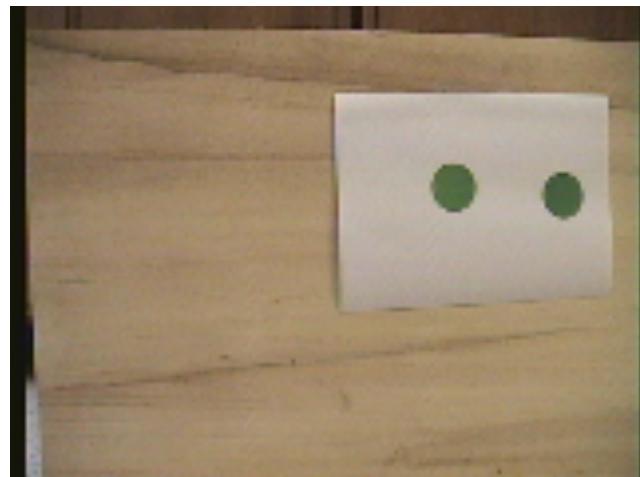
Image processing techniques



binarization in **RGB** space



erosion and dilation



binarization in **HSV** space



dilation

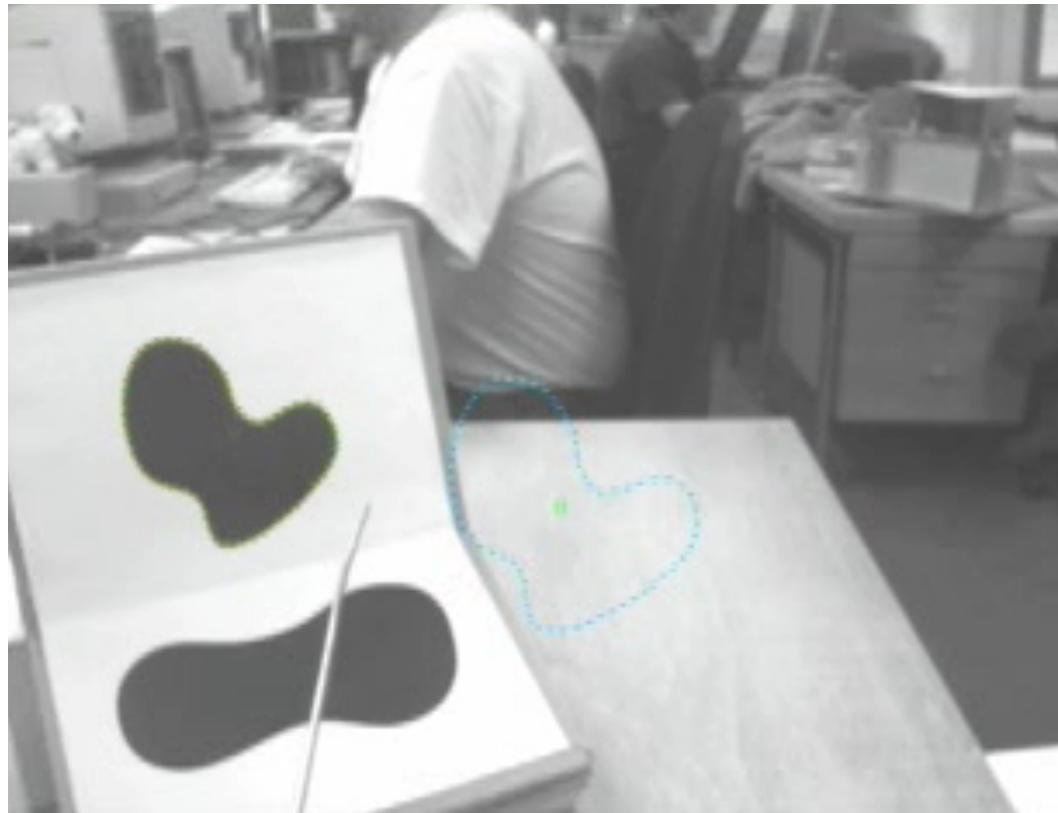


What is a feature?

- **image feature**: any interesting **characteristic** or **geometric structure** extracted from the image/scene
 - points
 - lines
 - ellipses (or any other 2D contour)
- **feature parameter(s)**: any **numerical quantity** associated to the selected feature in the image plane
 - coordinates of a point
 - angular coefficient and offset of a line
 - center and radius of a circle
 - area and center of a 2D contour
 - generalized **moments** of an object in the image
 - can also be defined so as to be scale- and rotation-invariant



Example of IBVS using moments



video

- avoids the problem of “finding **correspondences**” between points
- however, it is not easy to control the motion of all **6 dofs of the camera** when using only moments as features

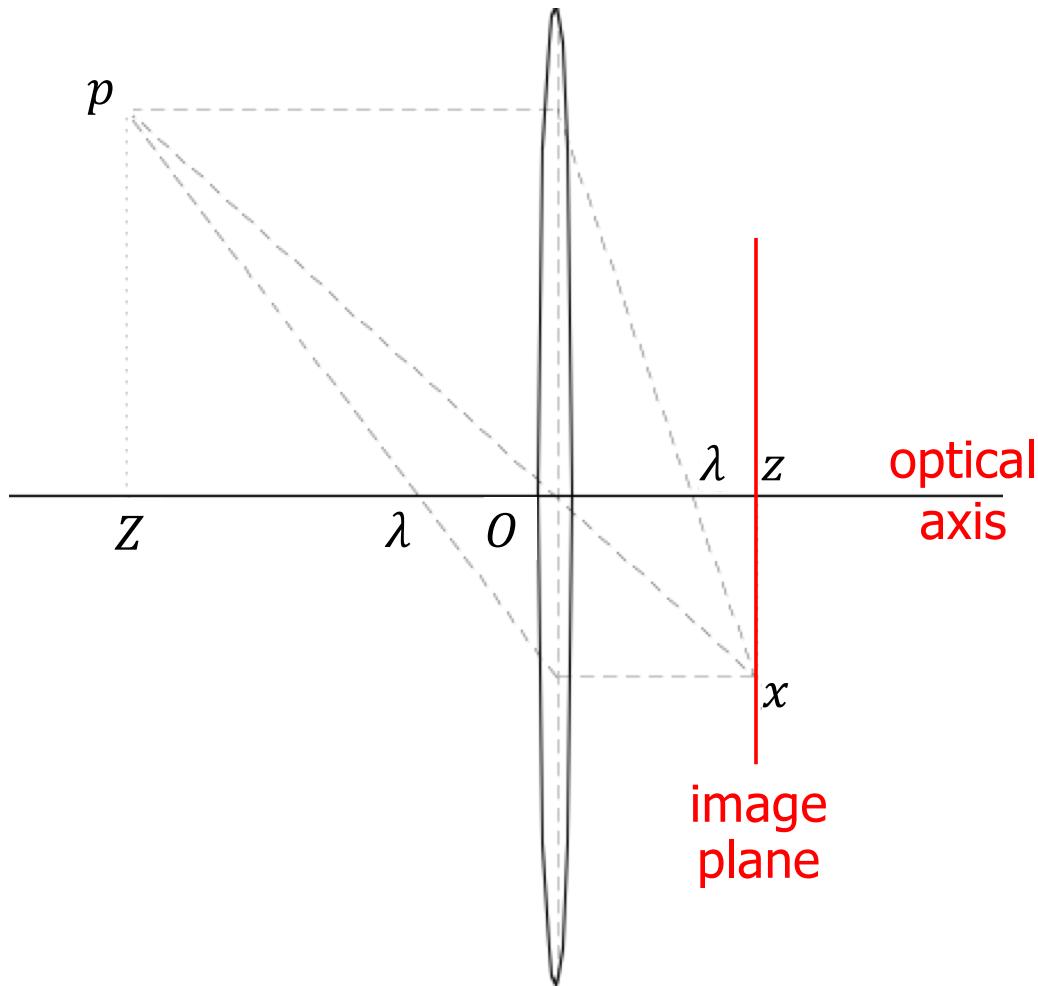


Camera model

- set of **lenses** to focus the incoming light
 - (converging) thin lenses
 - pinhole approximation
 - catadioptric lens or systems (combined with mirrors)
- matrix of light sensitive elements (pixels in **image plane**), possibly selective to **RGB** colors ~ human eye
- **frame grabber** “takes shots”: an electronic device that captures individual, digital still frames as output from an analog video signal or a digital video stream
 - board + software on PC
 - **frame rate** = output frequency of new digital frames
 - it is useful to randomly access a subset (area) of pixels



Thin lens camera model



- geometric/projection optics
- rays parallel to the optical axis are deflected through a point at distance λ (**focal length**)
- rays passing through the optical center O are **not** deflected

fundamental equation
of a thin lens

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{\lambda}$$

} dioptic
(optical)
power

proof? left as exercise ...



Pinhole camera model

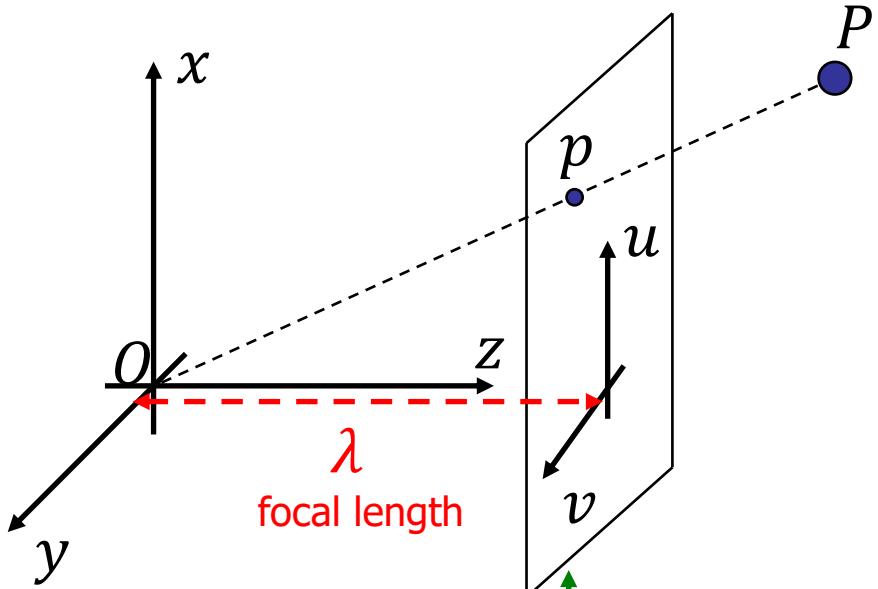


image plane with an array
of light sensitive elements
(actually located symmetrically
to the optical center O , but this
model avoids a change of signs...)

- when thin lens wideness is neglected
- all rays pass through optical center
- from the relations on similar triangles one gets the **perspective equations**

$$u = \lambda \frac{X}{Z} \quad v = \lambda \frac{Y}{Z}$$

to obtain these from discrete pixel values, an **offset** and a **scaling factor** should be included in each direction

sometimes **normalized** values (u', v') are used
(dividing by the focal length)

with

$$P = (X, Y, Z)$$

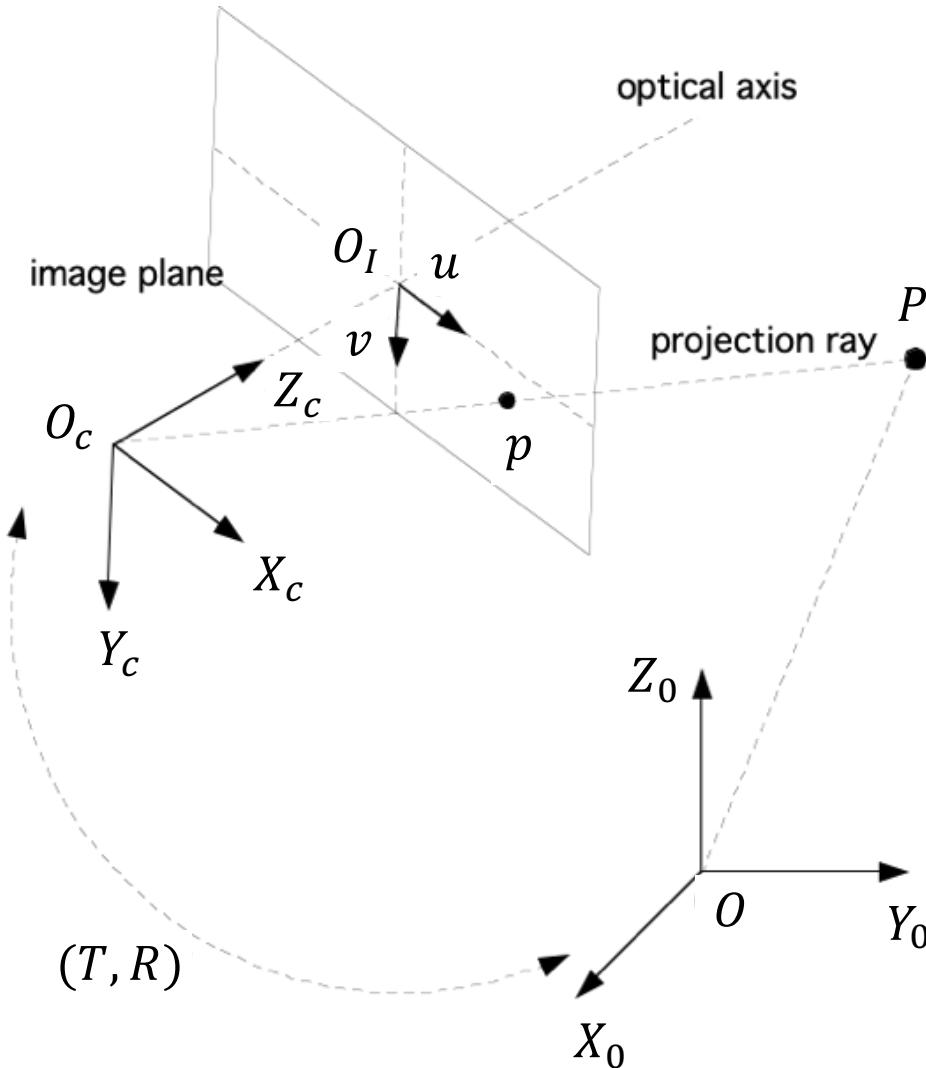
Cartesian point
(in camera frame)

$$p = (u, v, \lambda)$$

representative point
on the image plane



Reference frames of interest



- **absolute reference frame**
 $\mathcal{F}_0: \{O, \vec{X}_0, \vec{Y}_0, \vec{Z}_0\}$
- **camera reference frame**
 $\mathcal{F}_c: \{O_c, \vec{X}_c, \vec{Y}_c, \vec{Z}_c\}$
- **image plane reference frame**
 $\mathcal{F}_I: \{O_I, \vec{u}, \vec{v}\}$
- **position/orientation of \mathcal{F}_c w.r.t. \mathcal{F}_0**
 (T, R)
 (translation, rotation)



Interaction matrix

- given a set of feature(s) parameters $f = [f_1 \quad \cdots \quad f_k]^T \in \mathbb{R}^k$
- we look for the **(kinematic) differential relation** between **motion imposed to the camera** and **motion of features** on the image plane

$$\dot{f} = J(\cdot) \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

- $(V, \Omega) \in \mathbb{R}^6$ is the camera linear/angular velocity, a vector expressed in \mathcal{F}_c
- $J(\cdot)$ is a $k \times 6$ matrix, known as **interaction matrix**
- in the following, we consider mainly **point features**
 - other instances (areas, lines, ...) can be treated as extensions of the presented analysis



Computing the interaction matrix point feature, pinhole model

- from the perspective equations $u = \lambda \frac{X}{Z}$, $v = \lambda \frac{Y}{Z}$, we have

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{z} & 0 & -\frac{u}{z} \\ 0 & \frac{\lambda}{z} & -\frac{v}{z} \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = J_1(u, v, \lambda) \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix}$$

- the velocity $(\dot{X}, \dot{Y}, \dot{Z})$ of a point P in frame \mathcal{F}_c is **actually due to** the roto-translation (V, Ω) of the camera (P is assumed fixed in \mathcal{F}_0)
- kinematic relation between $(\dot{X}, \dot{Y}, \dot{Z})$ and (V, Ω)

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = -V - \Omega \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Note: ALL quantities are expressed in the camera frame \mathcal{F}_c
without explicit indication of subscripts



Computing the interaction matrix (cont)

point feature, pinhole model

- the last equation can be expressed in matrix form

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix} = J_2(X, Y, Z) \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

- combining things, the **interaction matrix** for a **point feature** is

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = J_1 J_2 \begin{bmatrix} V \\ \Omega \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{Z} & 0 & \frac{u}{Z} & \frac{uv}{\lambda} & -\left(\lambda + \frac{u^2}{\lambda}\right) & v \\ 0 & -\frac{\lambda}{Z} & \frac{v}{Z} & \lambda + \frac{v^2}{\lambda} & -\frac{uv}{\lambda} & -u \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

$$= J_p(u, v, Z) \begin{bmatrix} V \\ \Omega \end{bmatrix}$$



p = point (feature)

here, λ is assumed to be known



Comments

- the **interaction matrix** in the map

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = J_p(u, v, Z) \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

- depends on the actual value of the **feature** and on its **depth Z**
- since $\dim \ker J_p = 4$, there exist ∞^4 motions of the camera that are unobservable (for this feature) on the image
 - for instance, a translation along the projection ray
- when **more point features** are considered, the associated interaction matrices are stacked one on top of the other
 - p point features: the interaction matrix is $k \times 6$, with $k = 2p$



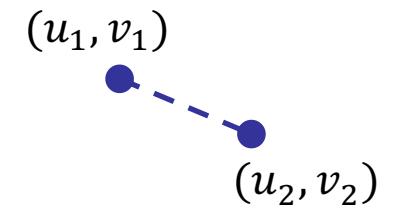
Other examples of interaction matrices

- distance between two point features

$$d = \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}$$

$$\dot{d} = \frac{1}{d} [u_1 - u_2 \quad v_1 - v_2 \quad u_2 - u_1 \quad v_2 - v_1] \begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \end{bmatrix}$$

$$= J_p(u_1, u_2, v_1, v_2) \begin{bmatrix} J_{p1}(u_1, v_1, Z_1) \\ J_{p2}(u_2, v_2, Z_2) \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix}$$



- image moments

$$m_{ij} = \iint_{\mathcal{R}(t)} x^i y^j dx dy$$

$\mathcal{R}(t)$ region of the image plane
occupied by the object

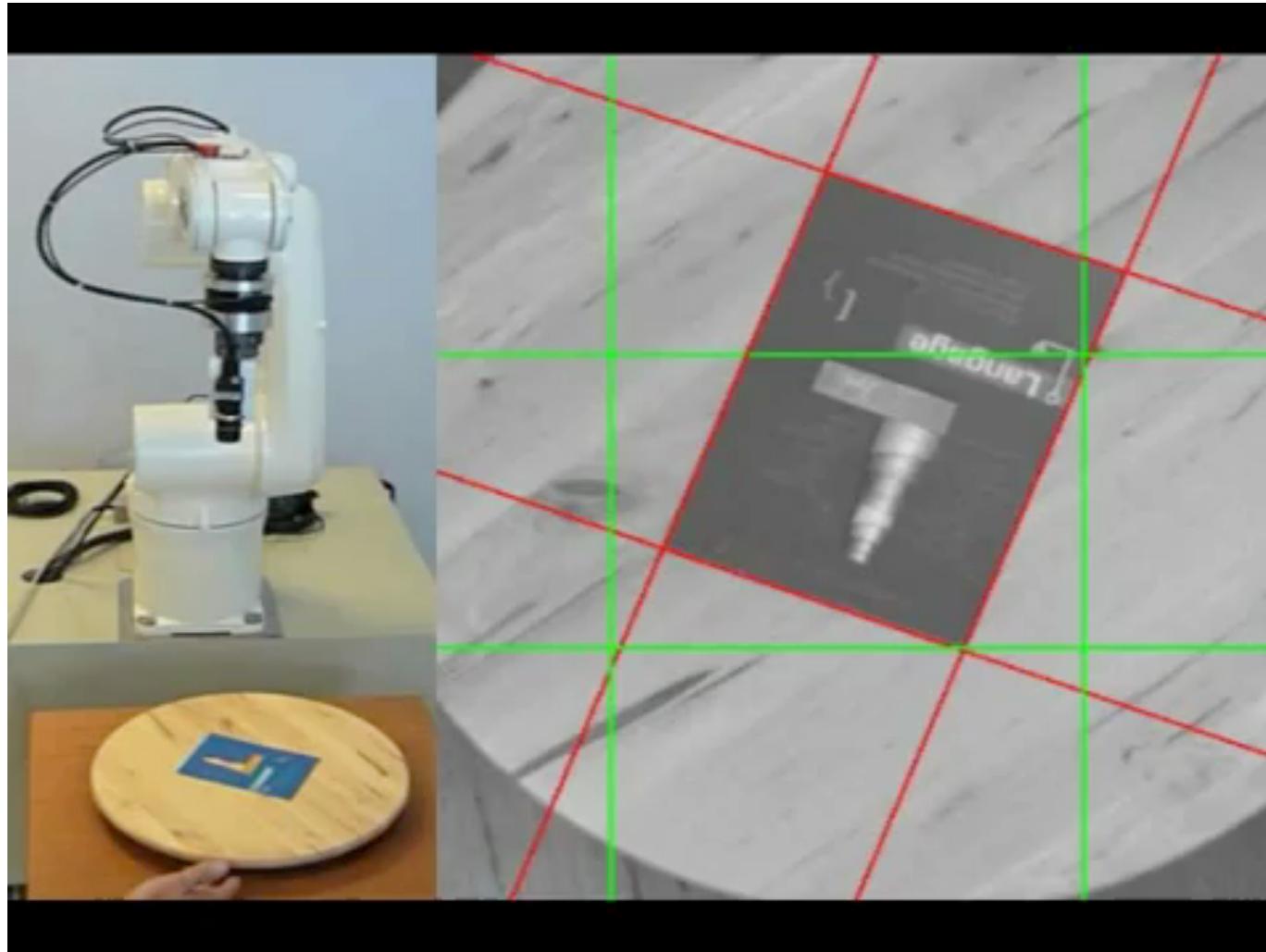
- useful for representing also qualitative geometric information ([area](#), [center](#), [orientation of an approximating ellipse](#), ...)
- by using Green formulas and the interaction matrix of a generic point feature

$$\rightarrow \dot{m}_{ij} = L_{ij} \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

