Robotics 2

Robots with kinematic redundancy

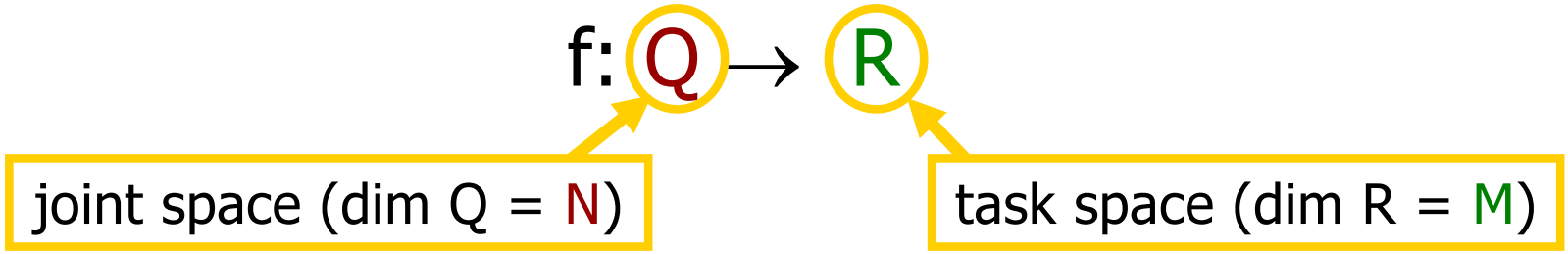
Prof. Alessandro De Luca

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

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 tasks and their dimensions

TASKS [for the 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	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

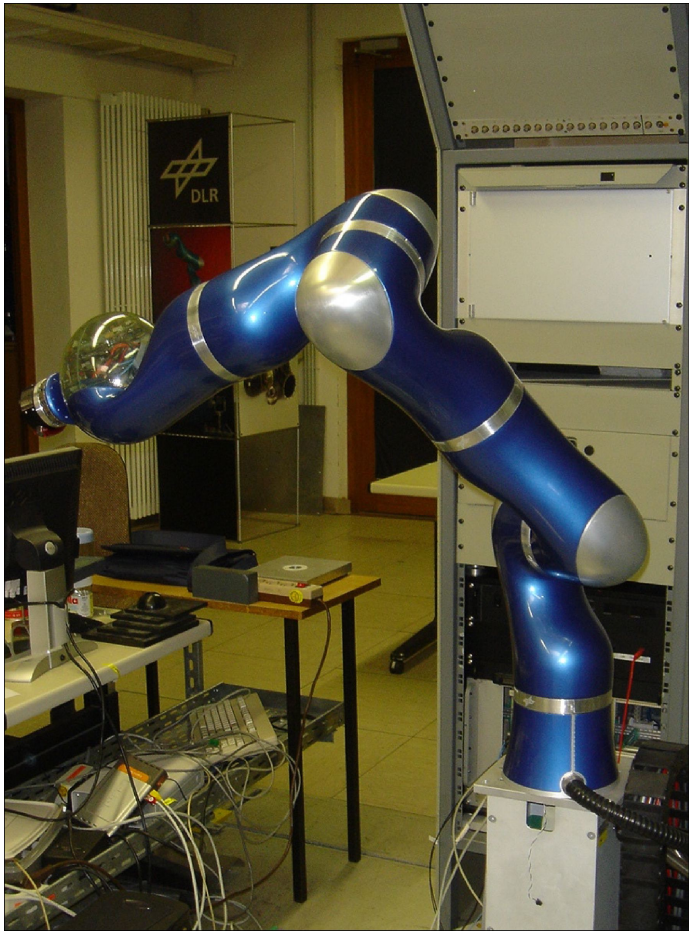
- 6R robot mounted on a linear track/rail
  - for positioning and orienting its end-effector in 3D space
- 6-dof robot used for arc welding tasks
  - task does not prescribe the final roll angle of the welding gun
- manipulator on a mobile base
- dexterous robot hands
- team of cooperating manipulators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

Uses of robot redundancy

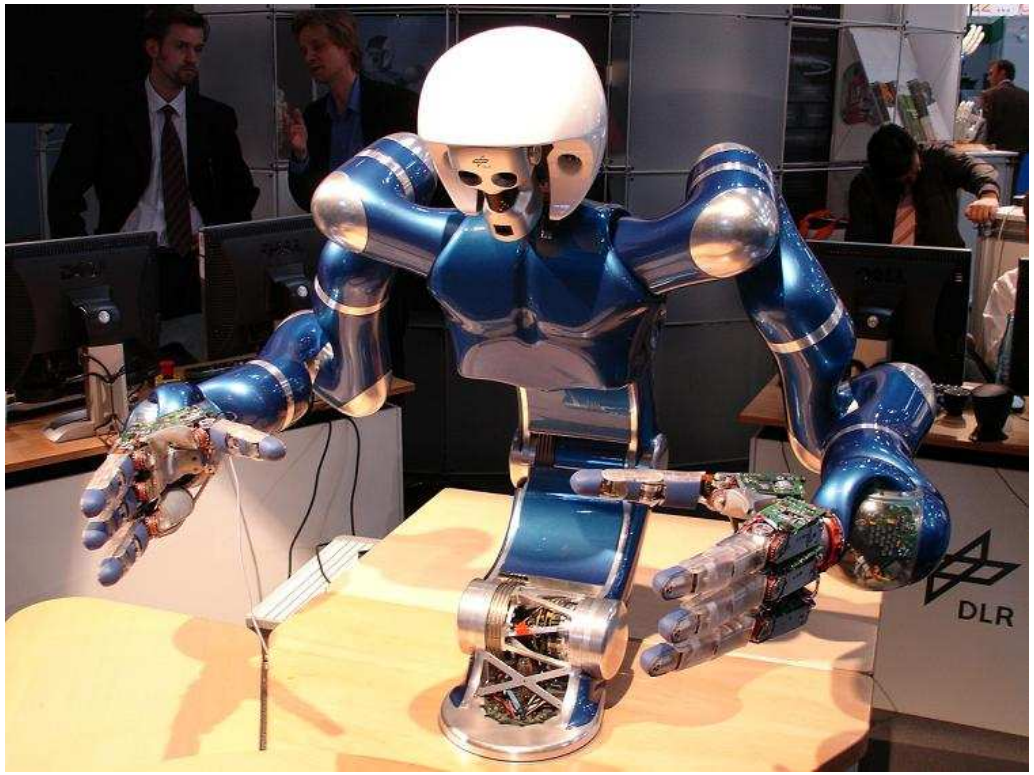
- avoid collision with obstacles (in Cartesian space) …
- … or kinematic singularities (in joint space)
- stay within the admissible joint ranges
- increase manipulability in specified directions
- uniformly distribute/limit joint velocities and/or accelerations
- minimize energy consumption or needed 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

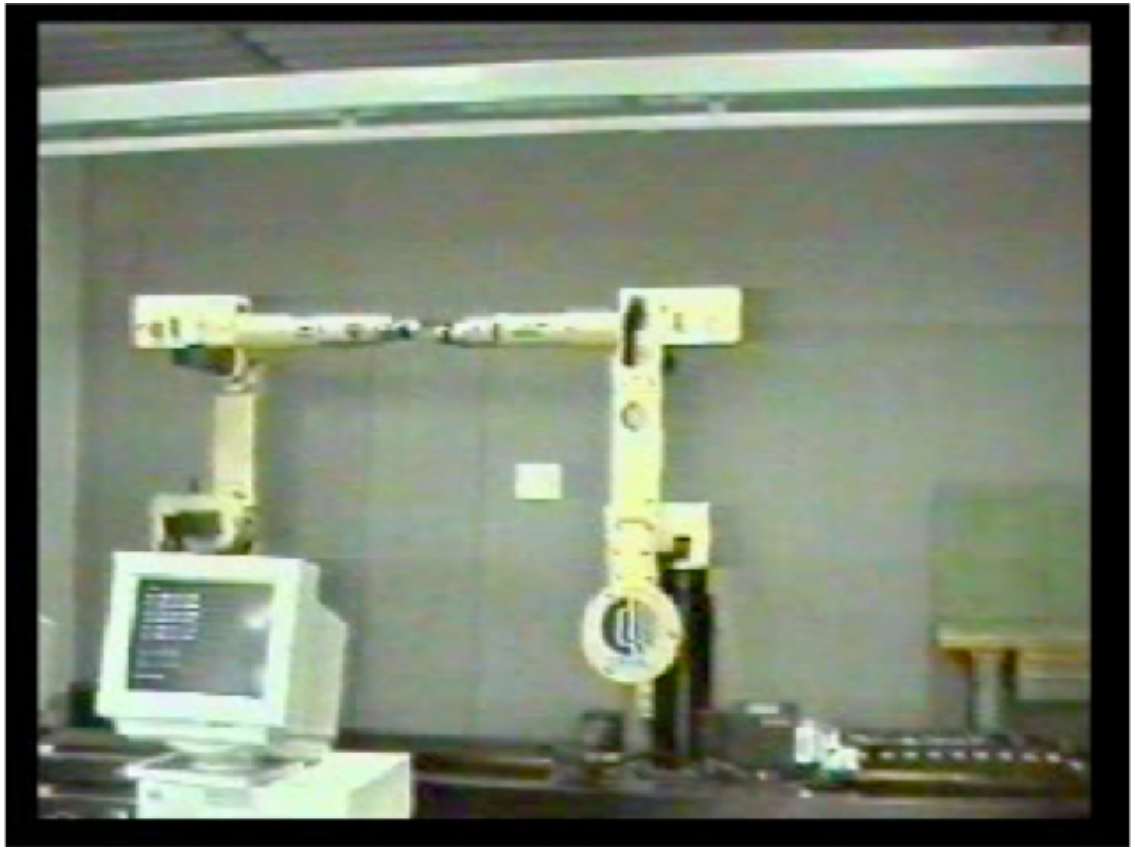
* = one joint is dependent on the motion of the other two

Justin carrying a trailer

video

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

Dual-arm redundancy



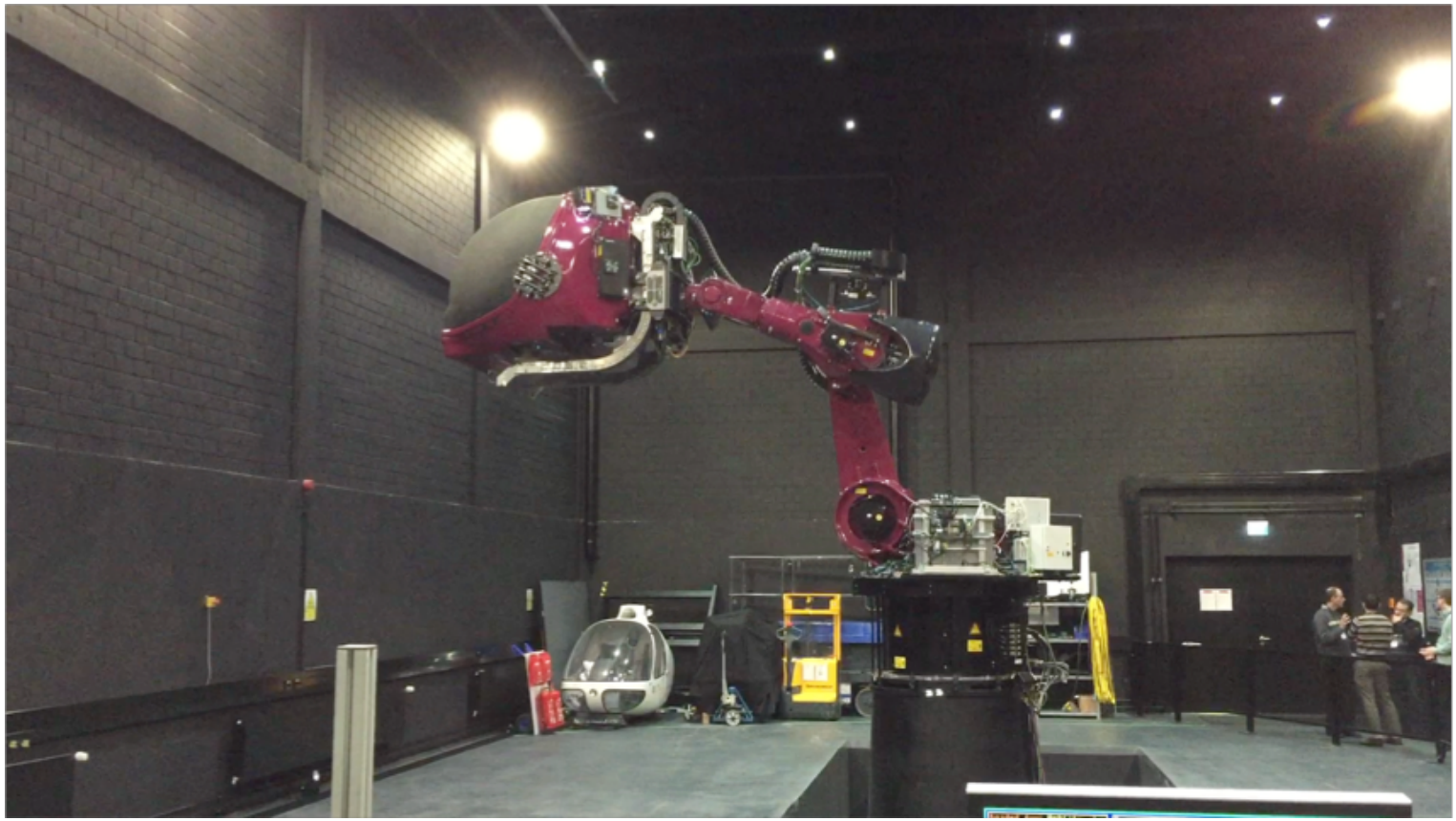
video

DIS, Uni Napoli

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

Motion cueing from redundancy

video



Max Planck Institute for Biological Cybernetics, Tübingen

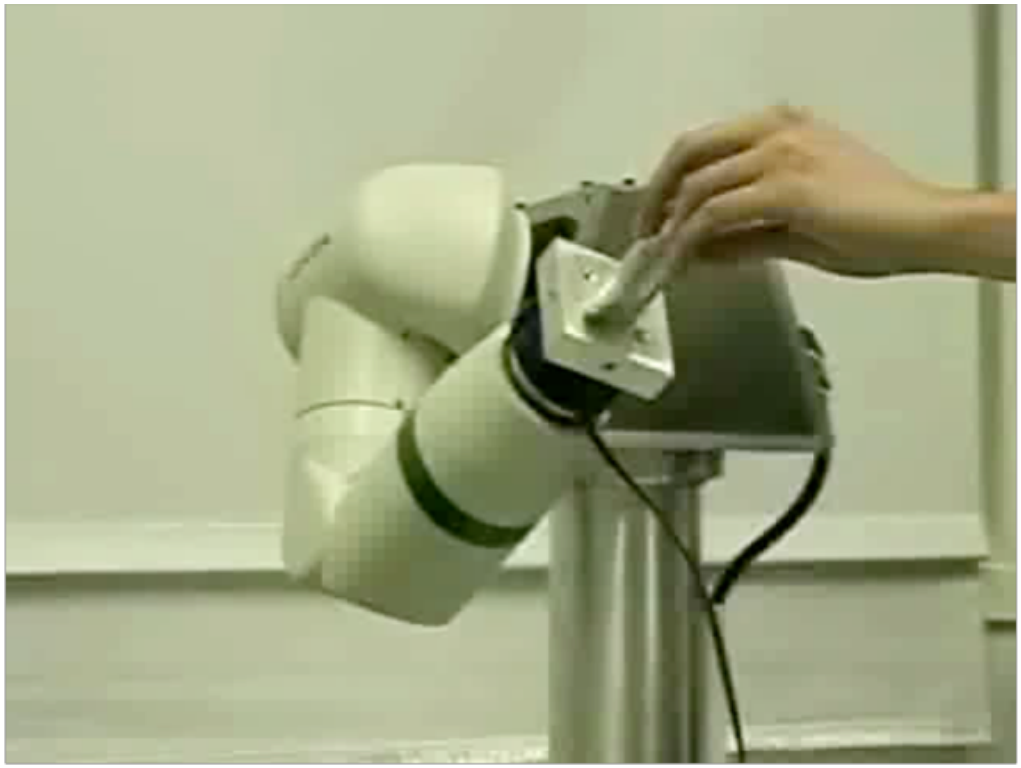
a 6R KUKA KR500 mounted on a linear track (+1P) with a sliding cabin (+1R), used as an emulation platform for dynamic human perception (N-M=2)

