Robots with kinematic redundancy
Part 2: Extensions

Prof. Alessandro De Luca
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

\[
\begin{align*}
\dot{q} & \in \mathbb{R}^n, \quad \dot{r}_k \in \mathbb{R}^{m_k}, \quad \dot{r}_k = J_k(q)q \\
& \quad k = 1, \ldots, p \\
\sum_{k=1}^{p} m_k & = m \leq n \\
\text{projector in the null-space of } k\text{-th task} & \end{align*}
\]

\[
\begin{align*}
P_k(q) & = I - J_k^\#(q)J_k(q) \\
p & \leq n
\end{align*}
\]

\[
\begin{align*}
\dot{r}_{A,k} & = \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \vdots \\ \dot{r}_k \end{pmatrix} \\
J_{A,k} & = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_k \end{pmatrix}
\end{align*}
\]

\[
\begin{align*}
P_{A,k} & = I - J_{A,k}^\#J_{A,k} \\
\text{projector in the null-space of the augmented Jacobian of the first } k \text{ tasks} & \end{align*}
\]

\[
\begin{align*}
J_iP_{A,k} & = O, \quad \forall i \leq k \\
\iff J_{A,k}P_{A,k} & = O
\end{align*}
\]
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

\[
\text{for } k = 1 \\
\begin{cases} 
\dot{q}_1 = \arg \min_{\dot{q} \in \mathbb{R}^n} \frac{1}{2} \| \dot{q} \|^2 \\
\quad \text{s.t. } J_1 \dot{q} = \dot{r}_1 
\end{cases} \xrightarrow{\text{for } k = 2} \\
\dot{q}_1 = J_1^\# \dot{r}_1
\]

\[
S_1 = \left\{ \dot{q}_1 + P_1 v_1, \; v_1 \in \mathbb{R}^n \right\}
\]

\[
\text{for } k = 2 \\
\begin{cases} 
\dot{q}_2 = \arg \min_{\dot{q} \in S_2} \frac{1}{2} \| \dot{q} \|^2 \\
S_2 = \left\{ \arg \min_{\dot{q} \in S_1} \frac{1}{2} \| J_2 \dot{q} - \dot{r}_2 \|^2 \right\}
\end{cases} \xrightarrow{\text{for } k = 2} \\
\dot{q}_2 = \dot{q}_1 + (J_2 P_1)^\# (\dot{r}_2 - J_2 \dot{q}_1)
\]

\[
S_2 = \left\{ \dot{q}_2 + P_{A,2} v_2, \; v_2 \in \mathbb{R}^n \right\}
\]
Recursive solution with priorities - 2

Generalizing to step $k$

Prioritized solution up to task $k - 1$

LQ problem for $k$-th task

Recursive formula

(Siciliano, Slotine: ICAR 1991)

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

\[ \mathbb{R}^n = S_0 \supseteq S_1 \supseteq \cdots \supseteq S_{p-1} \supseteq S_p \]
Recursive solution with priorities
properties and implementation

- the solution considering the first $k$ tasks with their priority
  \[
  \dot{q}_k = \dot{q}_{k-1} + (J_k P_{A,k-1})^\# (\dot{r}_k - J_k \dot{q}_{k-1})
  \]
satisfies also ("does not perturb") the previous $k - 1$ tasks
  \[
  J_{A,k-1} \dot{q}_k = J_{A,k-1} \dot{q}_{k-1}
  \]
since
  \[
  J_{A,k-1} (J_k P_{A,k-1})^\# = J_{A,k-1} P_{A,k-1} (J_k P_{A,k-1})^\# = O
  \]
  (Maciejewski, Klein: IJRR 1985): check the four defining properties of a pseudoinverse

- recursive expression also for the null-space projector
  \[
  P_{A,k} = P_{A,k-1} - (J_k P_{A,k-1})^\# J_k P_{A,k-1}
  \]
  \[
  P_{A,0} = I
  \]
  (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

\[ r = f(q) \quad \Rightarrow \quad \dot{r} = J(q)\dot{q} \quad \Rightarrow \quad \ddot{r} = J(q)\ddot{q} + J(q)\dot{q} \]

- rewritten in the form

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

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

the problem is formally equivalent to the previous one, with \textit{acceleration} in place of velocity commands

- for instance, in the null-space method

\[ \ddot{q} = J^\#(q)\ddot{x} + (I - J^\#(q)J(q))\ddot{q}_0 \]

\[ = \nabla_q H - K_D \dot{q} \]

solution with \textit{minimum acceleration norm } \( \|\ddot{q}\|^2 \)

needed to \textit{damp/stabilize} self-motions in the null space \((K_D > 0)\)
Dynamic redundancy resolution