(see F. Chaumette: "Image moments:
A general and useful set of features for visual servoing",
IEEE Trans. on Robotics, August 2004)



IBVS with straight lines as features



F. Chaumette, INRIA Rennes



Robot differential kinematics

- **eye-in-hand** case: camera is mounted on the **end-effector** of a manipulator arm (with fixed base or on a wheeled mobile base)
 - the motion (V, Ω) to be imposed to the camera coincides with the desired **end-effector linear/angular velocity** and is realized by choosing the manipulator **joint velocity** (or, more in general, the feasible **velocity commands** of a **mobile** manipulator)

$$\begin{bmatrix} V \\ \Omega \end{bmatrix} = J_m(q)\dot{q} = J_m(q)u$$

velocity
control
input

↑
Geometric Jacobian
of a manipulator

↑
... or the NMM Jacobian
of a mobile manipulator

for consistency with the previous interaction matrix, these Jacobians must be expressed in the camera frame \mathcal{F}_c

with $q \in \mathbb{R}^n$ being the robot configuration variables

- in general, an hand-eye calibration is needed for this Jacobian



Image Jacobian

- combining the step involved in the passage from the **velocity of point features** on the image plane to the **joint velocity/feasible velocity commands of the robot**, we finally obtain

$$\dot{f} = J_p(f, Z)J_m(q)u = J(f, Z, q)u$$

- matrix $J(f, Z, q)$ is called the **Image Jacobian** and plays the same role of a classical robot Jacobian
- we can thus apply one of the many standard kinematic (or even dynamics-based) control techniques for robot motion
- the (controlled) **output** is given by the vector of features in the image plane (the **task space!**)
- the **error** driving the control law is defined in this task space



Kinematic control for IBVS

- defining the error vector as $e = f_d - f$, the general choice

$$u = J^{\#}(\dot{f}_d + ke) + (I - J^{\#}J)u_0$$

minimum norm solution

term in $\ker J$ does not “move” the features

will **exponentially stabilize** the task error to zero (up to singularities, limit joint range/field of view, features exiting the image plane, ...)

- in the redundant case, vector u_0 (projected in the image Jacobian null space) can be used for optimizing behavior/criteria
- the error feedback signal **depends only on feature values**
- there is still a dependence of J on the **depths** Z of the features
 - use the constant and “known” values at the final **desired pose**
 - or, **estimate on line** their current values using a dynamic observer

$$J(f, Z^*, q)$$



Example with NMM

- mobile base (unicycle) + 3R manipulator
- eye-in-hand configuration
- 5 commands
 - linear and angular velocity of mobile base
 - velocities of the three manipulator joints
- task specified by 2 point features → 4 output variables



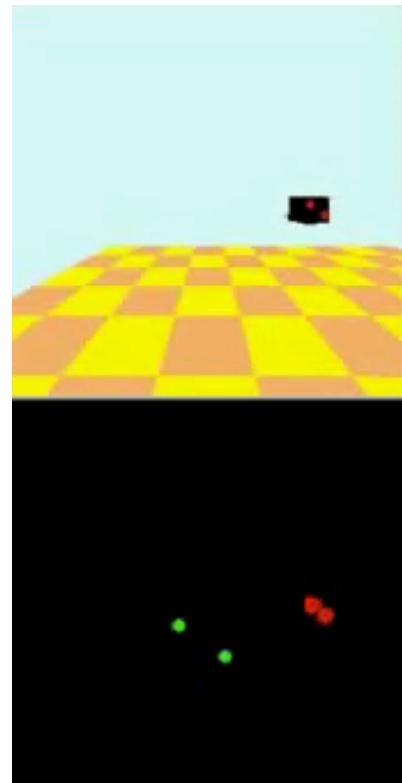
$$f = [f_{u1} \quad f_{v1} \quad f_{u2} \quad f_{v2}]^T$$

- $5 - 4 = 1$ degree of redundancy (w.r.t. the task)

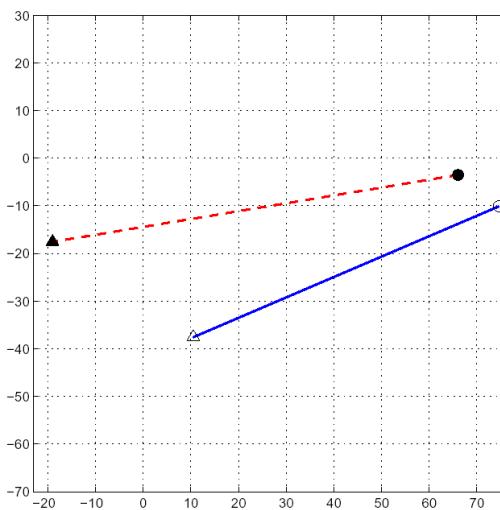


Simulation

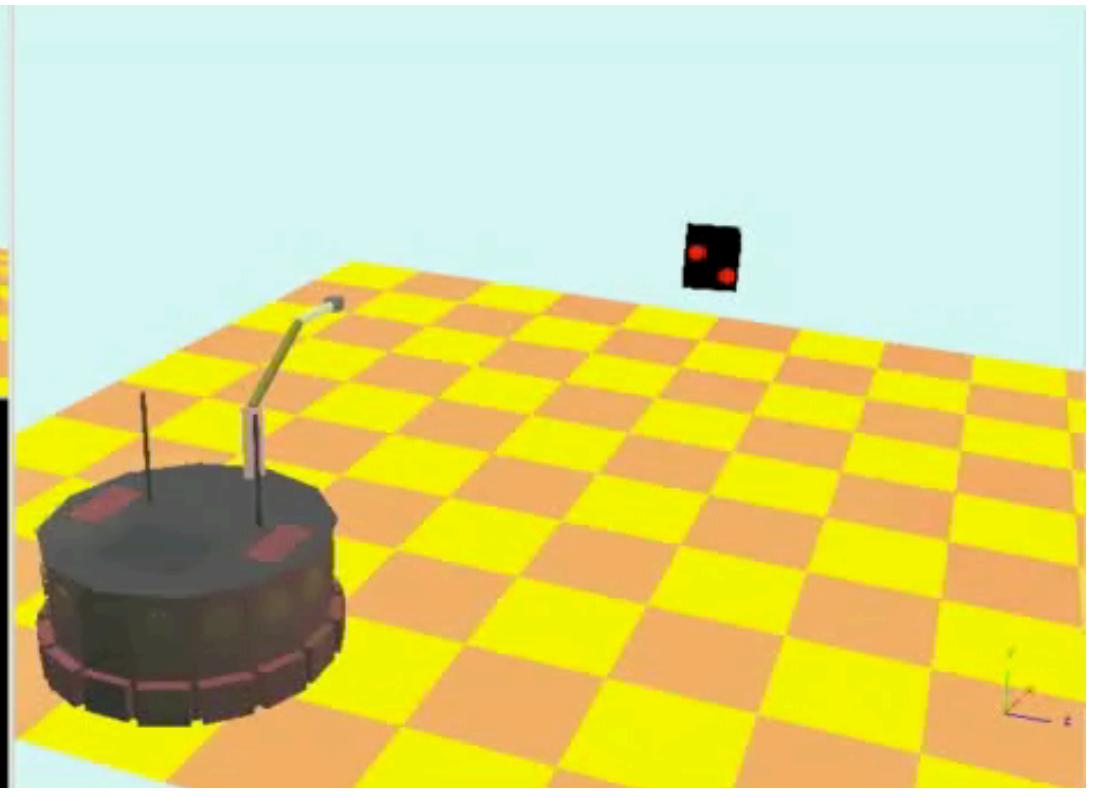
view from
camera



processed
image



behavior of
the 2 features



video

- simulation in Webots
- current value of Z is supposed to be known
- diagonal task gain matrix k (decoupling!)
- “linear paths” of features motion on the image plane



IBVS control with Task Sequencing

- approach: **regulate only one (some) feature** at the time, while keeping “**fixed**” the others by **unobservable motions** in the image plane

- Image Jacobians of single point features (e.g., $p = 2$)

$$\dot{f}_1 = J_1 u, \quad \dot{f}_2 = J_2 u$$

- the first feature f_1 is regulated during a first phase by using

$$u = J_1^\# k_1 e_1 + (I - J_1^\# J_1) u_0$$

- feature f_2 is then regulated by a command in the null-space of the first task

$$u = (I - J_1^\# J_1) J_2^T k_2 e_2$$

- in the first phase there are **two (more) degrees of redundancy** w.r.t. the “**complete**” case, which can be used, e.g., for improving robot alignment to the target

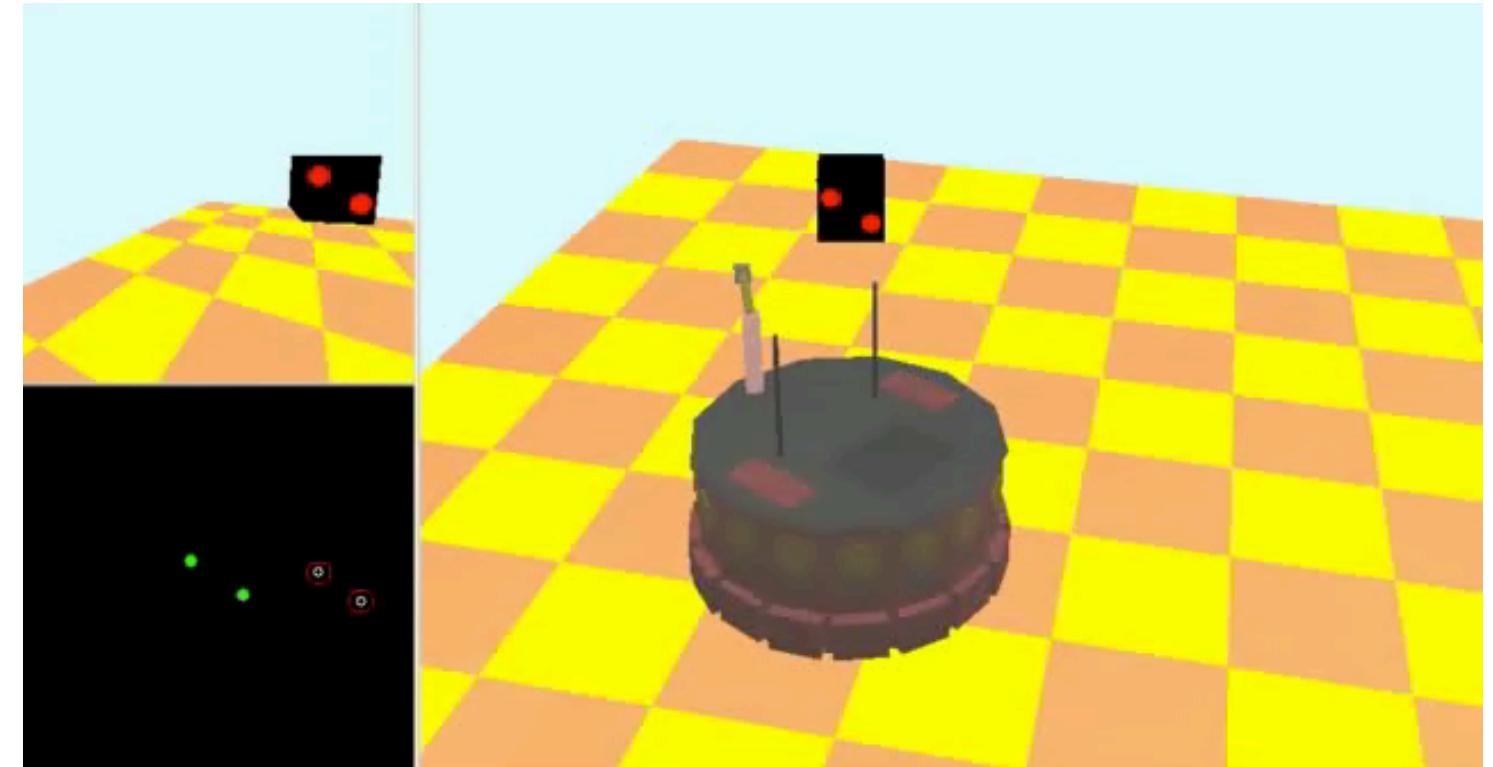
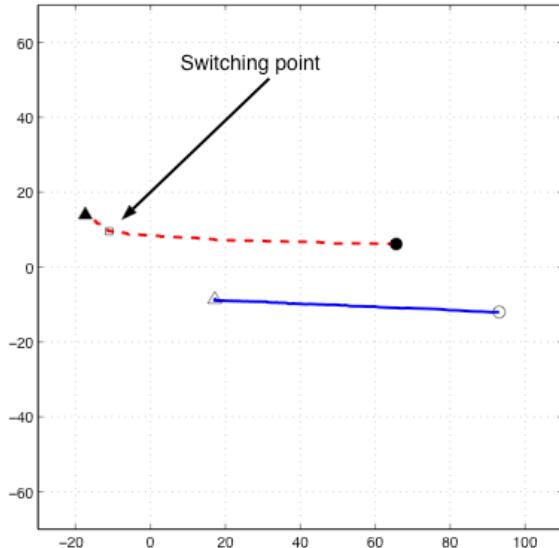
- if the complete case is **not** redundant: singularities are typically met without Task Sequencing, but possibly prevented with TS ...



Simulation with TS scheme

mobile base (unicycle) + polar 2R arm: Image Jacobian is square (4×4)

point feature 1
(regulated in
first phase)
point feature 2
(in second phase)



behavior of
the 2 features

- simulation in Webots
- current value of Z is supposed to be known



Experiments

Magellan (unicycle) + pan-tilt camera (same mobility of a polar 2R robot)



Video attachment to ICRA'08 paper

Visual Servoing with Exploitation of Redundancy:
An Experimental Study

A. De Luca

M. Ferri

G. Oriolo

P. Robuffo Giordano

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"

video

- comparison between **Task Priority (TP)** and **Task Sequencing (TS)** schemes
- both can handle singularities (of Image Jacobian) that are possibly encountered
- camera frame rate = 7 Hz



In many practical cases...

- the uncertainty in a number of relevant data may be large
 - focal length λ (**intrinsic** camera parameters)
 - **hand-eye calibration** (**extrinsic** camera parameters)
 - **depth** Z of point features
 -
- one can only compute an **approximation** of the Image Jacobian (both in its **interaction matrix** part, as well as in the **robot Jacobian** part)
- in the closed loop, error dynamics on features becomes
$$\dot{e} = -J \hat{J}^{\#} K e$$
 - **ideal** case: $J J^{\#} = I$ **real** case: $J \hat{J}^{\#} \neq I$
- it is possible to show that a **sufficient condition** for **local** convergence of the error to zero is

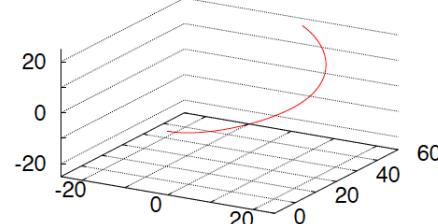
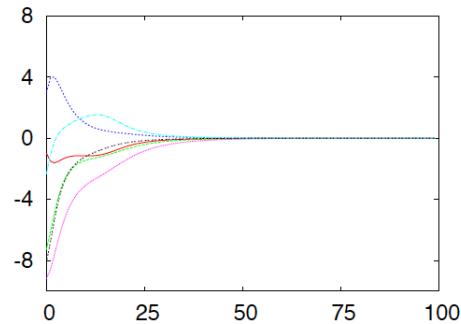
$$J \hat{J}^{\#} > 0$$



Approximate Image Jacobian

- use a constant Image Jacobian $\hat{J}(Z^*)$ that is computed at the desired target s^* (with a known, fixed depth Z^*)

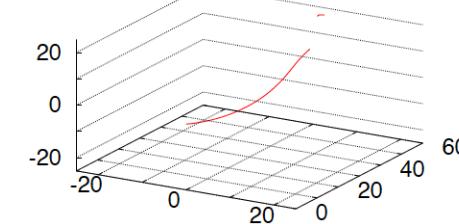
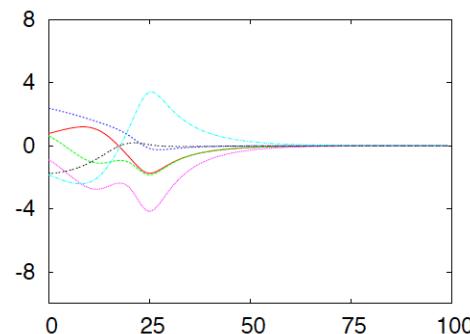
$$\dot{q} = J^\#(Z)Ke$$



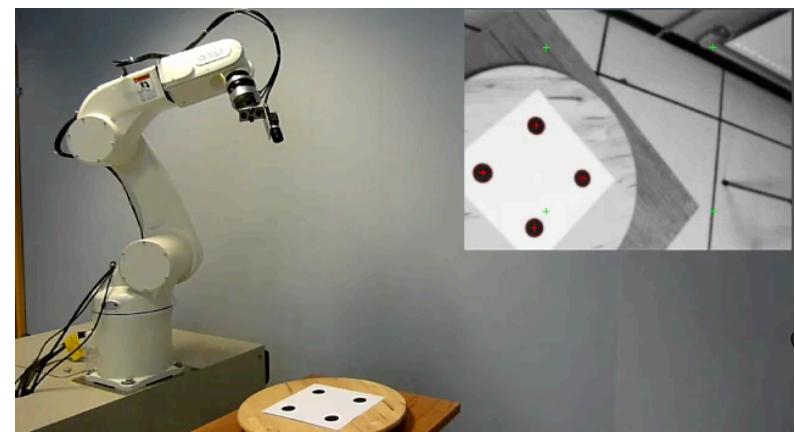
video



$$\dot{q} = \hat{J}^\#(Z^*)Ke$$



video



F. Chaumette, INRIA Rennes



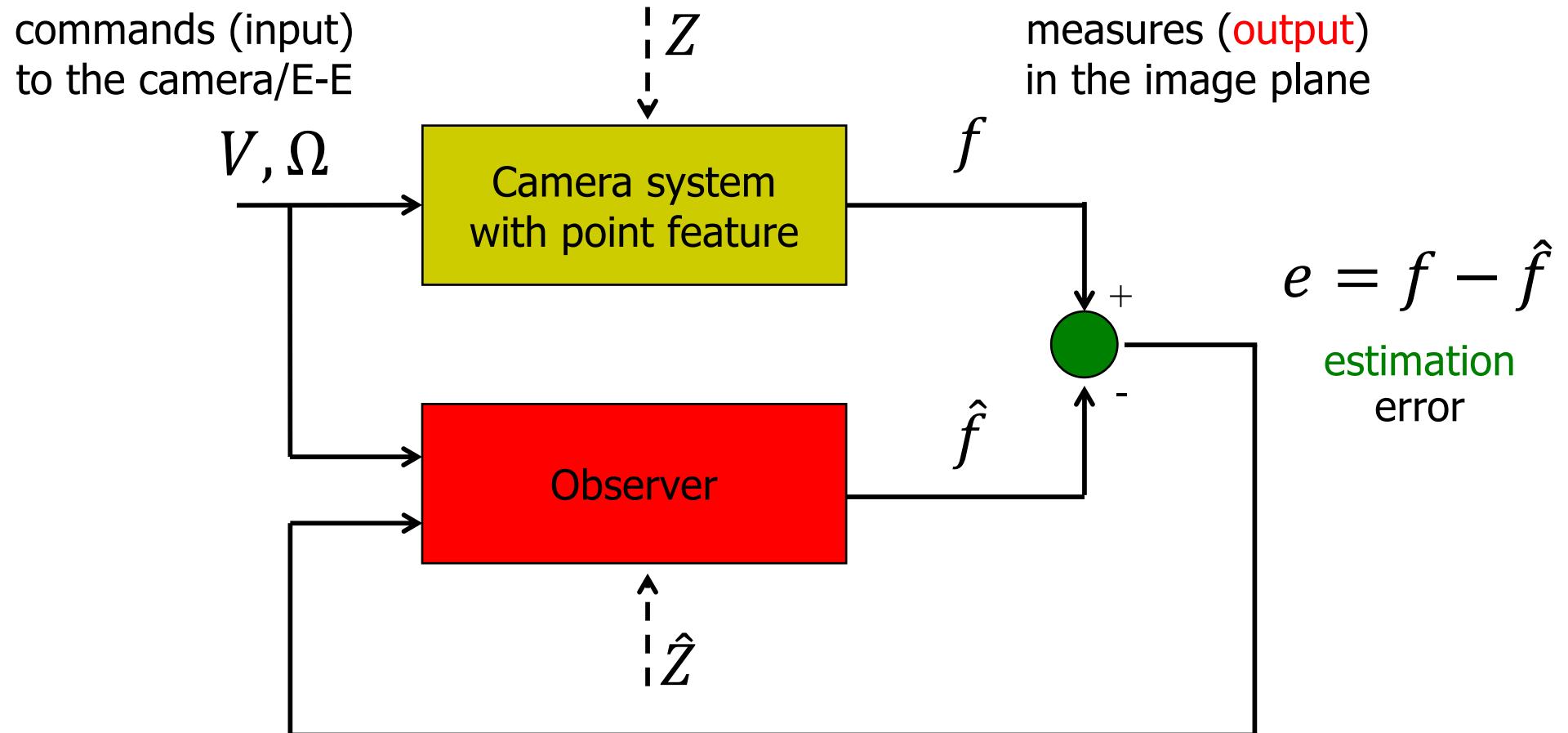
An observer of the depth Z

- it is possible to **estimate on line** the current value (possibly time-varying) of the depth Z , **for each** considered point feature, using a dynamic **observer**
- define $x = [u \quad v \quad 1/Z]^T$, $\hat{x} = [\hat{u} \quad \hat{v} \quad 1/\hat{Z}]^T$ as **current state** and **estimated state**, and $y = [u \quad v]^T$ as **measured output**
- a (nonlinear) observer of x with input $u_c = [V \quad \Omega]^T$
$$\dot{\hat{x}} = \alpha(\hat{x}, y)u_c + \beta(\hat{x}, y, u_c)$$
guarantees $\lim_{t \rightarrow \infty} \|x(t) - \hat{x}(t)\| = 0$ **provided that**
 - **linear velocity** of the camera is **not** zero
 - the linear velocity vector **is not aligned** with the projection ray of the considered point feature

