Task Priority Matrix Under Hard Joint Constraints

Maram Khatib
Dip. di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma
Roma 00185, Italy
khatib@diag.uniroma1.it

Khaled Al Khudir
School of Mechanical, Aerospace and Automotive Engineering
Coventry University
Coventry CV1 2JH, UK
ad5884@coventry.ac.uk

Alessandro De Luca
Dip. di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma
Roma 00185, Italy
deluca@diag.uniroma1.it

***Abstract*—We propose an extension to the Task Priority Matrix (TPM) method for redundancy resolution that includes also hard inequality joint constraints. This is done by combining TPM with a modified version of the basic Saturation in the Null Space (SNS) algorithm. Comparative simulations are reported for the 21-DOFs Romeo humanoid robot.**

***Index Terms*—Motion control, redundant robots, task priority, hard joint constraints.**

I. TASK PRIORITY CONTROL

For a kinematically redundant robot, the most common way to execute the desired Cartesian tasks with different assigned priorities is to use null-space projections [1]. This method is simple and ensures a strict priority. However, it requires recursions (from highest to lowest priority tasks, and possibly back) and suffers from joint velocity discontinuity during any change in the task priority order. Also, hard joint constraints could not be respected explicitly. The discontinuity problem can be solved by using suitable scaling factors so as to switch smoothly between the tasks [2]. Hard inequality constraints in the joint space are satisfied using the Saturation in the Null Space method (SNS) [3]. Alternatively, a unified optimization problem can be defined and then solved using the Hierarchical Complete Orthogonal Decomposition (HCOD) [4].

In [5], the Task Priority Matrix (TPM) method was proposed, which is simpler and faster than previous task priority approaches. Using TPM, redundancy resolution is totally separated from task priority handling. At the current $\boldsymbol{q}$, one has

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\boldsymbol{F}\dot{\boldsymbol{x}}, \tag{1}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ are the joint velocity commands,

$$\boldsymbol{J} = [\boldsymbol{J}_1^T \quad \dots \quad \boldsymbol{J}_i^T \quad \dots \quad \boldsymbol{J}_l^T]^T \tag{2}$$

is the augmented matrix with the Jacobians $\boldsymbol{J}_i \in \mathbb{R}^{m_i \times n}$ related to the l desired tasks ($\sum_{i=1}^{l} m_i = m$), and

$$\dot{\boldsymbol{x}} = [\dot{\boldsymbol{x}}_1^T \quad \dots \quad \dot{\boldsymbol{x}}_i^T \quad \dots \quad \dot{\boldsymbol{x}}_l^T]^T, \tag{3}$$

is the Stack of Tasks (SoT). The square matrix $\boldsymbol{F} \in \mathbb{R}^{m \times m}$ has to be computed in a specific way to impose the desired task priority. This is done depending only on the QR decomposition for the transpose of the augmented Jacobian $\boldsymbol{J}^T$, and using the Gauss-Jordan elimination method. In this case, any change in the task priority structure requires only to modify the $\boldsymbol{F}$ matrix in (1), which contains a compact but complete information on the different task dependencies. Note that the TPM method returns exactly the same solution obtained by the null-space projection method, but it is simpler and computationally faster. In [6], we extended and implemented the TPM method at the acceleration level to eliminate any discontinuities in the joint velocity due to any change in the task priorities or dependencies, and to include the consideration of robot dynamics for joint motion damping. However, joint inequality constraints have not been considered so far.

In this work, we propose another extension to the TPM method in order to include a particular class of inequality constraints in the priority stack of tasks, namely joint inequality constraints of the box type (i.e., limits on joint range and velocity). For this, we combine the basic SNS algorithm [3], designed to control a single Cartesian task under hard joint constraints, with the original TPM method. First, the $\boldsymbol{F}$ matrix is computed according to the desired priority of the Cartesian tasks, see [5] and [6] for details. Then, the modified basic SNS algorithm is applied as presented in Algorithm 1. The main difference w.r.t. the original SNS algorithm is in its core step, with the inverse differential solution computed now as

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_N + (\boldsymbol{J}\boldsymbol{W})^{\#}\boldsymbol{F}(\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}_N), \tag{4}$$

where $\dot{\boldsymbol{q}}_N$ and matrix $\boldsymbol{W} = diag\{W_{ii}\}$ with $0/1$ elements are used to apply saturated velocities on the joints that are overdriven by the simple pseudoinverse solution. According to Algorithm 1, joint inequality limits are always at first priority, and then the desired Cartesian tasks priority is imposed through the $\boldsymbol{F}$ matrix. When rank $(\boldsymbol{J}\boldsymbol{W}) < m$, the scale factor $s \leq 1$ is here uniformly applied to all tasks, regardless of their priority. When no joint limit is exceeded, i.e., $\boldsymbol{W} = \boldsymbol{I}$, the solution returned by Algorithm 1 is exactly as (1).

Our proposed solution can be seen as an alternative to the SNS algorithm for multiple tasks with priority [3]. When using the latter, however, the scaling factor is applied on each task separately and by the least amount.

