Kinematic Control of Redundant Robots with Online Handling of Variable Generalized Hard Constraints

Amirhossein Kazemipour*, Maram Khatib*, Khaled Al Khudir**, Claudio Gaz***, Alessandro De Luca*

***Abstract*—We present a generalized version of the Saturation in the Null Space (SNS) algorithm for the task control of redundant robots when hard inequality constraints are simultaneously present both in the joint and in the Cartesian space. These hard bounds should never be violated, are treated equally and in a unified way by the algorithm, and may also be varied, inserted or deleted online. When a joint/Cartesian bound saturates, the robot redundancy is exploited to continue fulfilling the primary task. If no feasible solution exists, an optimal scaling procedure is applied to enforce directional consistency with the original task. Simulation and experimental results on different robotic systems demonstrate the efficiency of the approach.**

I. Introduction

A robot manipulator is redundant with respect to a given task when the number of its joints is larger than that strictly needed to perform the task. The additional degrees of freedom allow for a greater flexibility in the execution of the primary task. Such redundancy is usually employed for achieving secondary goals, such as avoiding collisions with workspace obstacles, maximize manipulability, stay away from kinematic singularities, or minimize energy consumption [1]. The presence of joint and Cartesian inequality constraints is a critical issue in redundancy resolution. Robots should comply with hard constraints on position, velocity and acceleration in their joint motion, typically coming from actuator limitations. Inequality constraints on the Cartesian motion may be present because of the nature of the task, or sometimes suddenly appear due to the unstructured environment in which robots operate.

There are many ways to handle joint and Cartesian constraints in kinematic control of robots. Classical methods use artificial potentials [2], with a number of control points chosen along the kinematic chain being pushed away from the critical boundaries [3] and the associated control action taking place in the null space of the Jacobian of the primary task. This method is simple and effective, but highly parameter-dependent. Moreover, oscillatory behaviors may arise when activating/deactivating the evasive maneuvers in the proximity of the constraints [4]. In order to mitigate this undesired effect, the null-space projection term or the activation function may be designed in an incremental way [5], [6]. Nonetheless, the selection of suitable gains is still required. Furthermore, when multiple tasks are present, incorporating the avoidance scheme into the original Stack of Tasks (SoT) will give to each inequality constraint a different priority [5]–[8]. A framework dealing with adaptable Cartesian constraints with task scaling has been recently presented in [9]. However, this method is not able to manage a SoT with different priorities.

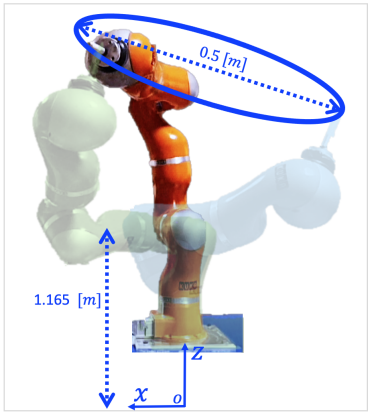

Figure 1: The KUKA LWR IV robot used for experimental evaluation. The world frame is placed on the lab floor. The desired end-effector task, the initial (solid orange), intermediate (shaded blue), and final (shaded orange) robot configurations of the first experiment are shown.

In order to deal with joint positional constraints, a common approach is to transform hard joint bounds into soft constraints, by adopting a suitable cost function whose minimization will keep the joint motions close to the center of their admissible ranges [10]. Alternatively, a weighted pseudo-inverse technique can be used [11], which further penalizes joint motions when they approach their limits. These techniques, however, do not guarantee that the hard inequality constraints will be always satisfied, and so they may result in unfeasible solutions. Recently, constrained optimization has been applied to the inverse kinematics of redundant robots, transforming the given tasks into a Least Squares (LS) problem and looking for solutions within a feasible convex set. A general formulation within this paradigm, which extends the priority framework also to inequality tasks, has been presented in [12].

The LS formulation has the advantage of explicitly including hard bounds into a numerically solvable Quadratic Programming (QP) problem. It allows to incorporate both joint and Cartesian motion limits as inequality constraints [13], [14]. However, these numerical approaches are computationally slower than analytical solutions [15]. Moreover, the feasibility of the task cannot be enforced, and the solution will be realizable only if the original (equality) task is compatible with the set of inequality constraints. Otherwise, the relaxation of these constraints in a least square sense leads to a physical violation of the hard limits. As an alternative to the previously mentioned QP approaches that minimize squared ℓ_2-norms, using the ℓ_1-norm minimization with suitable penalties offers

*Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Roma, Italy. Emails: amrkzp@gmail.com, {khatib, deluca}@diag.uniroma1.it

**School of Mechanical, Aerospace and Automotive Engineering, Coventry University, CV1 5FB Coventry, UK. Email: khaled.alkhudir@coventry.ac.uk

***Faculty of Science, Engineering and Computing, Department of Mechanical Engineering, Kingston University, SW15 3DW London, UK. Email: c.gaz@kingston.ac.uk

a computationally efficient solution for hierarchical robot task control [16].

The Saturation in the Null Space (SNS) algorithm introduced in [17] is capable of resolving part of the issues raised earlier, by linking QP to the SoT approach. This framework provides a simple yet efficient solution to resolve robot redundancy at velocity, acceleration, or torque levels that can be combined into a single solver [18]. If strictly needed, the SNS method utilizes a suitable task-scaling mechanism to recover feasibility of robot motion with respect to inequality constraints while preserving the geometric part of the primary equality task. This scaling strategy can be further extended in a predictive manner, by utilizing a model predictive control (MPC) approach to incorporate future evolution of the task [19]. In the original SNS algorithms, the inequality constraints on joint motion are regarded as hard bounds (i.e., they cannot be relaxed in a least-squares sense) and treated out of the SoT. So far, Cartesian bounds have not been treated explicitly, but rather approximated in the joint space as soft constraints. Accordingly, there is no guarantee that the robot will strictly comply also with the hard Cartesian constraints. On the other hand, in [20] both joint and Cartesian inequality constraints have been taken explicitly into account in the SoT for torque-controlled manipulators. In this approach, hard joint limits should always be given the highest priority over all other constraints. However, when both Cartesian and joint constraints are violated, the algorithm becomes unreliable because other lower priority limits will be treated as soft constraints.

Building on our preliminary results in [21], we generalize the original SNS algorithm in [17] with the following contributions.

- A single augmented vector is defined that considers all joint and Cartesian inequality constraints explicitly. This vector can be adapted online, without any parameter tuning phase, to follow any desired modification (addition or removal) in the set of task constraints.
- In the proposed algorithm, presented here at the velocity command level, all joint/Cartesian inequality constraints are treated equally. Accordingly, the hard bounds are always respected strictly. This is independent of the primary task (or of the SoT and its related priority management, when considering multiple equality tasks).
- The primary task $\dot{\boldsymbol{x}}$ is relaxed optimally by keeping its geometric direction, if and only if no feasible solution exists. Differently from [17], if $\dot{\boldsymbol{x}}$ exceeds any Cartesian constraint, it is saturated to its associated limit.
- The algorithm applies the saturation technique in both the joint and the Cartesian space, unlike [17].