Self-motion

video

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

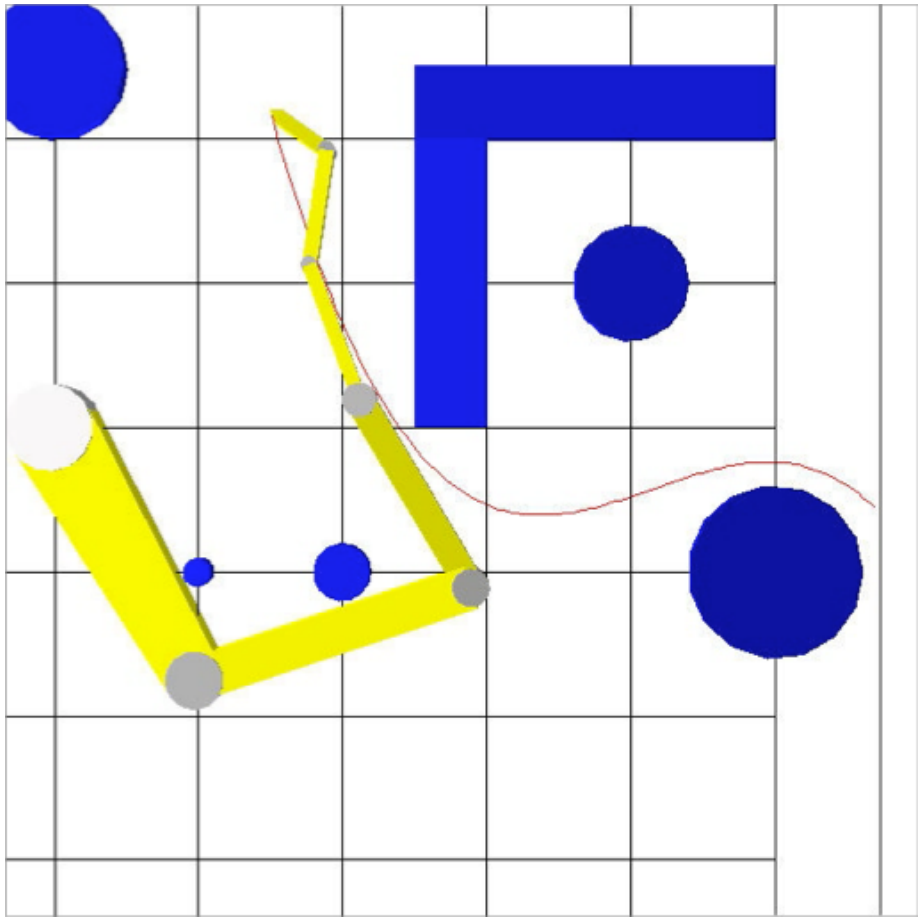
video



Nakamura's Lab, Uni Tokyo

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

Obstacle avoidance



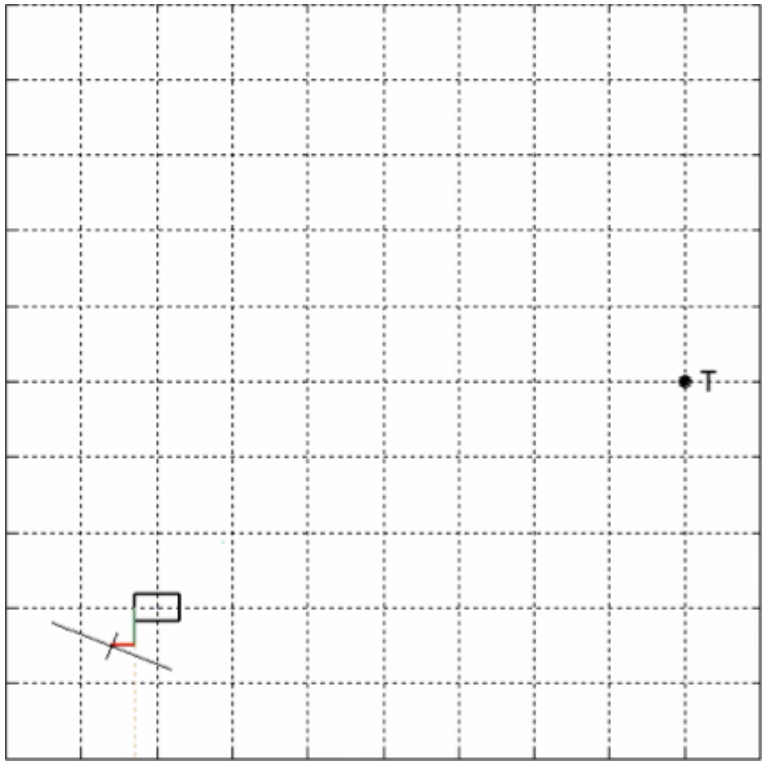
video

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

Mobile manipulators

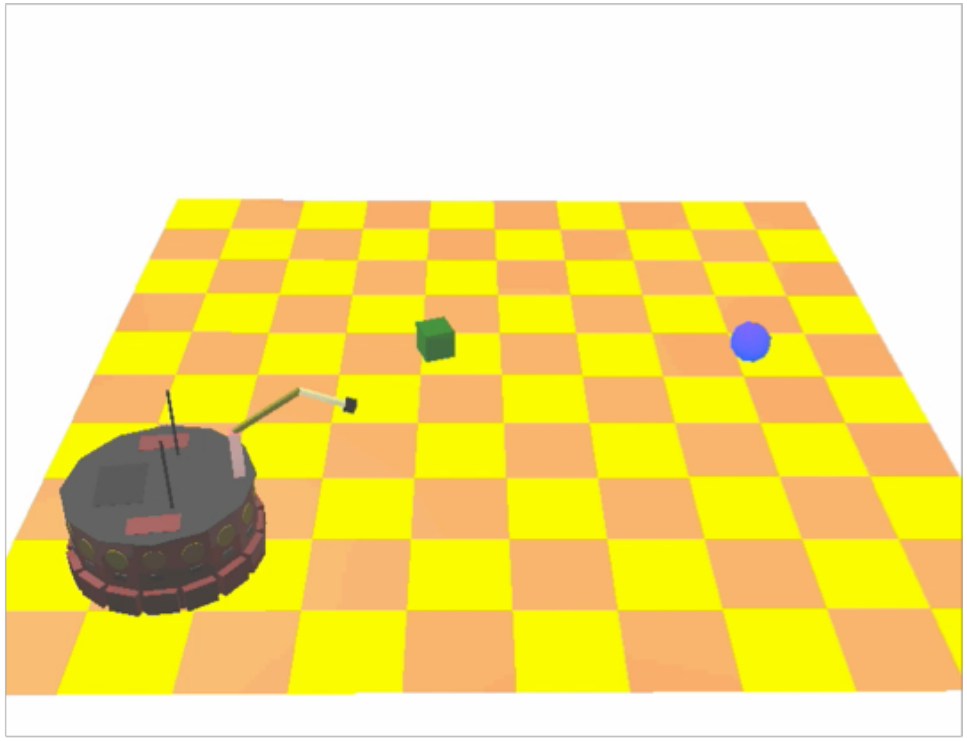
N_u available commands < N generalized coordinates!!

video



Unicycle + 2R planar arm (N=5, N_u=4):
E-E trajectory control on a circle,
with pointing task for first link (N_u-M=1)

video



Magellan + 3R arm (N=6, N_u=5):
E-E trajectory control on a circle,
with E-E pointing task (N_u-M=0!!)

Humanoid robots – HRP2

a hyper-redundant system, but with a few non-actuated dofs (at the base!)

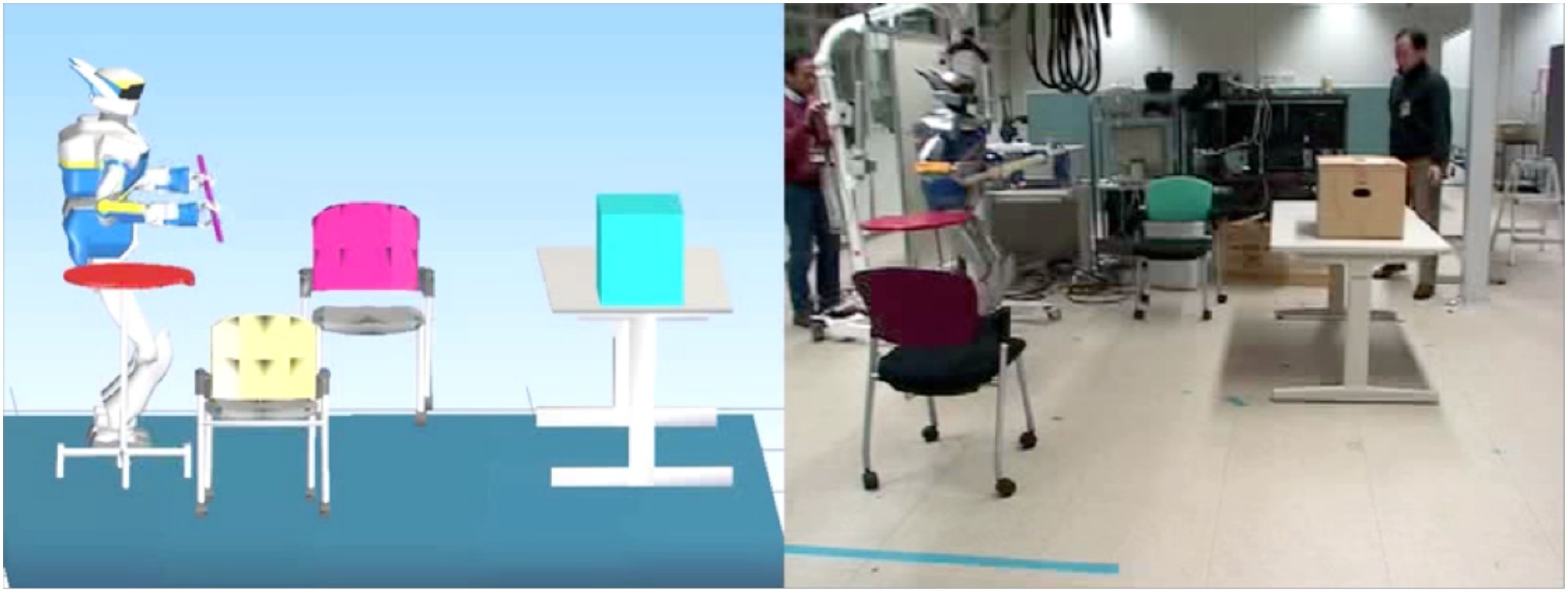
Dimensions		Height 1540mm Width 600mm Depth 340 mm (chest)
Total mass		58kg
Degrees of freedom		30 axes
Biped walk speed		up to 2 km/h
Hand grip		2 kgf
Sensors	Hinges	incremental encoders
	Vision	3-lens stereo camera
	Chest	3-axes gyro, 3-axes accelerometer
	Arms	6-axes force sensors
	Legs	6-axes force sensors
Motor Drivers		48V, 20 A (Imax), 2 axes/driver x 16
Power		NiMH Batteries 48V, 18Ah

built by Kawada Industries, Inc. for the Humanoid Robotics Project (HRP)
sponsored by the Japanese Ministry of Economy, Trade, and Industry

HRP2 humanoid robot in action

whole body motion/navigation avoiding obstacles

video



JRL CNRS-AIST Joint Research Lab, Toulouse-Tsukuba

E. Yoshida, C. Esteves, T. Sakaguchi, J.-P. Laumond, and K. Yokoi
"Smooth collision avoidance: Practical issues in dynamic humanoid motion"
IEEE Int. Conf. on Intelligent Robotics and Systems, 2006

Disadvantages of redundancy

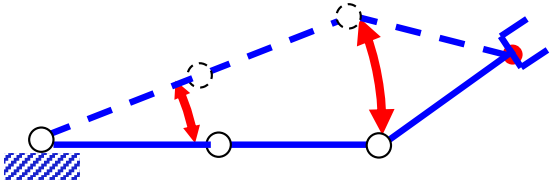
- potential benefits should be traded off against
  - a greater structural complexity of construction
    - mechanical (more links, transmissions, ...)
    - more actuators, sensors, ...
    - costs
  - more complicated algorithms for inverse kinematics and motion control

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 = constant)

N = 3 > 2 = M



r = constant
E-E position

- 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**

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

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

$\dot{q}(kT_s)$ ← ON-LINE

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

discrete-time form

- **Global**

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

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

2. null-space methods

   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 ≤ N-M further auxiliary tasks (when S = N-M, the problem has been "squared")

1 Jacobian-based methods

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

$J = \underbrace{\square}_{N}\}M$ $\qquad \dot{q} = K(q)\dot{r}$ $\qquad K = \underbrace{\square}_{M}\}N$

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

(➡ K = generalized inverse of J)

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

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)

Pseudoinverse

$$\dot{q} = J^{\#}(q)\dot{r}$$ ... a very common choice: $K = J^{\#}$

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

$$J\,J^{\#}\,J = J \qquad J^{\#}\,J\,J^{\#} = J^{\#}$$
$$\left(J\,J^{\#}\right)^{T} = J\,J^{\#} \qquad \left(J^{\#}\,J\right)^{T} = J^{\#}\,J$$

- if J is full (row) rank, $J^{\#} = J^{T}(J\,J^{T})^{-1}$; 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 the task is feasible ($\dot{r} = \mathcal{R}(J(q))$), there will be no task error

Weighted pseudoinverse

$$\dot{q} = J_W^{\#}(q)\dot{r}$$ another choice: $K = J_W^{\#}$

- if J is full (row) rank, $J_W^{\#} = W^{-1}J^T(JW^{-1}J^T)^{-1}$
- the solution $\dot{q}$ minimizes the weighted norm

$$\|\dot{q}\|_W^2 = \dot{q}^T W\, \dot{q}$$ $W > 0$, symmetric (often diagonal)

- large weight $W_i \Rightarrow$ small $\dot{q}_i$ (e.g., weights can be chosen proportionally to the inverse of the joint ranges)
- it is NOT a "pseudoinverse" (4th relation does not hold ...) but 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

$$\Sigma = \left(\begin{array}{ccc|c} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & 0_{M\times(N-M)} \\ & & \sigma_M & \end{array}\right) \qquad \begin{array}{c} \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\rho > 0 \\ \sigma_{\rho+1} = \cdots = \sigma_M = 0 \end{array}$$

singular values of J

where ρ = rank(J) ≤ M, so that their product is

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

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

$$Jv_i = \sigma_i u_i \quad (\mathrm{i} = 1,\cdots,\rho) \qquad\qquad Jv_i = 0 \quad (\mathrm{i} = \rho+1,\cdots,N)$$

Computation of pseudoinverses

- show that the pseudoinverse of J is equal to

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

  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

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

applies equally to square and non-square matrices

Singularity robustness
Damped Least Squares (DLS)

unconstrained minimization of a **suitable** objective function

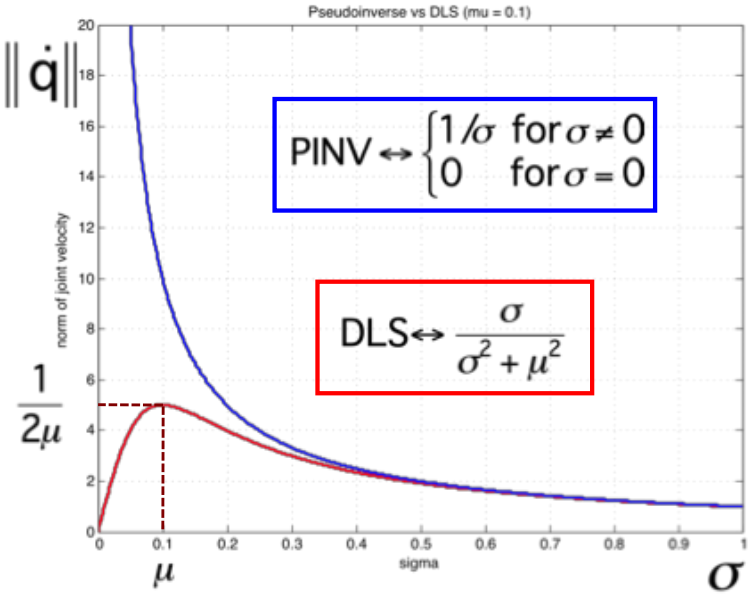
$$\min_{\dot{q}} H(\dot{q}) = \frac{\mu^2}{2}\|\dot{q}\|^2 + \frac{1}{2}\|\dot{r} - J\dot{q}\|^2$$

compromise between large joint velocity and task accuracy

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

- induces a robust behavior when crossing singularities, but in its basic version gives always a task error $\dot{e} = \mu^2(J\,J^T + \mu^2 I_M)^{-1}\dot{r}$ (as in the N=M case)
  - thus, J_{DLS} is not a generalized inverse K
  - using SVD: $J = U\,\Sigma\,V^T \Rightarrow J_{DLS} = V\Sigma_{DLS}U^T$, $\Sigma_{DLS} = \begin{pmatrix} \underbrace{diag\left\{\frac{\sigma_i}{\sigma_i^2+\mu_i^2}\right\}}_{\rho\times\rho} & 0_{(M-\rho)\times(M-\rho)} \\ 0_{(N-M)\times\rho} & 0_{(N-M)\times(M-\rho)} \end{pmatrix}$
  - choice of a variable damping factor $\mu^2(q) \geq 0$, as a function of the minimum singular $\sigma_m(q)$ value of J ≅ measure of distance to singularity
  - numerical filtering: introduces damping only/mostly in non-feasible directions for the task (see Maciejewski and Klein, *J of Rob Syst*, 1988)

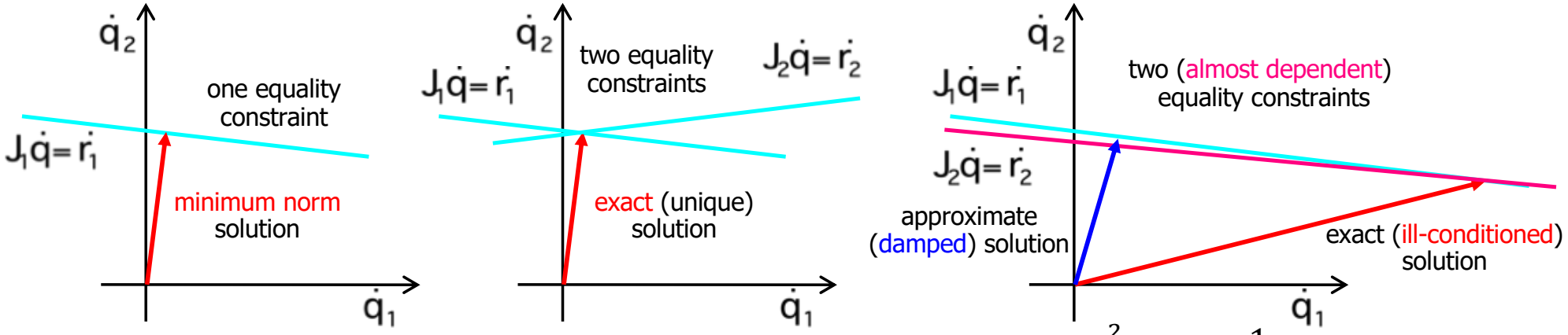
Behavior of DLS solution



a. comparison of joint velocity norm with PINV (pseudoinverse) or DLS solutions

- in a direction of a singular vector u, when the associated singular value $\sigma \longrightarrow 0$
- 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...)