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

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

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

Coriolis/centrifugal vector \( c(q, \dot{q}) \)

+ gravity vector \( g(q) \)

- 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 \( J \))

\(^{(o)}\) 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**

\[
H_2(\ddot{q}) = \frac{1}{2} \|\tau\|_M^{-2}^2 = \frac{1}{2} \tau^T M^{-2}(q)\tau = \frac{1}{2} \ddot{q}^T \dddot{q} + n^T(q, \dot{q})M^{-1}(q)\dddot{q} + \frac{1}{2} n^T(q, \dot{q})M^{-2}(q)n(q, \dot{q})
\]

**(inverse inertia weighted) torque norm**

\[
H_3(\ddot{q}) = \frac{1}{2} \|\tau\|_M^{-1}^2 = \frac{1}{2} \tau^T M^{-1}(q)\tau = \frac{1}{2} \ddot{q}^T M(q)\dddot{q} + n^T(q, \dot{q})M^{-1}(q)\dddot{q} + \frac{1}{2} n^T(q, \dot{q})M^{-1}(q)n(q, \dot{q})
\]
Closed-form solutions

minimum torque norm solution

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

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

minimum (squared inverse inertia weighted) torque norm solution

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

- good performance in general, to be preferred

minimum (inverse inertia weighted) torque norm solution

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

- 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)

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

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

versus

- 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 $K_P, K_D > 0$)

\[
\begin{align*}
\dot{r} &\rightarrow \dot{r}_d + K(r_d - r) \\
\ddot{r} &\rightarrow \ddot{r}_d + K_D(\dot{r}_d - \dot{r}) + K_P(r_d - r)
\end{align*}
\]

...in velocity-based...

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

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

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

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

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

\[ r = f(q) = f(q_b, q_m) \]

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

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

\[ = J_{NMM}(q) u \quad \text{Nonholonomic Mobile Manipulator (NMM) Jacobian} \ (M \times N_u) \]

- ... most previous results follow by just replacing

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

\[ \uparrow \quad \text{namely, the available velocity commands} \]
Mobile manipulators

video

Automatica Fair 2008

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

wheeled Justin with centered steering wheels
\( (N = 3 + 4 \times 2, N_u = 8) \)
“dancing” in controlled but otherwise passive mode

Robotics 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

\[ \dot{J}q = \dot{r} \quad Cq \leq d \quad \dot{q} \in \Omega \subseteq \mathbb{R}^n \]

within a given **convex** set

solution set, with **only equality** constraints

\[ S_{eq} = \arg \min_{\dot{q} \in \Omega} \frac{1}{2} \|J\dot{q} - \dot{r}\|^2 \]

given \( \dot{q}^* \in S_{eq} \) \( \Rightarrow \) \( S_{eq} = \{\dot{q} \in \Omega : J\dot{q} = J\dot{q}^*\} \)

solution set, with **only inequality** constraints

\[ S_{ineq} = \arg \min_{\dot{q} \in \Omega} \frac{1}{2} \|w\|^2 \]

s.t. \( C\dot{q} - w \leq d \quad w \in \mathbb{R}^m_+ \)

(non-negative) slack variables

given \( \dot{q}^* \in S_{ineq} \) \( \Rightarrow \) \( S_{ineq} = \Omega \cap \left\{ \begin{array}{l} c_j^T \dot{q} \leq d_j, \quad \text{if} \ c_j^T \dot{q}^* \leq d_j \\ c_j^T \dot{q} = c_j^T \dot{q}^*, \quad \text{if} \ c_j^T \dot{q}^* > d_j \end{array} \right\} \)

QP complete formulation

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

s.t. \( C\dot{q} - w \leq d \quad w \in \mathbb{R}^m_+ \)

(possibly with prioritization of constraints)
Equality and inequality linear constraints

Feasible convex area (from inequalities)

Active inequality constraint

Equality constraint

Minimum norm solution

Any priority order gives the same final solution

Set of possible minimum error solutions if...