The resulting control algorithm can be viewed as a general tool that can be easily used in any robot application, such as human-robot collaboration tasks [22]. Its main feature is in fact an overall efficiency and the adaptability to time-varying hard constraints that may be generated or deleted online based on sensor information. The validation of the basic algorithm is carried out with different simulations and experiments (illustrated also in the accompanying video).

The rest of the paper is organized as follows. Section I introduces the framework for incorporating generalized constraints. In Sec. III, the new kinematic control algorithm is presented at the velocity command level. Simulation results on a planar 6R manipulator and experiments on the 7R KUKA LWR IV robot results are reported and discussed in Sec. IV. Conclusions are summarized in Sec. V.

II. Problem formulation

Consider a robot manipulator with n joints and a single m-dimensional (primary) task, with $m < n$, to be performed by its end effector (EE) and defined by

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}), \qquad \boldsymbol{J}(\boldsymbol{q}) = \frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the joint position vector and the $m \times n$ task Jacobian matrix $\boldsymbol{J}$ has less rows than columns. Assuming that the robot is commanded by a kinematic control law at the velocity level, we solve the inverse kinematics as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}(\boldsymbol{q})\dot{\boldsymbol{x}}, \tag{2}$$

where $\boldsymbol{J}^{\#}$ is the Moore–Penrose pseudoinverse of $\boldsymbol{J}$. The command (2) is the minimum norm joint velocity corresponding to the desired task velocity $\dot{\boldsymbol{x}}$. It is the preferred solution in the absence of the constraints, either in the joint or in the Cartesian space, that we shall consider next.

Define the position, velocity, and acceleration limits of each joint, $j = 1, \ldots, n$, respectively as

$$\begin{aligned} Q_j^{min} &\leq q_j \leq Q_j^{max}, \\ V_j^{min} &\leq \dot{q}_j \leq V_j^{max}, \\ \Lambda_j^{min} &\leq \ddot{q}_j \leq \Lambda_j^{max}. \end{aligned} \tag{3}$$

Accordingly, inequality constraints can be defined at the velocity level for each joint as

$$\begin{aligned} &\dot{Q}_{min,j} = \\ &\max\left\{\frac{Q_j^{min} - q_j}{T}, V_j^{min}, -\sqrt{2\Lambda_j^{max}\left(q_j - Q_j^{min}\right)}\right\}, \\ &\dot{Q}_{max,j} = \\ &\min\left\{\frac{Q_j^{max} - q_j}{T}, V_j^{max}, \sqrt{2\Lambda_j^{max}\left(Q_j^{max} - q_j\right)}\right\}, \end{aligned} \tag{4}$$

where T is the sampling time. The velocity constraints in (4) is obtained by satisfying three requirements [17]: (1) the joint range limits should not be exceeded in the next sampled time instant; (2) the absolute joint velocity should be less than its specified limit; (3) the joint should be able to stop its motion before it reaches its closest joint range limit, assuming the maximal acceleration bound is respected.

Consider next r generic Cartesian control points distributed along the robot body, each of dimension $d_{cp,i} \in \{1, 2, 3\}$. Note that $d_{cp,i}$ represents the number of Cartesian directions along which we constrain the motion. Accordingly, each control point (for $i = 1, \ldots, r$) can be limited in motion either in all directions (in 3D case, it means $d_{cp,i} = 3$) or in selected directions only i.e., $d_{cp,i} < 3$. The desired position, velocity, and acceleration limits for each control point i can be defined as

$$\begin{aligned} \boldsymbol{P}_{cp,i}^{min} &\leq \boldsymbol{p}_{cp,i} \leq \boldsymbol{P}_{cp,i}^{max}, \\ \boldsymbol{V}_{cp,i}^{min} &\leq \dot{\boldsymbol{p}}_{cp,i} \leq \boldsymbol{V}_{cp,i}^{max}, \\ \boldsymbol{\Lambda}_{cp,i}^{min} &\leq \ddot{\boldsymbol{p}}_{cp,i} \leq \boldsymbol{\Lambda}_{cp,i}^{max}, \end{aligned} \tag{5}$$

where $\boldsymbol{p}_{cp,i} \in \mathbb{R}^{d_{cp,i}}$ is the position of the i-th control point. As done before for the joint space, the linear constraints for each control point can be defined, at the velocity level, as

$$\dot{\boldsymbol{P}}_{cp,i}^{min} = \max\left\{\frac{\boldsymbol{P}_{cp,i}^{min} - \boldsymbol{p}_{cp,i}}{T}, \boldsymbol{V}_{cp,i}^{min}, -\sqrt{2\boldsymbol{\Lambda}_{cp,i}^{max}\left(\boldsymbol{p}_{cp,i} - \boldsymbol{P}_{cp,i}^{min}\right)}\right\},$$
$$\dot{\boldsymbol{P}}_{cp,i}^{max} = \min\left\{\frac{\boldsymbol{P}_{cp,i}^{max} - \boldsymbol{p}_{cp,i}}{T}, \boldsymbol{V}_{cp,i}^{max}, \sqrt{2\boldsymbol{\Lambda}_{cp,i}^{max}\left(\boldsymbol{P}_{cp,i}^{max} - \boldsymbol{p}_{cp,i}\right)}\right\}, \tag{6}$$

To take into account all the inequality constraints in (3) and (5) while executing the desired task $\dot{\boldsymbol{x}}$, we define the augmented vector (of dimension $n + \sum_{i=1}^{r} d_{cp,i}$)

$$\boldsymbol{a} = \begin{pmatrix} \boldsymbol{q}^T & \boldsymbol{p}_{cp,1}^T & \boldsymbol{p}_{cp,2}^T & \cdots & \boldsymbol{p}_{cp,r}^T \end{pmatrix}^T, \tag{7}$$

and the augmented $(n + \sum_{i=1}^{r} d_{cp,i}) \times n$ matrix

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{J}_{cp,1}^T & \boldsymbol{J}_{cp,2}^T & \cdots & \boldsymbol{J}_{cp,r}^T \end{pmatrix}^T, \tag{8}$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix and $\boldsymbol{J}_{cp,i}$ is the $d_{cp,i} \times n$ Jacobian of the i-th control point position. Therefore, at a time instant $t = t_k = kT$, it is possible to define the generalized inequality constraints at the velocity level as

$$\boldsymbol{b}_{min}(t_k) \leq \dot{\boldsymbol{a}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \leq \boldsymbol{b}_{max}(t_k), \tag{9}$$

where $\boldsymbol{b}_{min}$ and $\boldsymbol{b}_{max}$ are the general limits augmented vectors and defined as

$$\boldsymbol{b}_{min} = \begin{pmatrix} \dot{Q}_{min,1} \ \ldots \ \dot{Q}_{min,n} & \dot{\boldsymbol{P}}_{cp,1}^{min^T} \ \ldots \ \dot{\boldsymbol{P}}_{cp,r}^{min^T} \end{pmatrix}^T,$$
$$\boldsymbol{b}_{max} = \begin{pmatrix} \dot{Q}_{max,1} \ \ldots \ \dot{Q}_{max,n} & \dot{\boldsymbol{P}}_{cp,1}^{max^T} \ \ldots \ \dot{\boldsymbol{P}}_{cp,r}^{max^T} \end{pmatrix}^T. \tag{10}$$

Satisfying the generalized inequality constraints in (9) leads to impose the original position and velocity bounds in a strict sense, taking into account the acceleration limits. Note that, when the robot control law is defined at the velocity level, acceleration limits can be treated only as soft constraints.