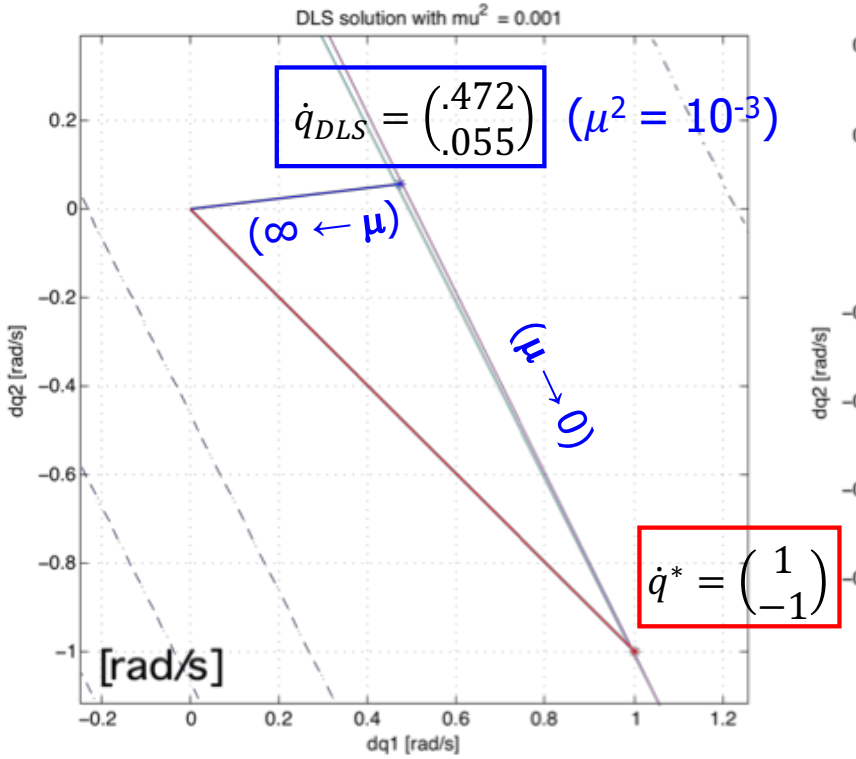
b. graphical interpretation of "damping" effect (here $M = N = 2$, for simplicity)

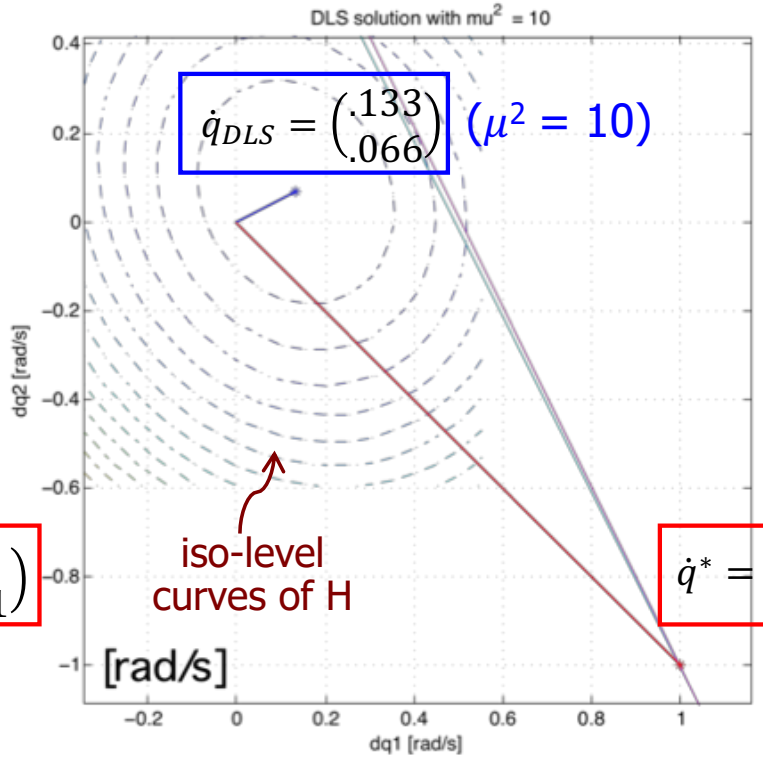


$$H(\dot{q}) = \frac{\mu^2}{2}\|\dot{q}\|^2 + \frac{1}{2}\|\dot{r} - J\dot{q}\|^2$$

Numerical example of DLS solution

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





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

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

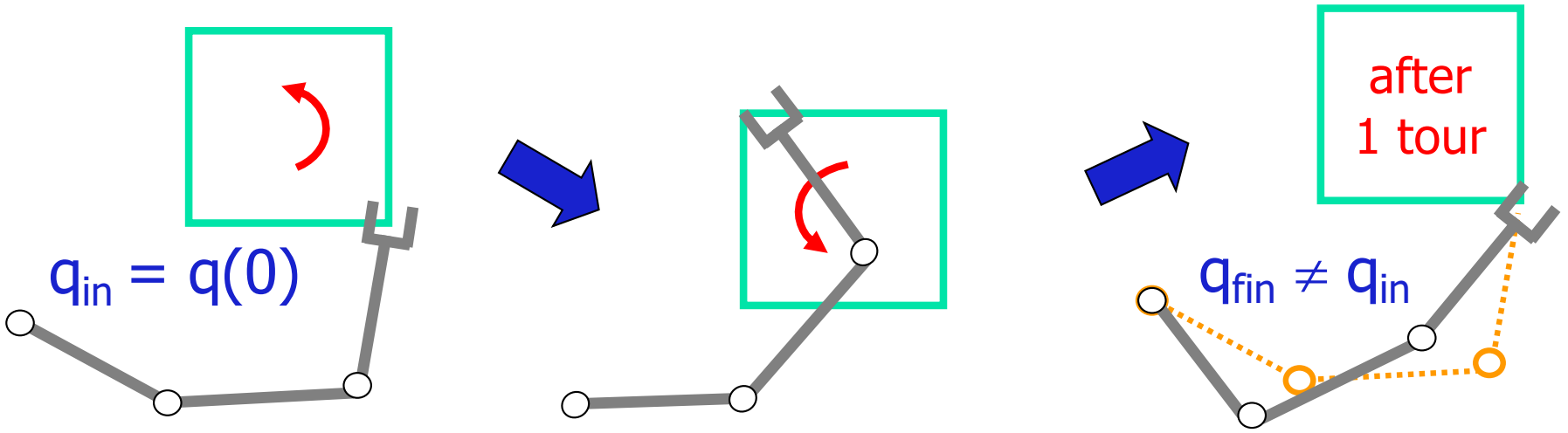
exact solution (μ=0)

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

μ^2	0	10^{-4}	10^{-3}	10^{-2}	10
$\|\dot{q}\|$	√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

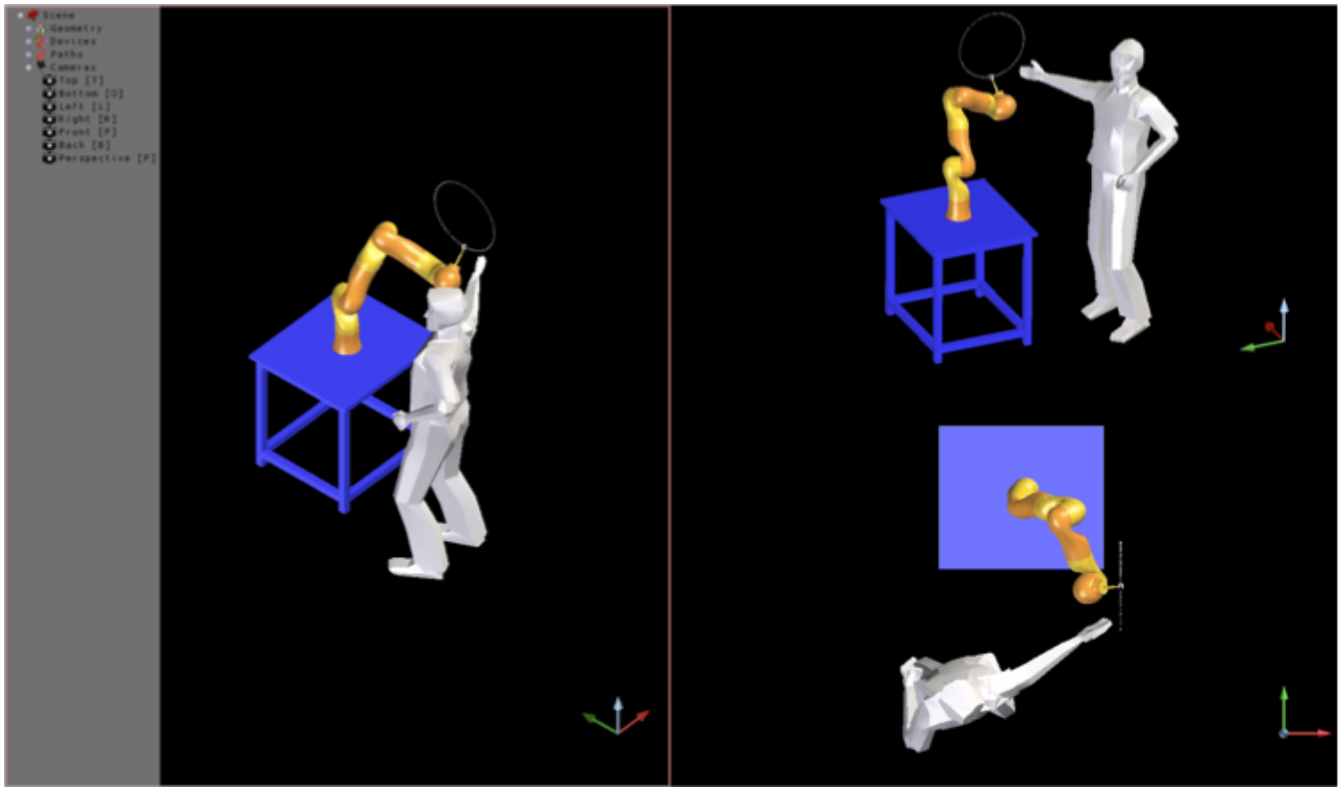
- no guarantee that singularities are globally avoided during task execution
  - despite joint velocities are kept to a minimum, this is only a local property and "avalanche" phenomena may occur
- typically lead to non-repeatable motion in the joint space
  - cyclic motions in task space do not map to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



$$q(t) = q(0) + \int_0^t K\big(q(\tau)\big)\dot{r}(\tau)\, d\tau$$

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 $J\dot{q} = \dot{r}$

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

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

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

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

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

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

even more in general...

$$\dot{q} = K_1\dot{r} + (I - K_2J)\dot{q}_0$$

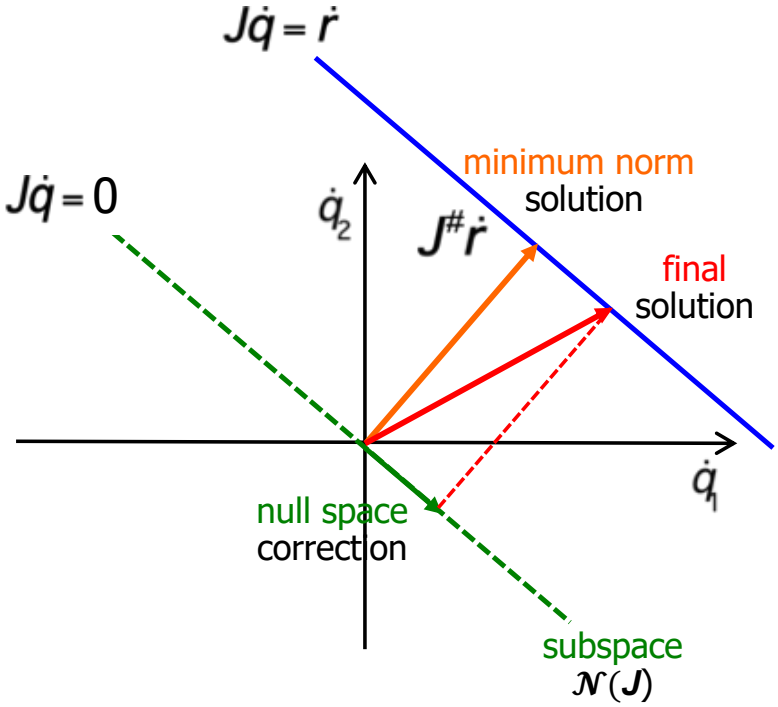
... but with less nice properties!

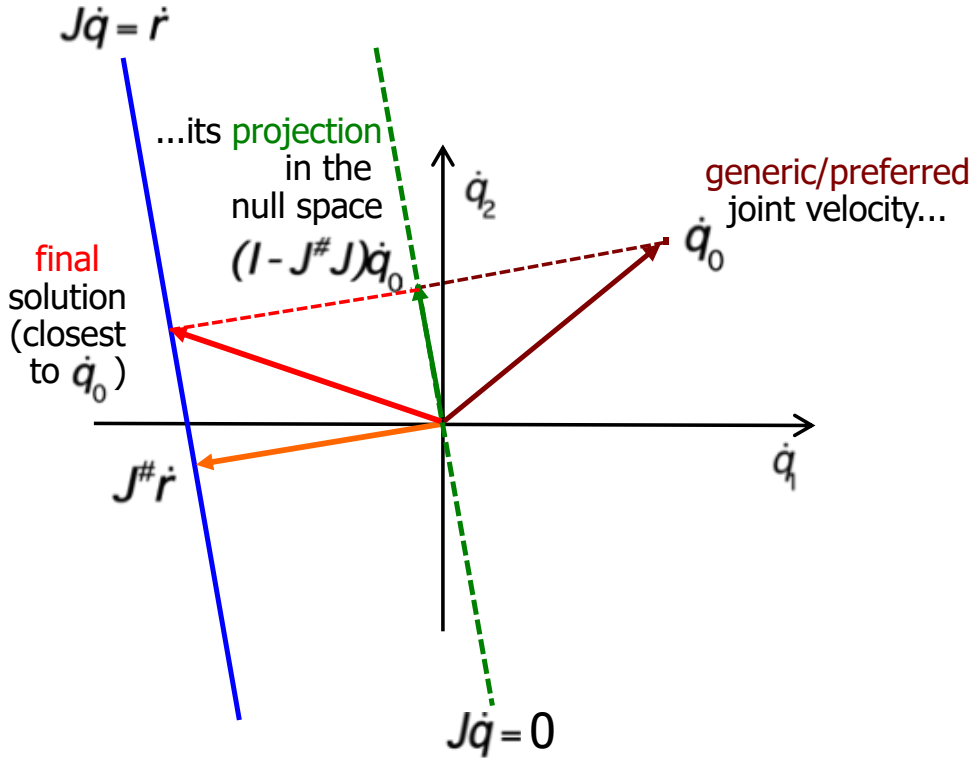
K_1, K_2 generalized inverses of J $(JK_iJ = J)$

