Robotics 2

Robots with kinematic redundancy

Part 2: Extensions

Prof. Alessandro De Luca

Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti

Sapienza Università di Roma

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

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

$$\dot{r}_k = J_k(q)\dot{q}$$

k-th task

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

projector in the null-space of k-th task

$$i < j \Rightarrow \text{task } i \text{ has higher priority than task } j$$

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

even larger!

$$\dot{r}_{A,k} = \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \vdots \\ \dot{r}_k \end{pmatrix}$$

stack of first k tasks

$$J_{A,k} = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_k \end{pmatrix}$$

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

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

$$\Longleftrightarrow \qquad J_{A,k} P_{A,k} = O$$

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

$\boxed{\text{for } k = 1}$

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

$$\Rightarrow \quad \dot{q}_1 = J_1^{\#} \dot{r}_1 \quad \Rightarrow \quad \mathcal{S}_1 = \{\dot{q}_1 + P_1 v_1, \; v_1 \in \mathbb{R}^n\}$$

$\boxed{\text{for } k = 2}$

$$\begin{cases} \dot{q}_2 = \arg \min_{\dot{q} \in \mathcal{S}_2} \frac{1}{2}\|\dot{q}\|^2 \\ \mathcal{S}_2 = \left\{ \arg \min_{\dot{q} \in \mathcal{S}_1} \frac{1}{2}\|J_2 \dot{q} - \dot{r}_2\|^2 \right\} \end{cases} \Rightarrow \quad \begin{aligned} \dot{q}_2 &= \dot{q}_1 + (J_2 P_1)^{\#} (\dot{r}_2 - J_2 \dot{q}_1) \\ \mathcal{S}_2 &= \{\dot{q}_2 + P_{A,2} v_2, \; v_2 \in \mathbb{R}^n\} \end{aligned}$$

Recursive solution with priorities - 2

generalizing to step k

$\dot{q}_{k-1}$

prioritized solution
up to task $k-1$

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

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

LQ problem
for k-th task

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

recursive formula
(Siciliano, Slotine:
ICAR 1991)

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

initialization

$$\overbrace{\begin{array}{r} \dot{q}_0 = 0 \\ P_{A,0} = I \end{array}}$$

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

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

Recursive solution with priorities
properties and implementation

- the solution considering the first k tasks with their priority

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

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

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

since

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

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

- recursive expression also for the null-space projector

$$\boxed{\boldsymbol{P}_{A,k} = \boldsymbol{P}_{A,k-1} - \left(\boldsymbol{J}_k \boldsymbol{P}_{A,k-1}\right)^{\#} \boldsymbol{J}_k \boldsymbol{P}_{A,k-1}} \qquad \boldsymbol{P}_{A,0} = \boldsymbol{I}$$

(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
(some still on-going research)

- 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
    - 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

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

- 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

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

needed to damp/stabilize self-motions in the null space ($K_D > 0$)

Dynamic redundancy resolution

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

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

$N \times N$ 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(°)
- closed-form expressions can be obtained by the solution formula(°) (assuming a full rank Jacobian J)

(°) in block *Kinematic redundancy - Part 1,* slide #26

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

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

(inverse inertia weighted) torque norm

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

Closed-form solutions

minimum torque norm solution

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

- 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

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

- good performance in general, to be preferred

minimum (inverse inertia weighted) torque norm solution

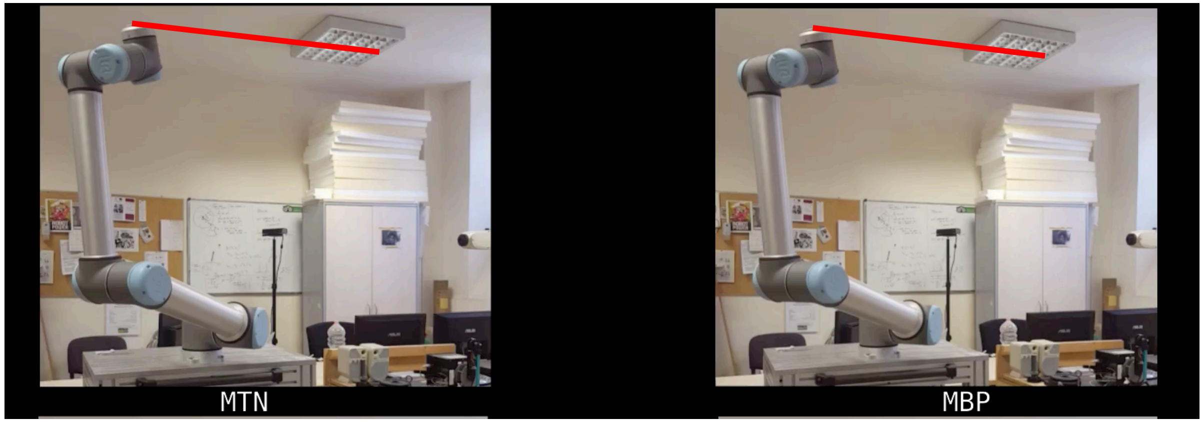
$$\frac{1}{2}\|\tau\|^2_{M^{-1}} \Rightarrow \tau_3 = J^T(q)\left(J(q)M^{-1}(q)J^T(q)\right)^{-1}\left(\ddot{r} - \dot{J}(q)\dot{q} + J(q)M^{-1}(q)n(q,\dot{q})\right)$$

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

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

Stabilizing the minimum torque solution

Universal Robots UR-10 (6-dof)