⇒ these are **persistent excitation** conditions (\sim observability conditions)



Block diagram of the observer





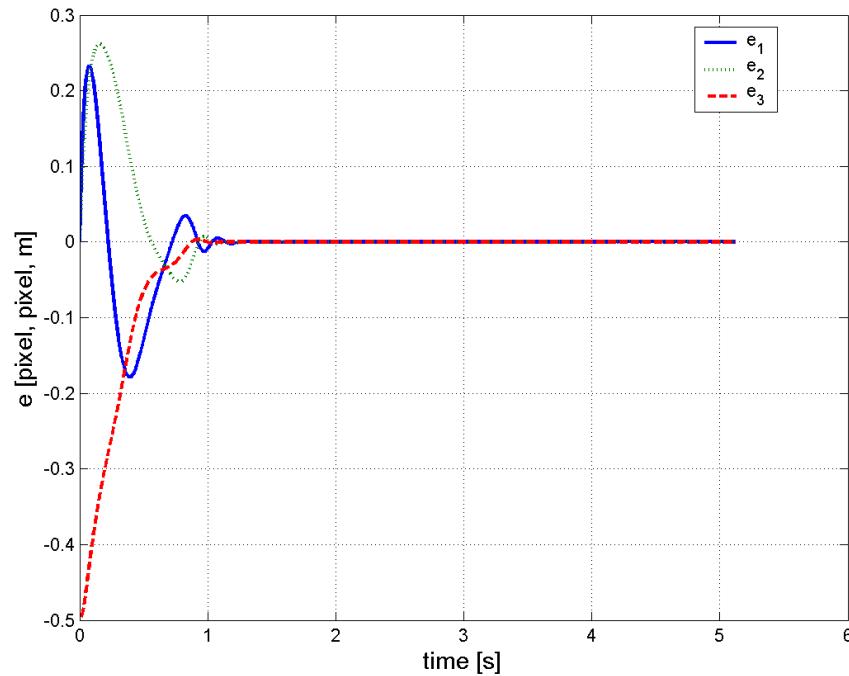
Results on the estimation of Z

real and estimated initial state

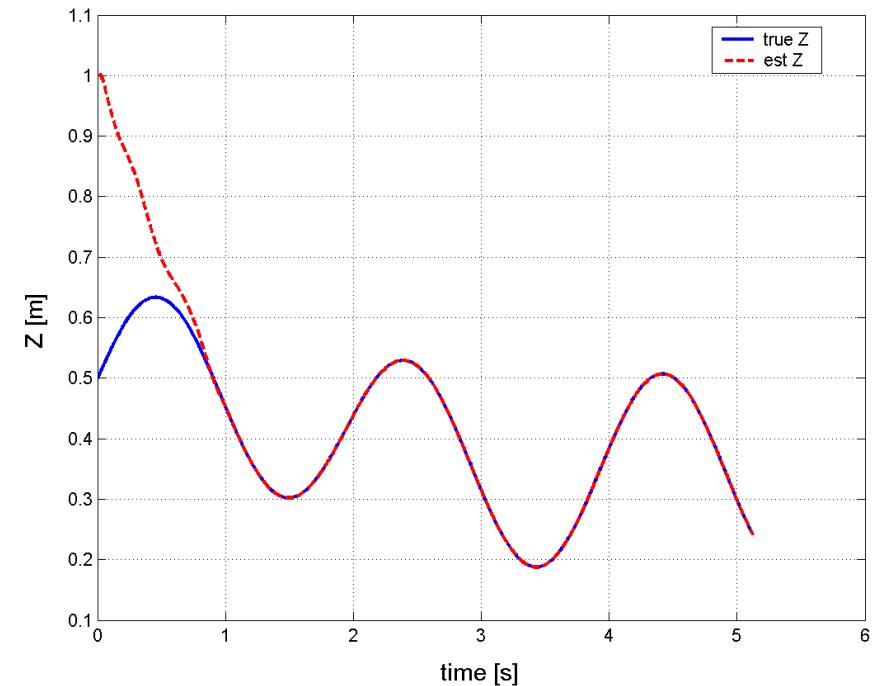
$$x(t_0) = [10 \quad -10 \quad 2]^T$$
$$\hat{x}(t_0) = [10 \quad -10 \quad 1]^T$$

$$v_x(t) = 0.1 \cos 2\pi t$$
$$v_z(t) = 0.5 \cos \pi t$$
$$\omega_x(t) = 0.6 \cos(\pi/2)t$$
$$\omega_z(t) = 1$$

open-loop
commands



estimation errors $e(t)$



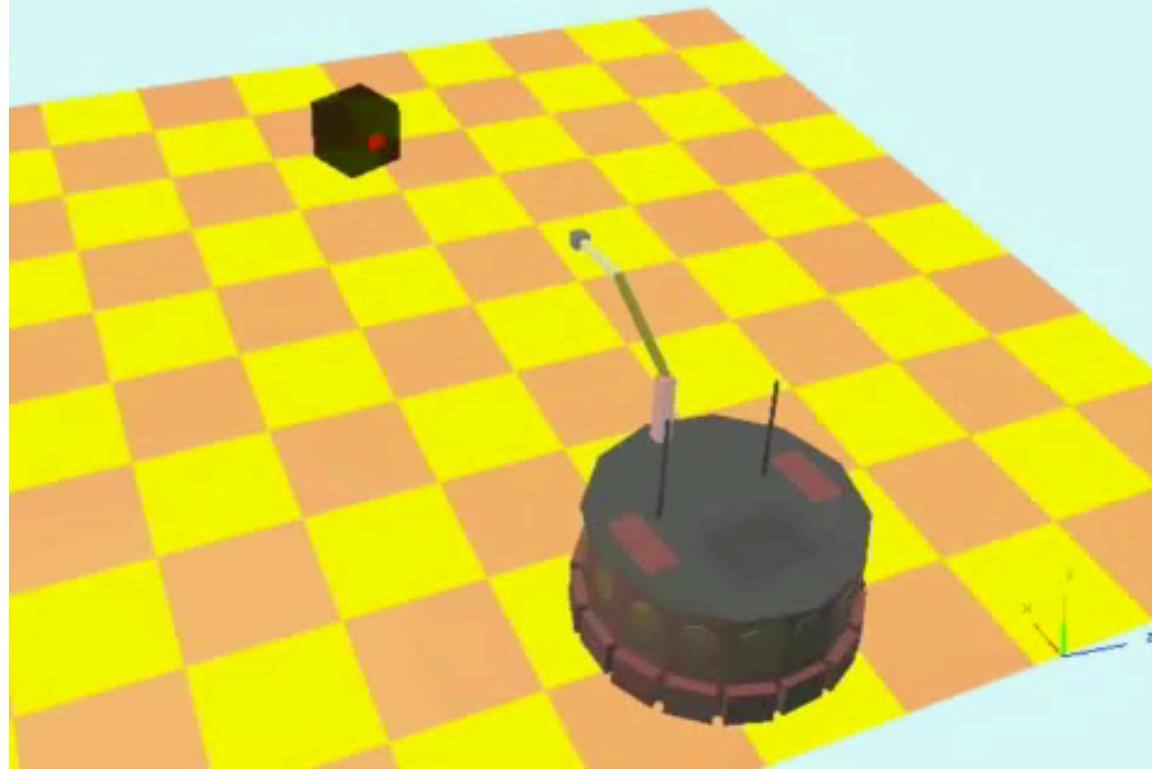
$Z(t)$ and $\hat{Z}(t)$



Simulation of the observer with stepwise displacements of the target

[video](#)

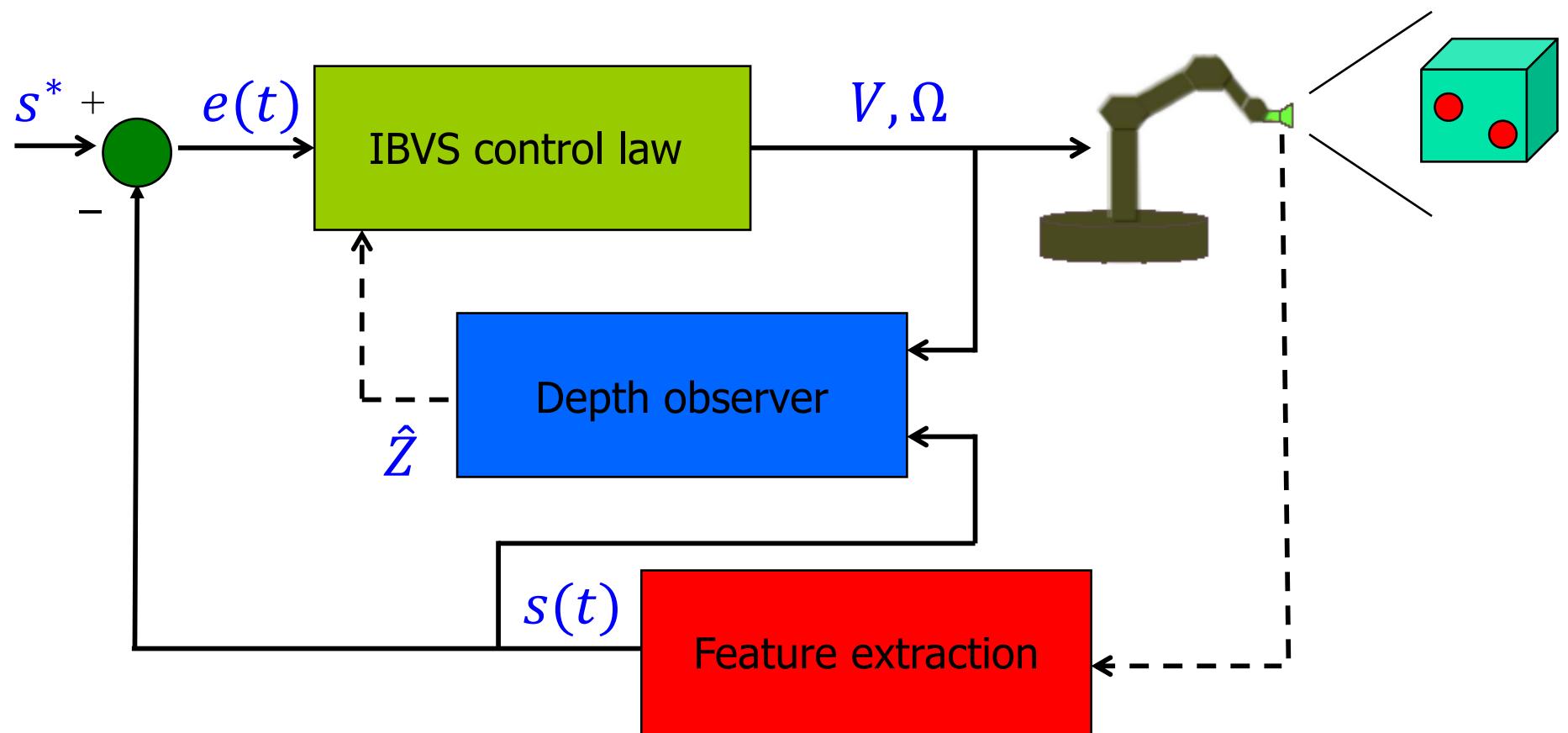
The crosshair represents the estimated value of the depth Z





IBVS control with observer

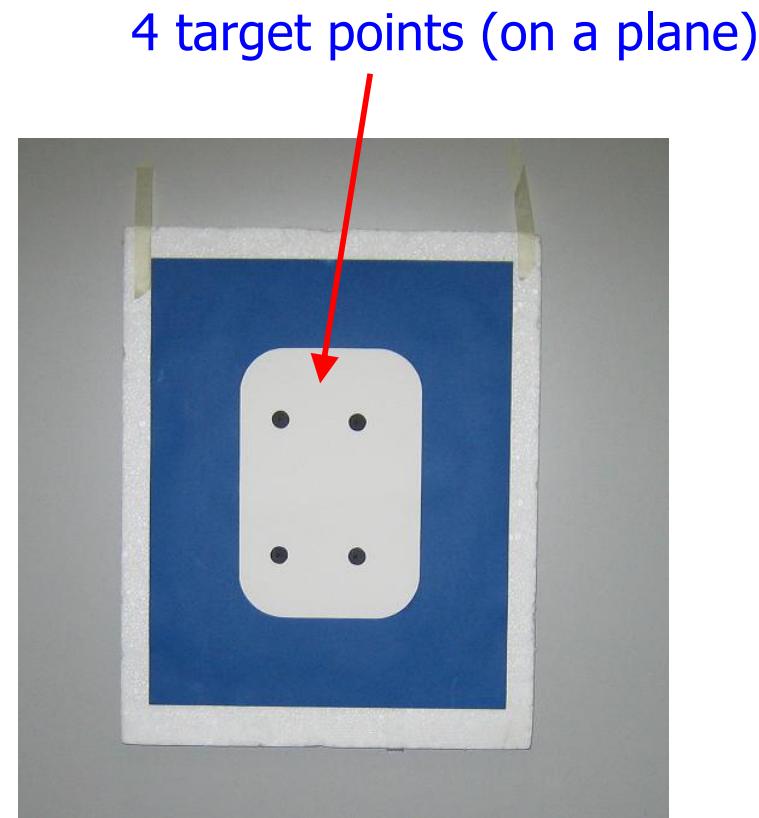
- the depth observer can be easily integrated on line with any IBVS control scheme





Experimental set-up

- visual servoing with fixed camera on a skid-steering mobile robot
- **very rough** eye-robot calibration
- **unknown** depths of target points (**estimated on line**)



a “virtual” 5th feature is also used
as the **average** of the four point features



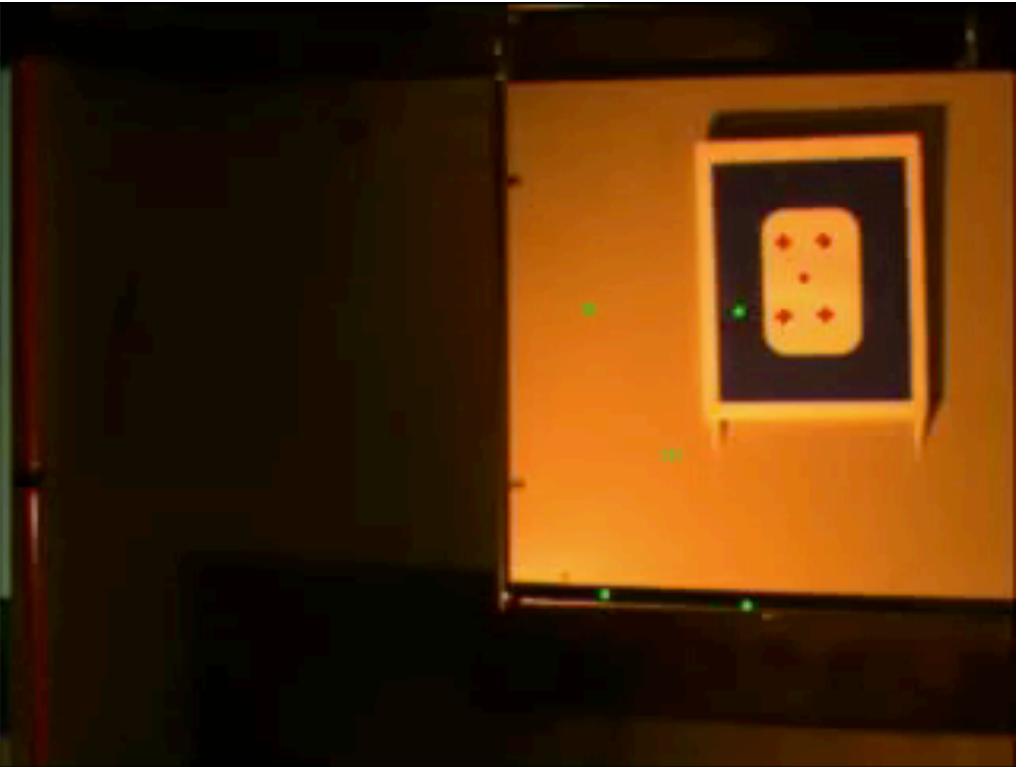
Experiments

- motion of features on the image plane is not perfect...
- the visual **regulation** task is however correctly **realized**

external view



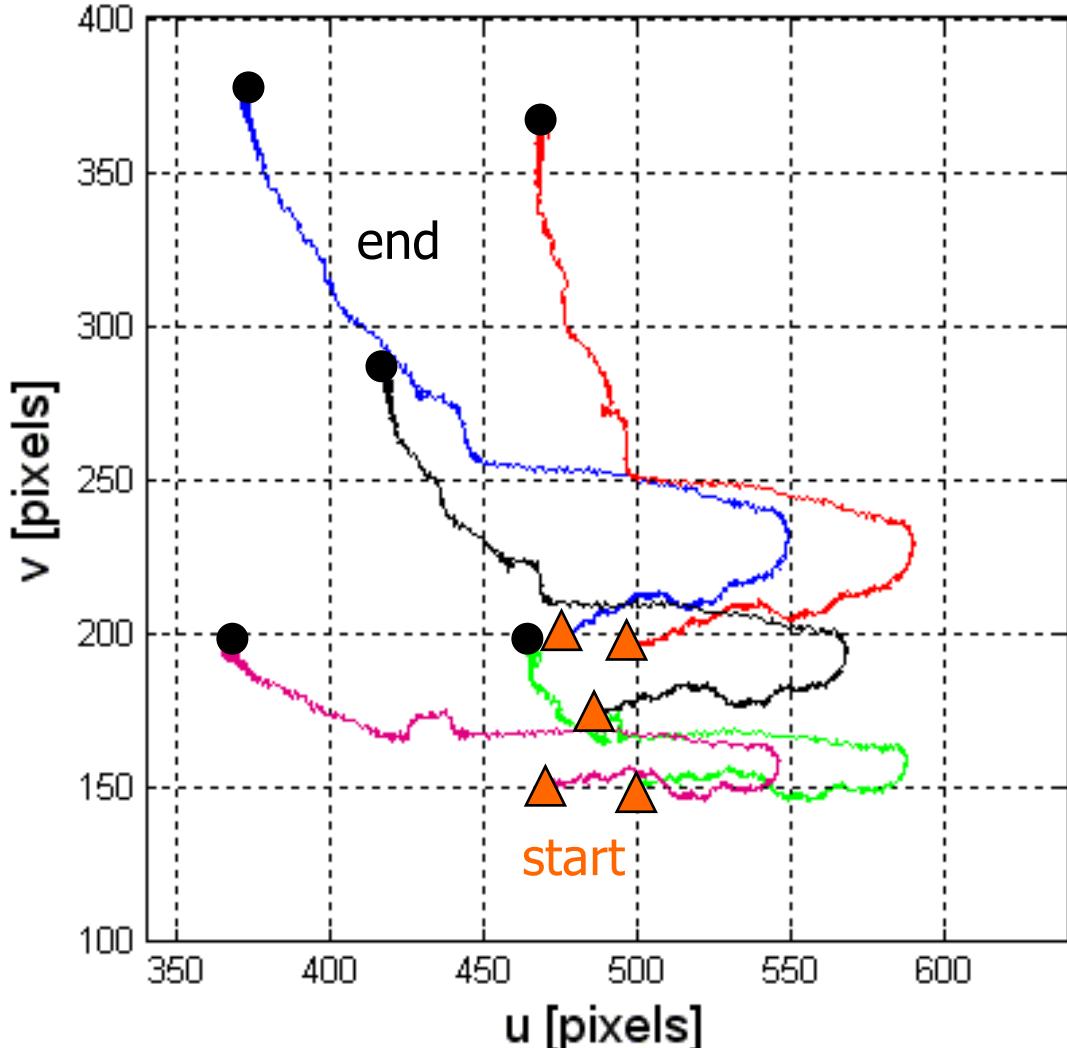
camera view



video (c/o Università di Siena)