...equality > inequalities

...inequalities > equality

NO exact solution here

NO exact solution here

Slack variables minimizing $\frac{1}{2} \|w\|^2$

$w_1 = w_2$

Solution if equality $\succ \{\text{ineq } \circledast, \text{ineq } \circledast\}$

Solution if $\text{ineq } \circledast > \text{ineq } \circledast$

Solution if $\text{ineq } \circledast > \text{ineq } \circledast$

Solution if $\text{ineq } \circledast$
an efficient task priority approach, with simultaneous inequality tasks handled as hard (cannot be violated) or soft (can be relaxed) constraints.
Equality and Inequality Tasks for the high-dof humanoid robot HRP2

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

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

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

IEEE Int. Conf. on Robotics and Automation (ICRA) 2009
Inclusion of hard limits in joint space
Saturation in the Null Space (SNS) method

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

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

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

contains saturated joint velocities
scaling factor
diagonal 0/1 matrix

Robotics 2
Geometric view on SNS operation in the space of velocity commands

the total correction to the original pseudoinverse solution is always in the null space of the Jacobian (up to task scaling, if present)
Illustrative example - 1

Consider a 4R robot with equal links of unitary length.

Task: end-effector Cartesian position

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

Manipulator configuration

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

differential map

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

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

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

task Jacobian

\[
J(q) = \begin{pmatrix}
-ls_1 - ls_{12} - ls_{123} - ls_{1234} & -ls_{123} - ls_{1234} & -ls_{123} - ls_{1234} & -ls_{1234} \\
lc_1 + lc_{12} + lc_{123} + lc_{1234} & lc_{12} + lc_{123} + lc_{1234} & lc_{123} + lc_{1234} & lc_{1234}
\end{pmatrix}
\]

Velocity limits

\( |\dot{q}_i| \leq V_i, \ i = 1 \ldots 4 \)

\( V_1 = V_2 = 2 \quad V_3 = V_4 = 4 \ \text{[rad/s]} \)
current configuration \( \mathbf{q} = (\pi/2 \ -\pi/2 \ \pi/2 \ -\pi/2)^T \)

associated Jacobian \( \mathbf{J} = (\mathbf{J}_1 \ \mathbf{J}_2 \ \mathbf{J}_3 \ \mathbf{J}_4) = \left( \begin{array}{cccc} -2 & -1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{array} \right) \)

desired end-effector velocity \( \dot{\mathbf{x}} = (\begin{array}{c} -4 \\ -1.5 \end{array})^T \)

\[ \dot{\mathbf{q}}_PS = \mathbf{J}^\# \dot{\mathbf{x}} = \left( \begin{array}{c} 2.4545 \\ -2.1364 \\ 1.2273 \\ -3.3636 \end{array} \right)^T \]

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

\[ \dot{\mathbf{q}}_PS = s\mathbf{J}^\# \dot{\mathbf{x}} = \left( \begin{array}{cccc} 2.0 & -1.74 & 1.0 & -2.74 \end{array} \right)^T \]

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

\[ \dot{\mathbf{x}}_{SNS} = \dot{\mathbf{x}} - \mathbf{J}_1 V_1 = (\begin{array}{cccc} \dot{\mathbf{q}}_2 \\ \dot{\mathbf{q}}_3 \\ \dot{\mathbf{q}}_4 \end{array}) \]

\[ \dot{\mathbf{q}}_{SNS} = \left( V_1 \left[ \begin{array}{cccc} \mathbf{J}_2 & \mathbf{J}_3 & \mathbf{J}_4 \end{array} \right] \right)^T \dot{\mathbf{x}}_{SNS} = \left( \begin{array}{cccc} 2 & -1.8333 & 1.8333 & -3.6667 \end{array} \right)^T \]
Joint velocity bounds

\[ Q_{\text{min},i} \leq q_i \leq Q_{\text{max},i} \]
\[ -V_{\text{max},i} \leq \dot{q}_i \leq V_{\text{max},i} \]
\[ -A_{\text{max},i} \leq \ddot{q}_i \leq A_{\text{max},i} \]

\[ \dot{Q}_{\text{min}}(t_k) \leq \dot{q} \leq \dot{Q}_{\text{max}}(t_k) \]

**Joint space limits**

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

\[ \dot{Q}_{\text{min},i} = \max \left\{ \frac{Q_{\text{min},i} - q_k,i}{T}, -V_{\text{max},i}, -\sqrt{2A_{\text{max},i} (q_k,i - Q_{\text{min},i})} \right\} \]

\[ \dot{Q}_{\text{max},i} = \min \left\{ \frac{Q_{\text{max},i} - q_k,i}{T}, V_{\text{max},i}, \sqrt{2A_{\text{max},i} (Q_{\text{max},i} - q_k,i)} \right\} \]

Smooth velocity bound “anticipates” the reaching of a hard limit.