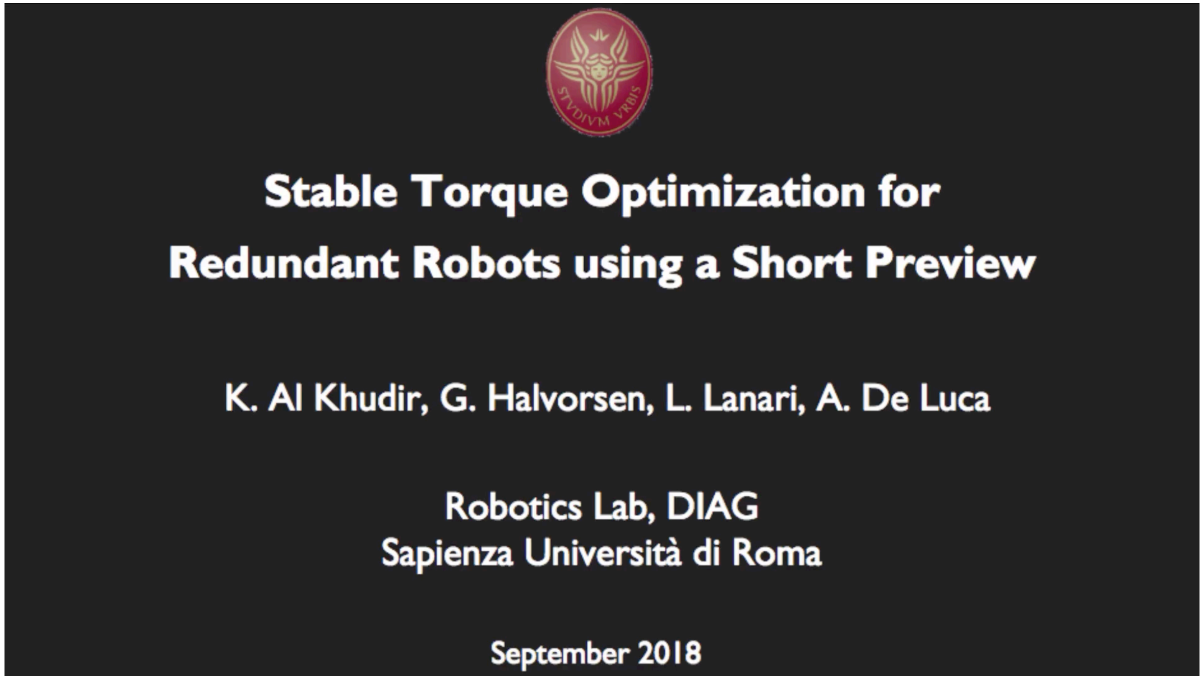
video

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

versus

video

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



- 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 $K_P, K_D > 0$)

$$\dot{r} \Rightarrow \dot{r}_d + K(r_d - r)$$ in velocity-based...

$$\ddot{r} \Rightarrow \ddot{r}_d + K_D(\dot{r}_d - \dot{r}) + K_P(r_d - r)$$ ...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

q_m

q_b

$$r = f(q) \in \mathbb{R}^M$$

(task output, e.g., the E-E pose)

kinematic model of the wheeled base (subject to nonholonomic constraints)

$$q = \begin{pmatrix} q_b \\ q_m \end{pmatrix} \in \mathbb{R}^N \qquad \begin{aligned} \dot{q}_b &= G(q_b)u_b \\ \dot{q}_m &= u_m \end{aligned}$$

$$u = \begin{pmatrix} u_b \\ u_m \end{pmatrix} \in \mathbb{R}^{N_u} \qquad N_u \leq N$$

Mobile manipulator Jacobian

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

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

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

$$= \boxed{J_{NMM}(q)}\,u$$

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

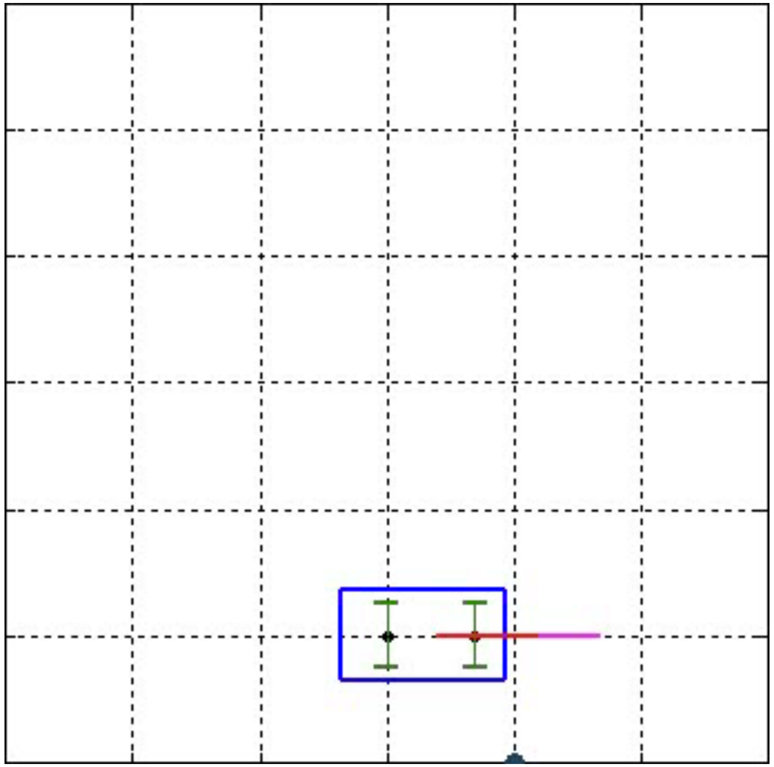
- … most previous results follow by just replacing

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

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

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

within a given convex set

solution set, with only equality constraints

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

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

solution set, with only inequality constraints

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

$$\text{s.t.} \quad \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{d} \leq \boldsymbol{w} \qquad \boldsymbol{w} \in \mathbb{R}^m_+$$