Accordingly, the problem we intend to solve can be formulated as a QP subject to linear equality and inequality constraints as follows

$$\min_{\dot{\boldsymbol{q}} \in \mathbb{R}^n,\, s \in [0,1]} \frac{1}{2}\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}} + \frac{1}{2}M(1-s)^2$$
$$\text{s.t.} \quad \boldsymbol{J}\dot{\boldsymbol{q}} = s\dot{\boldsymbol{x}}, \quad \boldsymbol{b}_{min} \leq \boldsymbol{A}\dot{\boldsymbol{q}} \leq \boldsymbol{b}_{max} \tag{11}$$

The parameter $M \gg 1$ serves as a penalty factor and can be used to favor the maximization of the task scaling factor s (ideally, $s = 1$) over the minimization of the squared norm of the joint velocity $\|\dot{\boldsymbol{q}}\|^2$.

III. The Generalized SNS Algorithm

We have revisited the original SNS algorithm in [17] so as to cover also the generalized constraints (9). We highlight here the main introduced differences.

The pseudo-code of the proposed scheme is presented as Algorithm 1. The method starts by initializing: a projection matrix $\boldsymbol{P} = \boldsymbol{I}$, a null-space joint velocity vector $\dot{\boldsymbol{q}}_N = \boldsymbol{0}$, a scaling factor $s^* = 0$, a null-space augmented velocity vector $\dot{\boldsymbol{a}}_N = \text{null}$, and an augmented saturation matrix $\boldsymbol{A}_{lim} = \text{null}$.

On the basis of the minimum norm velocity solution, the current commanded joint velocity is given by

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_N + (\boldsymbol{J}\,\boldsymbol{P})^{\#}\left(\dot{\boldsymbol{x}} - \boldsymbol{J}\,\dot{\boldsymbol{q}}_N\right), \tag{12}$$

Algorithm 1 Generalized Saturation in the Null Space (GSNS)

$\dot{\boldsymbol{q}}_N \leftarrow \boldsymbol{0},\ s^* \leftarrow 0,\ \boldsymbol{P} \leftarrow \boldsymbol{I},\ \boldsymbol{A}_{lim} \leftarrow \text{null},\ \dot{\boldsymbol{a}}_N \leftarrow \text{null}$
repeat
  limits_violated ← FALSE
  $\dot{\boldsymbol{q}} \leftarrow \dot{\boldsymbol{q}}_N + (\boldsymbol{J}\,\boldsymbol{P})^{\#}\left(\dot{\boldsymbol{x}} - \boldsymbol{J}\,\dot{\boldsymbol{q}}_N\right)$
  $\dot{\boldsymbol{a}} \leftarrow \boldsymbol{A}\,\dot{\boldsymbol{q}}$
  if $\exists\, h \in \left[1 : n + \Sigma_1^r d_{cp,i}\right] : (\dot{a}_h < b_{min,h}) \vee (\dot{a}_h > b_{max,h})$ **then**
    limits_violated ← TRUE
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{A}\,(\boldsymbol{J}\,\boldsymbol{P})^{\#}\,\dot{\boldsymbol{x}}$
    $\boldsymbol{\beta} \leftarrow \dot{\boldsymbol{a}} - \boldsymbol{\alpha}$
    $s_k \leftarrow \text{getTaskScalingFactor}(\boldsymbol{\alpha}, \boldsymbol{\beta})$
    $k \leftarrow \{\text{the most critical constraint}\}$
    if $s_k > s^*$ **then**
      $s^* \leftarrow s_k$
      $\dot{\boldsymbol{q}}_N^* \leftarrow \dot{\boldsymbol{q}}_N,\ \boldsymbol{P}^* \leftarrow \boldsymbol{P}$
    end if
    $\boldsymbol{A}_{lim} \leftarrow \text{concatenate}(\boldsymbol{A}_{lim}, \boldsymbol{A}_k)$
    $\dot{a}_N \leftarrow \begin{cases} \text{concatenate}(\dot{a}_N, b_{max,k}) & \text{if } (\dot{a}_h > b_{max,k}) \\ \text{concatenate}(\dot{a}_N, b_{min,k}) & \text{if } (\dot{a}_h < b_{min,k}) \end{cases}$
    $\boldsymbol{P} \leftarrow \boldsymbol{I} - (\boldsymbol{A}_{lim})^{\#}\,(\boldsymbol{A}_{lim})$
    if $\text{rank}(\boldsymbol{J}\boldsymbol{P}) < m \wedge k \notin \{\text{primary task}\}$ **then**
      $\dot{\boldsymbol{q}} \leftarrow \dot{\boldsymbol{q}}_N^* + (\boldsymbol{J}\,\boldsymbol{P}^*)^{\#}\left(s^*\dot{\boldsymbol{x}} - \boldsymbol{J}\,\dot{\boldsymbol{q}}_N^*\right)$
      limits_violated ← FALSE
    end if
  end if
  $\dot{\boldsymbol{q}}_N \leftarrow (\boldsymbol{A}_{lim})^{\#}\,\dot{a}_N$
until limits_violated = TRUE
$\dot{\boldsymbol{q}}_{SNS} \leftarrow \dot{\boldsymbol{q}}$

Algorithm 2 Optimal task scaling factor

function GETTASKSCALINGFACTOR($\boldsymbol{\alpha}, \boldsymbol{\beta}$)
  for $h \leftarrow 1 : n + \Sigma_1^r d_{cp,i}$ **do**
    $L_h \leftarrow b_{min,h} - \beta_h$
    $U_h \leftarrow b_{max,h} - \beta_h$
    if $\alpha_h < 0 \wedge L_h < 0$ **then**
      if $\alpha_h < L_h$ **then**
        $s_h \leftarrow L_h/\alpha_h$
      else
        $s_h \leftarrow 1$
      end if
    else if $\alpha_h > 0 \wedge U_h > 0$ **then**
      if $\alpha_h > U_h$ **then**
        $s_h \leftarrow U_h/\alpha_h$
      else
        $s_h \leftarrow 1$
      end if
    else
      $s_h \leftarrow 0$
    end if
  end for
  $s \leftarrow \min s_h$
return s
end function

which attempts to execute the desired task as efficiently as possible (with the lowest possible velocity norm) by enforcing the velocity of some overdriven joint/Cartesian constraints to saturation, thereby keeping the entire augmented velocity $\dot{\boldsymbol{a}}$ in (9) within the desired constraints box. If the solution in (12) is acceptable under the constraints (9), the algorithm terminates and outputs this velocity for controlling the robot at the current time instant t_k. On the other hand, if it violates one or more of the hard generalized constraints, the algorithm repeats the loop until an admissible solution is found. This is accomplished by first calling Algorithm 2 to determine the most critical constraint k, which corresponds to a constraint that has the smallest scaling factor s_k, among all limits (see Figure 2). Next, only the k-th joint/Cartesian constraint is saturated to its limit during each iteration, and $\dot{a}_N$ and $\boldsymbol{A}_{lim}$