Evolution of features



- motion of the 5 features (including the average point) in the image plane
- toward end, motion is \approx linear since the depth observers have already converged to the true values
- computed Image Jacobian is close to the actual one

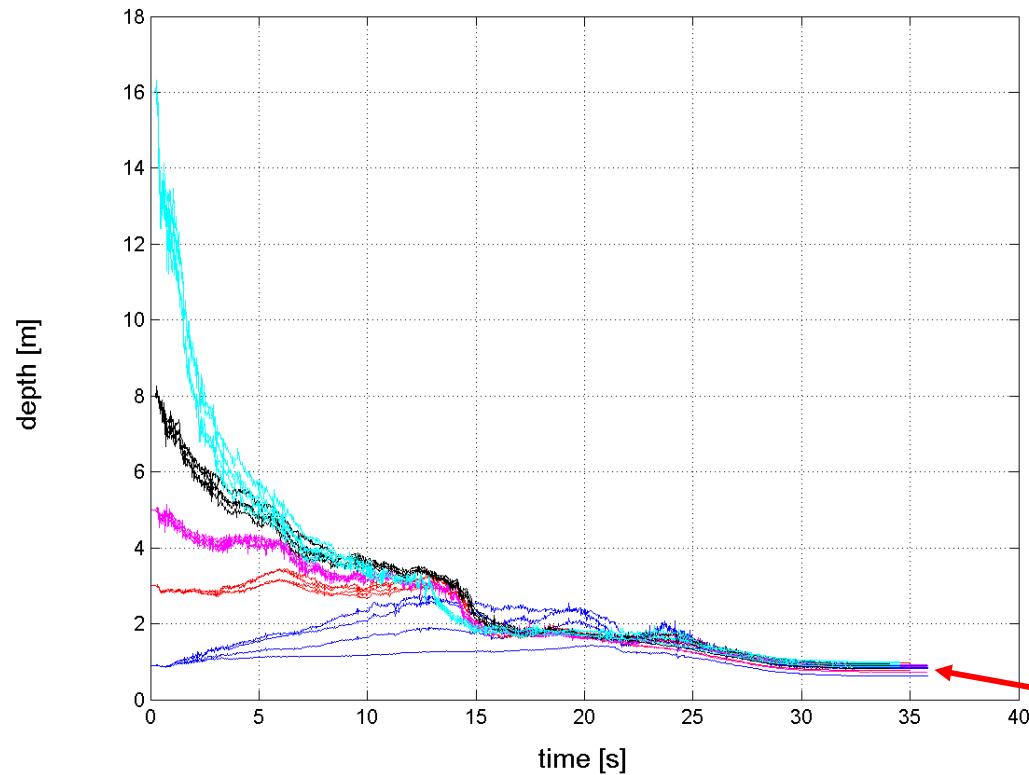
$$\hat{Z} \rightarrow Z \Rightarrow \hat{J} \rightarrow J$$

- “true” initial and final depths
- $$Z(t_0) \cong 4 \text{ m} \quad Z_d \cong 0.9 \text{ m}$$



Experiments with the observer

- the same task was executed with **five different initializations** for the depth observers, ranging between **0.9 m** (= true depth in the **final** desired pose) and **16 m** (much larger than in the true **initial** pose)



- initial values of depth estimates in the five tests

$$\left\{ \begin{array}{l} \hat{Z}_1(t_0) = 16 \text{ m} \\ \hat{Z}_2(t_0) = 8 \text{ m} \\ \hat{Z}_3(t_0) = 5 \text{ m} \\ \hat{Z}_4(t_0) = 3 \text{ m} \\ \hat{Z}_5(t_0) = 0.9 \text{ m} \end{array} \right. \quad \begin{array}{c} \text{cyan} \\ \text{black} \\ \text{magenta} \\ \text{red} \\ \text{blue} \end{array}$$

- true depths in initial pose

$$Z(t_0) \cong 4 \text{ m}$$

- true depths in final pose

$$Z_d \cong 0.9 \text{ m}$$

the evolutions of the **estimated** depths for the 4 point features



Visual servoing with Kuka robot set-up

- Kuka KR5 sixx R650 manipulator (6 revolute joints, spherical wrist)
- Point Grey Flea^{©2} CCD camera, eye-in-hand (mounted on a support)
- Kuka KR C2sr low-level controller (RTAI Linux and Sensor Interface)
- image processing and visual servoing on PC (Intel Core2 @2.66GHz)
- high-level control data exchange every **12ms** (via UDP protocol)



@ DIAG Robotics Laboratory

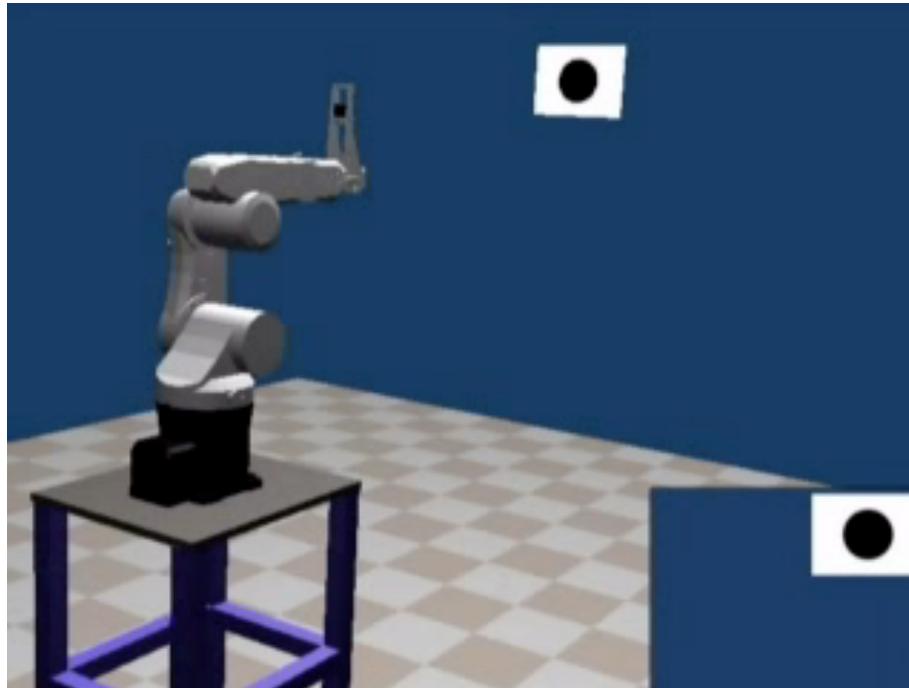


Visual servoing with Kuka robot

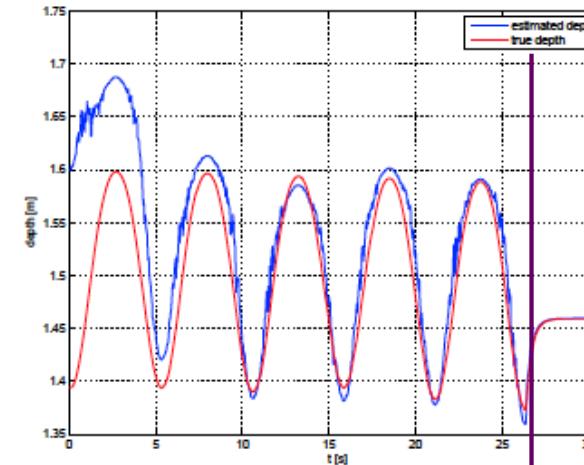
simulation and experiment with the observer

- depth estimation for a **fixed target** (single point feature)
- **simulation** using Webots first, then **experimental** validation
- **sinusoidal motion** of four robot joints so as to provide sufficient excitation

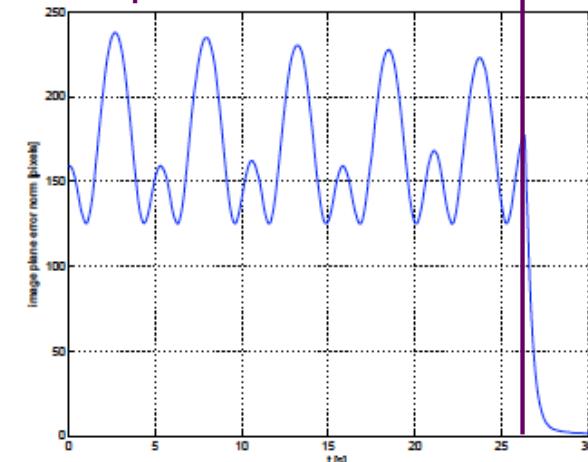
video



Webots simulation



Kuka experiment



true and
estimated depth

norm of
centering error
of feature point
[in pixels]

goes to zero
once control
is activated



Visual servoing with Kuka robot tracking experiment

- tracking a (slowly) time-varying target by visual servoing, including the depth observer (after convergence)



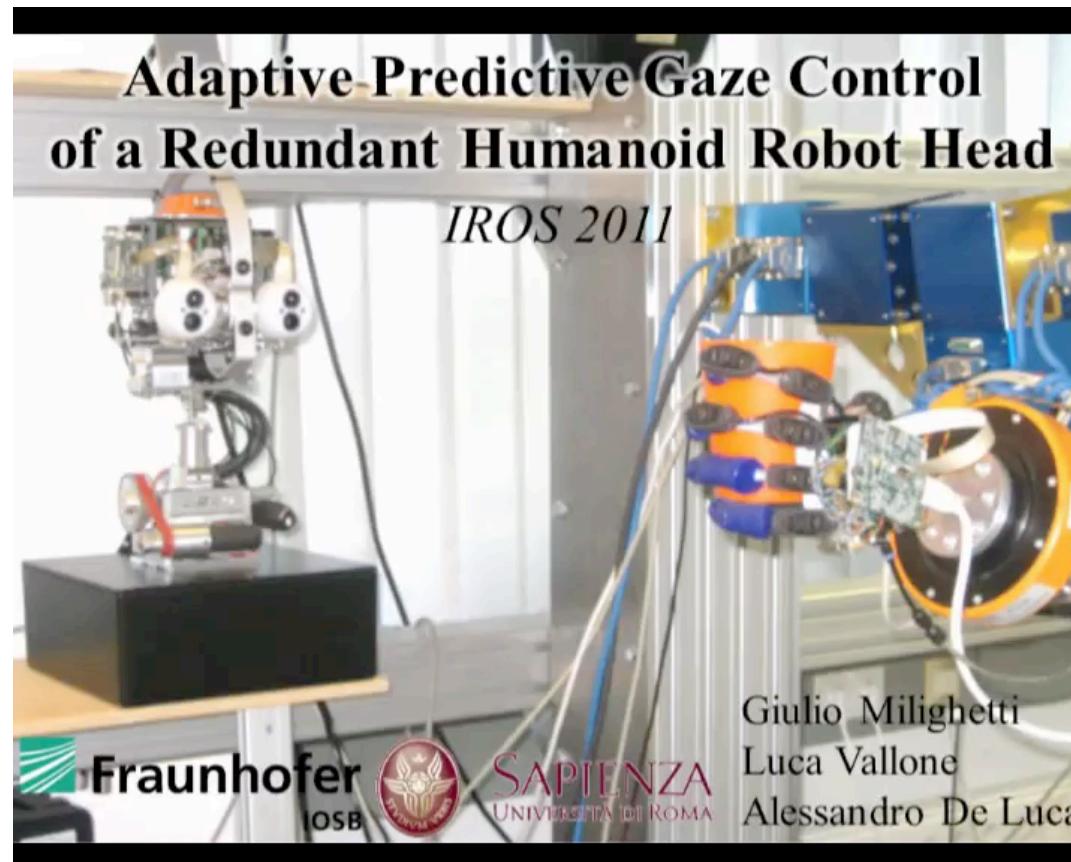
video



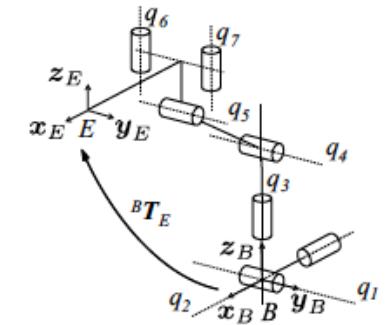
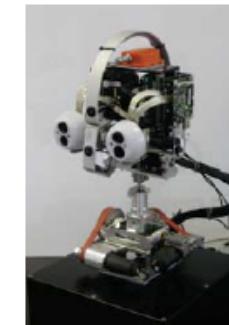
Gazing with humanoid head

stereo vision experiment

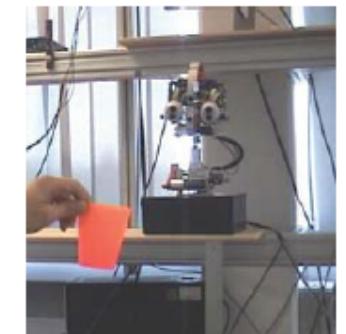
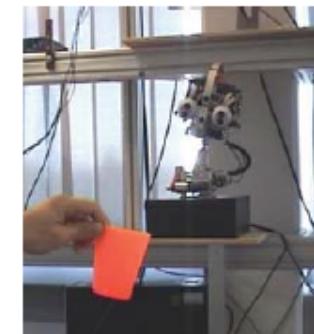
- gazing with a robotized head (7 joints, with $n = 6$ independently controlled) at a moving target (visual task dimension $m = 2$), using redundancy (to keep a better posture and avoid joint limits) and a predictive feedforward



video (c/o Fraunhofer IOSB, Karlsruhe)



$$\dot{\mathbf{q}} = \mathbf{J}_W^\dagger(\mathbf{q}) \left(\dot{\theta}_{fb} + \dot{\theta}_{ff} \right) - k_0 \left(\mathbf{I} - \mathbf{J}_W^\dagger(\mathbf{q}) \mathbf{J}(\mathbf{q}) \right) \nabla_{\mathbf{q}} H_0(\mathbf{q})$$



final head posture
without and with self-motions



Robotics 2

Detection and isolation of robot actuator faults

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Fault diagnosis problems - 1

- in the diagnosis of faults possibly affecting a (nonlinear) dynamic system various problems can be formulated
- **Fault Detection**
 - recognize that the malfunctioning of the (controlled) system is due to the occurrence of a fault (or not proper behavior) affecting some physical or functional component of the system
- **Fault Isolation**
 - discriminate which particular fault f has occurred out of a (large) class of potential ones, by distinguishing it from any other fault and from the effects of disturbances possibly acting on the system
- **Fault Identification**
 - determine the time profile (and/or class type) of the isolated fault f
- **Fault Accommodation**
 - modify the control law so as to compensate for the effects of the detected and isolated fault (possibly also identified)



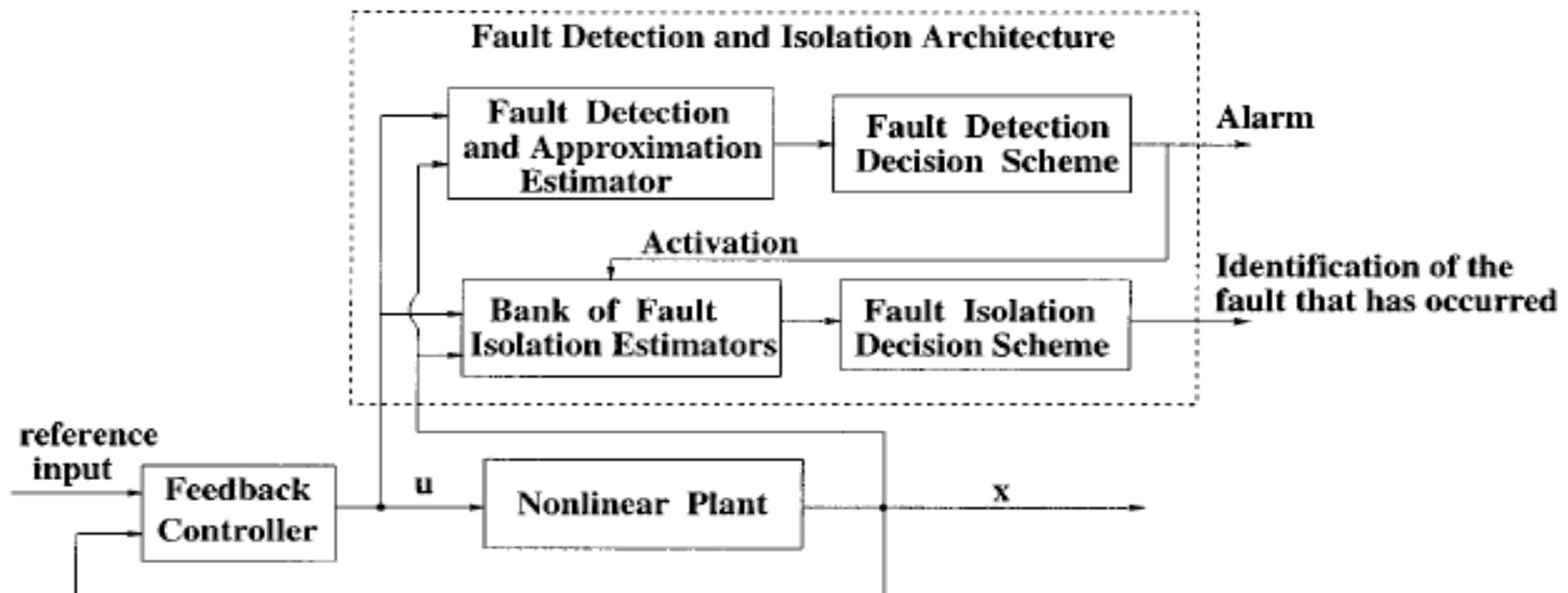
Fault diagnosis problems - 2

- FDI solution (simultaneous detection and isolation)
 - definition of an auxiliary dynamic system (**Residual Generator**) whose **output** will depend only on the presence of the fault f to be detected and isolated (and **not** on any other fault or disturbance) and will converge asymptotically to zero when $f \equiv 0$ (**stability**)
 - in case of many potential faults, each component r_i of the **vector r of residuals** will depend on one and only one associated fault f_i (possibly reproducing approximately its time behavior)
 - many of the FDI schemes are **model-based**: they use a nominal (fault- and disturbance-free) dynamic model of the system
- Fault Tolerant Control
 - **passive**: control scheme that is intrinsically robust to uncertainties and/or faults (typically having only moderate/limited effects)
 - **active**: control scheme involving a reconfiguration after FDI (with guaranteed performance for the faulted system)