(non-negative) slack variables

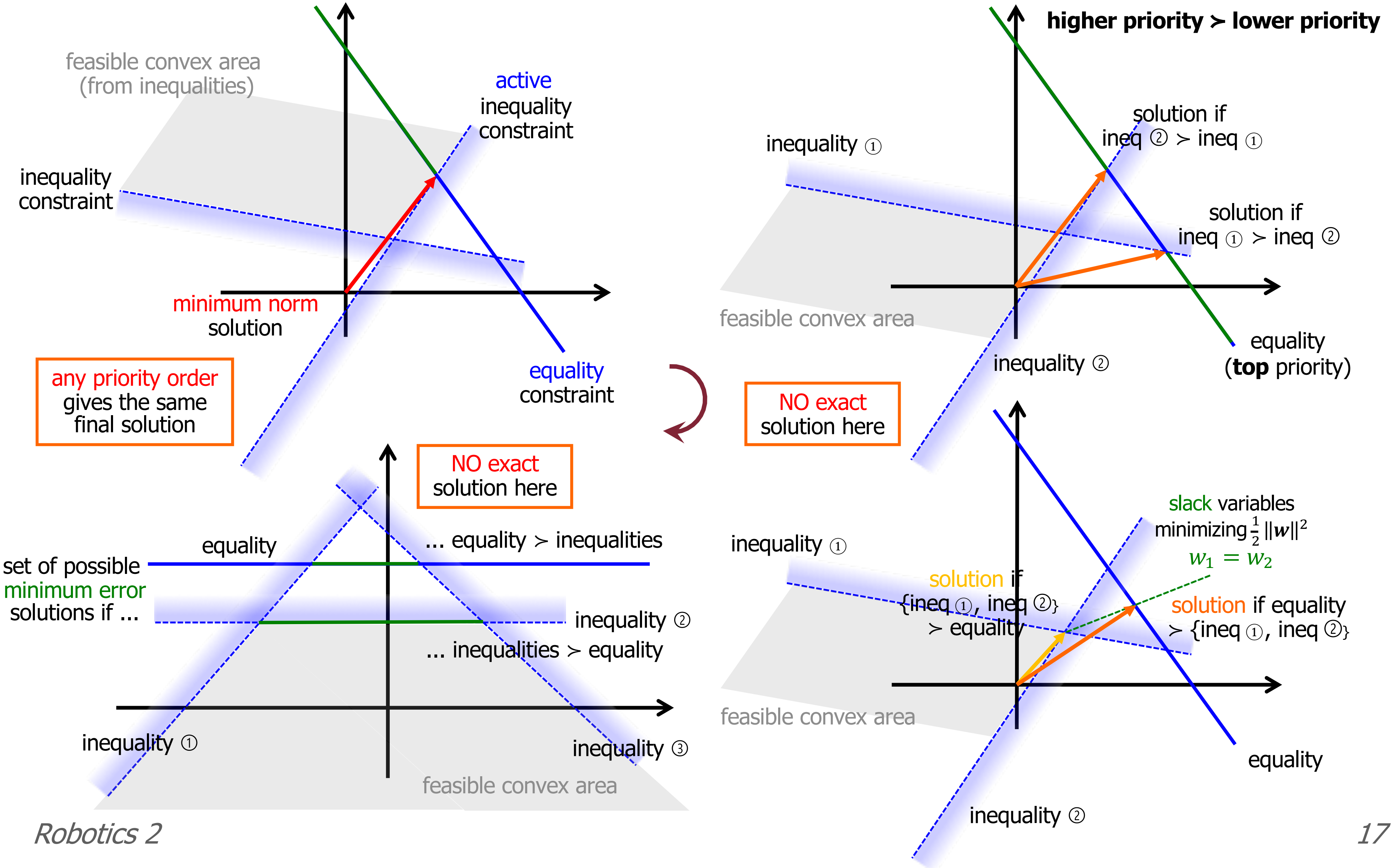
$$\text{given } \dot{\boldsymbol{q}}^* \in \mathcal{S}_{ineq} \quad \Rightarrow \quad \mathcal{S}_{ineq} = \Omega \cap \begin{cases} \boldsymbol{c}_j^T\dot{\boldsymbol{q}} \leq d_j, & \text{if } \boldsymbol{c}_j^T\dot{\boldsymbol{q}}^* \leq d_j \\ \boldsymbol{c}_j^T\dot{\boldsymbol{q}} = \boldsymbol{c}_j^T\dot{\boldsymbol{q}}^*,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

QP complete formulation

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

(possibly with prioritization of constraints)