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

$$\min_x H(x) = \frac{1}{2}(x - x_0)^T W (x - x_0)$$

$$\text{s.t. } \underset{M \times N}{J}\, x = y$$

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

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

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

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

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

necessary conditions +

$$\nabla_x L = \left(\frac{\partial L}{\partial x}\right)^T = W(x - x_0) + J^T \lambda = 0$$

$$\nabla_\lambda L = Jx - y = 0$$

sufficient condition for a minimum

$$\nabla_x^2 L = W > 0$$

$$x = x_0 - W^{-1} J^T \lambda$$

$$Jx_0 - JW^{-1}J^T \lambda - y = 0$$

$$\lambda = (JW^{-1}J^T)^{-1}(Jx_0 - y)$$

$M \times M$ invertible

$$x = x_0 - W^{-1}J^T (JW^{-1}J^T)^{-1}(Jx_0 - y)$$

Linear-Quadratic Optimization

application to robot redundancy resolution

PROBLEM

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

$\dot{q}_0$ is a "privileged" joint velocity

SOLUTION

$$\dot{q} = \dot{q}_0 - \underbrace{W^{-1}J^T(JW^{-1}J^T)^{-1}}_{J_W^{\#}}(J\dot{q}_0 - \dot{r})$$

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

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

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

Projected Gradient (PG)

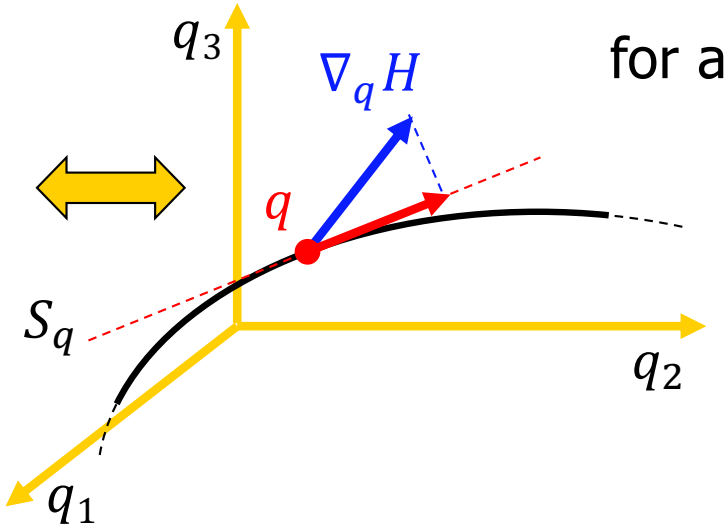
$$\dot{q} = J^{\#}\dot{r} + \left(I - J^{\#}J\right)\dot{q}_0$$

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 $H(q)$

projected gradient



for a fixed $\bar{r}$: $S_q = \{q \in \mathbb{R}^N : f(q) = \bar{r}\}$

$$\Rightarrow \dot{q} = \left(I - J^{\#}J\right)\nabla_q H$$

N-dimensional $\left(I - J^{\#}J\right)\nabla_q H = 0$

is a necessary condition
of constrained optimality

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)

$$q_i \in [q_{m,i}, q_{M,i}]$$

$$\bar{q}_i = \frac{q_{M,i} + q_{m,i}}{2}$$

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

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

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

also known as "clearance"

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

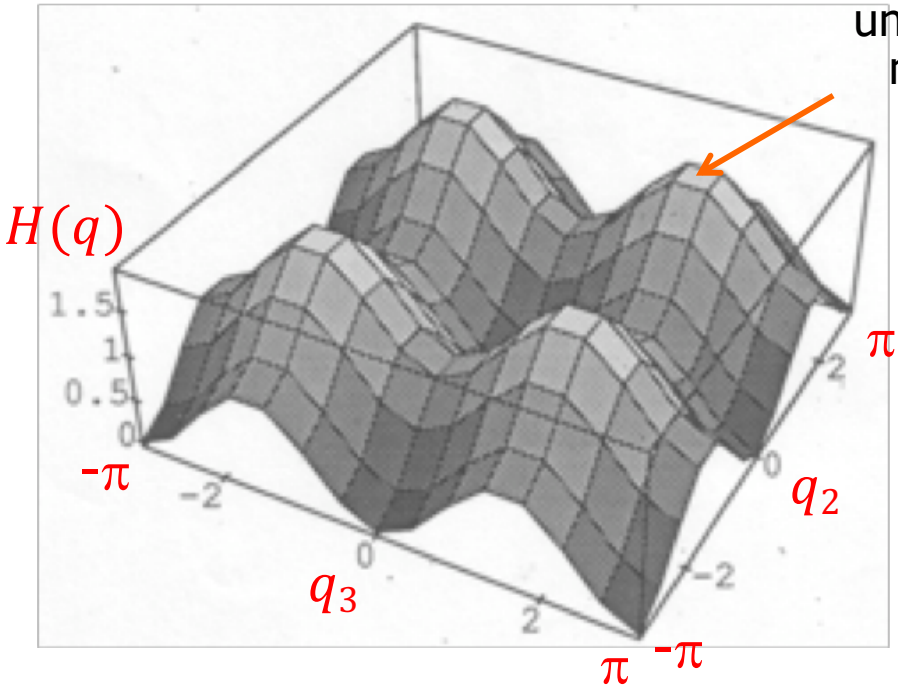
potential difficulties due to non-differentiability (this is a max-min problem)

Singularities of planar 3R arm

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

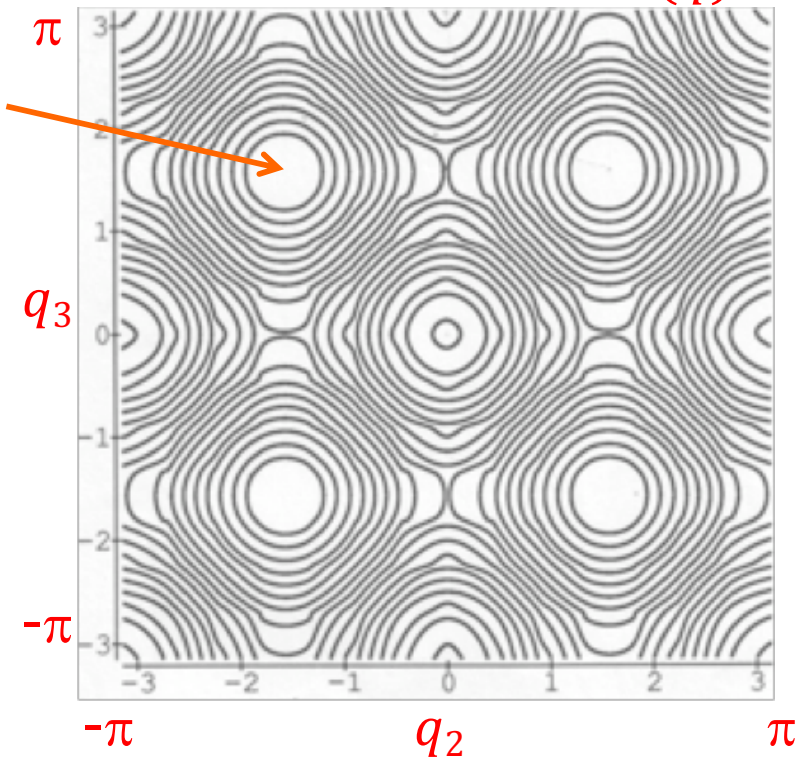
$$H(q) = \sin^2 q_2 + \sin^2 q_3$$

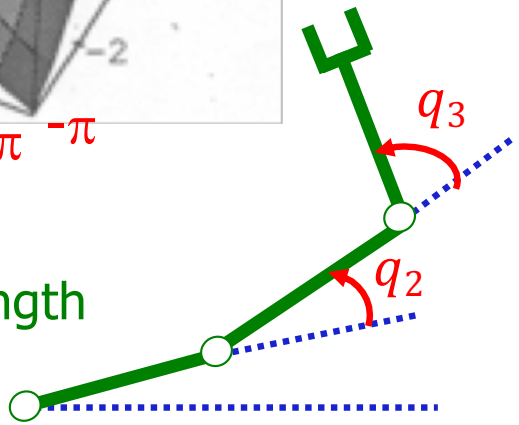
it is not H_{man} but has the same minima



unconstrained maxima of $H(q)$

iso-level curves of $H(q)$



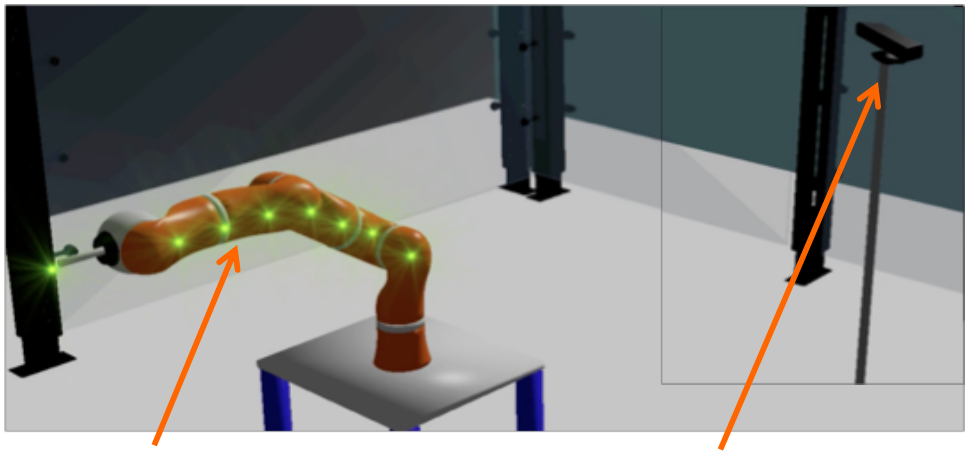


links of **equal** (unit) length

independent from q_1!

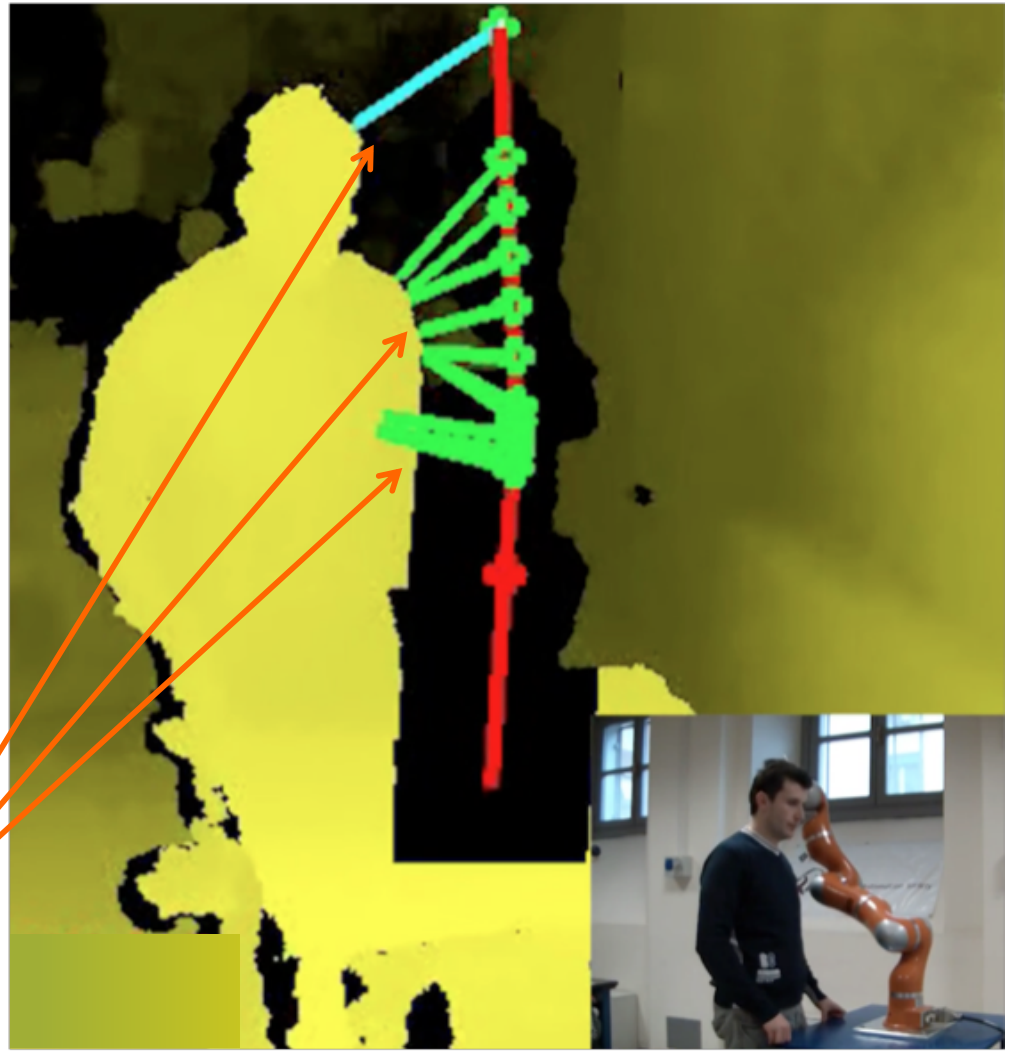
Minimum distance computation

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 depth 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

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

  but the pseudoinverse is needed anyway, and this is computationally intensive (SVD in the general case)
- in principle, the complexity of a redundancy resolution method should depend only from 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)

$$q = \begin{pmatrix} q_a \\ q_b \end{pmatrix} \begin{matrix} \} M \\ \} N-M \end{matrix} \quad \text{such that } \overbrace{J_a(q) = \frac{\partial f}{\partial q_a}}^{M \times M} \text{ is nonsingular}$$

- from the implicit function theorem, there exists then a 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)
- $q_a = g(r, q_b)$ are 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

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

$$(\neq \nabla_{q_b} H' \; !!)$$

$$\boxed{\nabla_{q_b} H' = 0}$$

is a "compact" (i.e., $N - M$ dimensional) necessary condition of constrained optimality

- algorithm

$$\dot{q}_b = \nabla_{q_b} H' \leftarrow$$ step in the gradient direction of the reduced $(N - M)$-dim space

$$J_a \dot{q}_a + J_b \dot{q}_b = \dot{r} \leftarrow$$ satisfaction of the M-dim task constraints

$$\dot{q}_a = J_a^{-1}(\dot{r} - J_b \dot{q}_b)$$

Comparison between PG and RG

- Projected Gradient (PG)

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

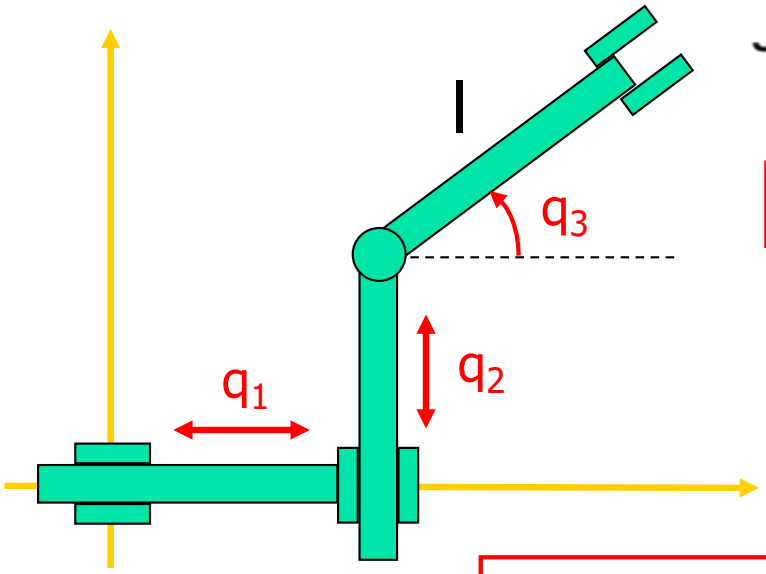
- Reduced Gradient (RG)

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

- 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 = cost & $N - M = 1$ ⇒ same (unique) direction for $\dot{q}$, but RG has automatically a larger optimization step size
- else ⇒ RG and PG methods provide always different evolutions

Analytic comparison

PPR robot



$$J = \left(\begin{array}{cc|c} 1 & 0 & -l\,s_3 \\ 0 & 1 & l\,c_3 \end{array}\right) = (J_a \mid J_b) \qquad q_a = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} \qquad q_b = q_3$$

RG:

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

$$\dot{q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \dot{r} + \begin{pmatrix} l\,s_3 \\ -l\,c_3 \\ 1 \end{pmatrix} \begin{pmatrix} l\,s_3 & -l\,c_3 & 1 \end{pmatrix} \nabla_q H$$

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

$$J^{\#} = \frac{1}{1+l^2} \begin{pmatrix} 1+l^2c_3^2 & l^2 s_3 c_3 \\ l^2 s_3 c_3 & 1+l^2 s_3^2 \\ -l\,s_3 & l\,c_3 \end{pmatrix} \qquad (I - J^{\#}J) = \frac{1}{1+l^2} \begin{pmatrix} l^2 s_3^2 & & \text{sym} \\ -l^2 s_3 c_3 & l^2 c_3^2 & \\ l\,s_3 & -l\,c_3 & 1 \end{pmatrix}$$

always < 1!!