Typical FDI architecture

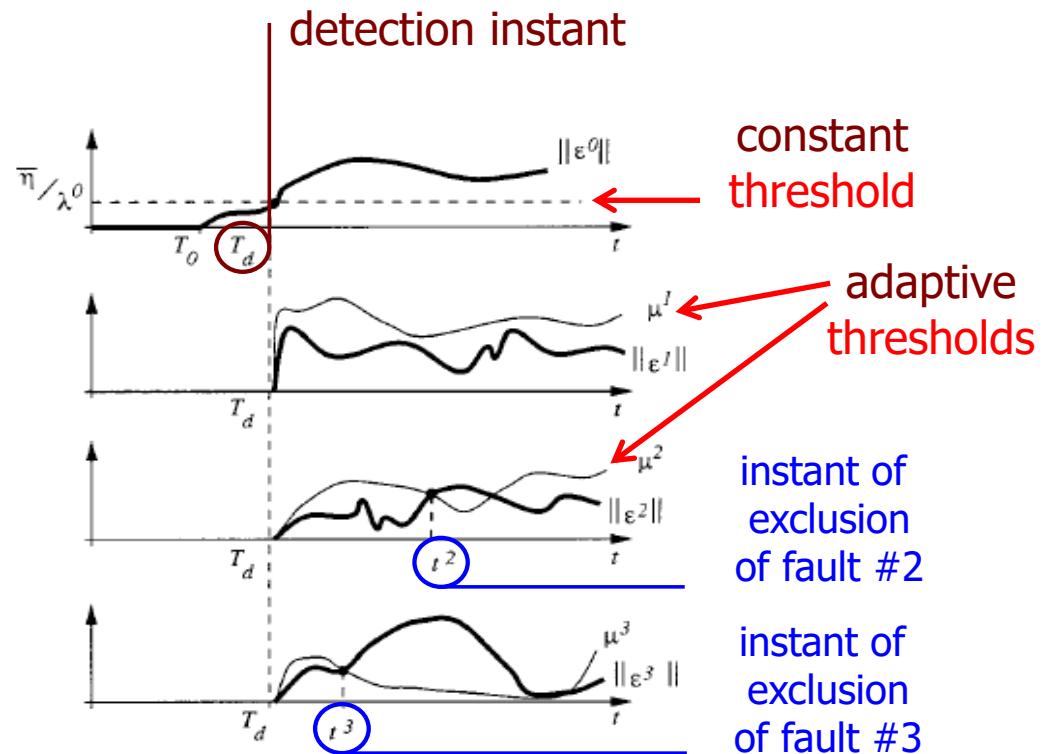
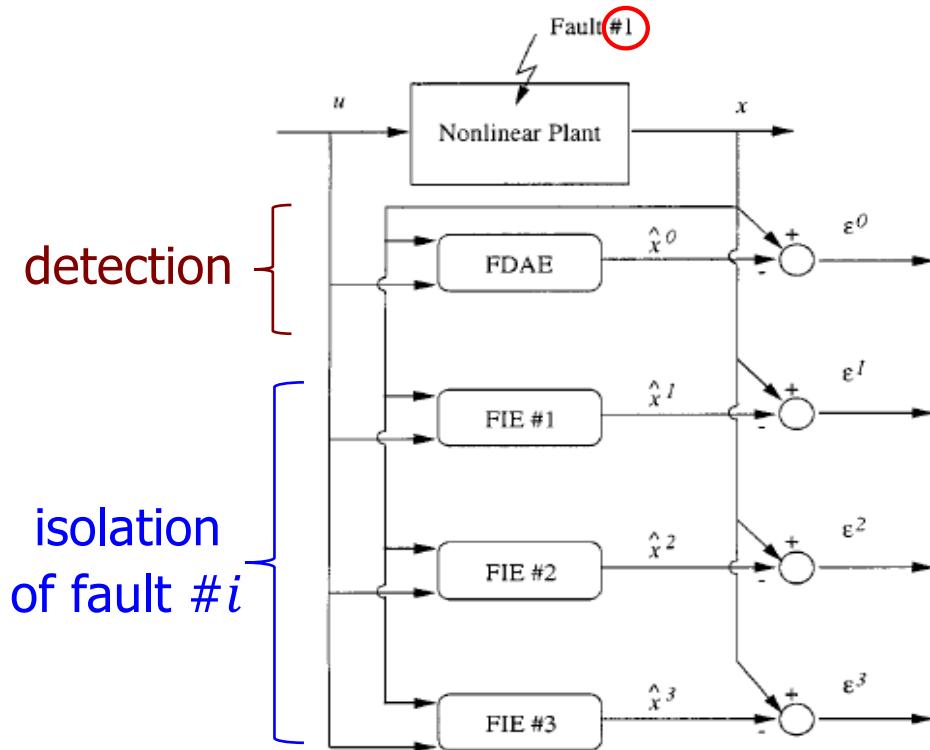
- bank of $n + 1$ (model-based) estimators
 - 1 for **detection** of a faulty condition
 - n for **isolation** of the specific (in general, **modeled**) fault





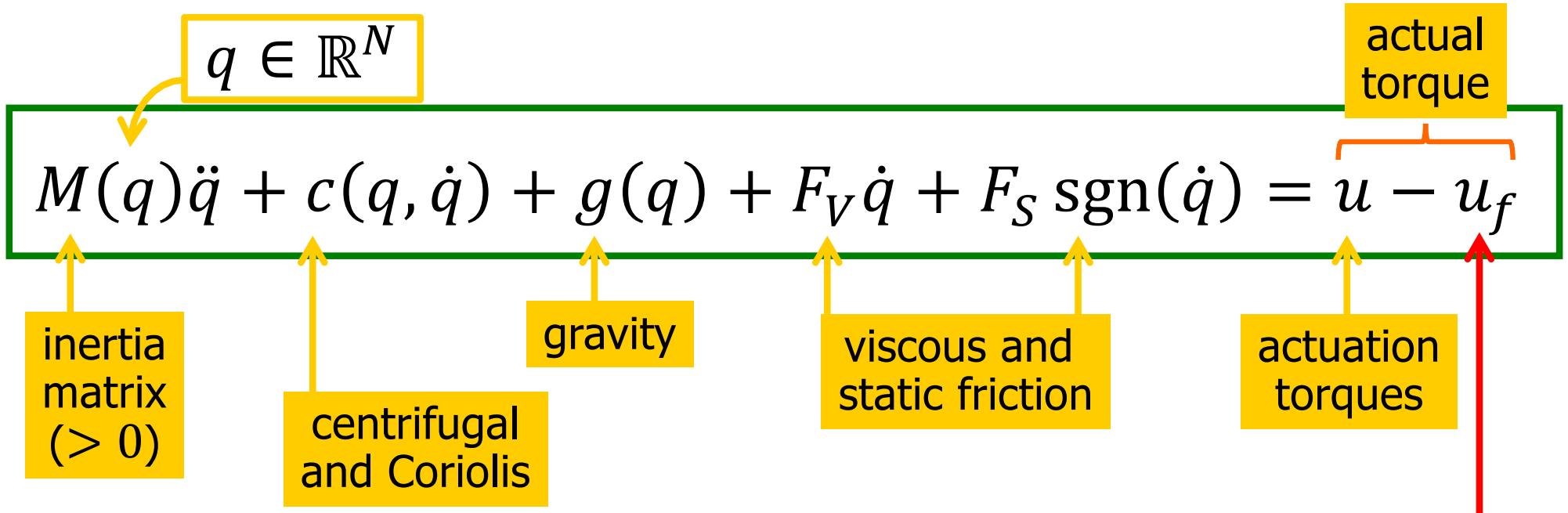
Some terminology

- fault types
 - instantaneous (abrupt), incipient (slow), intermittent, concurrent
- thresholds for detection/isolation (also adaptive)
 - delay times (w.r.t. the instant T_0 of fault start) vs. false alarms





Actuator faults in robots



vector of actuation faults (even concurrent on more axes)

- total fault $u_{f,i} = u_i$
- partial fault $u_{f,i} = \varepsilon u_i$ ($0 < \varepsilon < 1$)
- saturation $u_{f,i} = u_i - \operatorname{sgn}(u_i) u_{i,max}$
- bias $u_{f,i} = b_i$ Ex: ??
- block $u_{f,i} = \dots$
- ... any type!



Working assumptions

- signals and measurements available
 - the commanded input torque u , but obviously **not u_f** ...
 - a measure of the **full state** (q, \dot{q}) is available
 - can be relaxed: in practice, with an **estimate** of joint velocities
 - no further sensors are anyway necessary ("**sensorless**")
- the **robot dynamic model is known**
 - in the absence of faults, and neglecting disturbances
 - **no pre-specified model or type of faults** is needed
- **no dependence on/request of a specific input $u(t)$**
 - can be anything (open loop, linear or nonlinear feedback)
- **no dependence on/request of a specific motion $q_d(t)$**



Generalized momentum

$$p = M(q)\dot{q}$$

with associated dynamic equation

$$\dot{p} = u - u_f - \alpha(q, \dot{q})$$

decoupled components
relative to the single fault inputs

exploiting structure
of centrifugal and
Coriolis terms

$$\alpha_i = -\frac{1}{2}\dot{q}^T \frac{\partial M(q)}{\partial q_i} \dot{q} + g_i(q) + F_{V,i}\dot{q}_i + F_{S,i} \operatorname{sgn}(\dot{q}_i)$$

scalar expressions, for $i = 1, \dots, N$



FDI solution

- definition of a **vector of residuals**

$$r = K \left[\int (u - \alpha(q, \dot{q}) - r) dt - p \right]$$

$K > 0$
diagonal

- no need to compute joint accelerations nor to invert the robot inertia matrix $M(q)$
- with perfect model knowledge, the dynamics of r is

N **decoupled** filters,
with unitary gains and
time constants $\tau_i = 1/k_i$

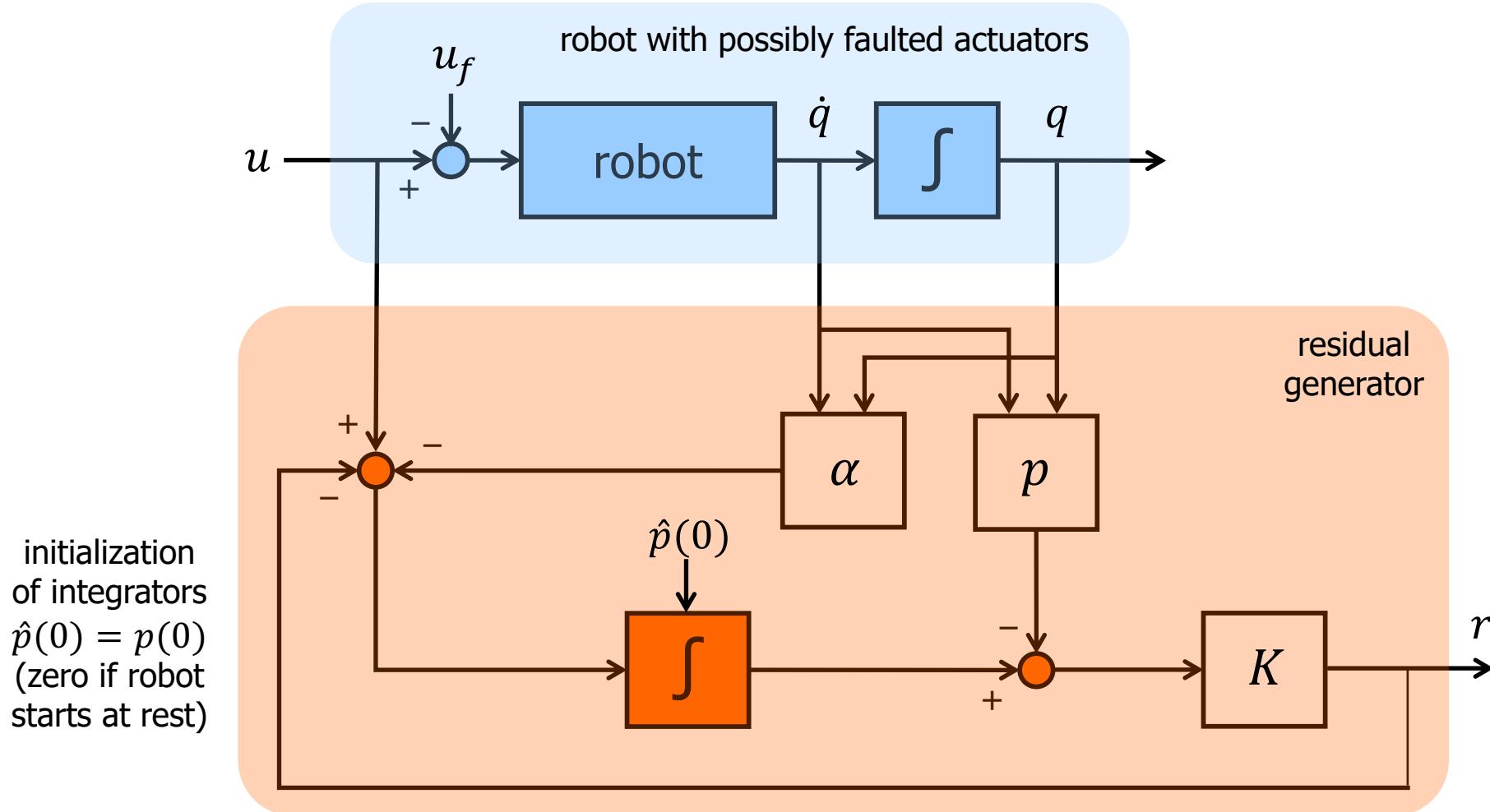
$$\dot{r} = -Kr + Ku_f$$

in the Laplace domain
$$\frac{r_i(s)}{u_{f,i}(s)} = \frac{k_i}{s + k_i} = \frac{1}{1 + \tau_i s}$$

for sufficiently large K , r reproduces the time behavior of u_f



Block diagram of the residual generator



$$r = K \left[\int (u - \alpha(q, \dot{q}) - r) dt - p \right]$$



Residual generator as “disturbance observer”

from the
block diagram...

$$\begin{aligned}\dot{\hat{p}} &= u - \alpha(q, \dot{q}) + K(p - \hat{p}) \\ r &= K(\hat{p} - p)\end{aligned}$$



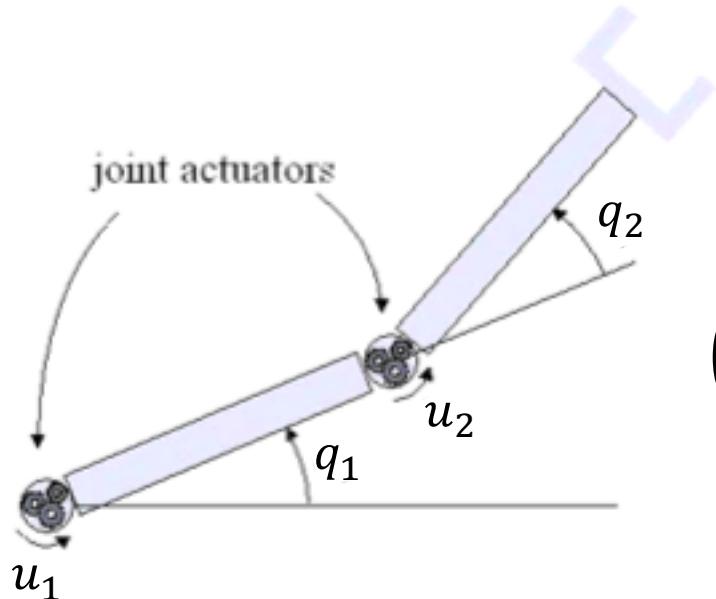
dynamic observer of the unknown actuation faults
($r \approx \rightarrow u_f$ = external disturbances)
with **linear** error dynamics (for constant u_f)

$$\begin{aligned}e_{obs} &= u_f - r \quad \rightarrow \quad \dot{e}_{obs} = \dot{u}_f - \dot{r} = \dot{u}_f - K(\dot{\hat{p}} - \dot{p}) \\ &= \dot{u}_f - K((u - \alpha - r) - (u - \alpha - u_f)) \\ &= \dot{u}_f - K(u_f - r) = \dot{u}_f - K e_{obs} \cong -K e_{obs}\end{aligned}$$



A worked-out example

- planar 2R robot under gravity



dynamic model (without friction)

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = u - u_f$$

$\overbrace{\quad\quad\quad}^{= S(q, \dot{q})\dot{q}}$

$$\begin{pmatrix} a_1 + 2a_2c_2 & a_3 + a_2c_2 \\ a_3 + a_2c_2 & a_3 \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + \begin{pmatrix} -a_2(2\dot{q}_1 + \dot{q}_2)\dot{q}_2 s_2 \\ a_2\dot{q}_1^2 s_2 \end{pmatrix}$$

$$+ \begin{pmatrix} a_4c_1 + a_5c_{12} \\ a_5c_{12} \end{pmatrix} = \begin{pmatrix} u_1 - u_{f,1} \\ u_2 - u_{f,2} \end{pmatrix}$$

computation of the residual vector

$$r = K \left[\int (u - \alpha(q, \dot{q}) - r) dt - p \right]$$

$$p = M(q)\dot{q}$$

$$\alpha_1 = g_1(q) = a_4c_1 + a_5c_{12}$$

$$\alpha_2 = -\frac{1}{2} \dot{q}^T \frac{\partial M(q)}{\partial q_2} \dot{q} + g_2(q)$$

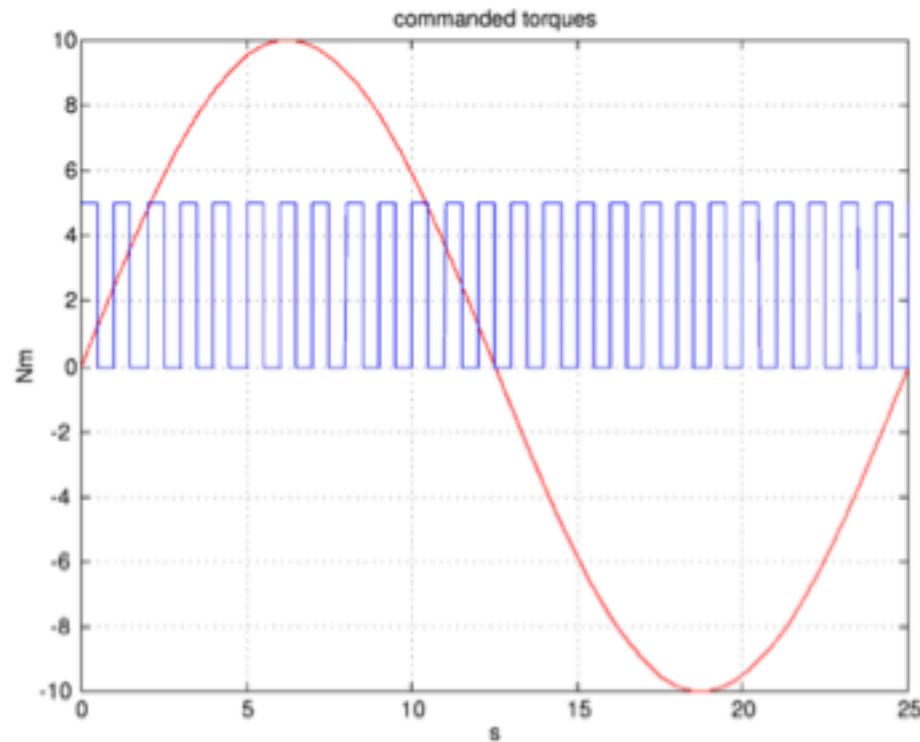
$$= a_2(\dot{q}_1 + \dot{q}_2)\dot{q}_1 s_2 + a_5c_{12}$$



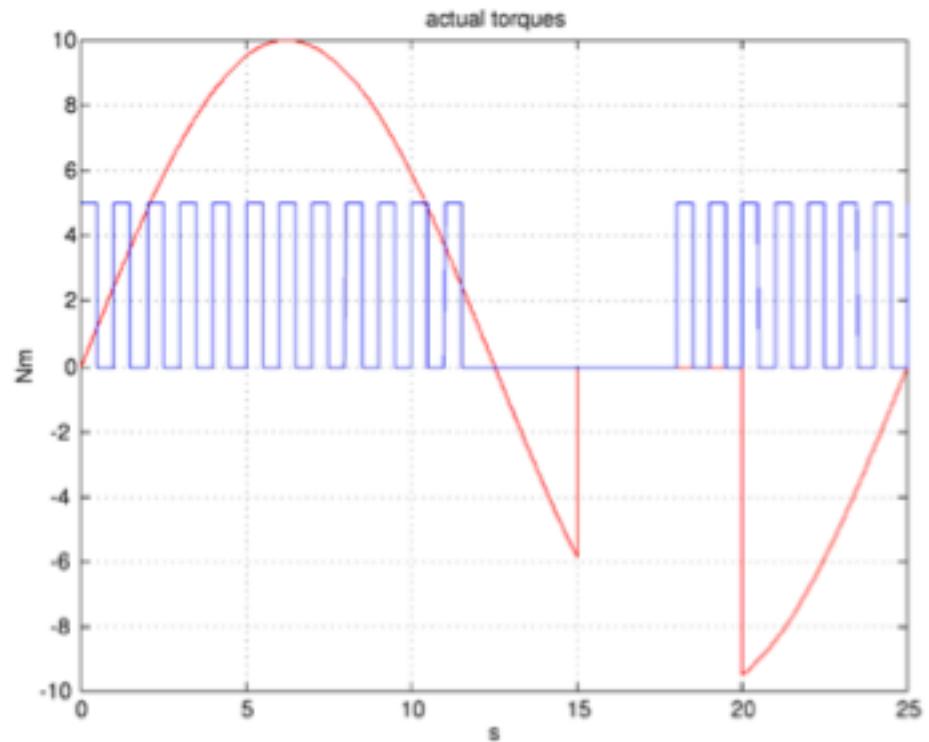
Faults on both actuators

(total, intermittent, concurrent)

commanded torques (in open loop)



actual (faulted) torques



= first joint (fault for $t \in [15 \div 20]$ sec)



= second joint (fault for $t \in [12 \div 18]$ sec)