Equality and inequality linear constraints

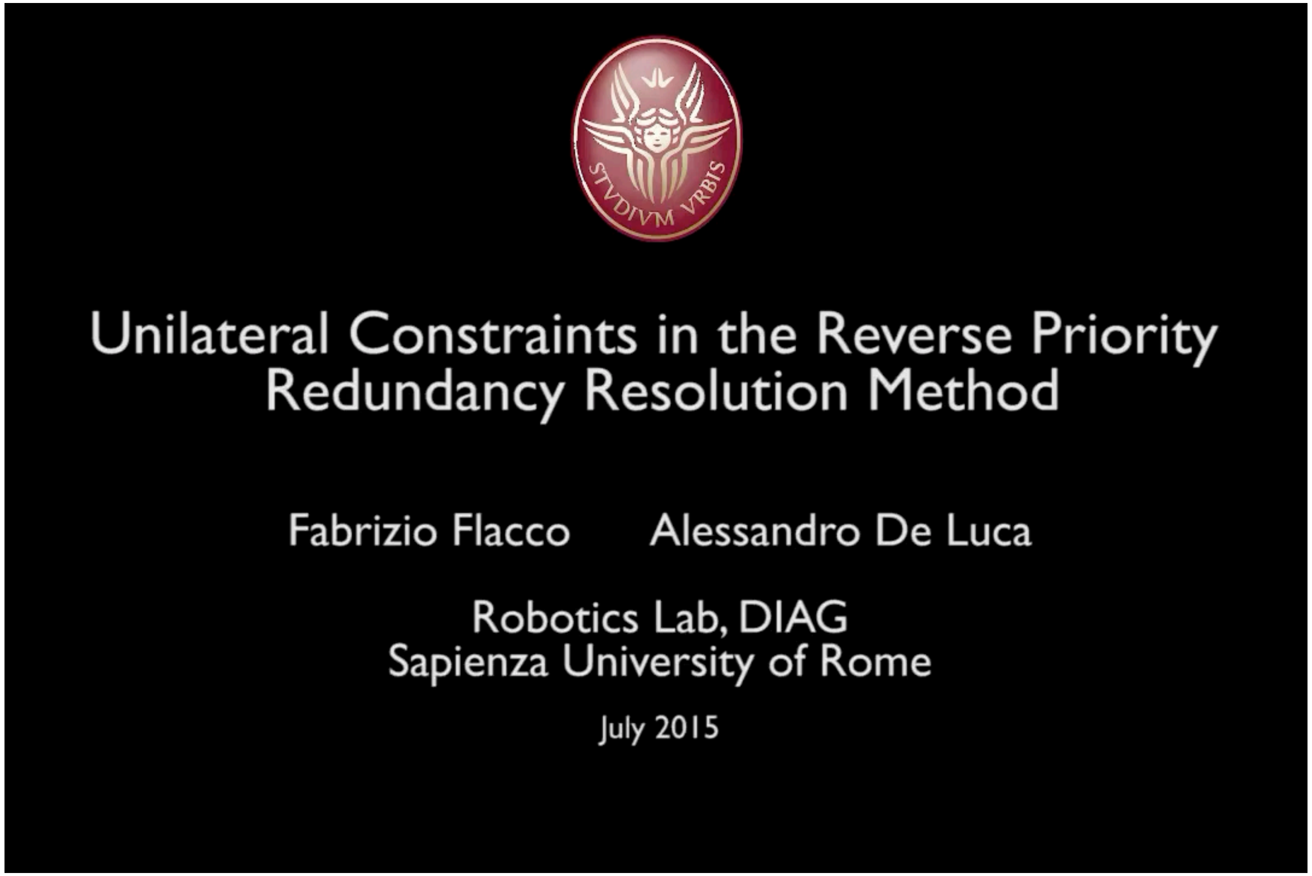


Equality and Inequality Tasks

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



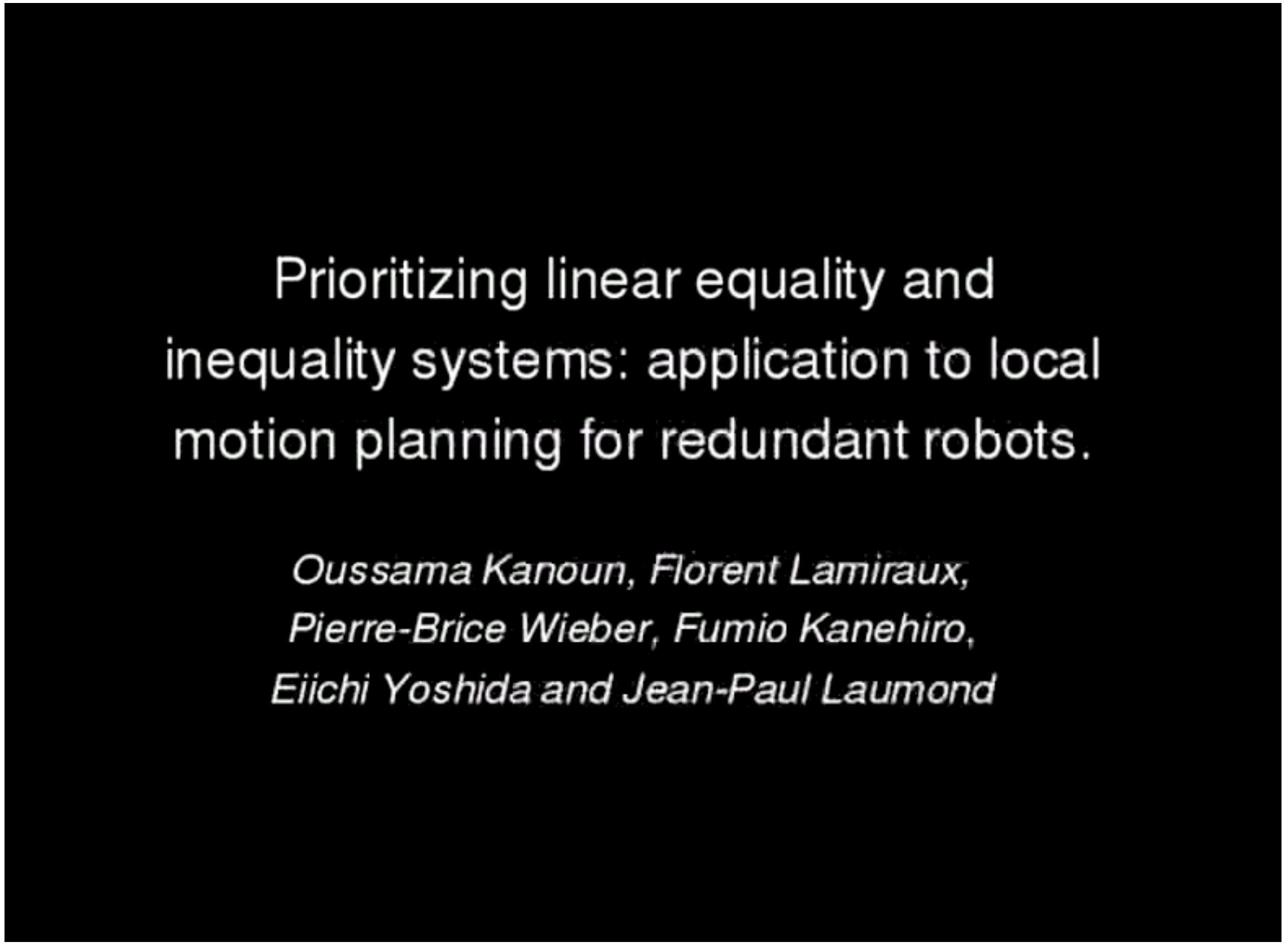
IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) 2015

Equality and Inequality Tasks

for the high-dof humanoid robot HRP2

- a systematic task priority approach, with several simultaneous tasks

video



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

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

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)

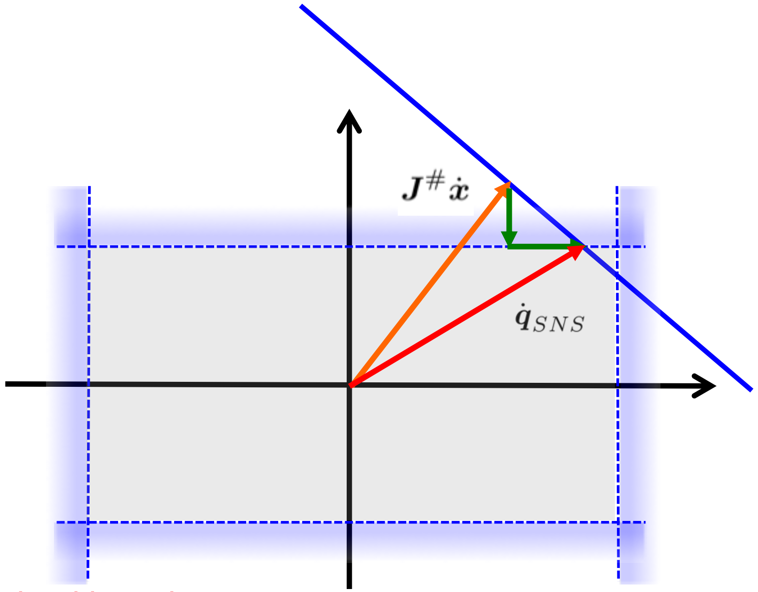
⬇

- saturate one overdriven joint command at a time, until a feasible and better performing solution is found ⇒ 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