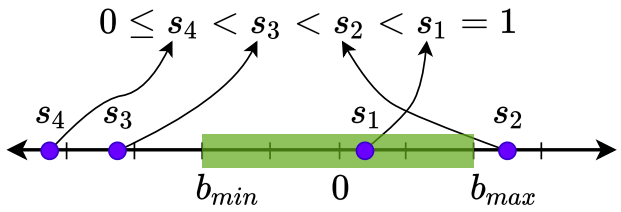

Figure 2: The task scaling factor associated with each constraint is computed in Algorithm 2. The factor is maximum (equal to 1 for the original task) when the corresponding velocity falls within the admissible interval, i.e., $b_{min,h} \le \dot{a}_h \le b_{max,h}$. In all other cases, the scaling factor is less than 1, and the constraint becomes more critical as the associated velocity moves further away from the boundaries of the interval.

are updated accordingly. Then, the solution (12) is recomputed, and the procedure is repeated.

At line 18, the projection matrix $\boldsymbol{P}$ is constructed according to the current saturated constraints as

$$\boldsymbol{P} = \boldsymbol{I} - (\boldsymbol{A}_{lim})^{\#} (\boldsymbol{A}_{lim}), \tag{13}$$

where $\boldsymbol{A}_{lim}$ incorporates the generalized active constraints. At line 16, $\boldsymbol{A}_k$ corresponds to the k-row of the augmented matrix $\boldsymbol{A}$ defined in (8). At the current iteration, if the most critical constraint is associated with the k-th joint, then the k-th row of the identity matrix $\boldsymbol{I}_{n\times n}$ is extracted and augmented to $\boldsymbol{A}_{lim}$. For the Cartesian constraints, a similar procedure is followed. For instance, if the most critical constraint is associated with the x-direction of the k-th control point, then the corresponding row of its Jacobian matrix $\boldsymbol{J}^x_{cp,k}$ is taken out and concatenated to $\boldsymbol{A}_{lim}$.

At the end of each iteration, the algorithm checks if the robot is still redundant in executing the primary task under the currently active set of constraints ($\text{rank}(\boldsymbol{JP}) \ge m$). This check fails when the task redundancy of the robot is exhausted, and there is no way to perform $\dot{\boldsymbol{x}}$ under the given constraints. In this case, the solution with the highest task scaling factor obtained so far (the value s_k that is closest to 1) is applied

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}^*_N + (\boldsymbol{J}\,\boldsymbol{P}^*)^{\#} (s^* \dot{\boldsymbol{x}} - \boldsymbol{J}\,\dot{\boldsymbol{q}}^*_N). \tag{14}$$

This choice preserves the geometry of the task, although it scales down the task speed by the (optimal) factor s^*.

Note that, if the current most critical constraint is associated with $\dot{\boldsymbol{x}}$, i.e., $k \in \{\text{primary task}\}$, e.g., when a control point coincides with the end-effector, then there is no need to scale down the task since its violated part is saturated to its limit, i.e., by modifying $\dot{a}_N$ and $\boldsymbol{A}_{lim}$ accordingly. In this way, differently from [17], Algorithm 1 is able to saturate the constraints in both the joint and the Cartesian space. Finally, we remark here that the proposed algorithm can be used to manage easily multiple Cartesian tasks with equal priority. This is essential for collision avoidance purposes when avoiding any obstacle in the work space is considered as a Cartesian task, e.g., in [3].

IV. Numerical Results

The efficiency of the new algorithm has been evaluated in simulations, on planar manipulators, and with experiments on a KUKA LWR IV robot ($n = 7$), see Figs. 1 and 3. For the presented case studies, a stabilizing feedback action is integrated into the desired EE velocity $\dot{\boldsymbol{x}}$ of the primary task to compensate for any numerical errors as

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_d + \boldsymbol{K}_p(\boldsymbol{x}_d - \boldsymbol{f}_{ee}(\boldsymbol{q})), \tag{15}$$

where $\boldsymbol{x}_d(\sigma)$ is the desired parametrized Cartesian path, $\sigma(t)$ is the timing law of the path parameter, $\boldsymbol{f}_{ee}(\boldsymbol{q})$ is the robot direct kinematics, and $\boldsymbol{K}_p > 0$ is the (diagonal) control gain matrix of dimension m. A suitable rest-to-rest timing law is considered in each case. The actual motions of the robots are shown in the accompanying video.

A. Simulations

1) 6R planar manipulator: A verification of Algorithm 1 has been done first through a MATLAB simulation. The EE of a 6R planar manipulator should track a 2D linear path ($m = 2$) in $T_{interval} = 10$ [s] with a 5^{th}-order polynomial timing law, see Fig. 3. The control gain matrix is set to $\boldsymbol{K}_p = \text{diag}\{5, 5\}$ and the sampling time is $T = 1$ [ms]. The initial robot configuration is chosen as

$$\boldsymbol{q}_0 = \begin{pmatrix} 30 & -30 & -30 & 60 & -30 & -30 \end{pmatrix}^T \text{ [deg]}. \tag{16}$$

In this example, the joint limits in (3) are equal and symmetric for all joints $j = 1, \ldots, 6$, where

$$\begin{aligned} Q_j^{max} &= -Q_j^{min} = 90 \ \text{[deg]}, \\ V_j^{max} &= -V_j^{min} = 15 \ \text{[deg/s]}, \\ \Lambda_j^{max} &= -\Lambda_j^{max} = 30 \ \text{[deg/s}^2\text{]}. \end{aligned} \tag{17}$$

As for Cartesian constraints, we considered $r = 5$ control points (each with $d_{cp,i} = 1$) along the robot body, located at the joints $i = 2, \ldots, 6$. The Cartesian limits in (3) are the same for all control points, and are imposed only along the y-direction:

$$\begin{aligned} P_{cp,i}^{max,y} &= -P_{cp,i}^{min,y} = 1 \ \text{[m]}, \\ V_{cp,i}^{max,y} &= -V_{cp,i}^{min,y} = 0.5 \ \text{[m/s]}, \\ \Lambda_{cp,i}^{max,y} &= -\Lambda_{cp,i}^{min,y} = 1 \ \text{[m/s}^2\text{]}. \end{aligned} \tag{18}$$

The EE begins its motion on the desired path without initial errors. As shown in Fig. 4, the position error is kept to zero throughout the task execution, except when the task scaling is active (i.e., $s^* < 1$) to comply with the saturated phases occurring in the joint and Cartesian motion —see Figs. 5 and 6. The robot is capable of completing the primary task while satisfying all hard inequality constraints (many of which are saturated).