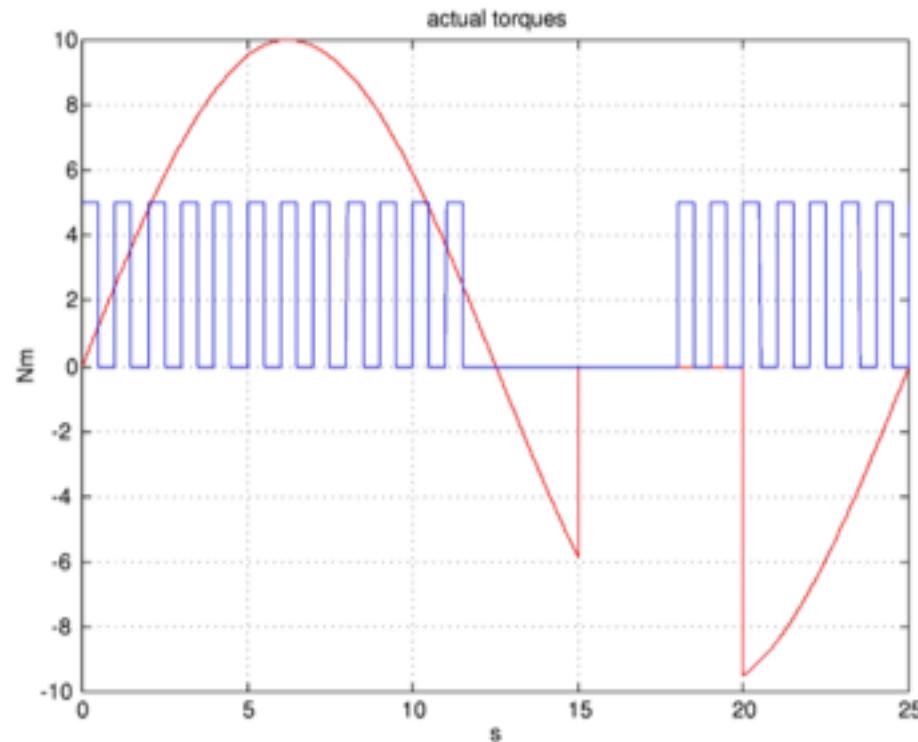


time interval of
fault **concurrence**
 $t \in [15 \div 18]$ sec



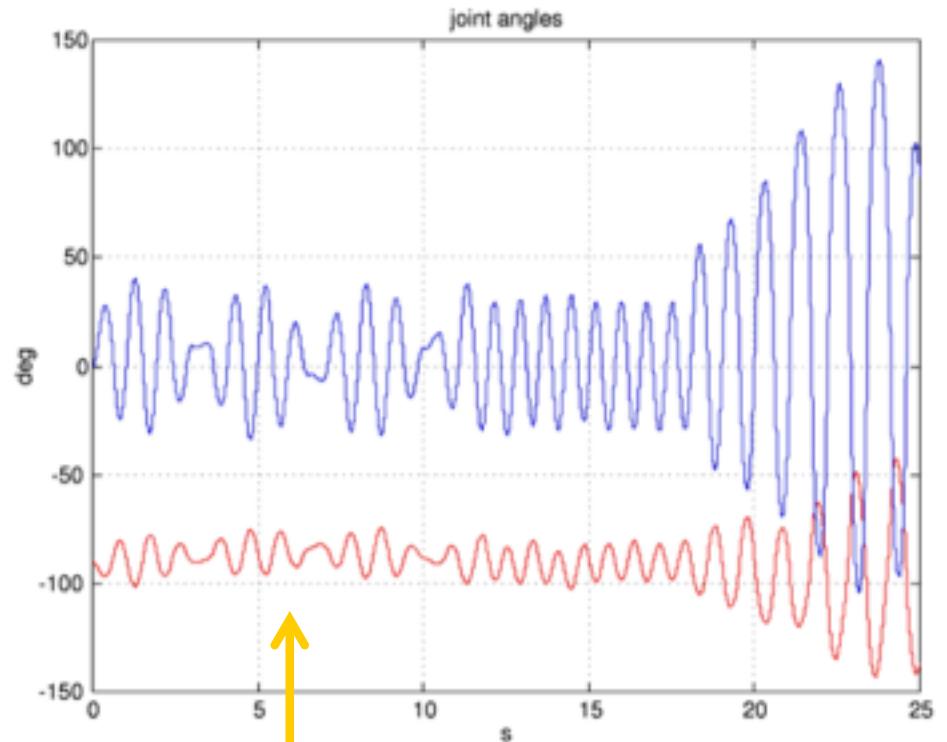
First simulation

actual torques (to the robot)



- = first joint
- = second joint

(measured) joint positions

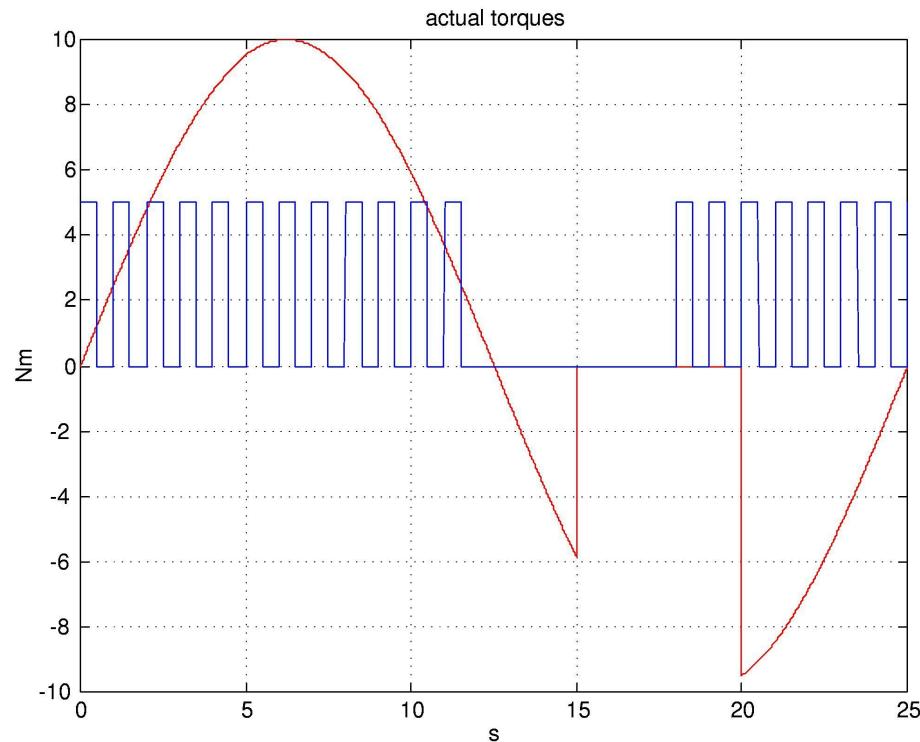


no clear evidence of faults in the dynamic evolution of the system!



First simulation – FDI

actual torques (to the robot)

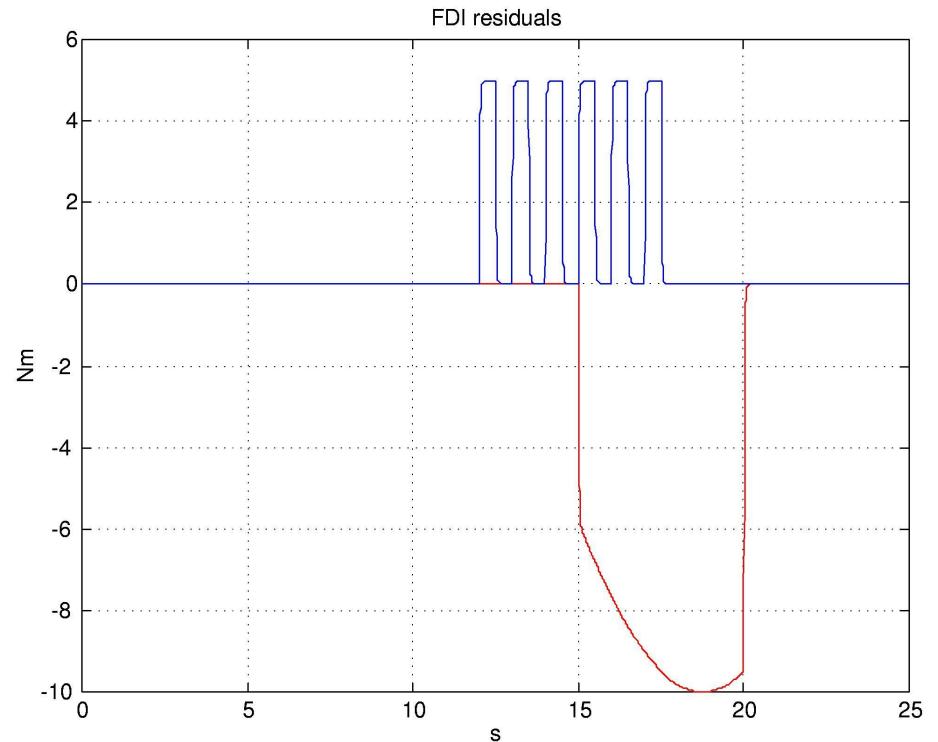


— = first joint

— = second joint

$$K = \text{diag}\{50, 50\}$$

residuals



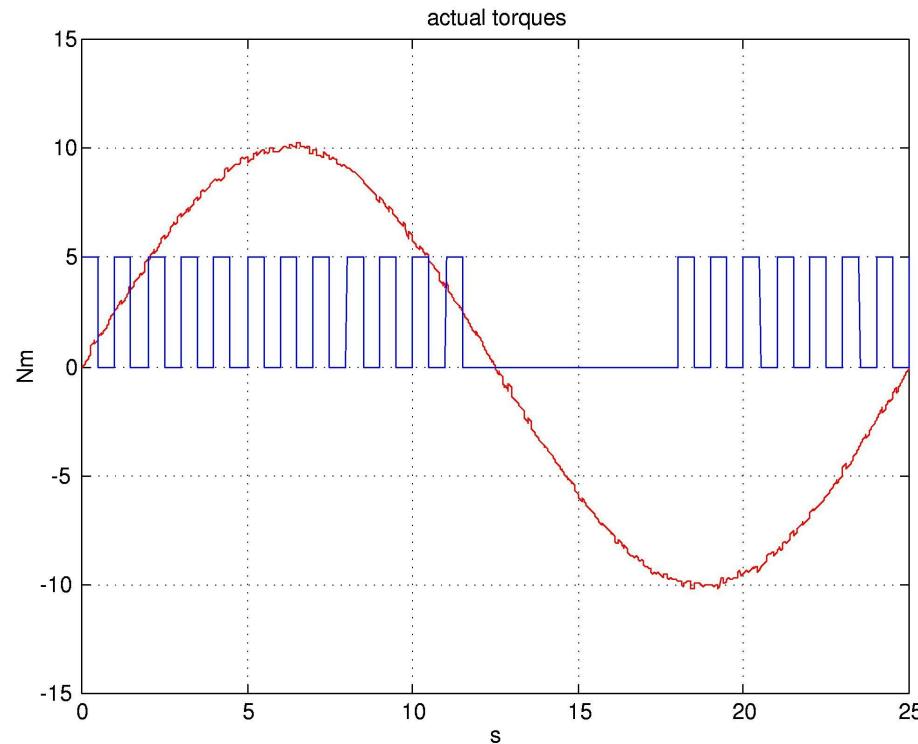
residuals reconstruct the
“missing” parts of the torques
(identification property!)



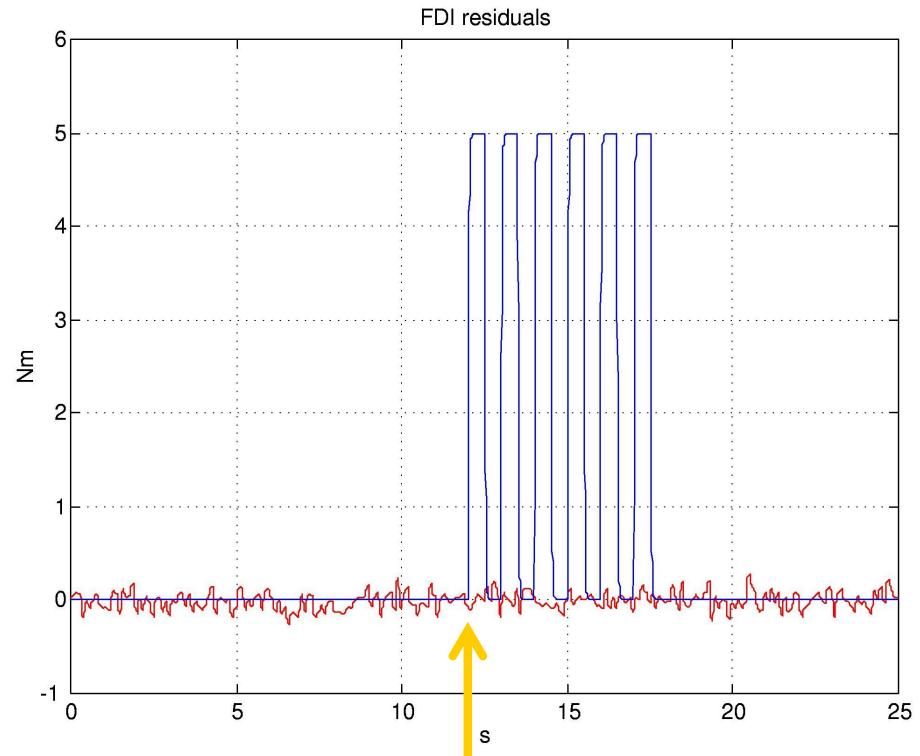
Second simulation – FDI

(total fault on second actuator, added noise on first channel)

actual torques (to the robot)



residuals



— = first joint

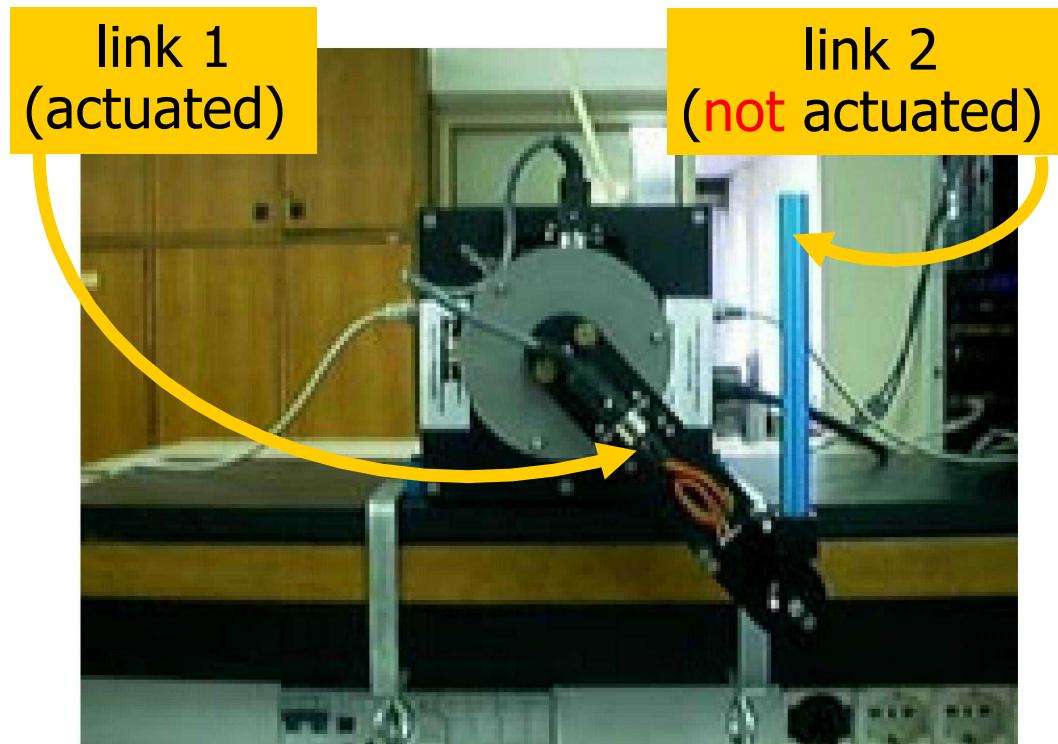
— = second joint (fault for $t \in [12 \div 18]$ sec)

residual r_1 is not affected by faulty actuation, while residual r_2 is not affected by the disturbance on first channel (decoupling property)

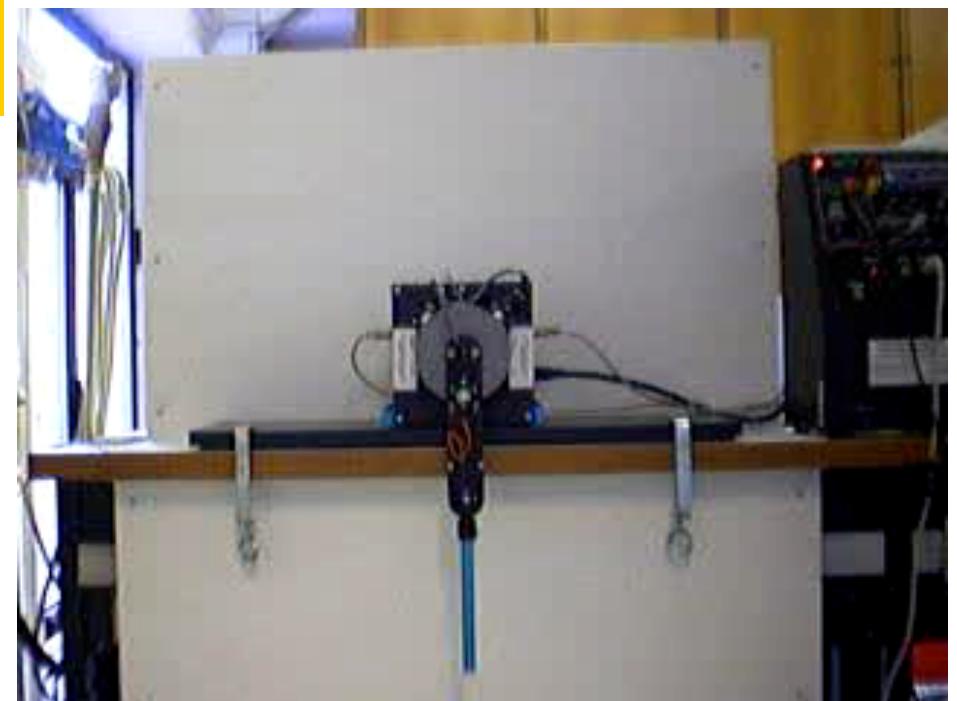


Experimental setup

Quanser Pendubot



with encoders on both joints



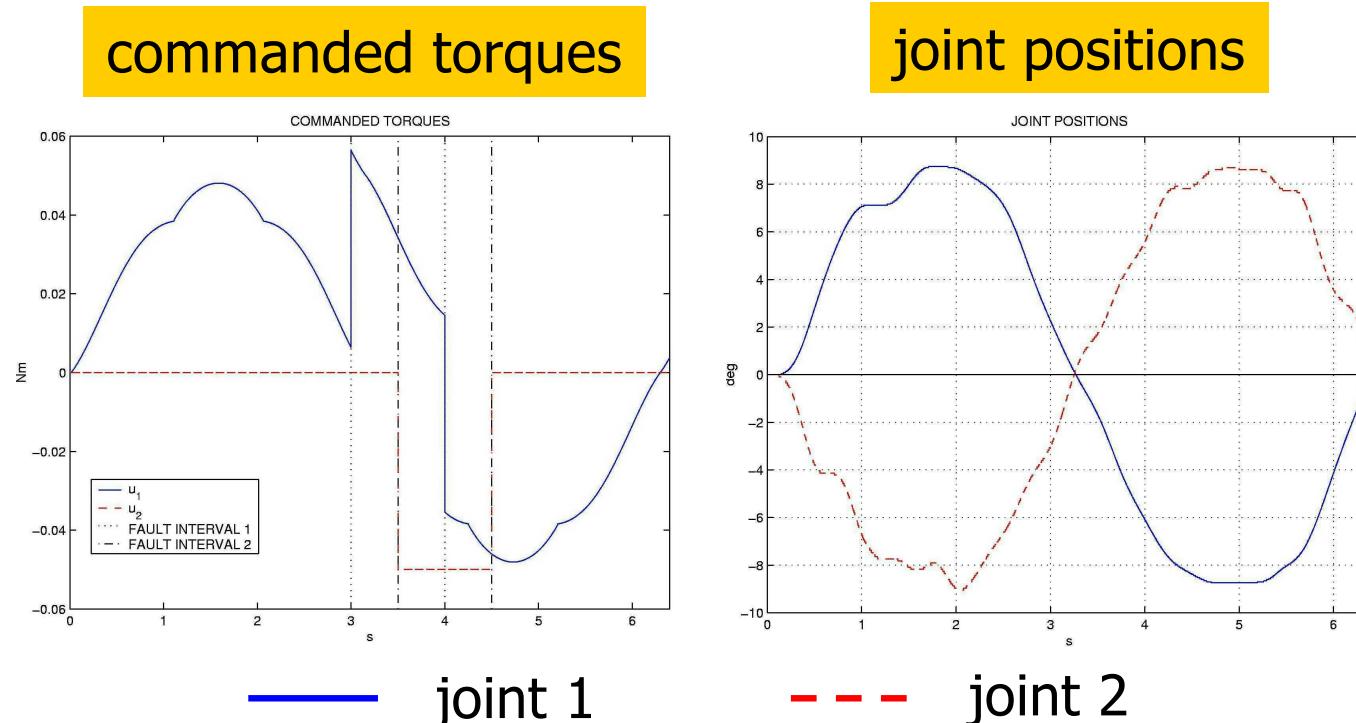
nonlinear control for swing-up

sampling time $T_c = 1$ ms, residual gains $K_i = 50$,
practical thresholds of fault detection $\cong 10^{-2} \div 10^{-3}$ Nm