Area with admissible velocity:

\[ -1.5 \leq q_i \leq 2 \text{ [rad]} \]
\[ -1.5 \leq \dot{q}_i \leq 1.5 \text{ [rad/s]} \]
\[ -3 \leq \ddot{q}_i \leq 3 \text{ [rad/s^2]} \]

Robotics 2
Algorithm 1

\[ W = I, \dot{q}_N = 0, s = 1, s^* = 0 \]

repeat

limit\_exceeded = FALSE
\[ \ddot{q} = \dot{q}_N + (JW)^\# (\dot{x} - J\dot{q}_N) \]

if \[ \{ \exists \ i \in [1:n] : \dot{q}_i < \dot{Q}_{\text{min},i} \text{ OR } \dot{q}_i > \dot{Q}_{\text{max},i} \} \] then

limit\_exceeded = TRUE

\[ a = (JW)^\# \dot{x} \]
\[ b = \ddot{q} - a \]

getTaskScalingFactor(a, b) (*call Algorithm 2*)

if \{task scaling factor\} > \(s^*\) then

\[ s^* = \{\text{task scaling factor}\} \]
\[ W^* = W, \dot{q}_N = \dot{q}_N \]

end if

\[ j = \{\text{the most critical joint}\} \]
\[ W_{jj} = 0 \]
\[ \dot{q}_{N,j} = \begin{cases} \dot{Q}_{\text{max},j} & \text{if } \dot{q}_j > \dot{Q}_{\text{max},j} \\ \dot{Q}_{\text{min},j} & \text{if } \dot{q}_j < \dot{Q}_{\text{min},j} \end{cases} \]

if \text{rank}(JW) < m then

\[ s = s^*, W = W^*, \dot{q}_N = \dot{q}_N^* \]
\[ \ddot{q} = \dot{q}_N + (JW)^\# (s\dot{x} - J\dot{q}_N) \]

limit\_exceeded = FALSE  (*outputs solution*)

end if

end if

until limit\_exceeded = TRUE

\[ \dot{q}_{SNS} = \ddot{q} \]

\[ initial\text{ization} \]
\[ W : \text{diagonal matrix with } (j,j) \text{ element} \]
\[ = 1 \text{ if joint } j \text{ is enabled} \]
\[ = 0 \text{ if joint } j \text{ is disabled} \]

\[ \dot{q}_N : \text{vector with saturated velocities in correspondence of disabled joints} \]

\[ s : \text{current task scale factor} \]

\[ s^* : \text{largest task scale factor so far} \]
SNS at velocity level

Algorithm 1

\[ W = I, \dot{q}_N = 0, s = 1, s^* = 0 \]

repeat

\[ \text{limit.exceeded} = \text{FALSE} \]
\[ \ddot{q} = \dot{q}_N + (JW)^\#(\dot{x} - J\dot{q}_N) \]

if \{ \exists i \in [1:n]: \dot{q}_i < \dot{q}_{\text{min},i} \text{ OR } \dot{q}_i > \dot{q}_{\text{max},i} \} then

\[ \text{limit.exceeded} = \text{TRUE} \]

\[ a = (JW)^\# \dot{x} \]
\[ b = \ddot{q} - a \]
\[ \text{getTaskScalingFactor}(a, b) (*\text{call Algorithm 2}*) \]

if \{ \text{task scaling factor} > s^* \} then

\[ s^* = \{ \text{task scaling factor} \} \]
\[ W^* = W, \dot{q}_N = \dot{q}_N \]
end if

\[ j = \{ \text{the most critical joint} \} \]
\[ W_{jj} = 0 \]
\[ \dot{q}_{N,j} = \begin{cases} \dot{q}_{\text{max},j} & \text{if } \dot{q}_j > \dot{q}_{\text{max},j} \\ \dot{q}_{\text{min},j} & \text{if } \dot{q}_j < \dot{q}_{\text{min},j} \end{cases} \]

if \text{rank}(JW) < m then

\[ s = s^*, W = W^*, \dot{q}_N = \dot{q}_N^* \]
\[ \ddot{q} = \dot{q}_N + (JW)^\#(s\dot{x} - J\dot{q}_N) \]
\[ \text{limit.exceeded} = \text{FALSE} \hspace{1cm} (*\text{outputs solution}*) \]
end if

end if

until \text{limit.exceeded} = \text{TRUE}

\[ \dot{q}_{\text{SNS}} = \ddot{q} \]
Algorithm 1

\[
W = I, \dot{q}_N = 0, s = 1, s^* = 0
\]

repeat

limit\_exceeded = FALSE
\[
\dot{\tilde{q}} = \dot{q}_N + (JW)^\# (\dot{x} - J\dot{q}_N)
\]

if \(\exists i \in [1:n]: \dot{q}_i < \dot{Q}_{\text{min}, i} \text{ OR } \dot{q}_i > \dot{Q}_{\text{max}, i}\) then

limit\_exceeded = TRUE

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

b = \dot{q} - a

g\_{\text{getTaskScalingFactor}}(a, b) (*call Algorithm 2*)