2) Hyper-redundant planar manipulator: In order to assess the computational efficiency of our algorithm, we have considered a hyper-redundant planar robot with n revolute joints, with n varying from 20 to 200. We have chosen a link length of $l_j = 6/n$ [m], so that the total length of the robot is equal to that of the previous 6R planar robot. The robot starts from rest in the stretched configuration $\boldsymbol{q} = \boldsymbol{0}$, and its EE needs to track a 2D linear path in $T_{interval} = 5$ [s] with a 5^{th}-order polynomial timing law. In order to induce a large number $(n - m)$ of joint/Cartesian saturations during task execution, the joint position limits are chosen as

$$Q_j^{max} = -Q_j^{min} = \frac{800}{n} \ \text{[deg]}, \tag{19}$$

while the remaining joint and Cartesian constraints are the same as in (17) and (18).

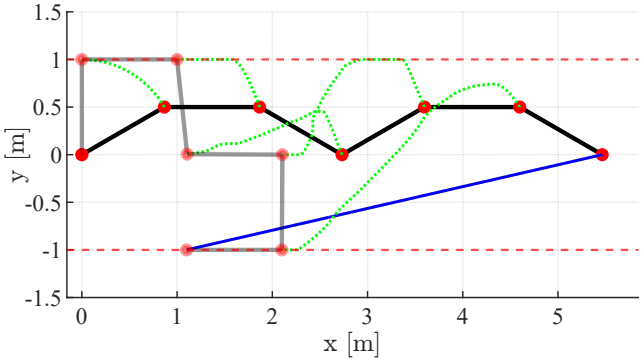

Figure 3: Simulation. The 6R planar arm is shown in its initial (black) and final (gray) configurations. The robot joints (and the end effector) are represented by red circles. The desired end-effector path is the blue line, to be traced from right to left. The Cartesian position bounds are indicated by the two dashed red lines. The dotted green lines are the paths of the chosen control points during task execution.

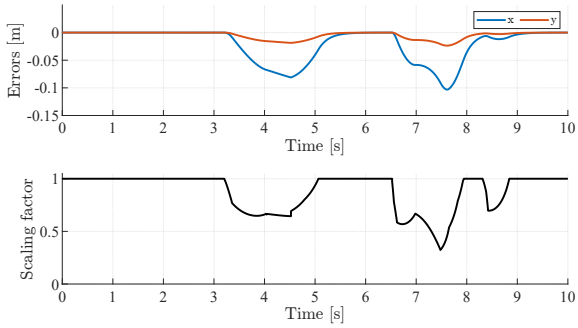

Figure 4: Simulation. The end-effector x and y position errors and the associated task scaling factor.

For comparison, we benchmarked our algorithm (GSNS) against a state-of-the-art QP active-set solver *qpOASES* [23], which solves the optimization problem as stated in (11). The simulations were conducted using MATLAB R2022a on an Intel Core i9-10900k CPU 3.7 GHz and 64 GB of RAM. The average execution time needed by the two algorithms are shown in Fig. 7. It can be clearly recognized that the GSNS consistently performs faster than *qpOASES*; the difference in execution times becomes larger as the number of degrees of freedom increases.

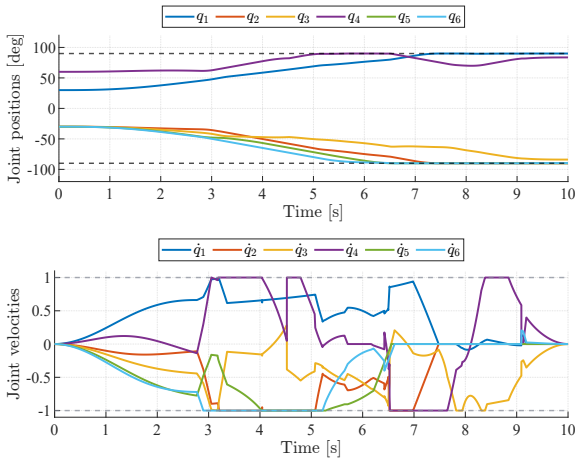

Figure 5: Simulation. Evolution of the position and velocity of the joints during task execution. The bounds on the joint motion are indicated by the dashed grey lines.

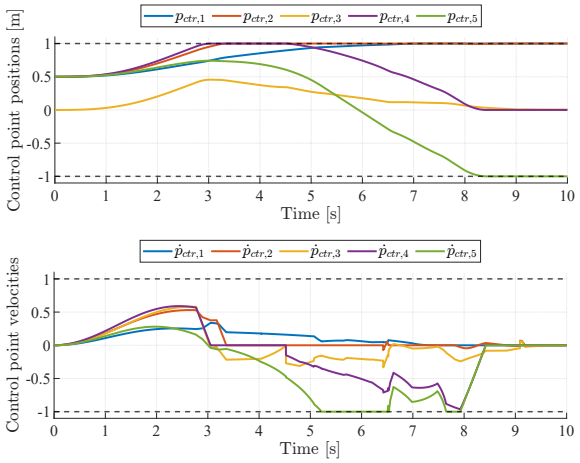

Figure 6: Simulation. Evolution of the position and velocity of the control points along the y-direction. The Cartesian bounds on the motion of the control points are indicated by the dashed grey lines.

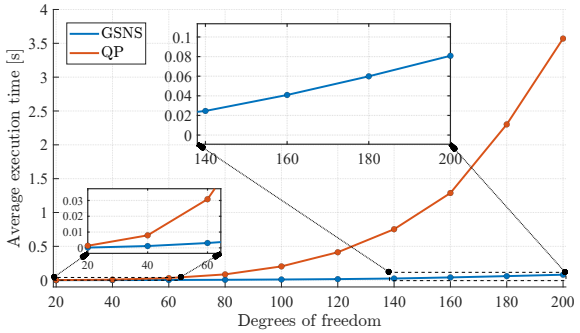

Figure 7: Simulation. Comparison of execution time for GSNS and a local QP solver (qpOASES).

B. Experiments with the KUKA LWR robot

The proposed Algorithm 1 has been implemented in C++ to perform experimental evaluations with a KUKA LWR IV robot ($n = 7$). A position control mode through the KUKA FRI library is used, with sampling time $T = 5$ [ms]. The results of two experiments are presented.