Joint range limits

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

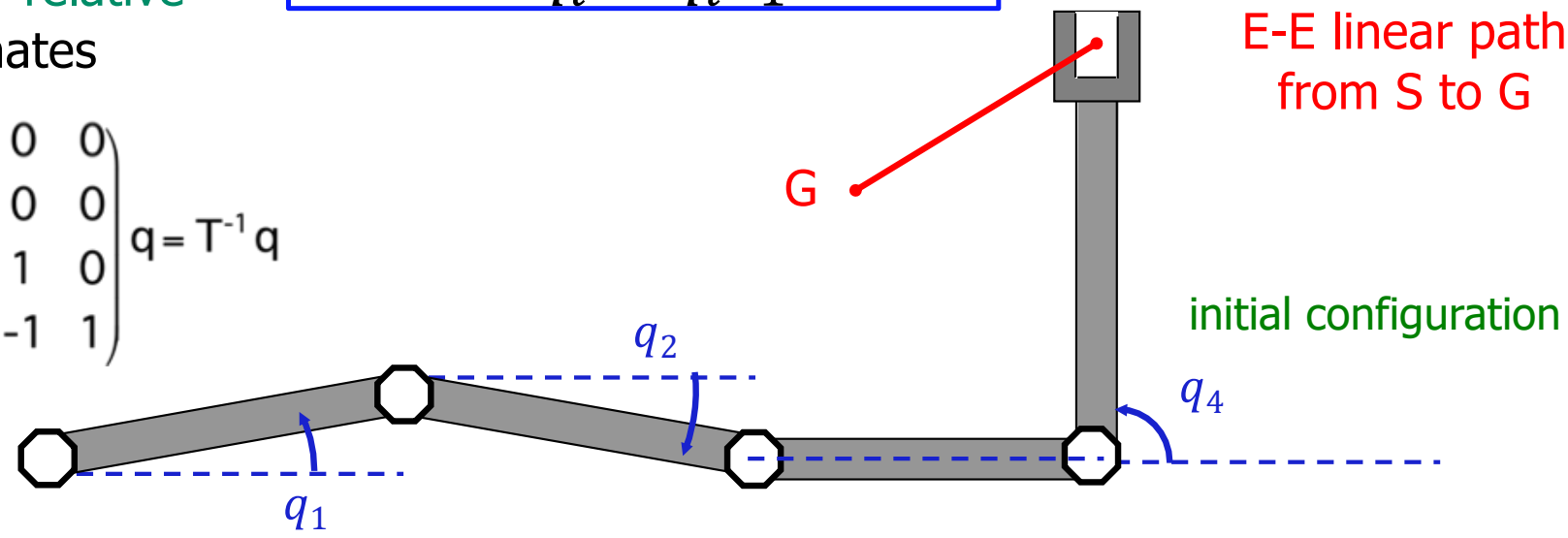
absolute ⇔ relative coordinates

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

$$-90° \leq \theta_i \leq 90°$$

$$\Updownarrow$$

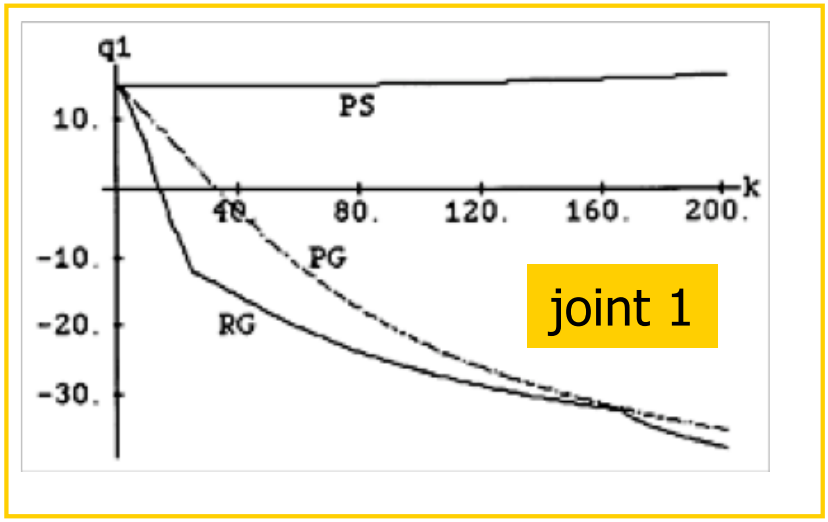
$$-90° \leq q_i - q_{i-1} \leq 90°$$

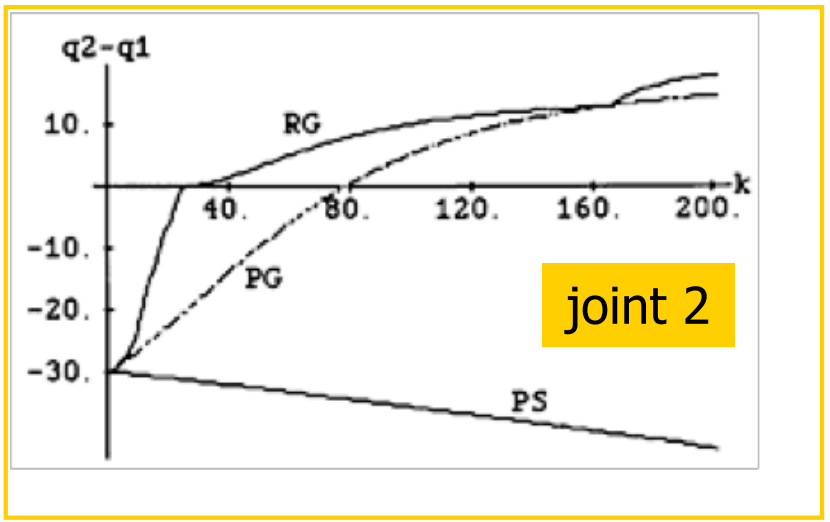


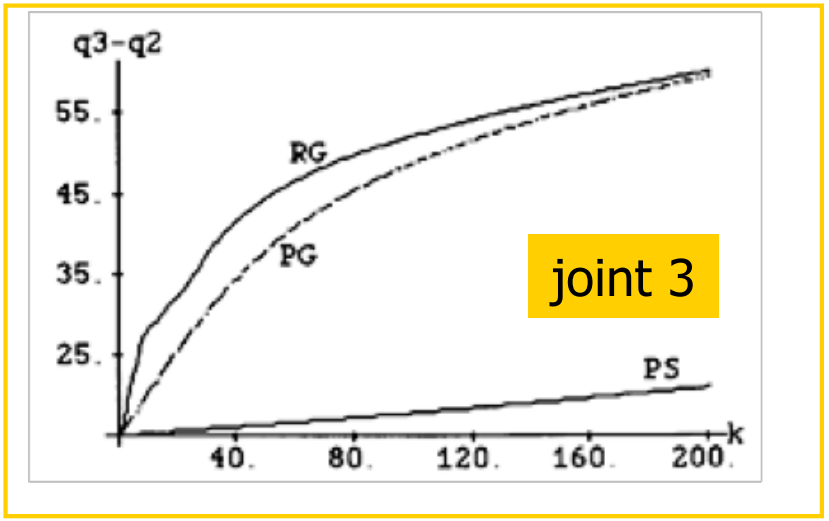
numerical comparison among pseudoinverse (PS), projected gradient (PG), and reduced gradient (RG) methods

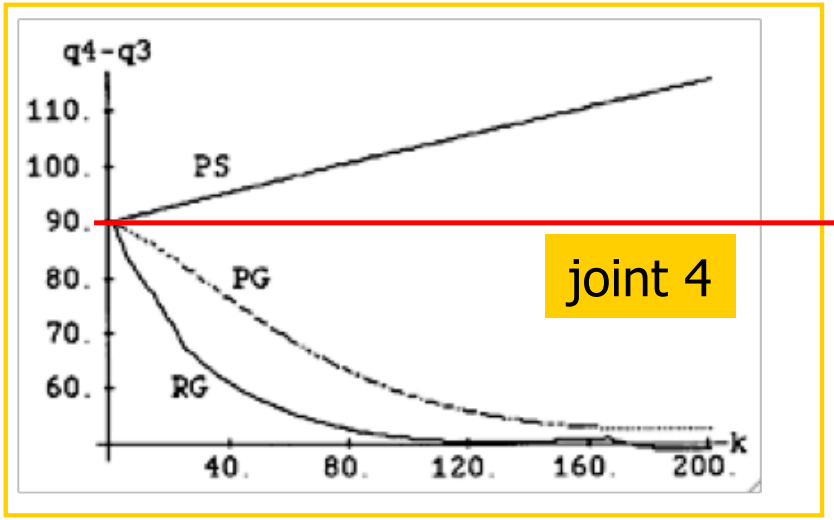
Numerical results

minimizing distance from mid joint range







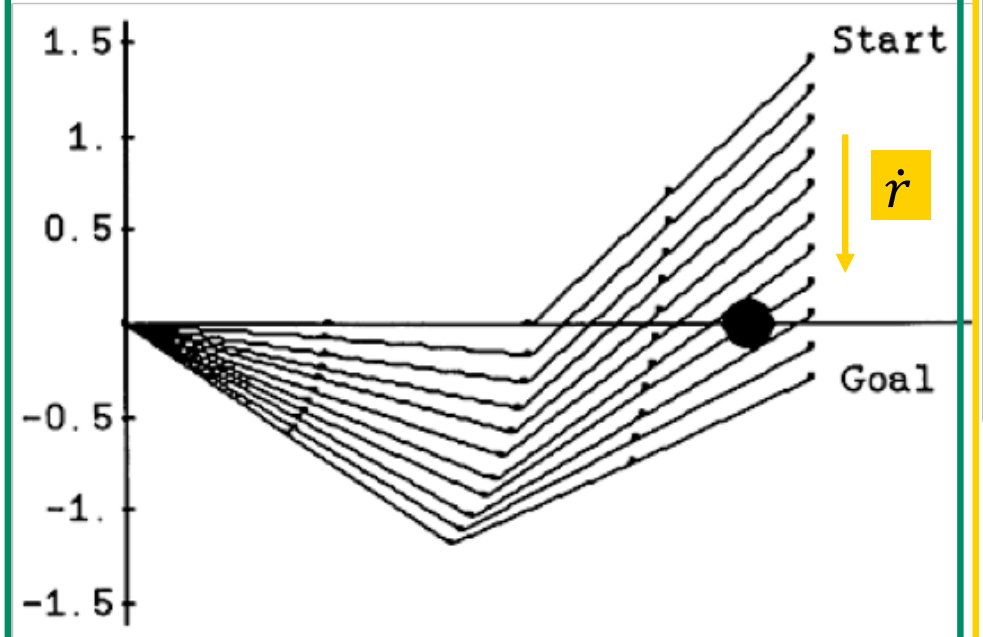


steps of numerical simulation

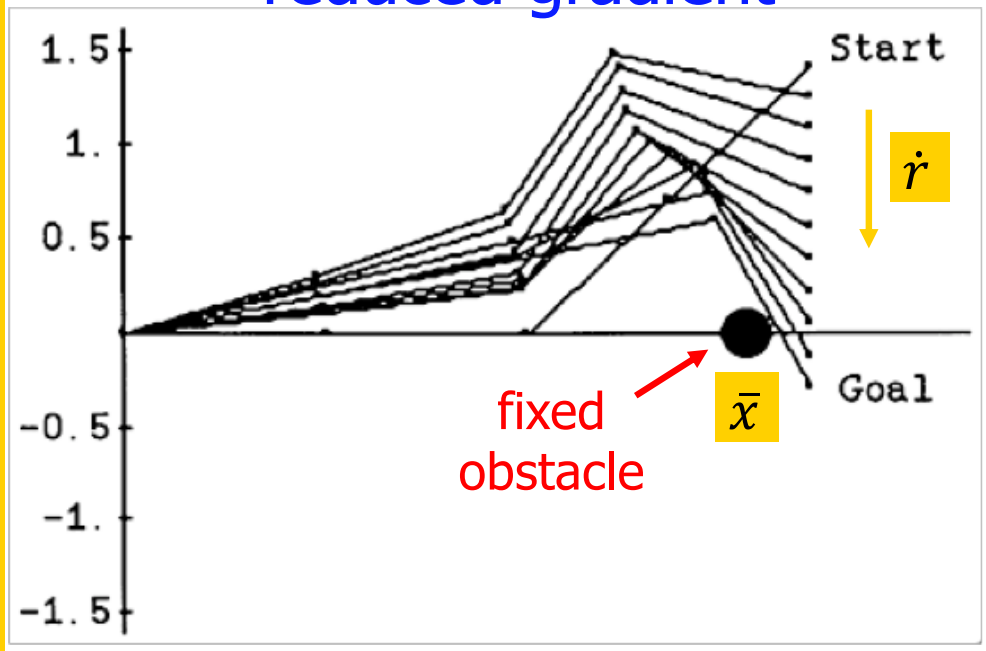
Numerical results

obstacle avoidance

pseudoinverse $\dot{q} = J^{\#}\dot{r}$



reduced gradient



constrained maximization of

$$H(q) = \left| \bar{x}\, sin\, q_4 - \sum_{i=1}^{3} l_i\, sin(q_4 - q_i) \right|$$

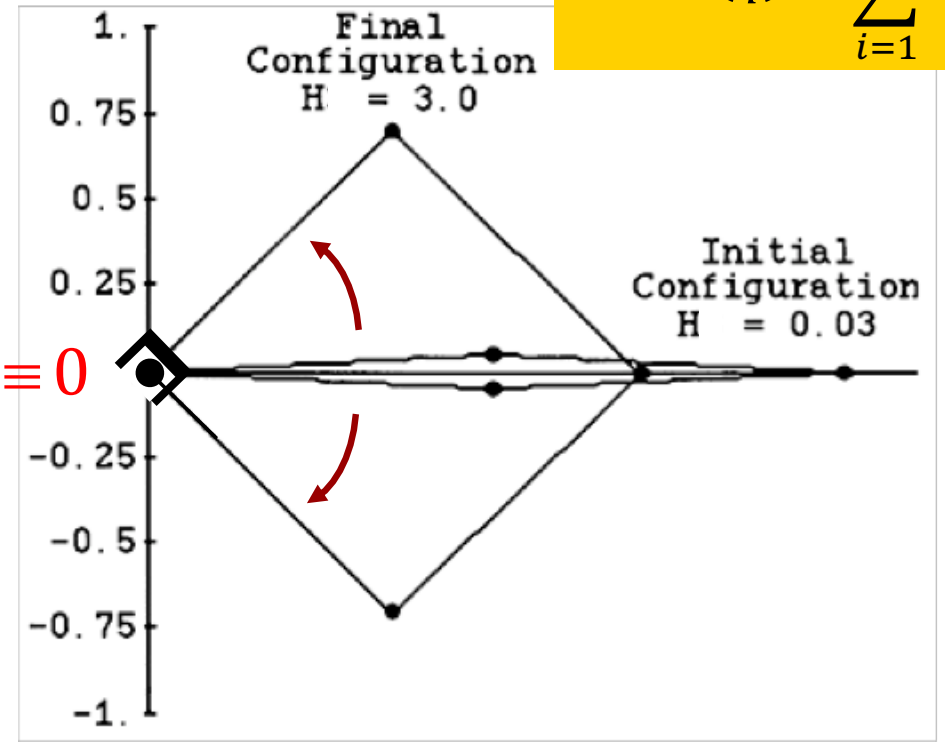
Numerical results

self-motion for escaping singularities

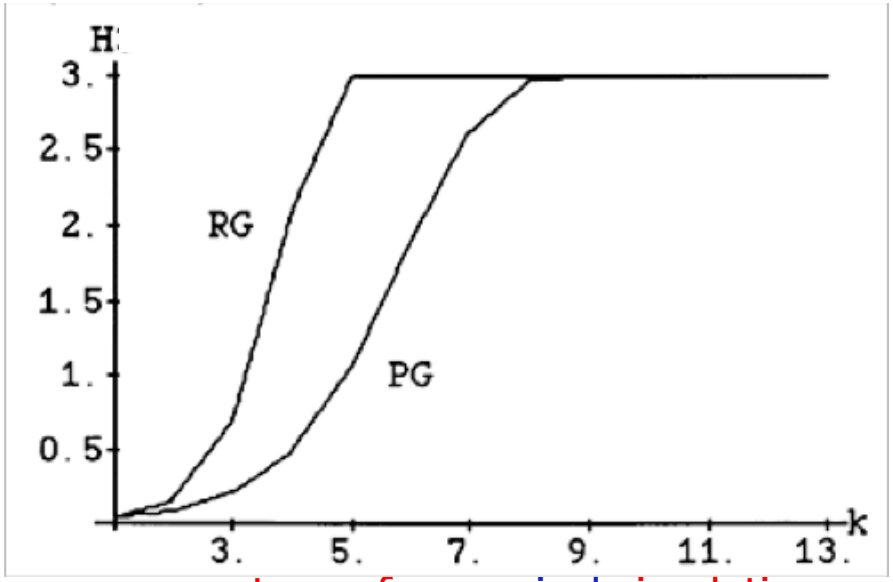
(optimal)