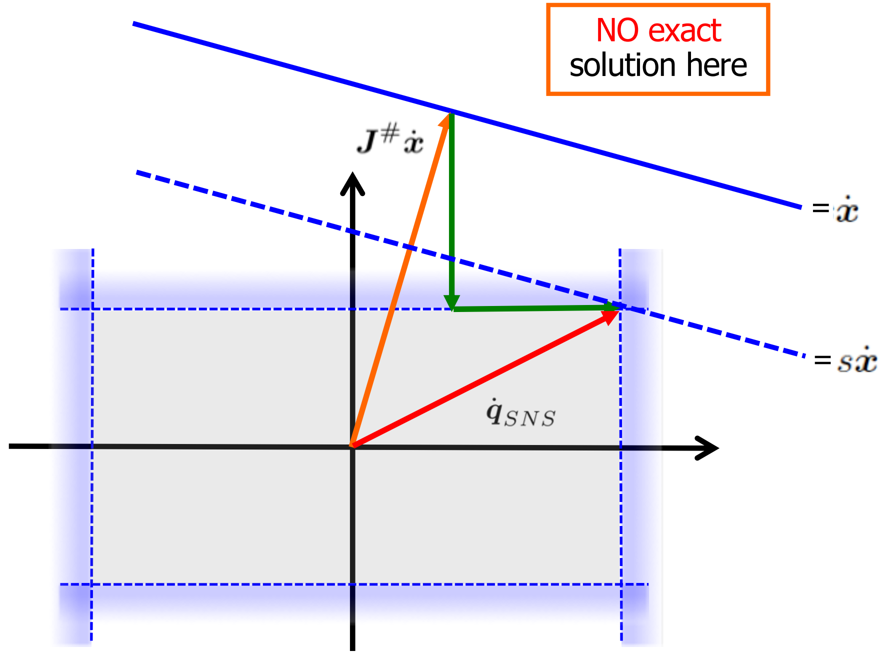
$$\dot{q}_{SNS} = (JW)^{\#} \underbrace{s}_{\text{scaling factor}} \dot{x} + \left(I - (J\underbrace{W}_{\text{diagonal 0/1 matrix}})^{\#} J\right) \underbrace{\dot{q}_N}_{\text{contains saturated joint velocities}}$$

Geometric view on SNS operation

in the space of velocity commands



hard bounds
(box inequality constraints)

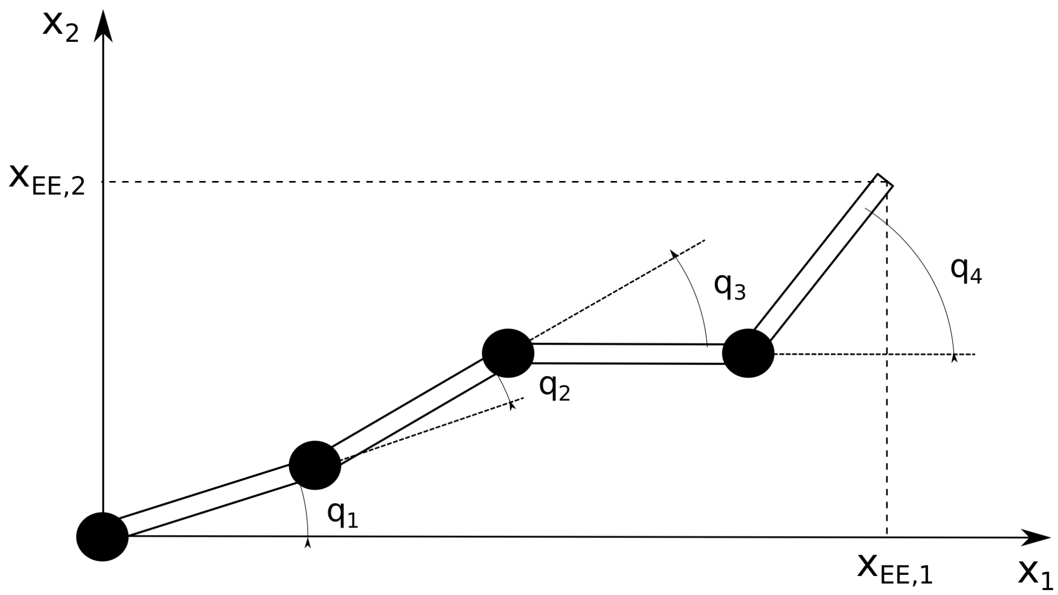


hard bounds
(box inequality constraints)

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: end-effector Cartesian position

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

manipulator configuration

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

differential map

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

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

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

task Jacobian

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

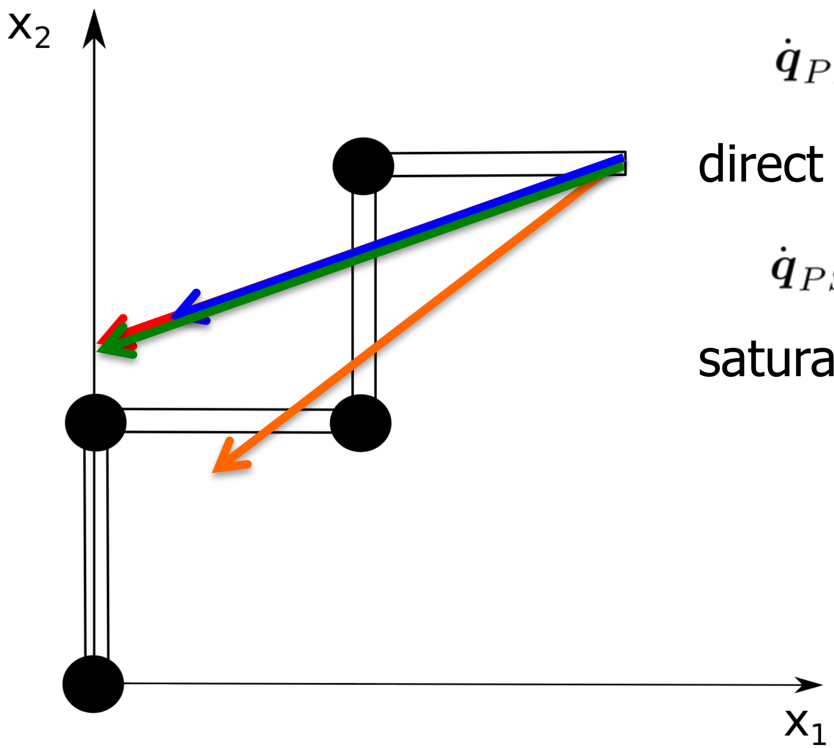
velocity limits $|\dot{q}_i| \leq V_i\,, i = 1 \ldots 4$ $\quad \boxed{V_1 = V_2 = 2 \quad V_3 = V_4 = 4\ [\mathrm{rad/s}]}$