1) Cartesian constraints at the elbow: In the first experiment, the desired EE task is to track three times a 3D circle ($m = 3$), starting on the path with the initial joint configuration

$$\boldsymbol{q}_0 = (13.50 \;\; -7.76 \;\; 55.16 \;\; 79.70 \;\; 0 \;\; -6.19 \;\; 0)^T \;\; [\text{deg}].$$

Defining the world frame on the lab floor (see, Fig. 1), the desired circular path is centered at $\boldsymbol{C} = (0 \;\; 0.5 \;\; 1.5)^T$ [m], with a radius of 0.25 [m]. The timing law on the path has a trapezoidal velocity profile, with maximum acceleration $\ddot{\sigma} = 0.15$ [m/s^2] and cruise velocity $\dot{\sigma} = 0.15$ [m/s]. The control gain matrix is set to $\boldsymbol{K}_p = \text{diag}(30, 30, 30)$. The robot joint limits are set to

$$\boldsymbol{Q}^{max} = -\boldsymbol{Q}^{min} = (170 \;\; 105 \;\; 170 \;\; 120 \;\; 170 \;\; 85 \;\; 170)^T \;\; [\text{deg}],$$
$$\boldsymbol{V}^{max} = -\boldsymbol{V}^{min} = (20 \;\; 22 \;\; 20 \;\; 26 \;\; 26 \;\; 36 \;\; 36)^T \;\; [\text{deg/s}],$$
$$\boldsymbol{\Lambda}^{max} = -\boldsymbol{\Lambda}^{min} = (30 \;\; 30 \;\; 30 \;\; 30 \;\; 30 \;\; 30 \;\; 30)^T \;\; [\text{deg/s}^2].$$

A single control point of dimension $d_1 = 2$ is considered at the robot elbow (joint 4), which has to satisfy the temporal constraints

$$\begin{aligned} \boldsymbol{p}_{cp_x,1} &\leq 0.15 \;\; [\text{m}], \quad 16 \leq t \leq 22 \;\; [\text{s}], \\ \boldsymbol{p}_{cp_y,1} &\leq 0.2 \;\; [\text{m}], \quad\;\; 5 \leq t \leq 10 \;\; [\text{s}], \end{aligned} \tag{20}$$

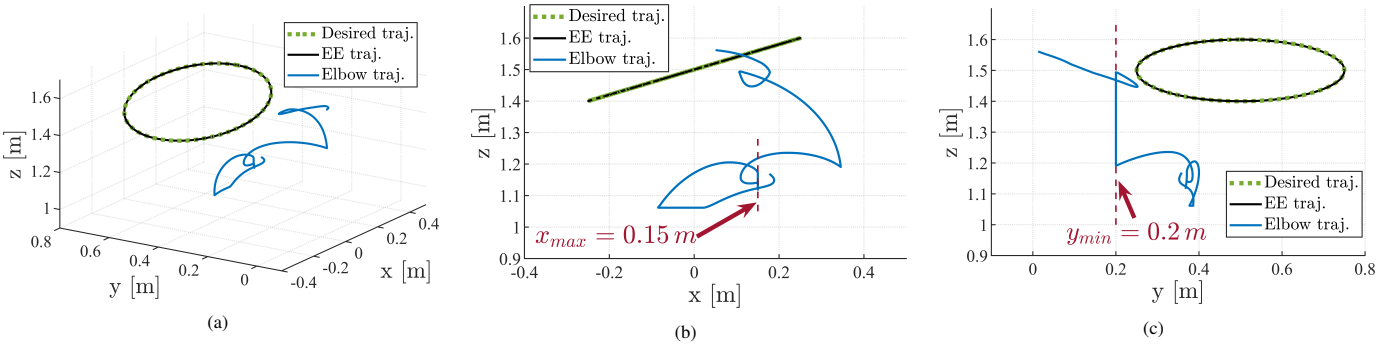

Figure 8: First experiment. (a) The motion executed by the end effector (in black) coincides with the desired circular path (dashed green). The position of the robot elbow (blue traces) satisfies the temporal constraints (20), both on the x-axis (b) and on the y-axis (c).

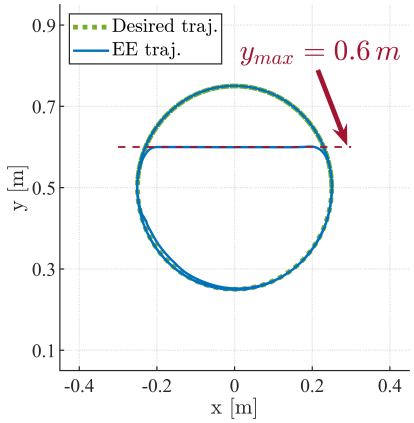

Figure 9: Second experiment. The end-effector motion (in blue) coincides with the desired circular path (dashed green) in the first and third rounds, while its position saturates the temporal constraint (22) during the second round.

and the permanent constraints

$$-0.1 \leq \dot{\boldsymbol{p}}_{cp_x,1} \leq 0.1, \quad -0.1 \leq \dot{\boldsymbol{p}}_{cp_y,1} \leq 0.1 \ [\mathrm{m/s}],$$
$$-0.5 \leq \ddot{\boldsymbol{p}}_{cp_x,1} \leq 0.5, \quad -0.5 \leq \ddot{\boldsymbol{p}}_{cp_y,1} \leq 0.5 \ [\mathrm{m/s^2}]. \tag{21}$$

Figure 8 shows how the robot executes the desired task by complying with the Cartesian constraints. In Fig. 11(a), the errors on the primary task are zero, except when no feasible solution exists under the considered hard constraints. In this case, the robot task is scaled down, i.e., $s^* < 1$, while keeping the EE velocity direction tangent to the desired path. In fact, the EE motion in Fig. 8 keeps nicely the geometry of the original path.

The corresponding evolution of the joints in Fig. 11(b) satisfies the hard joint limits at all times. The frequent saturation in position of joints 2, 3 and 6, as well as of all joint velocities (except for joint 7) clearly illustrates how Algorithm 1 exploits the available joint motion capabilities. The motion of the robot elbow (control point) is shown in Fig. 11(c). When the inequality constraints (20) are activated/deactivated (i.e., the shadowed areas), the elbow reacts properly and saturates, if necessary, to stay in the limits.

Figure 10a depicts the execution time of our algorithm (GSNS) and a local QP solver (qpOASES), solving the optimization problem stated in (11) with the penalty factor of $M = 10^3$. Both algorithms are implemented as C++ codes. The results indicate that GSNS achieves a higher computational speed than qpOASES. Figure 10b shows the number of iterations performed by GSNS at each instant during task execution. As it can be seen, the execution time of the GSNS rises with the increase in the number of iterations. However, the GSNS average execution time is only $18\,\mu$s, which is over an order of magnitude faster than that of qpOASES, running at $283\,\mu$s.

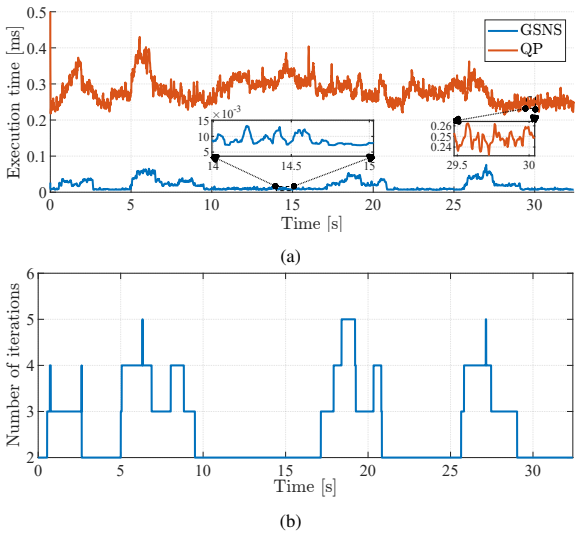

Figure 10: First experiment. Panel (a) shows the comparison of execution times for GSNS and a local QP solver (qpOASES). The number of iterations for GSNS over time is shown in panel (b).