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

s^* = \{task scaling factor\}

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

end if

j = \{the most critical joint\}

W_{jj} = 0

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

if rank(JW) < m then

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

\[
\dot{\tilde{q}} = \dot{q}_N + (JW)^\# (s\dot{x} - J\dot{q}_N)
\]

limit\_exceeded = FALSE (*outputs solution*)

end if

end if

until limit\_exceeded = TRUE

\[
\dot{q}_{\text{SNS}} = \dot{\tilde{q}}
\]
Task scaling factor
Algorithm 2

function getTaskScalingFactor(a, b)
  for i = 1 → n do
    Smmin,i = (Qmin,i − bi) / ai
    Smax,i = (Qmax,i − bi) / ai
    if Smmin,i > Smax,i then
      {switch Smmin,i and Smax,i}
    end if
  end for
  smax = min_i {Smax,i}
  smin = max_i {Smmin,i}
  the most critical joint = argmin_i {Smax,i}
  if smin > smax .OR. smax < 0 .OR. smin > 1 then
    task scaling factor = 0
  else
    task scaling factor = smax
  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

<table>
<thead>
<tr>
<th>Axis</th>
<th>Range of motion, software-limited</th>
<th>Velocity without payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 (J1)</td>
<td>+/-170°</td>
<td>100°/s</td>
</tr>
<tr>
<td>A2 (J2)</td>
<td>+/-120°</td>
<td>110°/s</td>
</tr>
<tr>
<td>E1 (J3)</td>
<td>+/-170°</td>
<td>100°/s</td>
</tr>
<tr>
<td>A3 (J4)</td>
<td>+/-120°</td>
<td>130°/s</td>
</tr>
<tr>
<td>A4 (J5)</td>
<td>+/-170°</td>
<td>100°/s</td>
</tr>
<tr>
<td>A5 (J6)</td>
<td>+/-120°</td>
<td>160°/s</td>
</tr>
<tr>
<td>A6 (J7)</td>
<td>+/-170°</td>
<td>160°/s</td>
</tr>
</tbody>
</table>

7-dof KUKA LWR IV

- \( \theta_{max} = (170, 120, 170, 120, 170, 120, 170) \text{ [deg]} \)

- \( \nu_{max} = (100, 110, 100, 130, 130, 180, 180) \text{ [deg/s]} \)

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

- \( T = 1 \text{ [ms]} \)
Simulation results

For increasing \( V \)

Requested task
Move the end-effector through six desired Cartesian positions along linear paths with constant speed \( V \)

\[
\dot{x} = V \frac{x_r - x}{\|x_r - x\|}
\]

Task redundancy degree = \( 7 - 3 = 4 \)

Robot starts at the configuration

\( q(0) = (0, 45, 45, 45, 0, 0, 0) \) [deg]

(with a small initial approaching phase)
KUKA LWR IV with hard joint-space limits

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

Fabrizio Flacco  Alessandro De Luca  Oussama Khatib

Robotics Lab, DIAG  Artificial Intelligence Lab
Sapienza Università di Roma  Stanford University

July 2014

IEEE Transactions on Robotics 2015
Variations of the SNS method

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

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

Attached video to IROS 2012

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

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

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


Appendix A - Recursive Task Priority
proof of recursive expression for null-space projector

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

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

\[
\begin{pmatrix} A \\ B \end{pmatrix}^\# = (A^\# - TBA^\#) T
\]

\[
T = E^\# + X(I - EE^\#) \quad X \text{ is irrelevant here}
\]

\[
E = B(I - A^\# A)
\]

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

\[
= I - \begin{pmatrix} J_{A,k-1}^\# - TJ_k J_{A,k-1}^\# & T \end{pmatrix} \begin{pmatrix} J_{A,k-1} \\ J_k \end{pmatrix}
\]

\[
= I - J_{A,k-1}^\# J_{A,k-1} + TJ_k J_{A,k-1}^\# J_{A,k-1} - TJ_k
\]

\[
P_{A,k-1} - TJ_k P_{A,k-1}
\]

- (ii) \( T = (J_k P_{A,k-1})^\# + X(I - (J_k P_{A,k-1}) (J_k P_{A,k-1})^\#) \)

\[
\Rightarrow \quad TJ_k P_{A,k-1} = (J_k P_{A,k-1})^\# J_k P_{A,k-1}
\]

\[
(i) + (ii) \Rightarrow \text{Q.E.D.}
\]

- 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)