Illustrative example - 2

current configuration $\boldsymbol{q} = \begin{pmatrix} \pi/2 & -\pi/2 & \pi/2 & -\pi/2 \end{pmatrix}^T$

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

desired end-effector velocity $\dot{\boldsymbol{x}} = \begin{pmatrix} -4 & -1.5 \end{pmatrix}^T$



2.0 -2.0

$$\dot{\boldsymbol{q}}_{PS} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} = \begin{pmatrix} 2.4545 & -2.1364 & 1.2273 & -3.3636 \end{pmatrix}^T$$

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

$$\dot{\boldsymbol{q}}_{PS} = s\boldsymbol{J}^{\#}\dot{\boldsymbol{x}} = \begin{pmatrix} 2.0 & -1.74 & 1.0 & -2.74 \end{pmatrix}^T$$

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

$$\dot{\boldsymbol{x}}_{SNS} = \dot{\boldsymbol{x}} - J_1 V_1 = \begin{pmatrix} J_2 & J_3 & J_4 \end{pmatrix} \begin{pmatrix} \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{pmatrix}$$

$$\dot{\boldsymbol{q}}_{SNS} = \left(V_1 \quad \left[\begin{pmatrix} J_2 & J_3 & J_4 \end{pmatrix}^{\#} \dot{\boldsymbol{x}}_{SNS} \right]^T \right)^T$$

$$= \begin{pmatrix} 2 & -1.8333 & 1.8333 & -3.6667 \end{pmatrix}^T$$

Joint velocity bounds

joint space limits

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

joint velocity bounds

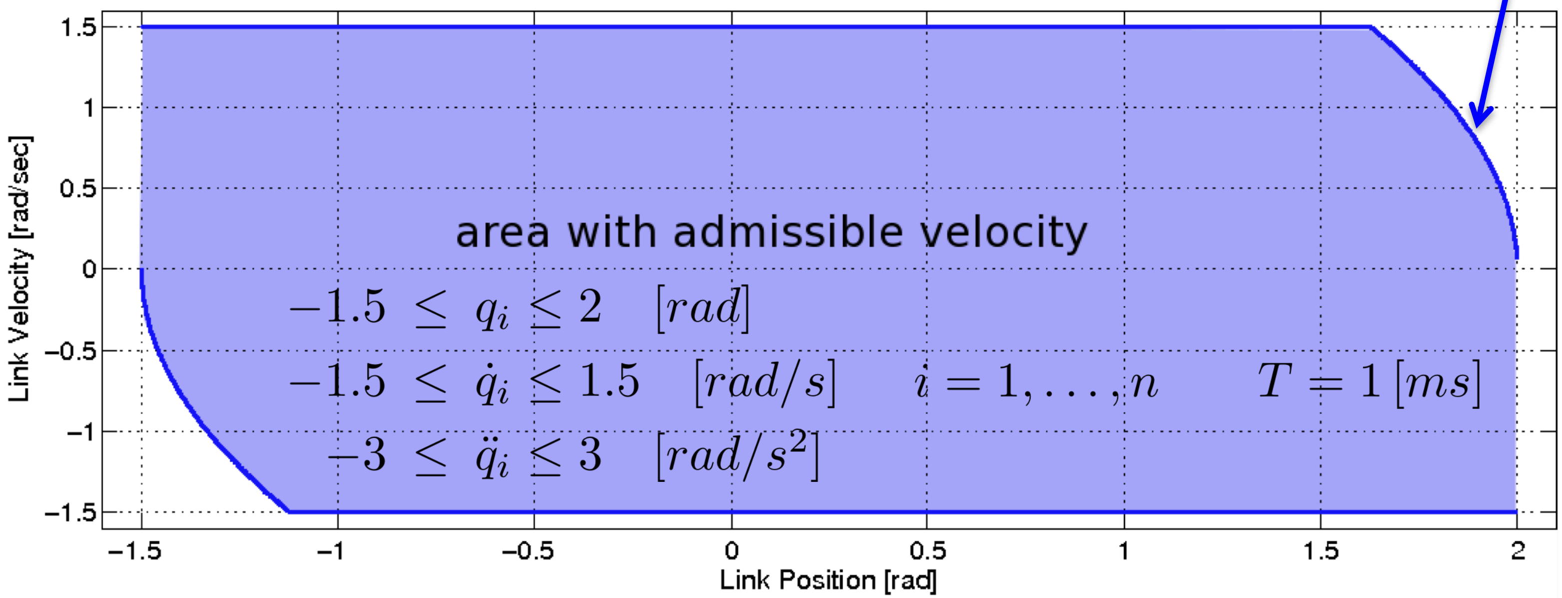
$$\boxed{\dot{\boldsymbol{Q}}_{min}(t_k) \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{Q}}_{max}(t_k)}$$

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

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

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

smooth velocity bound “anticipates” the reaching of a hard limit



SNS at velocity level

Algorithm 1

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

repeat

limit_exceeded = FALSE

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

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

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

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

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

if {task scaling factor} $> s^*$ **then**

$s^* =$ {task scaling factor}

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

end if

$j =$ {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 rank($\boldsymbol{J}\boldsymbol{W}$) $< m$ **then**

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

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

limit_exceeded = FALSE (**outputs solution**)

end if

end if

until limit_exceeded = TRUE

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

initialization

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

$\dot{\boldsymbol{q}}_N$: 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

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

repeat

limit_exceeded = FALSE

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

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

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

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

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

if {task scaling factor} $> s^*$ **then**

s^* = {task scaling factor}

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

end if

j = {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 rank($\boldsymbol{J}\boldsymbol{W}$) $< m$ **then**