2) Cartesian constraints at the end-effector: In the second experiment, a different single control point of dimension $d_1 = 2$ is chosen, placed coincident with the robot end effector. The primary task of dimension $m = 3$ requires the EE to follow three times a circular path as in the previous scenario, but the circle is rotated around the y-axis through its diameter so as to align with the xy-plane. The initial joint configuration is chosen as

$$\boldsymbol{q}_0 = (13.19 \ -8.59 \ 54.78 \ 89.04 \ -0.069 \ -10.09 \ 0)^T \ [\mathrm{deg}],$$

with the robot starting on the desired path. In this second case, we raise the maximum velocity and acceleration along the EE trajectory to $\dot{\sigma} = 0.65\,[\mathrm{m/s}]$ and $\ddot{\sigma} = 0.65\,[\mathrm{m/s^2}]$, respectively. This is to push the robot closer to its physical limits. Accordingly, we use the (symmetric) joint limits provided by the manufacturer

$$\boldsymbol{Q}^{max} = (170 \ 120 \ 170 \ 120 \ 170 \ 120 \ 170)^T \ [\mathrm{deg}],$$
$$\boldsymbol{V}^{max} = (100 \ 110 \ 100 \ 130 \ 130 \ 180 \ 180)^T \ [\mathrm{deg/s}],$$

with $\boldsymbol{Q}^{min} = -\boldsymbol{Q}^{max}$ and $\boldsymbol{V}^{min} = -\boldsymbol{V}^{max}$.

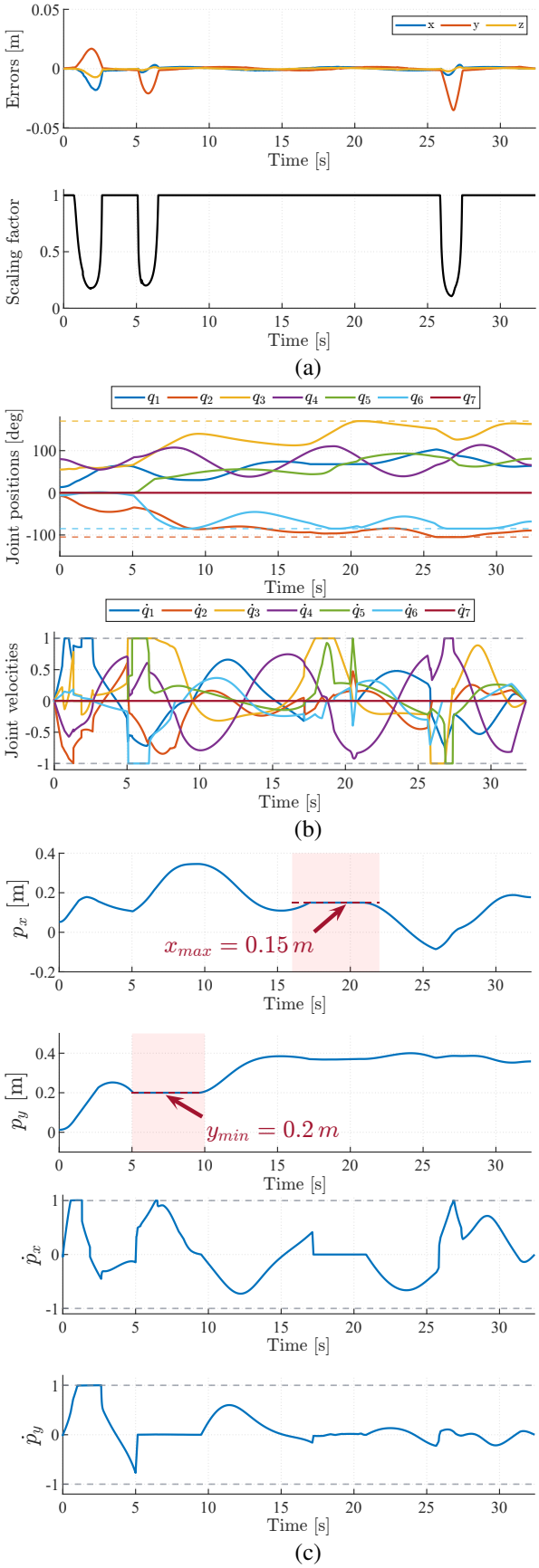

Figure 11: First experiment. (a) Error components on the primary task and the related optimal scaling factor. (b) Joint positions and normalized velocities. (c) The elbow (control point) position and normalized velocity components. The shadowed areas in pink represent the activation period of the constraints (20). Dashed lines indicate the associated bounds.

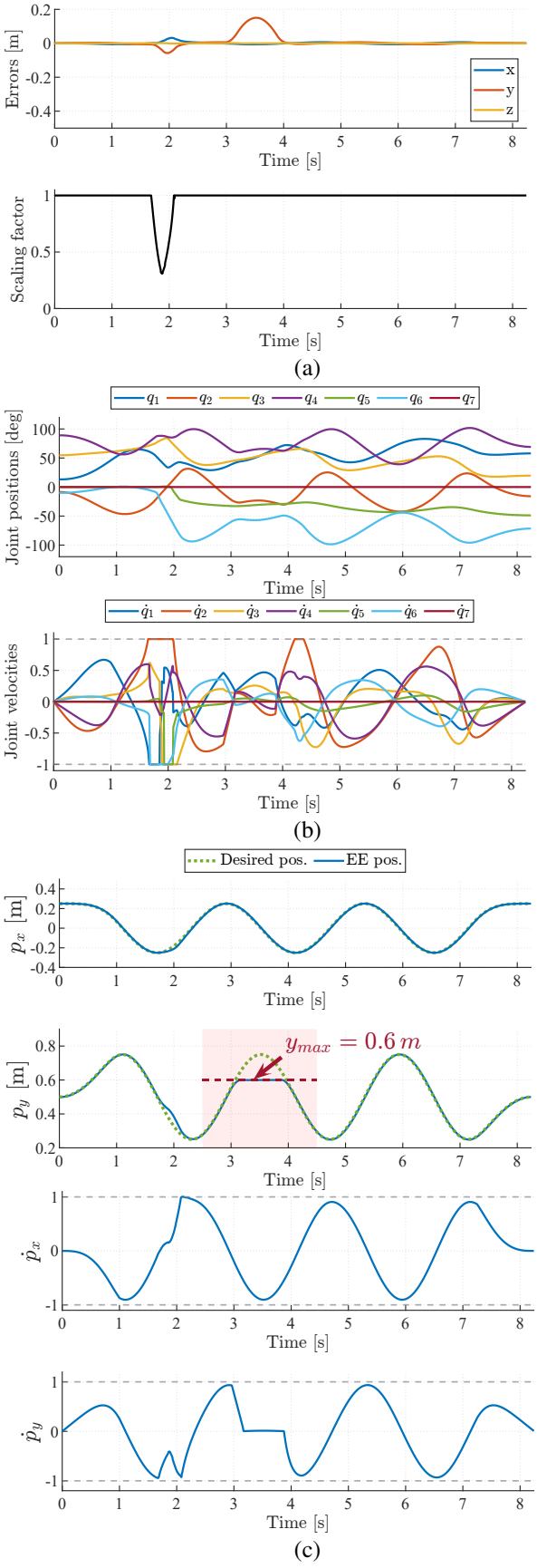

Figure 12: Second experiment. (a) Error components on the primary task and the related optimal scaling factor. (b) Joint positions and normalized velocities. (c) The end-effector (control point) position and normalized velocity components. The shadowed area in pink represents the activation period of the constraint (22). Dashed lines indicate the associated bounds.