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

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



$r \equiv 0$



steps of numerical simulation

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

3 Task augmentation methods

- an auxiliary task is added (task augmentation)

$$S \updownarrow \boxed{f_y(q) = y} \qquad S \; ? \; N - M$$

corresponding to some desirable feature for the solution

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

J_A: N rows, $M + S$ columns

- a solution is chosen still in the form

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

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

Augmenting the task …

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

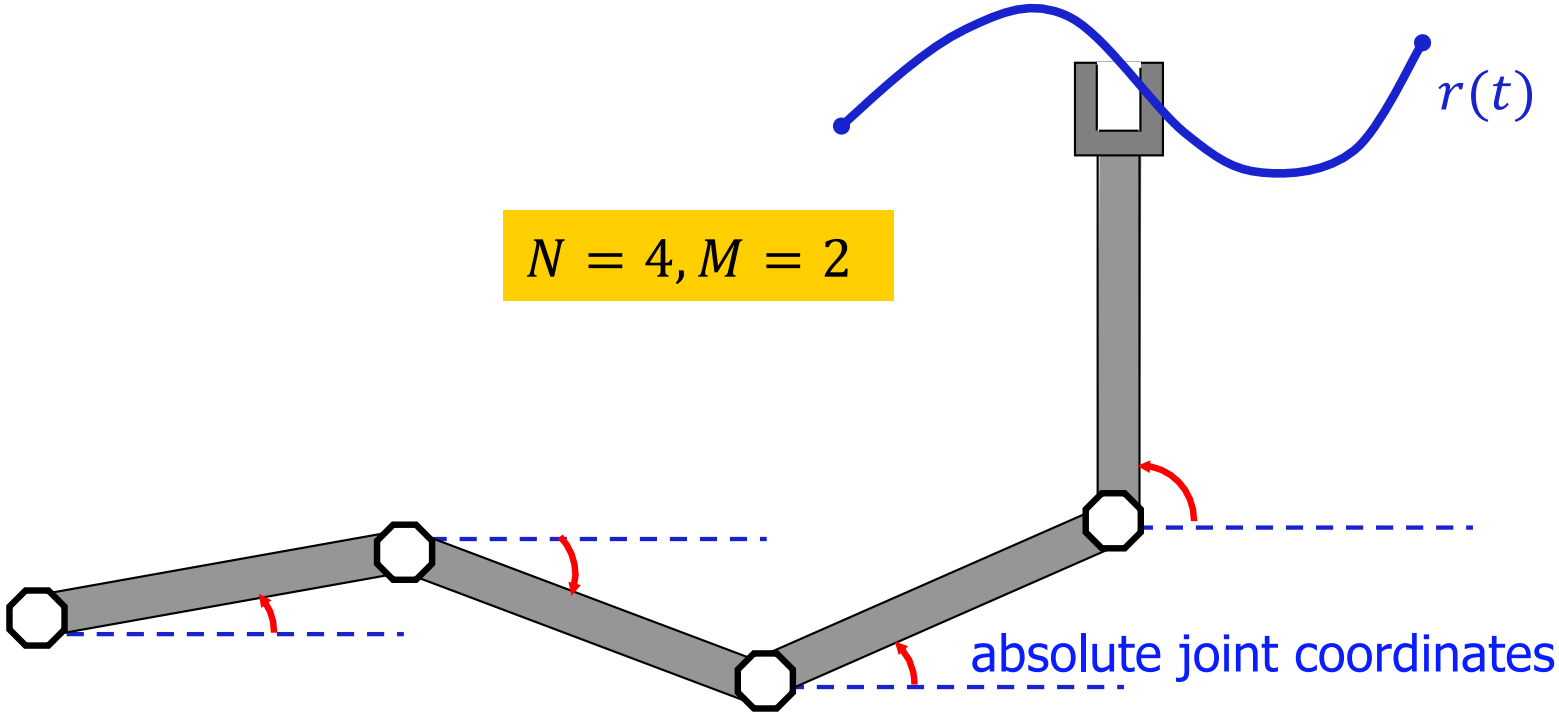
$$\rho(J) = M \quad \rho(J_y) = S \quad \text{but} \quad \rho(J_A) < M + S$$

it should always be $\mathcal{R}(J^T) \cap \mathcal{R}(J_y^T) = \emptyset$

difficult to be obtained globally!

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

Augmented task
example



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

last link to be held vertical

Extended Jacobian ($S = N - M$)

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

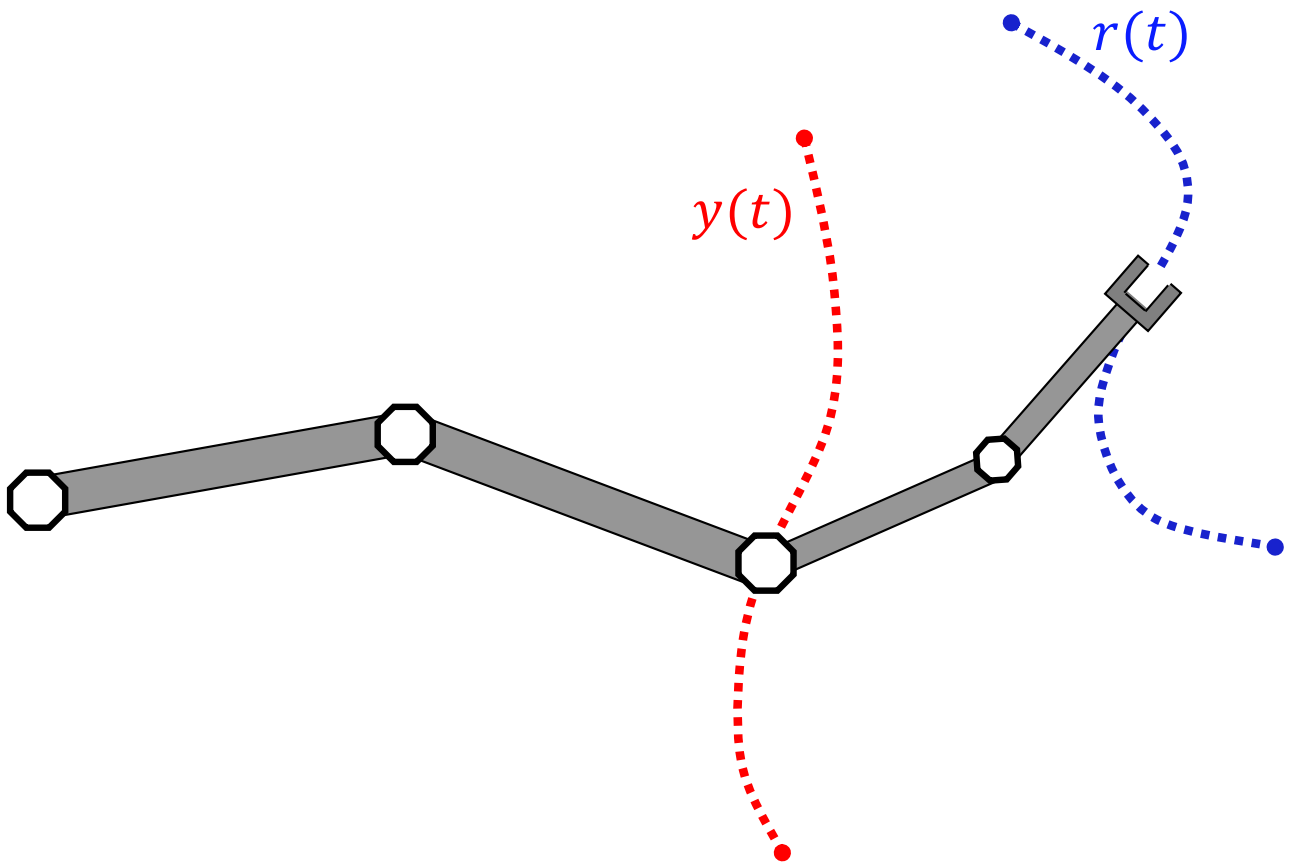
$$\dot{q} = J_A^{-1}(q)\dot{r}_A$$

- 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 #39), this scheme guarantees that constrained optimality is locally preserved during task execution
- in the vicinity of algorithmic singularities, the execution of both the original task as well as the auxiliary task(s) are affected by errors (when using a DLS inversion)

• 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



$N = 4, M = 2$

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

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

$$J_A = \left(\begin{array}{c|c} * & * \\ * & 0 \end{array}\right) \quad 4\times4$$

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

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

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

$$\dot{r}_2 = J_2\dot{q} = J_2\,J_1^{\#}\dot{r}_1 + J_2\left(I - J_1^{\#}J_1\right)v_1 = J_2\,J_1^{\#}\dot{r}_1 + J_2P_1v_1$$

the general solution for v_1 takes the usual form

$$v_1 = (J_2P_1)^{\#}\left(\dot{r}_2 - J_2\,J_1^{\#}\dot{r}_1\right) + \left(I - (J_2P_1)^{\#}(J_2P_1)\right)v_2$$

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

Task Priority (continue)

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

$$\dot{q} = J_1^{\#}\dot{r}_1 + P_1(J_2P_1)^{\#}\left(\dot{r}_2 - J_2J_1^{\#}\dot{r}_1\right) + P_1\left(I - (J_2P_1)^{\#}(J_2P_1)\right)v_2$$

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

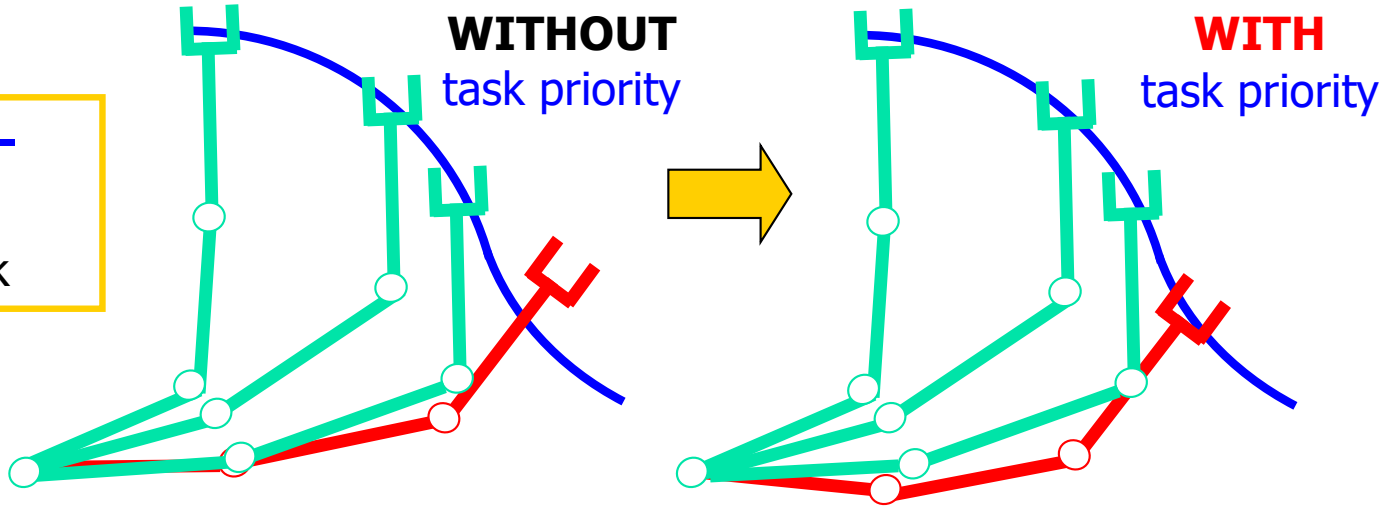
$= (J_2P_1)^{\#}$

possibly $= 0$

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

task 1: follow —

task 2: vertical third link



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

$\dot{q} \in \mathbb{R}^n$

$\dot{r}_k \in \mathbb{R}^{m_k}$

$k = 1, \ldots, p$

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

k-th task

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

projector in the null-space of k-th task

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

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

even larger!

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

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

for $k = 1$

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

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

for $k = 2$

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

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

$$\dot{q}_k = \arg \min_{\dot{q} \in \mathcal{S}_k} \frac{1}{2} \|\dot{q}\|^2$$

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

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

initialization

$$\dot{q}_0 = 0$$
$$P_{A,0} = I$$

over the steps, the search set is progressively reduced

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

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

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

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

- recursive expression also for the null-space projector

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

(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
      - in particular for humanoid robots in multiple complex 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

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

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

Dynamic redundancy resolution
as Linear-Quadratic optimization problems

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

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

$N \times N$ symmetric inertia matrix, positive definite for all q

Coriolis/centrifugal vector $c(q,\dot{q})$ + gravity vector $g(q)$

input torque vector (provided by the motors)

M-dimensional acceleration task

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

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

minimum torque norm

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

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

minimum torque (squared inverse inertia weighted) norm

Three dynamic solutions

- by applying the general formula for LQ optimization problems(°), check that the following closed-form expressions are obtained (in the assumption of full rank Jacobian J)

(°) **in slide #31**

minimum torque solution

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

- good for short trajectories (in fact, it is still only a "local" solution!)
- for longer trajectories it may lead to a torque "explosion" (whipping effect)

minimum (squared inverse inertia weighted) torque solution

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

- good performance in general, to be preferred

minimum (simple inverse inertia weighted) norm of the torque ...
a solution with leading $J^T(q)$ term: what is its physical interpretation?

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

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 (with 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

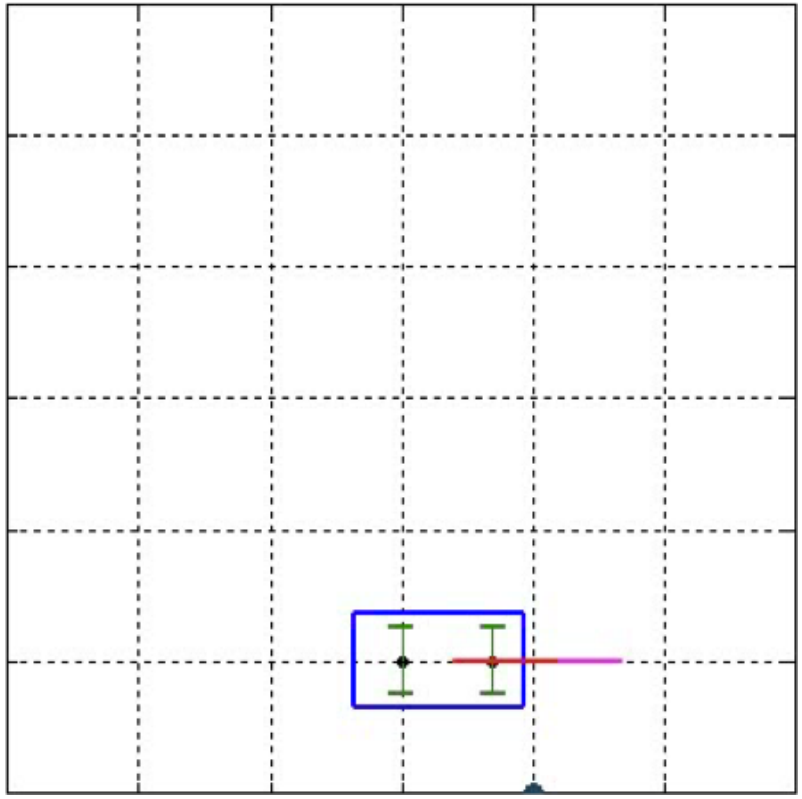
$$J \implies J_{NMM} \qquad \dot{q} \implies u$$

(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

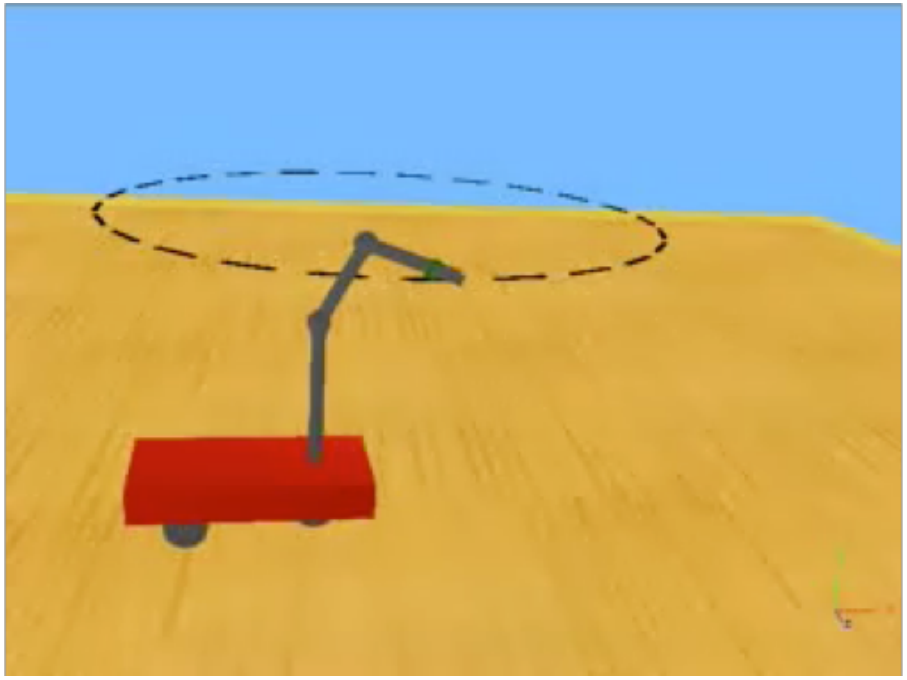
wheeled Justin: base with centered steering wheels ($N = 3 + 4{\times}2, N_u = 8$)
"dancing" in controlled but otherwise passive mode