II. RESULTS

A comparison between the TPM in (1) and the proposed Algorithm 1 is done through MATLAB simulations, using a simplified version of robot Romeo with 21 DOFs (see Fig. 1) and considering the same desired Cartesian tasks and other simulation parameters presented in [5]. For Algorithm 1, we

Algorithm 1 SNS-TPM algorithm

```
$\boldsymbol{W} = \boldsymbol{I}$, $\dot{\boldsymbol{q}}_N = \boldsymbol{0}$, $s = 1$, $s^* = 0$
**repeat**
    limit_exceeded = FALSE
    $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_N + (\boldsymbol{J}\boldsymbol{W})^{\#}\, \boldsymbol{F}\, (\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}_N)$
    **if** $\left\{ \begin{array}{l} \exists\ i \in [1\!:\!n] : \\ \dot{q}_i < \dot{Q}_{min,i} \text{ .OR. } \dot{q}_i > \dot{Q}_{max,i} \end{array} \right\}$ **then**
        limit_exceeded = TRUE
        $\boldsymbol{a} = (\boldsymbol{J}\boldsymbol{W})^{\#}\, \boldsymbol{F}\dot{\boldsymbol{x}}$
        $\boldsymbol{b} = \dot{\boldsymbol{q}} - \boldsymbol{a}$
        getTaskScalingFactor($\boldsymbol{a}$, $\boldsymbol{b}$)
        **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$\}$
        $W_{jj} = 0$
        $\dot{q}_{N,j} = \left\{ \begin{array}{ll} \dot{Q}_{max} & \text{if } \dot{q}_j > \dot{Q}_{max} \\ \dot{Q}_{min} & \text{if } \dot{q}_j < \dot{Q}_{min} \end{array} \right.$
        **if** rank$(\boldsymbol{J}\boldsymbol{W}) < m$ **then**
            $s = s^*$, $\boldsymbol{W} = \boldsymbol{W}^*$, $\dot{\boldsymbol{q}}_N = \dot{\boldsymbol{q}}_N^*$
            $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_N + (\boldsymbol{J}\boldsymbol{W})^{\#}\, \boldsymbol{F}\, (s\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}_N)$
            limit_exceeded = FALSE   (*outputs solution*)
        **end if**
    **end if**
**until** limit_exceeded = TRUE
```

considered the joint velocity limits given in the official data sheet of Romeo. Using Algorithm 1, task errors increase when a scale factor $s < 1$ needs to be applied to recover feasibility, see Fig. 2. However, Fig. 3 shows that all hard joint velocity limits are respected along the whole motion. While using TPM alone, the joint velocities would violate their limits frequently. See the accompanying video for a complete understanding.

References

[1] J.-J. Slotine and B. Siciliano, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th Int. Conf. on Advanced Robotics*, 1991, pp. 1211–1216.

[2] J. Lee, N. Mansard, and J. Park, "Intermediate desired value approach for task transition of robots in kinematic control," *IEEE Trans. on Robotics*, vol. 28, no. 6, pp. 1260–1277, 2012.

[3] 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.

[4] 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.

[5] F. Flacco, "The tasks priority matrix: A new tool for hierarchical redundancy resolution," in *Proc. 16th IEEE-RAS Int. Conf. on Humanoid Robots*, 2016, pp. 1–7.

[6] 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.

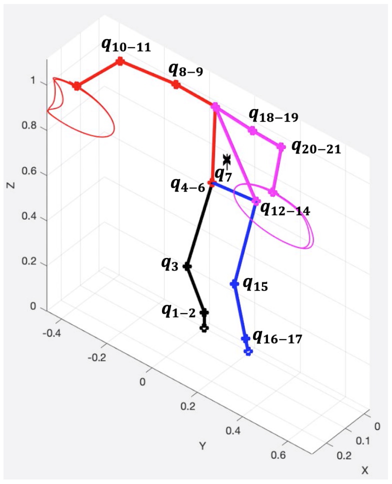

Fig. 1. Snapshot during Matlab simulation using Romeo.

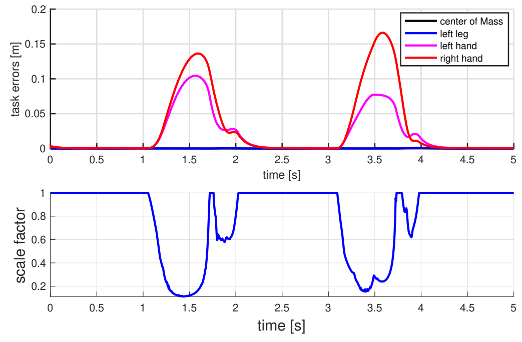

Fig. 2. Algorithm 1: Task errors (above) and the applied scale factor (below).

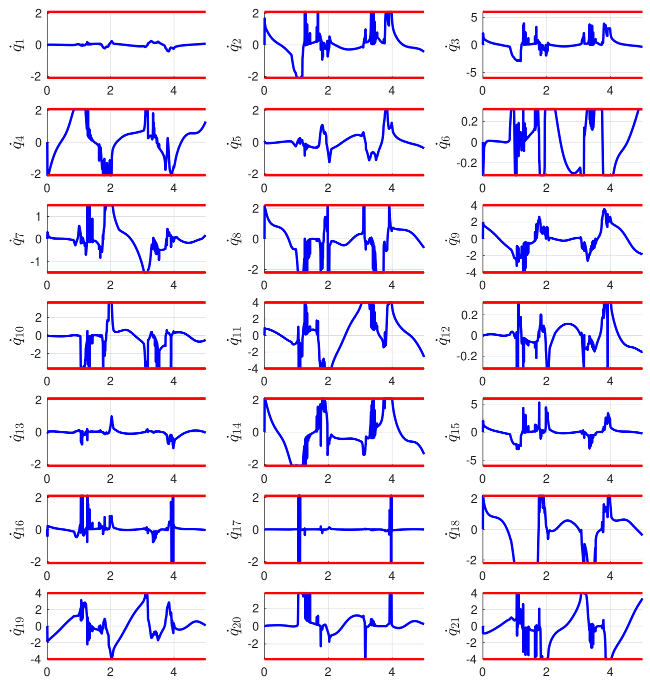

Fig. 3. Algorithm 1: Commanded joint velocities (in blue). The red lines represent the enforced joint limits.