Joint acceleration limits are set to $\boldsymbol{\Lambda}_j^{max} = -\boldsymbol{\Lambda}_j^{min} = 300\ [\mathrm{deg/s^2}]$, for $j = 1, \ldots, 7$. The temporal constraint

$$\boldsymbol{p}_{cp_y,1} \leq 0.6\ [\mathrm{m}], \quad 2.5 \leq t \leq 4.5\ [\mathrm{s}], \tag{22}$$

is imposed to the control point together with the permanent constraints

$$\begin{aligned} &-0.7 \leq \dot{\boldsymbol{p}}_{cp_x,1} \leq 0.7, \quad -0.7 \leq \dot{\boldsymbol{p}}_{cp_y,1} \leq 0.7\ [\mathrm{m/s}], \\ &-1.5 \leq \ddot{\boldsymbol{p}}_{cp_x,1} \leq 1.5, \quad -1.5 \leq \ddot{\boldsymbol{p}}_{cp_y,1} \leq 1.5\ [\mathrm{m/s^2}]. \end{aligned} \tag{23}$$

Figure 9 shows the execution of the task using Algorithm 1. In Fig. 12(a), the errors on the EE task increase around $t = 2$ [s], where the task scaling factor is applied in order to be able to satisfy the velocity limits of joints 1, 2 and 6 (see Fig. 12(b)). The EE error along the y direction between $t = 3$ and $t = 4$ [s] is large, because of the simultaneous activation of the (hard) temporal constraint (22), which is in fact inconsistent with the primary task. In this case, there is no use in scaling down the task as done instead in the former event. Moreover, while the EE position saturates at the imposed maximum limit in the y-direction, a complete fulfilment of the other task components along the x and z axes is still kept, see Fig. 12(c).

V. Conclusions

We have presented a generalized null-space saturation algorithm for the kinematic control of redundant robots to realize a primary task under hard inequality constraints in the joint and Cartesian spaces. All hard constraints are equally enforced and the task is automatically scaled in an optimal fashion when no feasible solution exists. The presented case studies have proven the efficiency of the approach to handle any possible simultaneous (de-)activation of joint and/or Cartesian inequality constraints without any oscillatory behavior. The ability to handle time-dependent constraints makes the method easy to be integrated in any sensor-based strategy, e.g., for online/dynamic collision avoidance in human-robot collaborative applications [22]. As for the original SNS algorithm [17], one can consider moving the commands to the acceleration level, making them suitable for torque-controlled robotic systems. However, a proper extension of the velocity-based SNS technique to acceleration or torque levels requires further investigation since the task-scaling strategy at the acceleration level is not always sufficient to ensure the satisfaction of the geometrical path due to the additional kinematic terms that arise. Furthermore, the problem of hierarchical task control is not addressed in this work. Nevertheless, similar to the original SNS, the presented algorithm can be extended to include multiple operational tasks with priorities, keeping the entire set of hard inequality constraints out of the stack of equality tasks.

References

[1] S. Chiaverini, G. Oriolo, and A. A. Maciejewski, "Redundant robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2016, pp. 221–242.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, Eds. Springer, 1986, pp. 396–404.

[3] M. Khatib, K. Al Khudir, and A. De Luca, "Task priority matrix at the acceleration level: Collision avoidance under relaxed constraints," *IEEE Robotics and Automation Lett.*, vol. 5, no. 3, pp. 4970–4977, 2020.

[4] ——, "Task priority matrix under hard joint constraints," in *Proc. 2nd Italian Conf. on Robotics and Intelligent Machines*, 2020, pp. 173–174.

[5] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Trans. on Robotics*, vol. 25, no. 3, pp. 670–685, 2009.

[6] E. Simetti and G. Casalino, "A novel practical technique to integrate inequality control objectives and task transitions in priority based control," *J. of Intelligent and Robotic Systems*, vol. 84, no. 1, pp. 877–902, 2016.

[7] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *Int. J. of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[8] ——, "A whole-body control framework for humanoids operating in human environments," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 2641–2648.

[9] E. Magrini, F. Ferraguti, A. J. Ronga, F. Pini, A. De Luca, and F. Leali, "Human-robot coexistence and interaction in open industrial cells," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101846, 2020.

[10] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[11] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators," *IEEE Trans. on Robotics and Automation*, vol. 11, no. 2, pp. 286–292, 1995.

[12] O. Kanoun, F. Lamiraux, and 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.

[13] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[14] E. M. Hoffman, A. Laurenzi, L. Muratore, N. G. Tsagarakis, and D. G. Caldwell, "Multi-priority Cartesian impedance control based on quadratic programming optimization," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2018, pp. 309–315.

[15] F. Flacco and A. De Luca, "Fast redundancy resolution for high-dimensional robots executing prioritized tasks under hard bounds in the joint space," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 2500–2506.

[16] A. S. Sathya, G. Pipeleers, W. Decré, and J. Swevers, "A weighted method for fast resolution of strictly hierarchical robot task specifications using exact penalty functions," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3057–3064, 2021.

[17] F. Flacco, A. De Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Trans. on Robotics*, vol. 31, no. 3, pp. 637–654, 2015.

[18] A. Ziese, M. D. Fiore, J. Peters, U. E. Zimmermann, and J. Adamy, "Redundancy resolution under hard joint constraints: A generalized approach to rank updates," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2020, pp. 7447–7453.

[19] M. Faroni, M. Beschi, N. Pedrocchi, and A. Visioli, "Predictive inverse kinematics for redundant manipulators with task scaling and kinematic constraints," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 278–285, 2018.

[20] J. D. M. Osorio, F. Allmendinger, M. D. Fiore, U. E. Zimmermann, and T. Ortmaier, "Physical human-robot interaction under joint and Cartesian constraints," in *Proc. 19th Int. Conf. on Advanced Robotics*, 2019, pp. 185–191.

[21] A. Kazemipour, M. Khatib, K. Al Khudir, and A. De Luca, "Motion control of redundant robots with generalised inequality constraints," in *Proc. 3rd Italian Conf. on Robotics and Intelligent Machines*, 2021, pp. 138–140.

[22] M. Khatib, K. Al Khudir, and A. De Luca, "Human-robot contactless collaboration with mixed reality interface," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 102030, 2021.

[23] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.