Mobile manipulators

issues in acceleration control with steering wheels

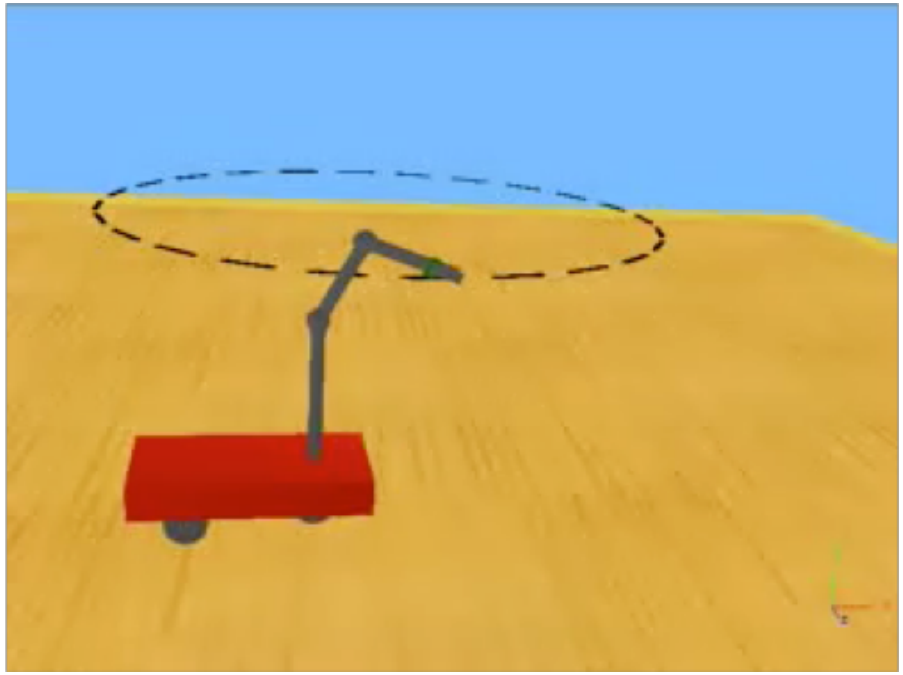
- velocity commands of steering wheels do not affect the task velocity
- solution: at the task acceleration level, using mixed commands
  - here 5: joint (3) and base linear (1) accelerations + steering velocity (1)

video



without null-space stabilizing term

video



with null-space stabilizing term

car-like (rear-drive speed + front steering) + 3R arm ($N = 7, N_u = 5$):
E-E trajectory control on a circle ($N_u - M = 2$)

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

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

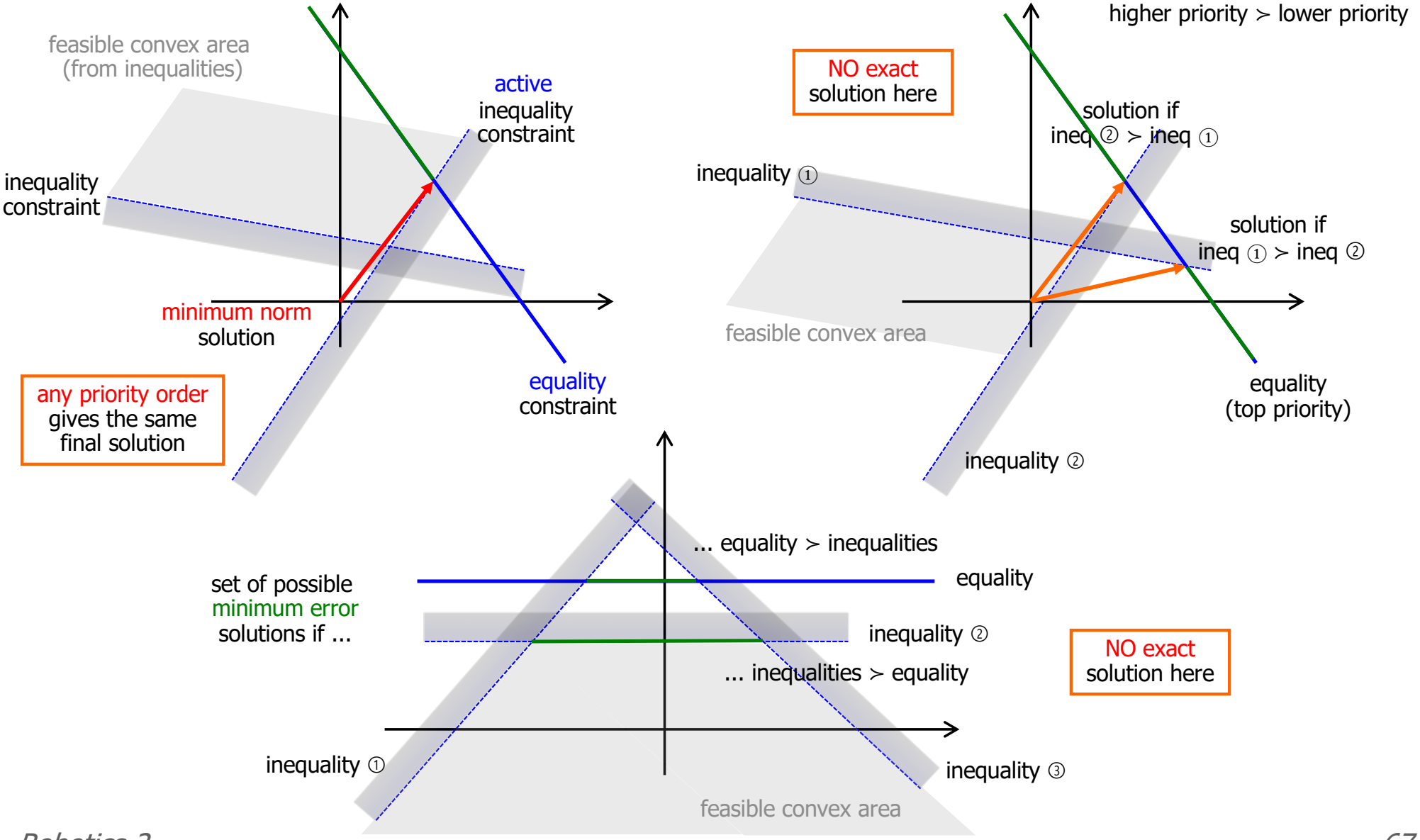
QP complete formulation

$$\min_{\dot{\boldsymbol{q}} \in \Omega} \frac{1}{2}\|\boldsymbol{J}\dot{\boldsymbol{q}} - \dot{\boldsymbol{r}}\|^2 + \frac{1}{2}\|\boldsymbol{w}\|^2$$

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

(possibly, also with prioritization of constraints)

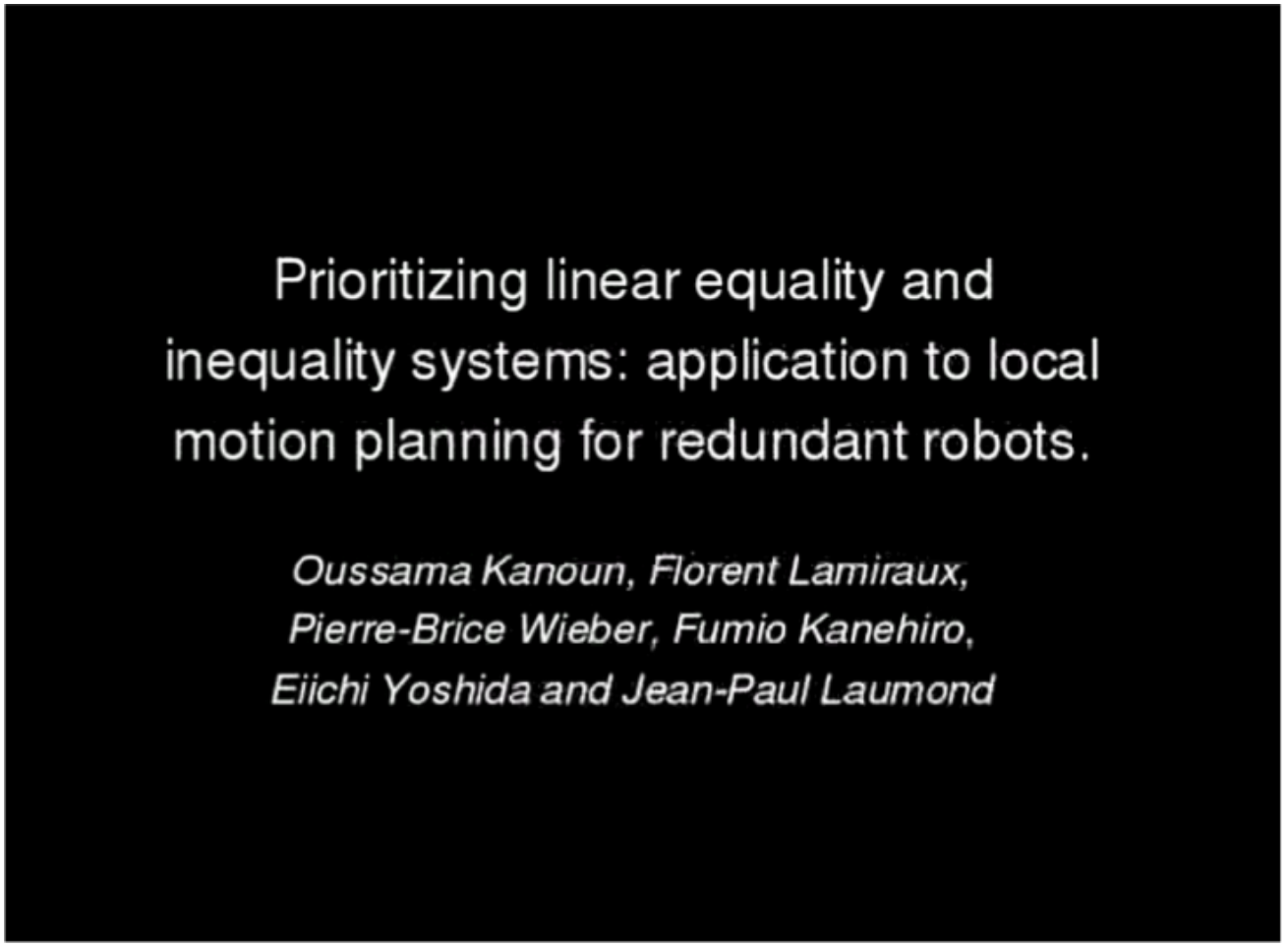
Equality and inequality linear constraints



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 that preserves task direction with minimal scaling

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

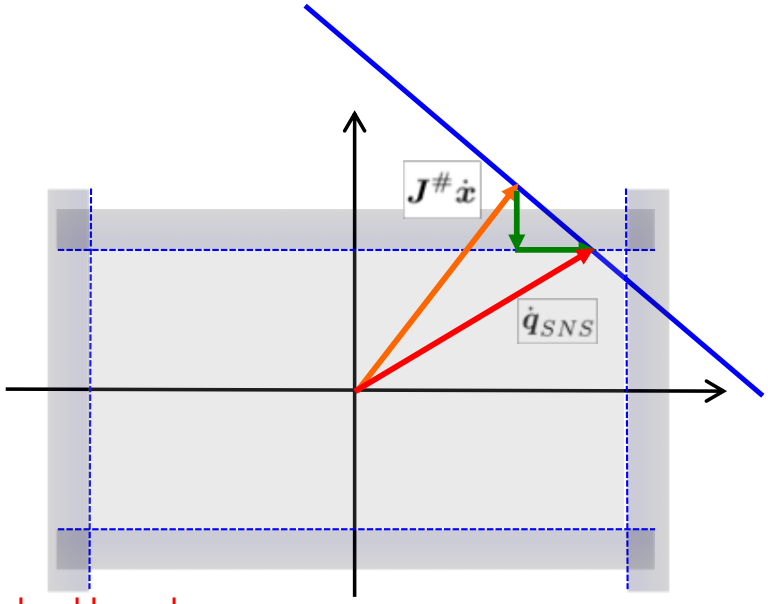
contains saturated joint velocities

scaling factor

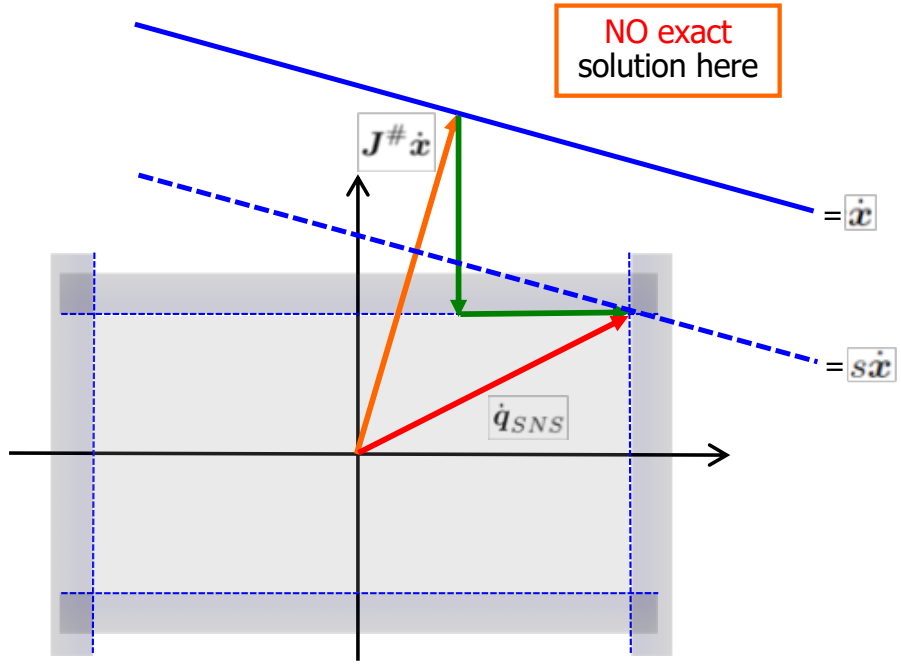
diagonal 0/1 matrix

Geometric view on SNS operation

in the space of velocity commands



hard bounds
(box inequality constraints)

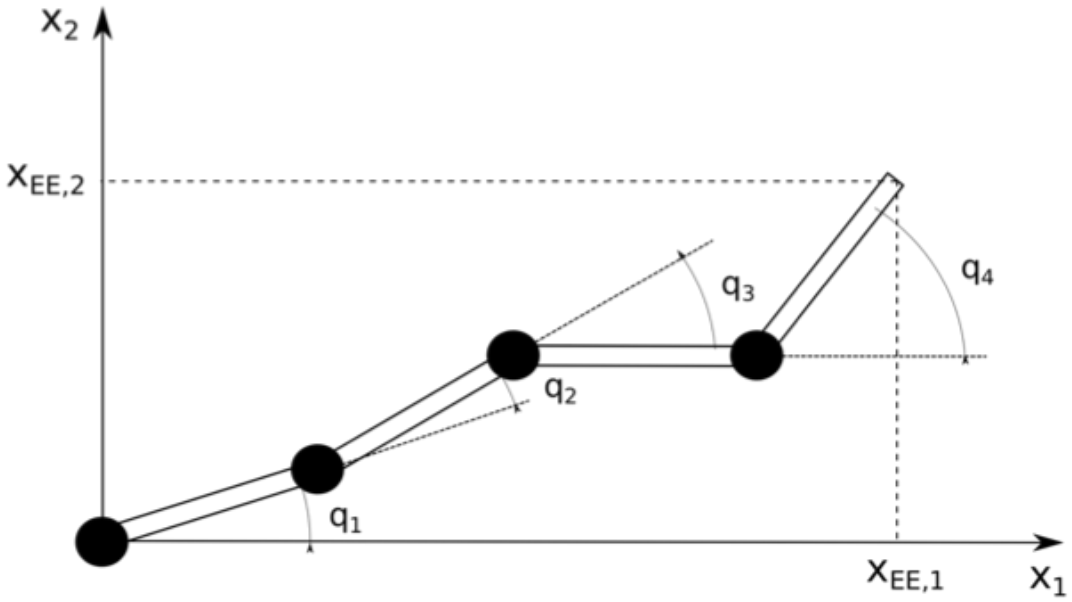


hard bounds
(box inequality constraints)