First experiment

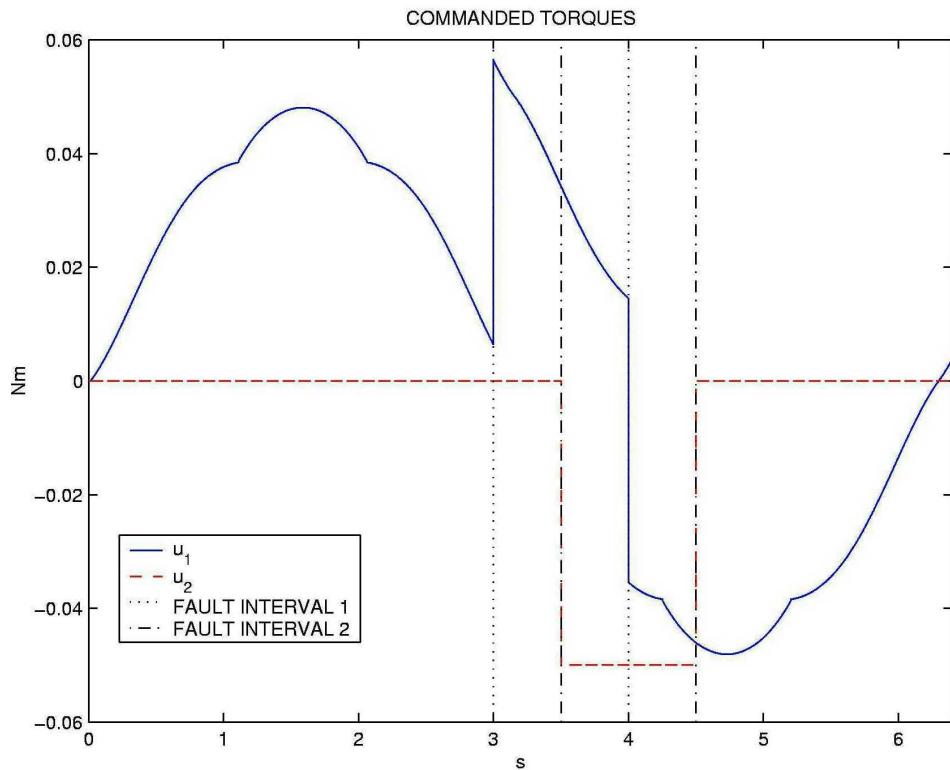
- motor 1 driven by sinusoidal voltage of period 2π sec (open loop)
- **bias fault** on u_1 for $t \in [3 \div 4]$ sec
- **total fault** on second joint for $t \in [3.5 \div 4.5]$ sec (a constant torque is requested, but **no motor at the joint to provide 0.05 Nm...**)
- **fault concurrency** for $t \in [3.5 \div 4]$ sec



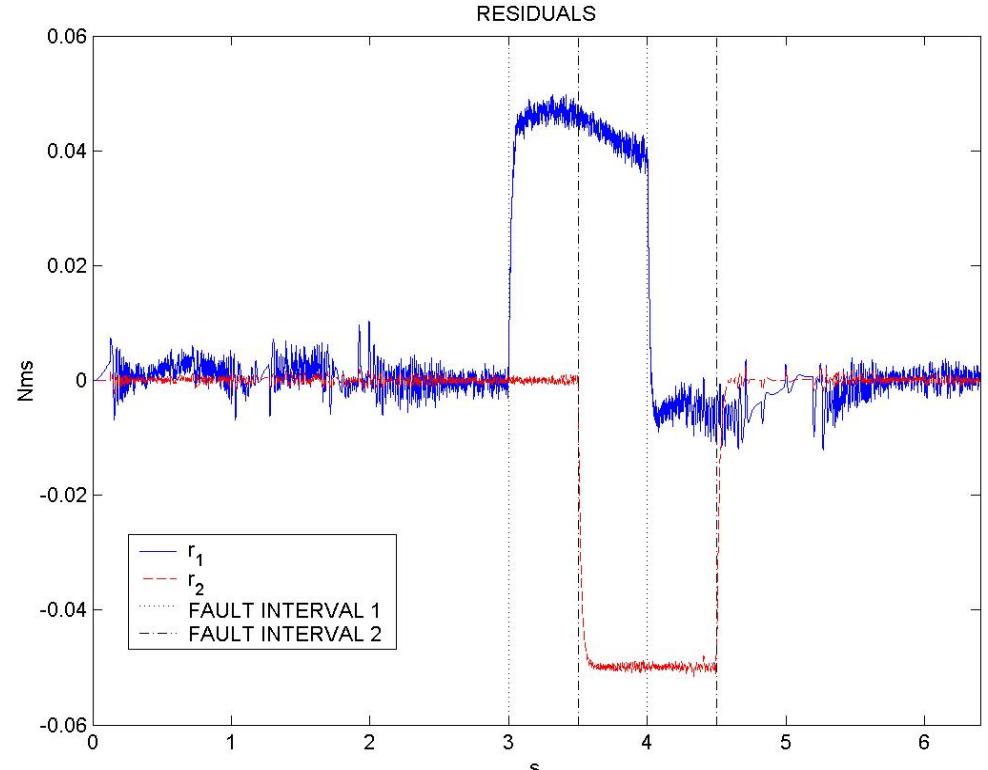


First experiment – FDI

commanded torques



residuals

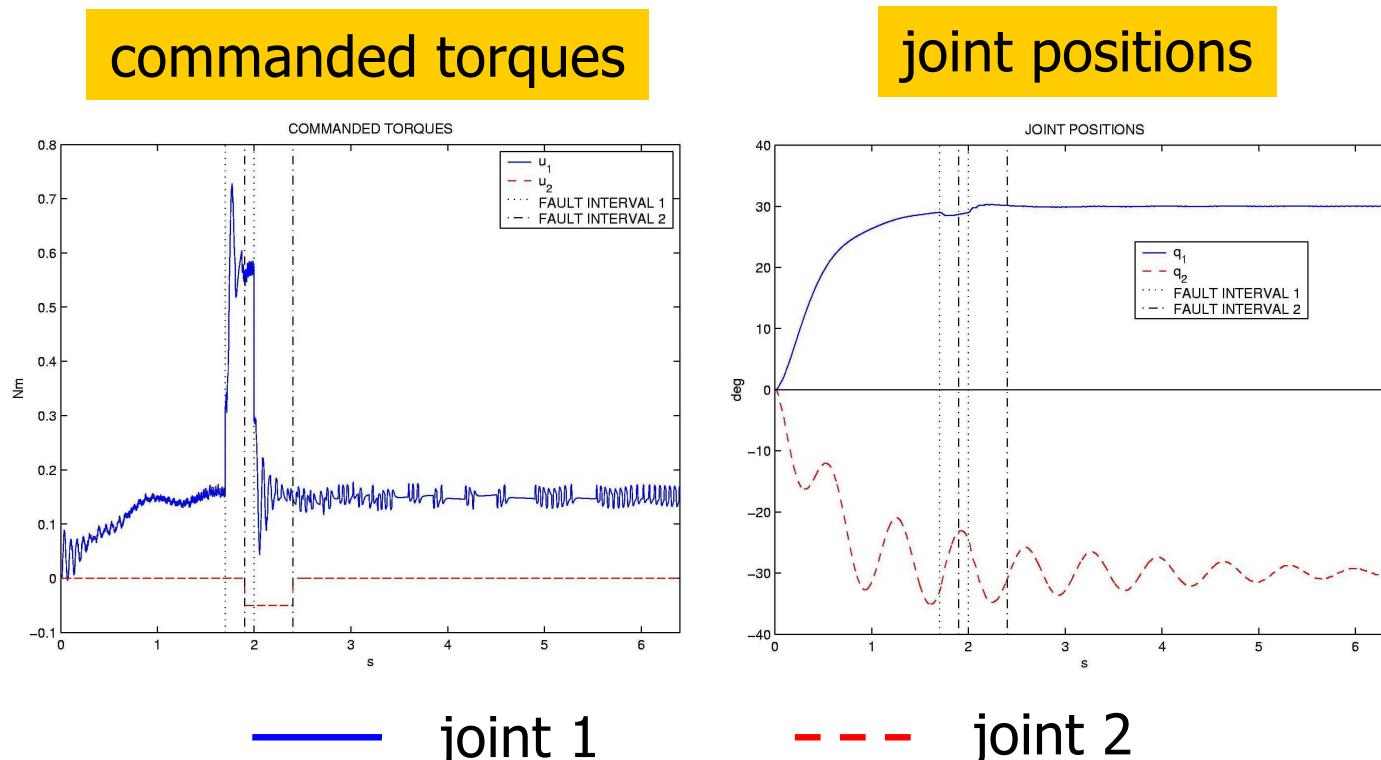


— joint 1

- - - joint 2

Second experiment

- position regulation of the first joint at $q_{d1} = 30^\circ$ (**PID control**)
- **50% power loss** on motor 1 for $t \in [1.7 \div 2]$ sec
- **total fault** on joint 2 for $t \in [1.9 \div 2.4]$ sec (**no motor...**)
- **fault concurrency** for $t \in [1.7 \div 1.9]$ sec

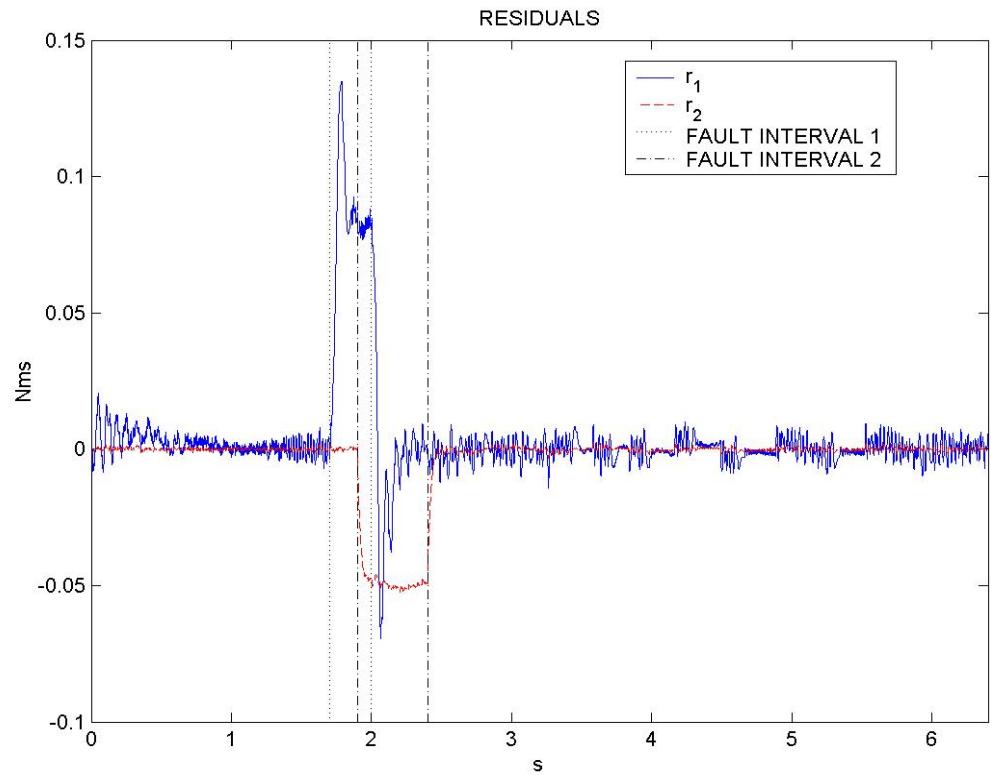
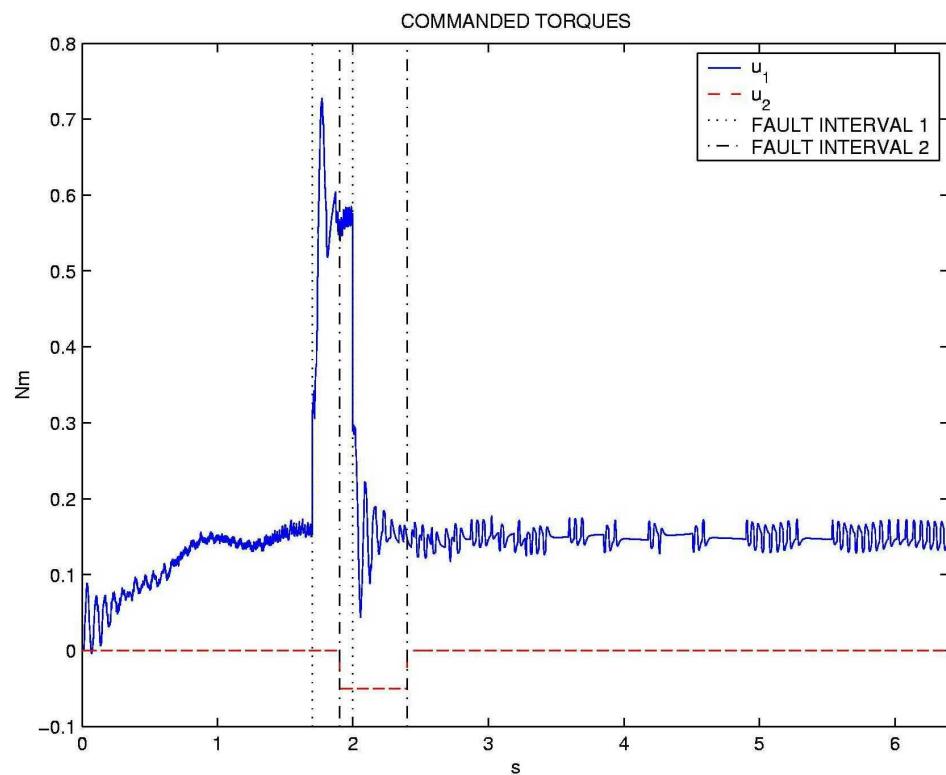




Second experiment – FDI

commanded torques

residuals

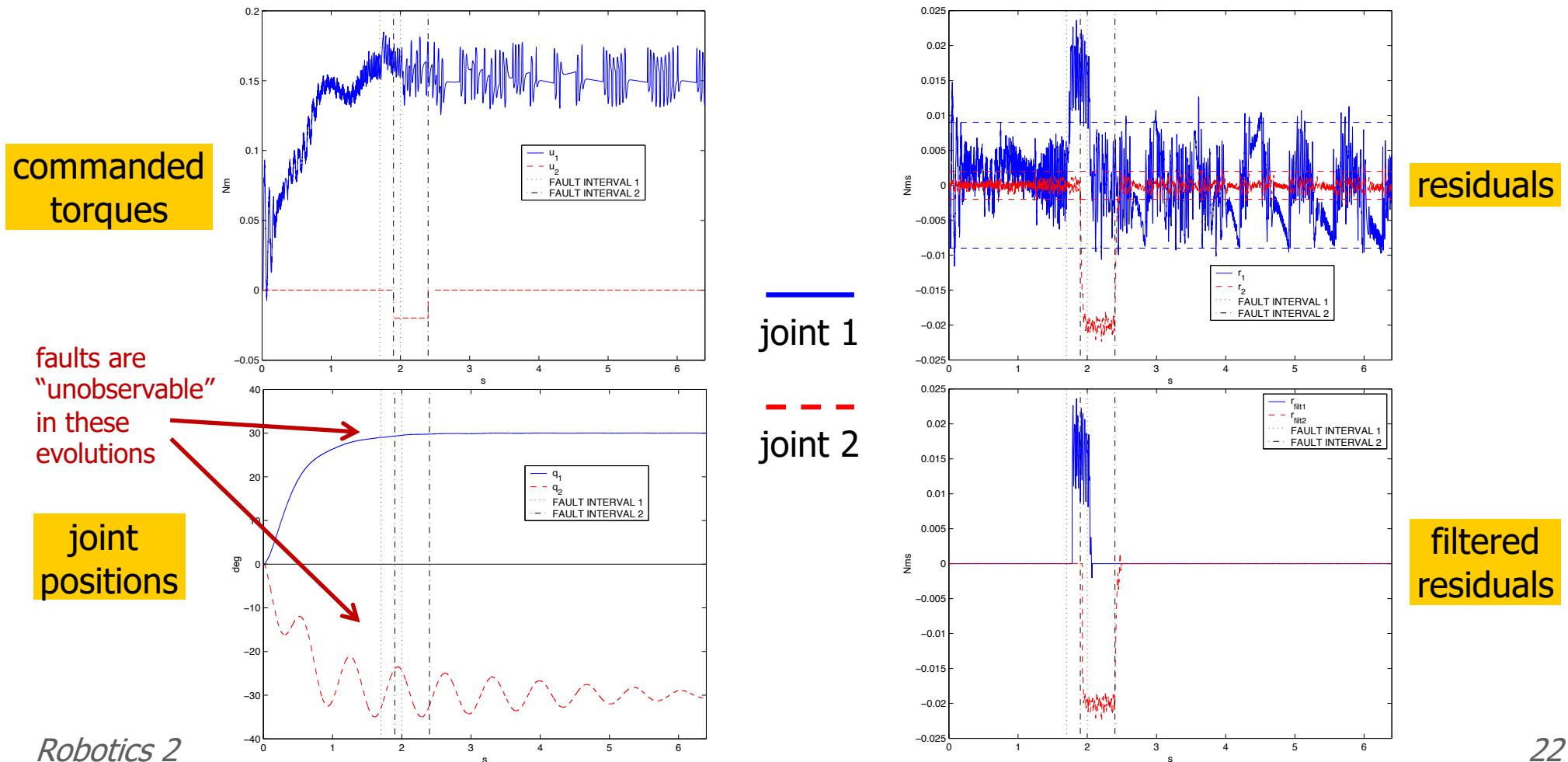


— joint 1

- - - joint 2

Third experiment – FDI

- same as in second experiment, but with only **10% power loss** on motor 1
 - due to noisy PWM signals driving the DC motor, a **dynamic filtering** of residuals is used, staying above [below] a threshold ($r_{1,thres} = 9 \cdot 10^{-3}$ Nm, $r_{2,thres} = 2 \cdot 10^{-3}$ Nm) for a time $T_{set} = 0.02$ s [$T_{reset} = 0.03$ s] before detecting a fault [reset to normal operation]





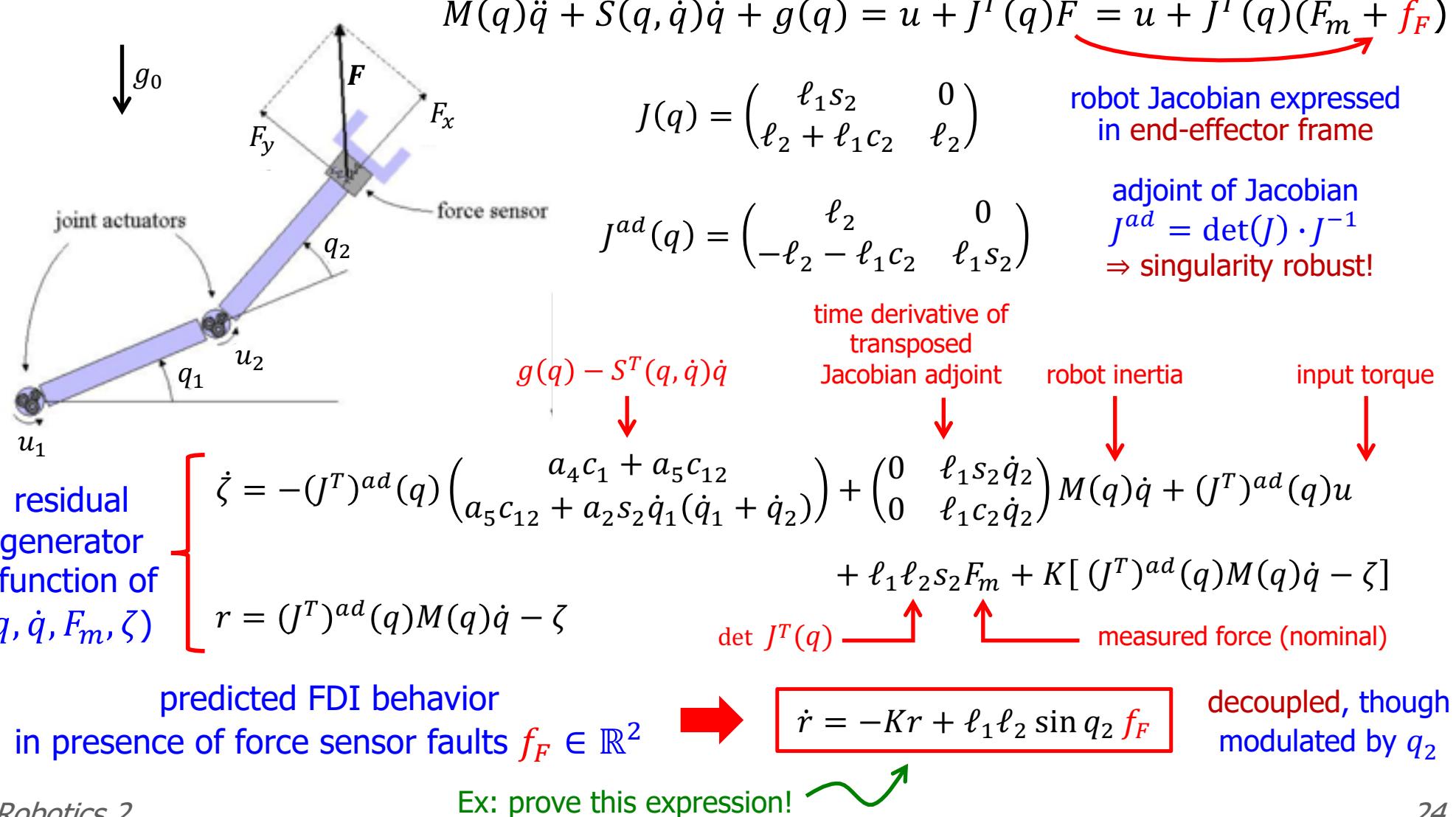
Extensions

- FDI method based on generalized momentum is easily extended to the presence of **flexible transmissions** (elastic joints), **actuator dynamics**, ...
- the scheme can be made **adaptive**, so as to handle parametric uncertainties in the robot dynamic model
- the method can be modified for detection and isolation of significant classes of **sensor faults** (e.g., faults in force/torque sensor at the wrist)
 - applies to all faults that instantaneously affect robot **acceleration** or **torque** (i.e., occurring at the second-order differential level)
- assuming **non-concurrency** (at most a single fault occurs at the same time) of a given set of faults, **relaxed FDI conditions** have been derived
 - of interest when the necessary conditions for multiple FDI are violated
 - involves processing of **continuous** residuals + **discrete** logic for isolation
- the same FDI-type approach has been applied also for **compensation of unmodeled friction** (treated as a “permanent fault” on the system)
- combination of **model-** and **signal-based** approaches to FDI