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

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

limit_exceeded = FALSE (**outputs solution**)

end if

end if

until limit_exceeded = TRUE

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

compute the joint velocity with initialized values

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

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

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

repeat

limit_exceeded = FALSE

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

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

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

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

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

if {task scaling factor} $> s^*$ **then**

$s^* =$ {task scaling factor}

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

end if

$j =$ {the most critical joint}

$W_{jj} = 0$

$$\dot{q}_{N,j} = \left\{ \begin{array}{ll} \dot{Q}_{max,j} & \text{if } \dot{\bar{q}}_j > \dot{Q}_{max,j} \\ \dot{Q}_{min,j} & \text{if } \dot{\bar{q}}_j < \dot{Q}_{min,j} \end{array} \right.$$

if rank$(\boldsymbol{J}\boldsymbol{W}) < m$ **then**

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

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

limit_exceeded = FALSE *(*outputs solution*)*

end if

end if

until limit_exceeded = TRUE

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

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 no joint limit is exceeded

Task scaling factor

Algorithm 2

---

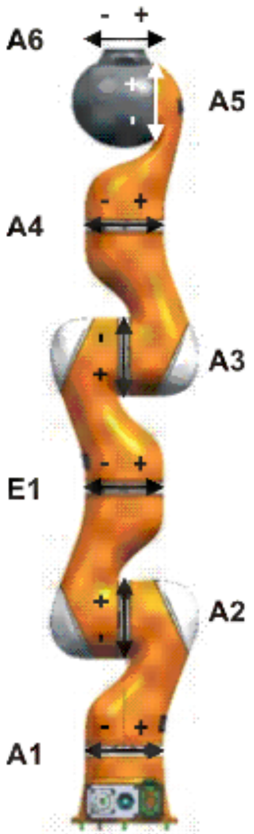
function getTaskScalingFactor($\boldsymbol{a}$, $\boldsymbol{b}$)
for $i = 1 \rightarrow n$ **do**
  $S_{min,i} = \left(\dot{Q}_{min,i} - b_i\right)/a_i$
  $S_{max,i} = \left(\dot{Q}_{max,i} - b_i\right)/a_i$
  if $S_{min,i} > S_{max,i}$ **then**
    {switch $S_{min,i}$ and $S_{max,i}$}
  end if
end for
$s_{max} = \min_i \{S_{max,i}\}$
$s_{min} = \max_i \{S_{min,i}\}$
the most critical joint $= \text{argmin}_i \{S_{max,i}\}$
if $s_{min} > s_{max}$ .OR. $s_{max} < 0$ .OR. $s_{min} > 1$ **then**
  task scaling factor $= 0$
else
  task scaling factor $= s_{max}$
end if

---

yields the best task scaling factor (i.e., closest to the ideal value = 1) for the most critical joint in the current joint velocity solution

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

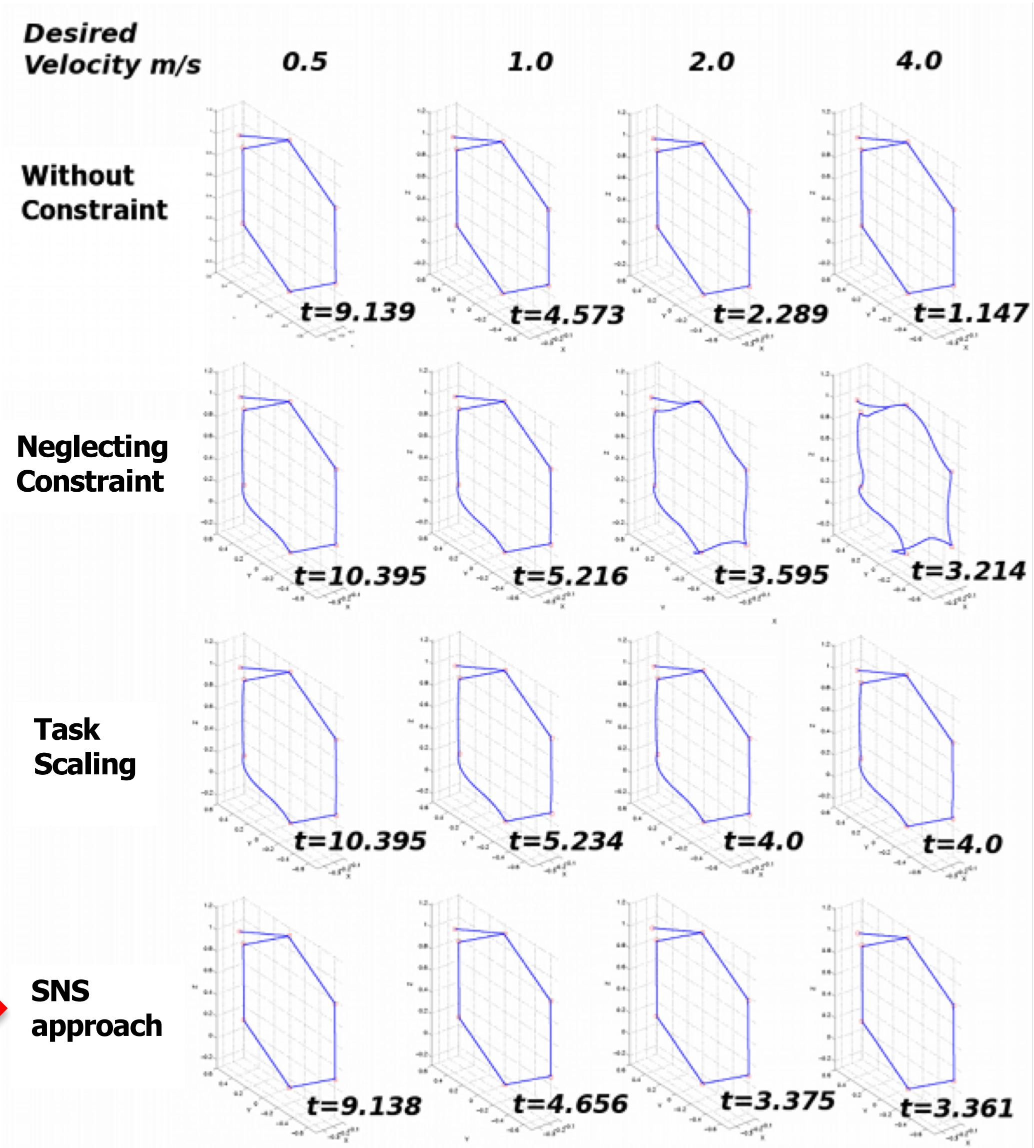
$$\boldsymbol{Q}_{max} = (170, 120, 170, 120, 170, 120, 170) \text{ [deg]}$$