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

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

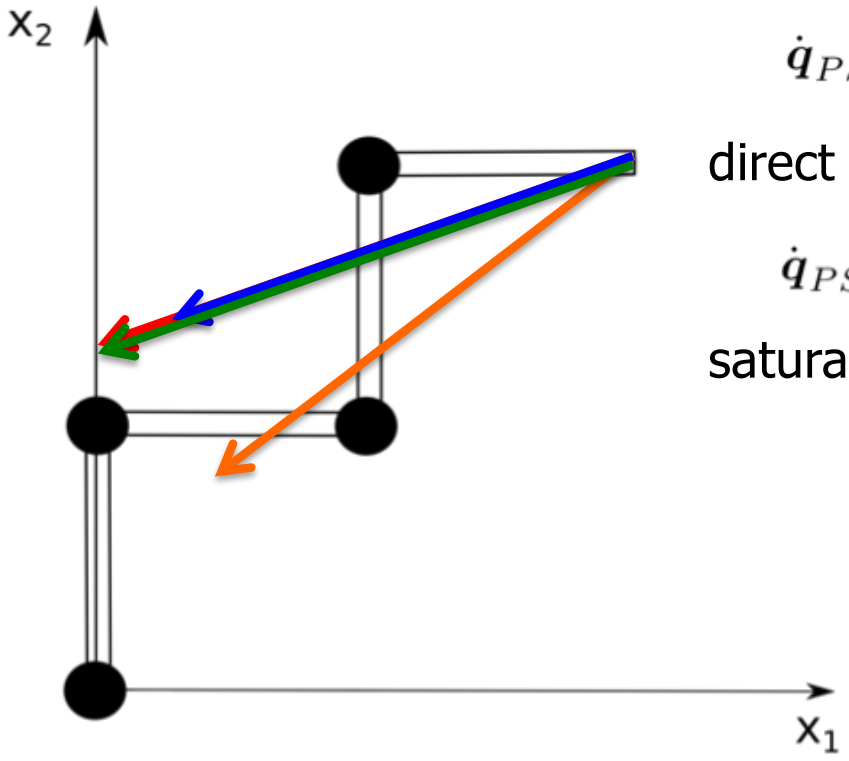
velocity limits $|\dot{q}_i| \leq V_i\,, i = 1 \ldots 4$ $\quad V_1 = V_2 = 2 \quad V_3 = V_4 = 4\ [\mathrm{rad/s}]$

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



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

2.0 -2.0

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? $\dot{q}_1 = V_1 = 2$

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

$$\dot{\boldsymbol{q}}_{SNS} = \begin{pmatrix} V_1 & \begin{pmatrix} J_2 & J_3 & J_4 \end{pmatrix}^{\#} \dot{\boldsymbol{x}}_{SNS} \end{pmatrix} = \begin{pmatrix} 2 & -1.8333 & 1.8333 & -3.6667 \end{pmatrix}^T$$

Joint velocity bounds

joint space limits

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

joint velocity bounds

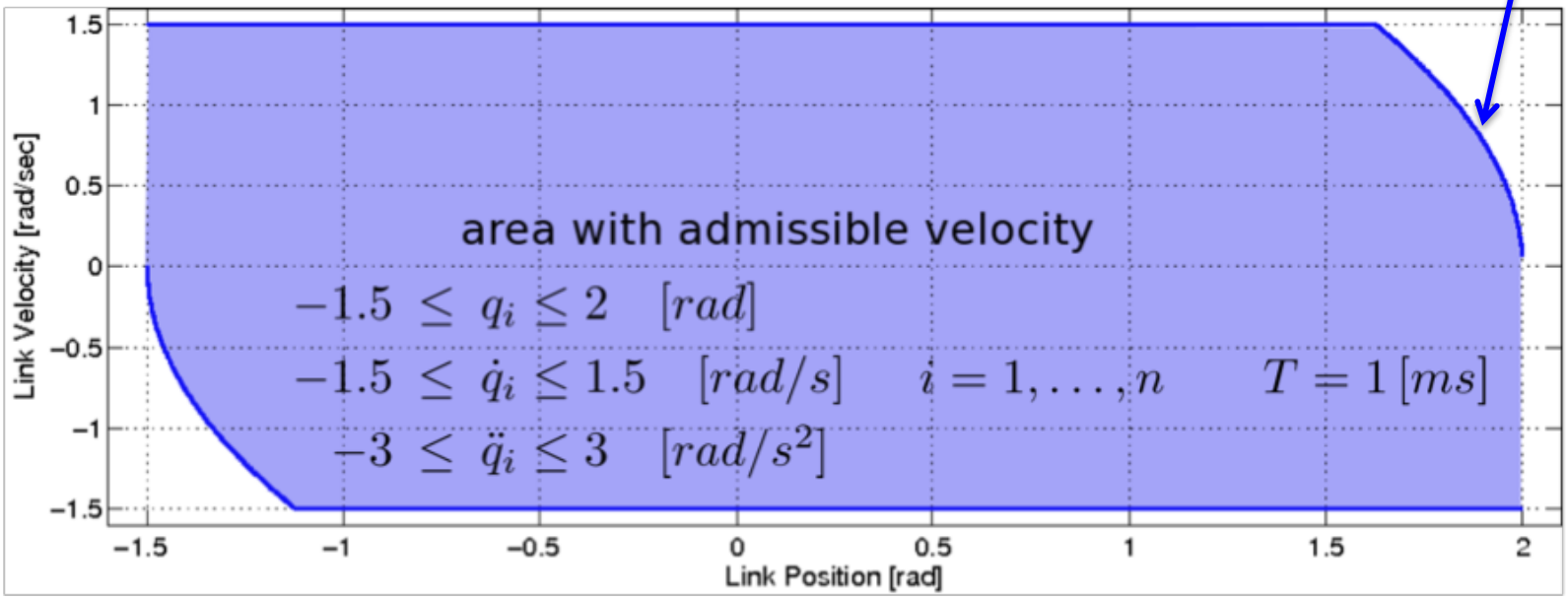
$$\dot{\boldsymbol{Q}}_{min}(t_k) \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{Q}}_{max}(t_k)$$

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
  $\dot{\bar{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_N + (\boldsymbol{JW})^{\#} (\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}_N)$
  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
    $\boldsymbol{a} = (\boldsymbol{JW})^{\#} \dot{\boldsymbol{x}}$
    $\boldsymbol{b} = \dot{\bar{\boldsymbol{q}}} - \boldsymbol{a}$
    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$(\boldsymbol{JW}) < m$ **then**
      $s = s^*$, $\boldsymbol{W} = \boldsymbol{W}^*$, $\dot{\boldsymbol{q}}_N = \dot{\boldsymbol{q}}_N^*$
      $\dot{\bar{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_N + (\boldsymbol{JW})^{\#} (s\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}_N)$
      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

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

  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

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

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

    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($\boldsymbol{JW}$) $< m$ **then**

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

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

      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

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

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

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

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

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($\boldsymbol{JW}$) $< m$ **then**

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

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

limit_exceeded = FALSE *(*outputs solution*)*

end if

end if

until limit_exceeded = TRUE

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

check if the 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

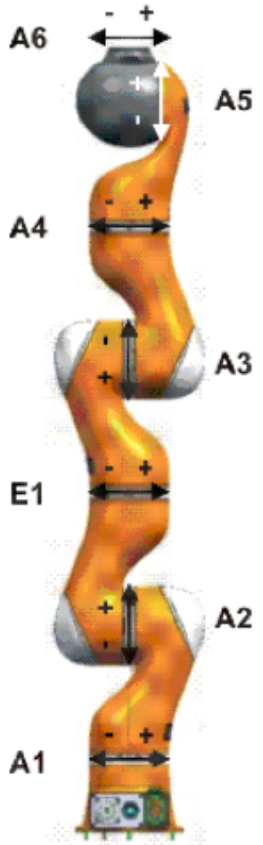
Algorithm 2

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* 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

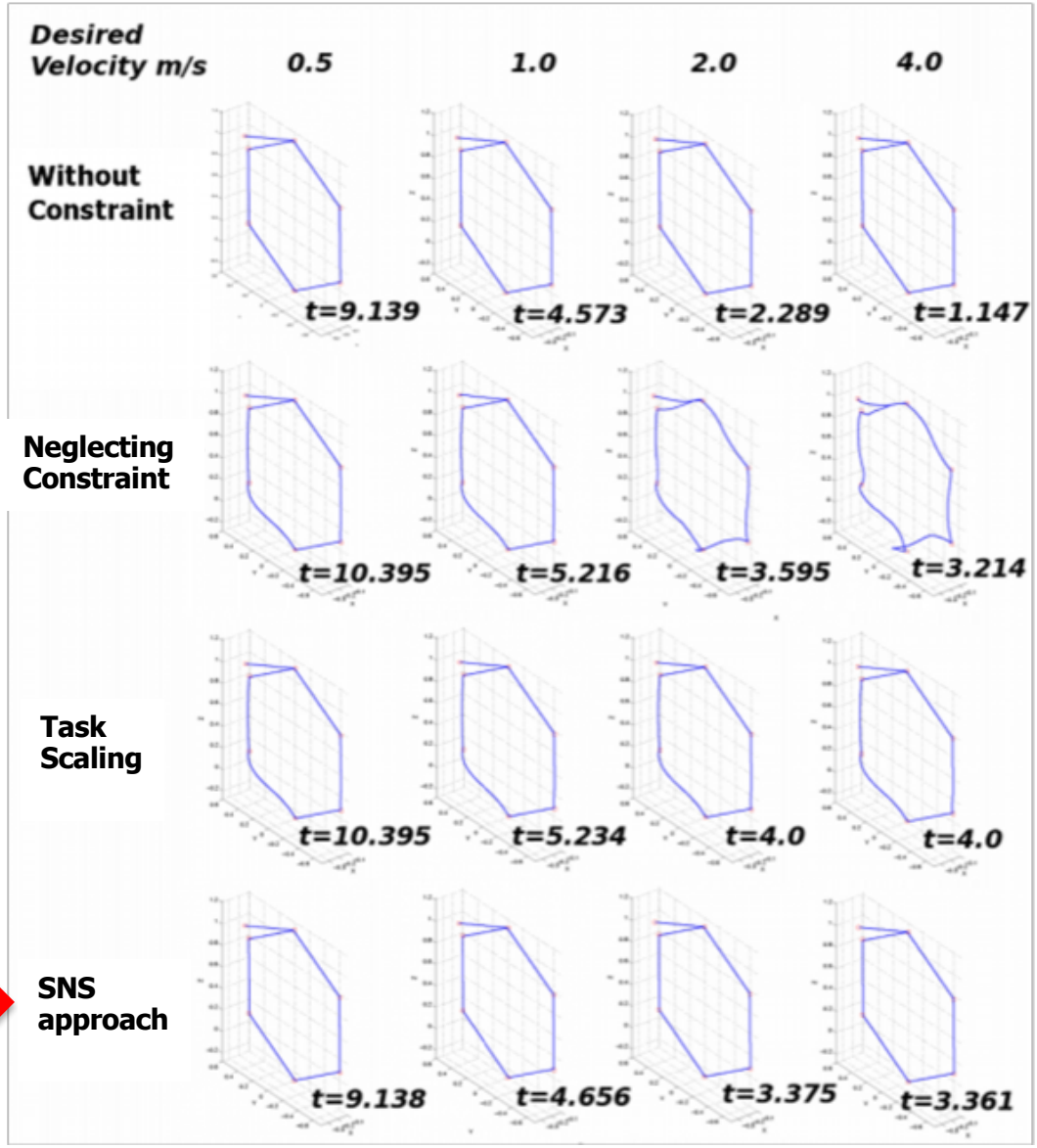
$$\boldsymbol{Q}_{max} = (170, 120, 170, 120, 170, 120, 170) \text{ [deg]}$$

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

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

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

Simulation results



for increasing V

requested task

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

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

task redundancy degree = 7 – 3 = 4

robot starts at the configuration

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

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

IEEE IROS 2012

video

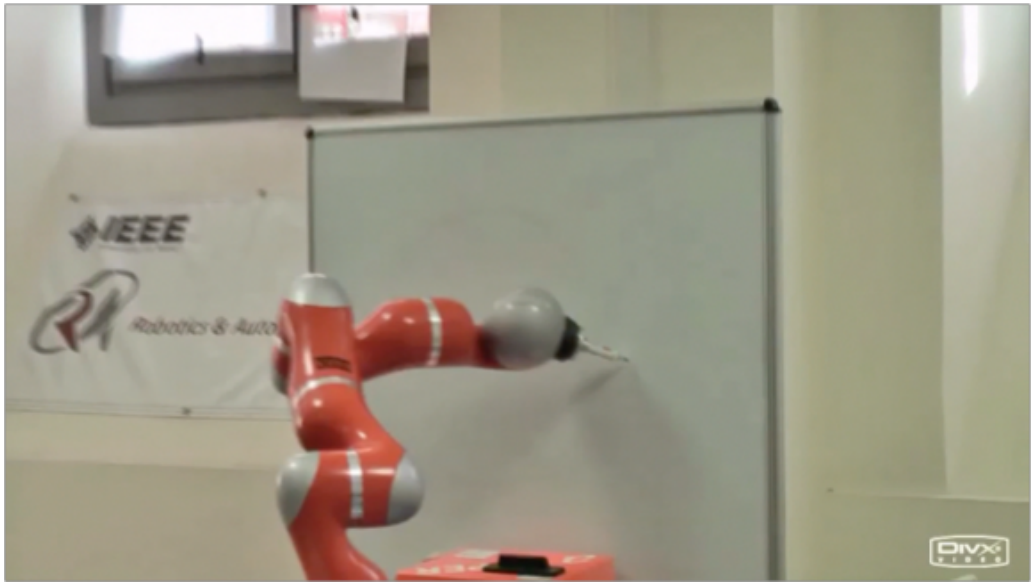
Global solution for repeatable motion

video (IEEE ICRA 2013)

- for cyclic tasks: a "bidirectional" probabilistic search in the reduced space of extra degrees of freedom (dimension $N - M$)
- including also collision avoidance (or other task constraints)

- experiment with KUKA LWR4 at DIAG Robotics Lab (Dec 2012)

video



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," http://hal.archives-ouvertes.fr/hal-00751924 (including software), 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, L. Lanari, G. Oriolo, "Control of redundant robots on cyclic trajectories," *IEEE Int. Conf. on Robotics and Automation*, pp. 500-506, 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
- M. Cefalo, G. Oriolo, M. Vendittelli, "Planning safe cyclic motions under repetitive task constraints," *IEEE Int. Conf. on Robotics and Automation,* pp. 3807-3812, 2013

Appendix A - Recursive Task Priority

proof of recursive expression for null-space projector

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

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

$$\begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{pmatrix}^{\#} = \begin{pmatrix} \boldsymbol{A}^{\#} - \boldsymbol{T}\boldsymbol{B}\boldsymbol{A}^{\#} & \boldsymbol{T} \end{pmatrix}$$

$$\boldsymbol{T} = \boldsymbol{E}^{\#} + \boldsymbol{X}\left(\boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^{\#}\right) \qquad \boldsymbol{E} = \boldsymbol{B}(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A})$$

$\boldsymbol{X}$ is irrelevant here

- (i)

$$\begin{aligned} \boldsymbol{P}_{A,k} &= \boldsymbol{I} - \boldsymbol{J}_{A,k}^{\#}\boldsymbol{J}_{A,k} = \boldsymbol{I} - \begin{pmatrix} \boldsymbol{J}_{A,k-1} \\ \boldsymbol{J}_k \end{pmatrix}^{\#} \begin{pmatrix} \boldsymbol{J}_{A,k-1} \\ \boldsymbol{J}_k \end{pmatrix} \\ &= \boldsymbol{I} - \begin{pmatrix} \boldsymbol{J}_{A,k-1}^{\#} - \boldsymbol{T}\boldsymbol{J}_k\boldsymbol{J}_{A,k-1}^{\#} & \boldsymbol{T} \end{pmatrix} \begin{pmatrix} \boldsymbol{J}_{A,k-1} \\ \boldsymbol{J}_k \end{pmatrix} \\ &= \boldsymbol{I} - \boldsymbol{J}_{A,k-1}^{\#}\boldsymbol{J}_{A,k-1} + \boldsymbol{T}\boldsymbol{J}_k\boldsymbol{J}_{A,k-1}^{\#}\boldsymbol{J}_{A,k-1} - \boldsymbol{T}\boldsymbol{J}_k \\ &= \boldsymbol{P}_{A,k-1} - \boldsymbol{T}\boldsymbol{J}_k\boldsymbol{P}_{A,k-1} \end{aligned}$$

- (ii)

$$\boldsymbol{T} = \left(\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}\right)^{\#} + \boldsymbol{X}\left(\boldsymbol{I} - \left(\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}\right)\left(\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}\right)^{\#}\right)$$

$$\Rightarrow \quad \boldsymbol{T}\boldsymbol{J}_k\boldsymbol{P}_{A,k-1} = \left(\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}\right)^{\#}\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}$$

- (i) + (ii) ⇒ Q.E.D.

- if k-th task is scalar

$$\boldsymbol{J}_k = \text{single row } \boldsymbol{j}_k^T$$

$$\boldsymbol{P}_{A,k} = \boldsymbol{P}_{A,k-1} - \frac{\boldsymbol{P}_{A,k-1}\boldsymbol{j}_k\boldsymbol{j}_k^T\boldsymbol{P}_{A,k-1}}{\|\boldsymbol{P}_{A,k-1}\boldsymbol{j}_k\|^2}$$

(Greville formula)