Isolation of F/T sensor faults

- planar 2R robot with **fault** on force **measure** of sensor on the end-effector

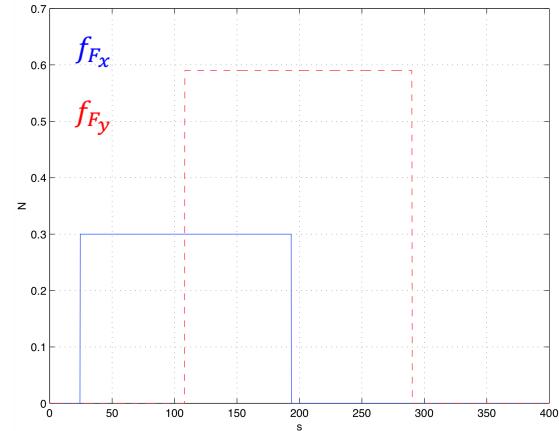




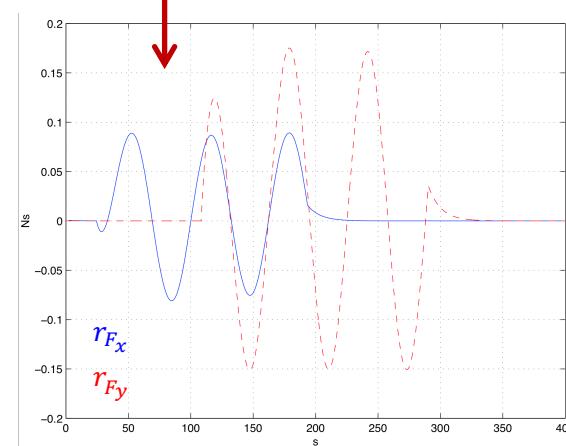
Isolation of F/T sensor faults

- simulation on the 2R robot

bias faults
on the two components
of force sensor measures
0.3N on f_{F_x} in $t \in [25 \div 190]$
0.6N on f_{F_y} in $t \in [109 \div 285]$



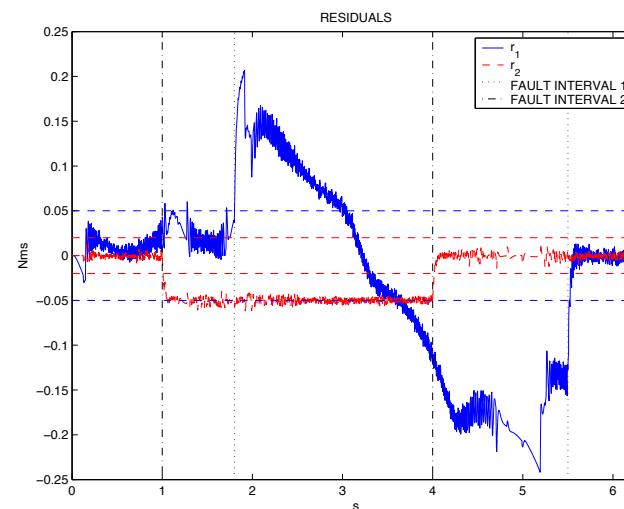
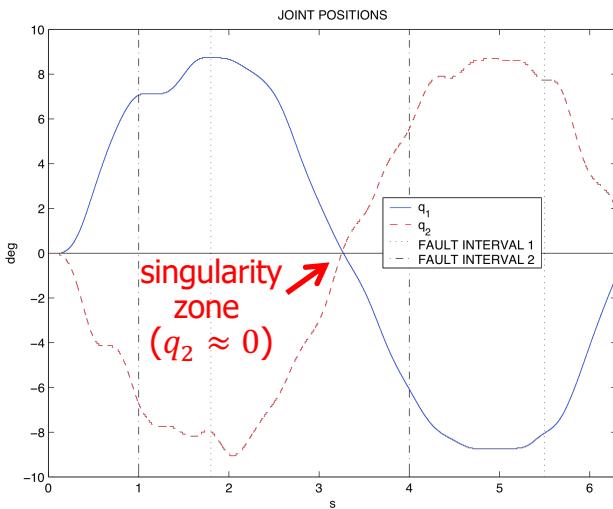
q_2 is tracking a sinusoid ($A = \pi/8$ rad, $\omega = 0.1$ rad/s)



FDI residual
components
(with $K = 0.1I$)

- experiment on the Pendubot (no force sensor and no contact!)

evolution
of joint
variables



residuals
for emulated bias
measurement faults
-1N on F_x in $t \in [1.8 \div 5.5]$
0.05N on F_y in $t \in [1 \div 4]$

$(J^T)^{ad} \rightarrow \text{diag}\{s_2, 1\} J^{-T}$
in previous scheme



Isolation of non-concurrent faults

- faults of the actuators **AND** faults of the force sensor components (possibly occurring **simultaneously**) **CANNOT** be detected **AND** isolated
 - for a mechanical system with N dofs, the **max # of faults allowing FDI** is $N!$
- with **non-concurrency**, e.g., 2 actuator + 2 F/T sensor faults in 2R robot

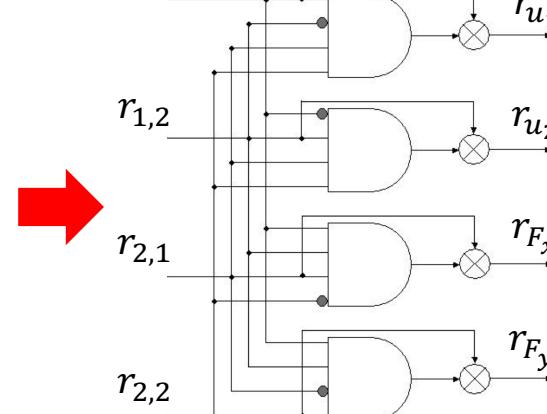
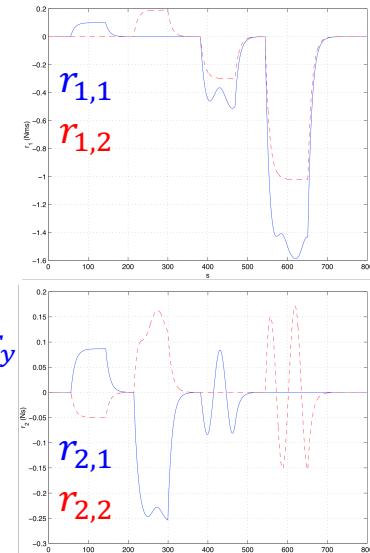
dependence of residuals on considered faults

residual fault	$r_{1,1}$	$r_{1,2}$	$r_{2,1}$	$r_{2,2}$
f_{u_1}	1	0	1	1
f_{u_2}	0	1	1	1
f_{F_x}	1	1	1	0
f_{F_y}	1	1	0	1

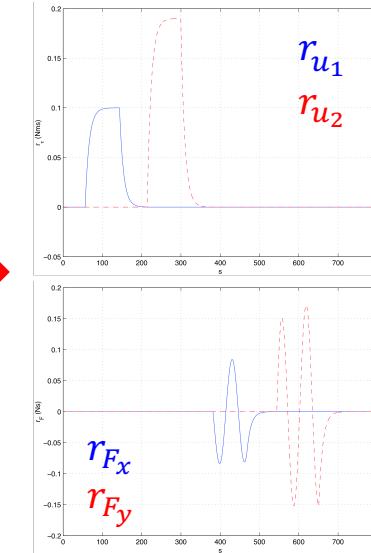
isolation matrix

$r_{2,1} \ r_{2,2}$	11	10	01	00
$r_{1,1} \ r_{1,2}$				
10	f_{u_1}	NA	NA	NA
01	f_{u_2}	NA	NA	NA
11	NC	f_{F_x}	f_{F_y}	NA
00	NA	NA	NA	no fault

time sequence of non-concurrent bias faults:
 $f_{u_1} \rightarrow f_{u_2} \rightarrow f_{F_x} \rightarrow f_{F_y}$



isolation logics



hybrid residuals allowing isolation of 4 faults

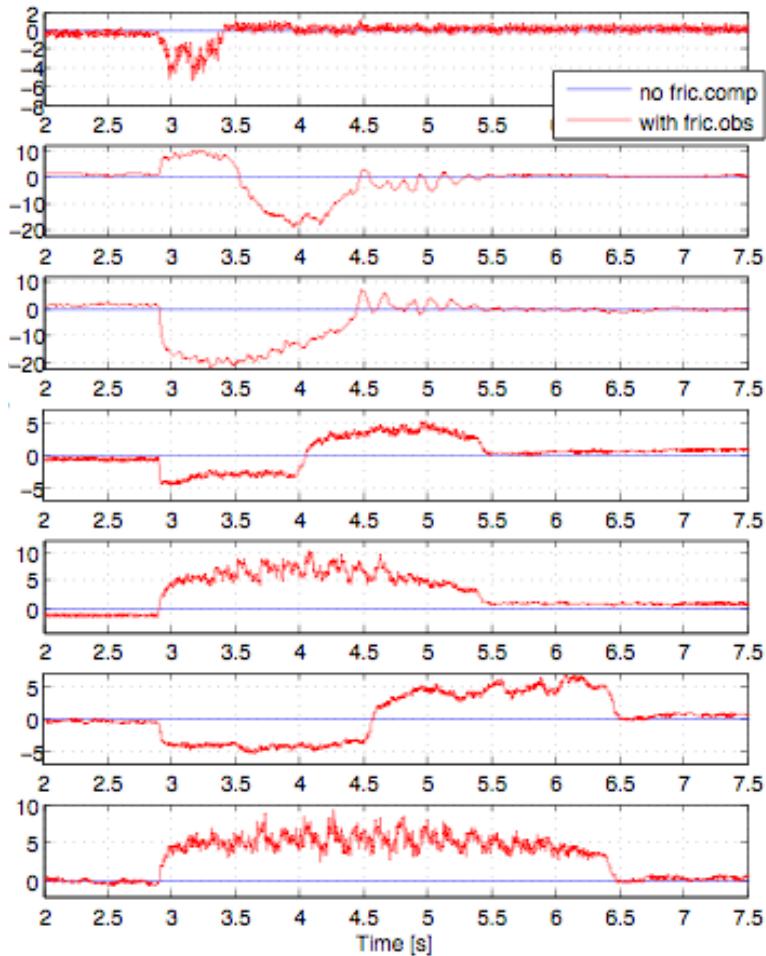


Experiments on friction compensation

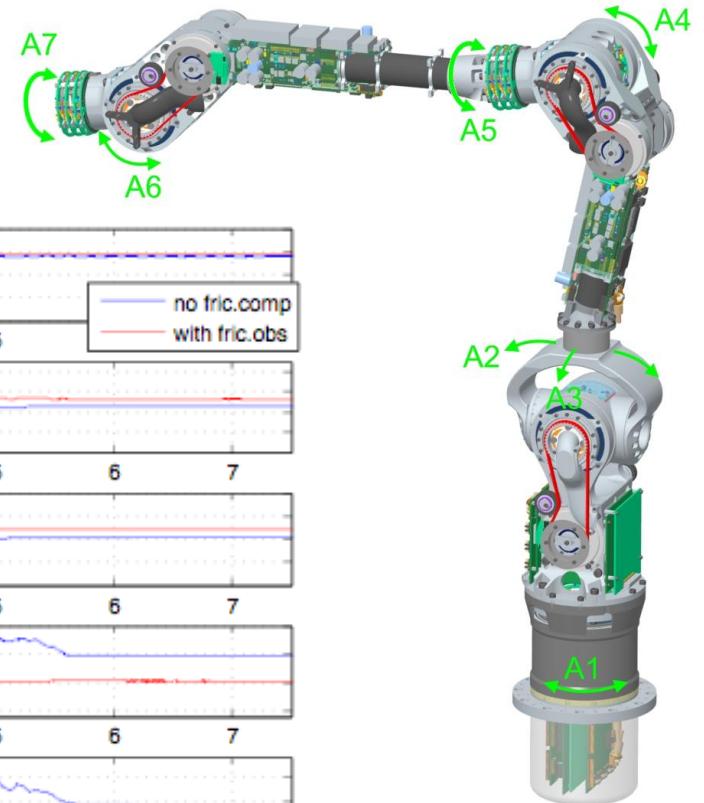
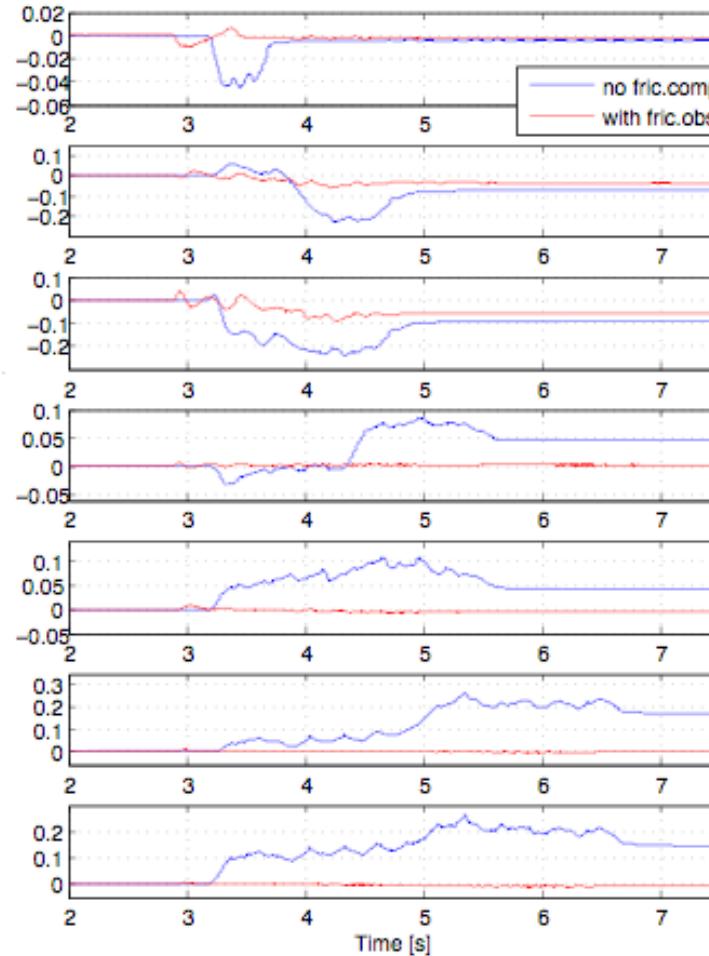
- results on the DLR 7R medical robot

used then on-line
in control law...

friction estimate via residuals



position error

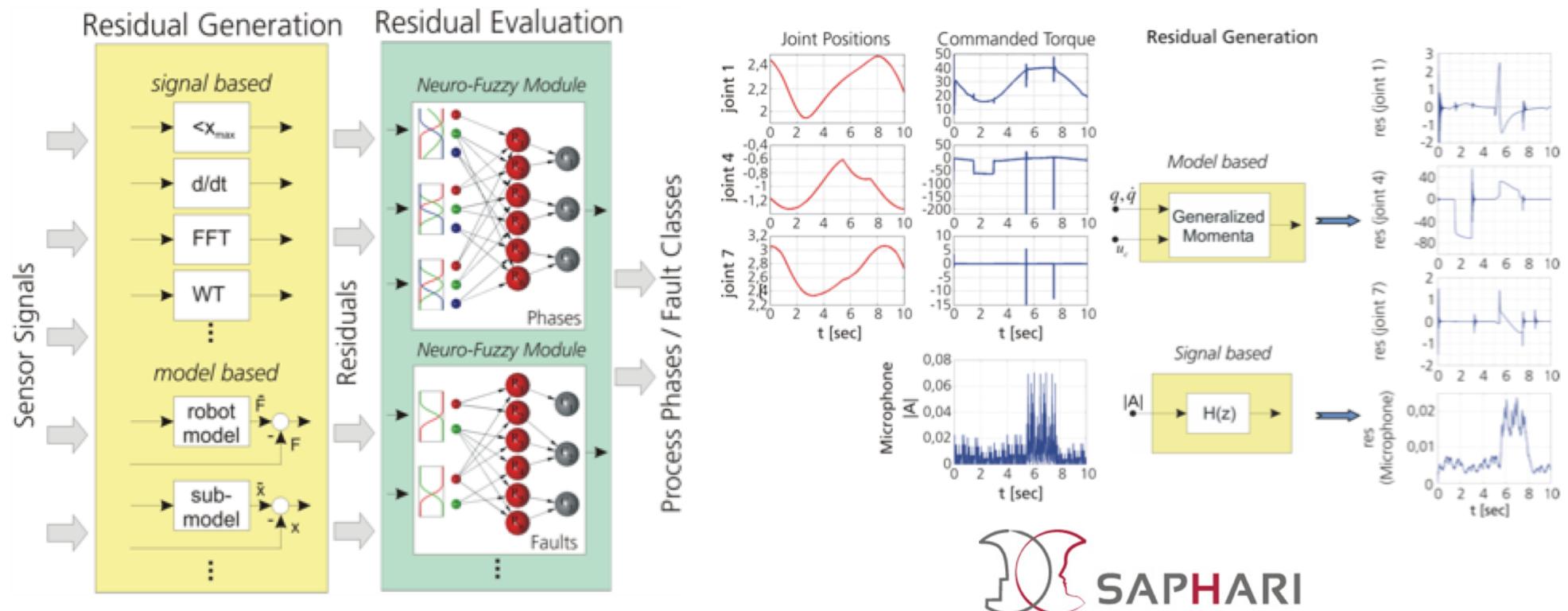


HD at the joints
⇒ elastic joint
dynamic model



Model- and signal-based FDI

- detection and isolation features can be enhanced by combining multiple sensor inputs and different approaches
 - model-based (exact, but require accurate models)
 - signal-based (approximate, but without special requirements)
- so as to obtain the “best of both worlds”





Bibliography

- X. Zhang, M. Polycarpou, T. Parisini, "Robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems," *IEEE Trans. on Automatic Control*, vol. 47, no. 4, pp. 576-592, 2002.
- A. De Luca, R. Mattone, "Actuator failure detection and isolation using generalized momenta," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 634-639, 2003.
- A. De Luca, R. Mattone, "An adapt-and-detect actuator FDI scheme for robot manipulators," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 4975-4980, 2004.
- A. De Luca, R. Mattone, "An identification scheme for robot actuator faults," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp.1127-1131, 2005.
- R. Mattone, A. De Luca, "Relaxed fault detection and isolation: An application to a nonlinear case study," *Automatica*, vol. 42, no. 1, pp. 109-116, 2006.
- R. Mattone, A. De Luca, "Nonlinear fault detection and isolation in a three-tank heating system," *IEEE Trans. on Control Systems Technology*, vol. 14, no. 6, pp. 1158-1166, 2006.
- L. Le Tien, A. Albu-Schäffer, A. De Luca, G. Hirzinger, "Friction observer and compensation for control of robots with joint torque measurements," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3789-3795, 2008.
- C. Gaz, A. Cristofaro, A. De Luca, "Detection and isolation of actuator faults and collisions for a flexible robot arm," *Proc. 59th IEEE Conf. on Decision and Control*, pp. 2684-2689, 2020.