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

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

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

Simulation results



for increasing V

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

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

task redundancy degree = 7 – 3 = 4

robot starts at the configuration

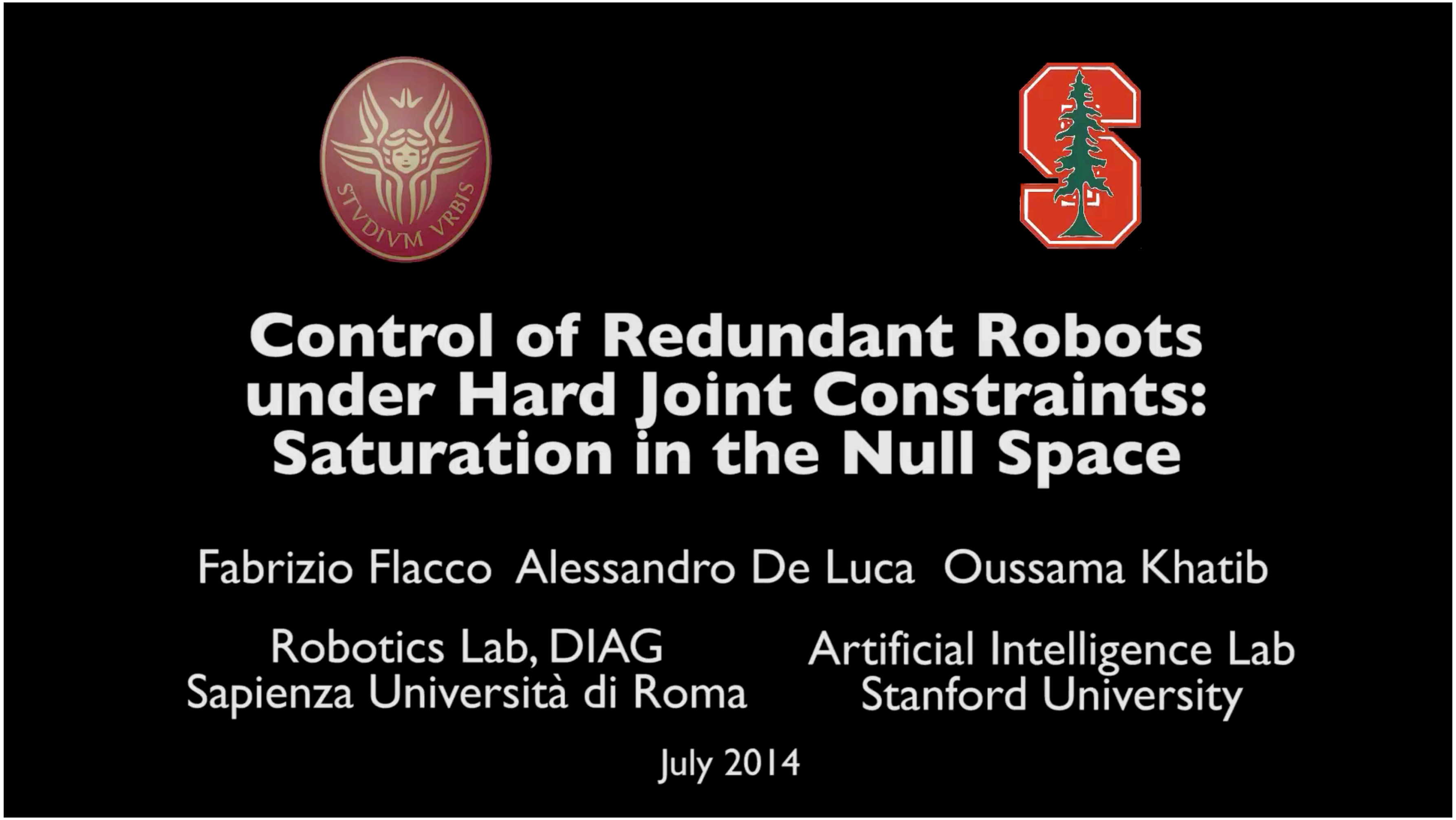
$\boldsymbol{q}(0) = (0, 45, 45, 45, 0, 0, 0)$ [deg]

(with a small initial approaching phase)

Experimental results

KUKA LWR IV with hard joint-space limits

video



IEEE Transactions on Robotics 2015

Variations of the SNS method

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

video

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

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)

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 redundnat robots using a short preview,” *IEEE Robotics and Automation Lett.*, vol 4, no, 2, pp. 2046-2057, 2019

Appendix A - Recursive Task Priority

proof of recursive expression for null-space projector

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

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

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

$$E = B(I - A^{\#}A)$$

- (i) $$P_{A,k} = I - J_{A,k}^{\#} J_{A,k} = I - \begin{pmatrix} J_{A,k-1} \\ J_k \end{pmatrix}^{\#} \begin{pmatrix} J_{A,k-1} \\ J_k \end{pmatrix}$$

$$= I - \begin{pmatrix} J_{A,k-1}^{\#} - TJ_kJ_{A,k-1}^{\#} & T \end{pmatrix} \begin{pmatrix} J_{A,k-1} \\ J_k \end{pmatrix}$$

$$= I - J_{A,k-1}^{\#} J_{A,k-1} + TJ_kJ_{A,k-1}^{\#} J_{A,k-1} - TJ_k$$

$$= P_{A,k-1} - TJ_kP_{A,k-1}$$

- (ii) $$T = (J_kP_{A,k-1})^{\#} + X\left(I - (J_kP_{A,k-1})(J_kP_{A,k-1})^{\#}\right)$$

$$\Rightarrow \quad TJ_kP_{A,k-1} = (J_kP_{A,k-1})^{\#} J_kP_{A,k-1}$$

- (i) + (ii) ⇒ Q.E.D.

- if k-th task is scalar

$$J_k = \text{single row } j_k^T$$

$$P_{A,k} = P_{A,k-1} - \frac{P_{A,k-1} j_k j_k^T P_{A,k-1}}{\|P_{A,k-1} j_k\|^2}$$

(Greville formula)