A Predictive Technique for the Real-Time Trajectory Scaling Under High-Order Constraints

Corrado Guarino Lo Bianco, Marco Faroni, Manuel Beschi, and Antonio Visioli

***Abstract*—Modern robotic systems must be able to react to unexpected environmental events. To this purpose, planning techniques for the real-time generation/modification of trajectories have been developed in recent times. In the frequent case of applications which require following a predefined path, the assigned timing law must be inspected in real time so as to verify whether it satisfies the system constraints, or conversely, if it must be scaled in order to obtain a feasible trajectory. The problem has been addressed in several ways in the literature. One of the known approaches, based on the use of nonlinear filters, is revised in this article in order to return feasible solutions under any circumstances. Differently from alternative strategies, it manages constraints up to the torque derivatives and has evaluation times compatible with the ones required by modern control systems. The proposed technique is validated through simulations and real experiments. Comparisons are proposed with an algorithm based on a model predictive technique and with an alternative scaling system.**

***Index Terms*—Jerk bounded trajectories, manipulator dynamics, real-time trajectory scaling, robot kinematics, robot motion.**

I. Introduction

MODERN sensing devices equipping new robotic systems allow them to perceive the environment and react in real time to unexpected situations. Such reaction capability has encouraged developing algorithms for the real-time generation of trajectories whose computational times must be compatible with the sampling rates of discrete time controllers. Typically, planned trajectories must be time optimal, although almost-minimum-time solutions are often accepted because of the real-time requirement. Obviously, generated trajectories must take into account the physical limits of the system.

The first real-time trajectory planners for multiaxis robotic systems appeared in relatively recent times [1]–[3]. These early works devised a feasible point-to-point motion by taking into account kinematic constraints on joint speeds, accelerations, and jerks. Recent works reconsidered the problem by taking into account constraints up to the nth derivative of the generated signal [4] or by considering arbitrary initial and final interpolating conditions and asymmetric jerk bounds [5].

An alternative planning problem concerns the generation of trajectories along assigned paths, by adopting the so-called path-velocity decomposition [6]. This problem appears when the application requires to strictly follow an assigned path. Solutions provided in early works were all based on concepts deriving from the Pontryagin maximum principle so that minimum-time trajectories were obtained via numerical integration techniques. The first works considered trajectories in the joint space subject to torque constraints [7]–[9]. In [10], a similar technique was used to limit the maximum feeding voltage of the joint motors, while in [11], the approach was modified so as to manage torque-constrained problems. The planner proposed in [12] was similarly able to limit the actuators speeds and torques. The methodology was later extended in [13] by smoothing the trajectory through the adoption of continuous acceleration profiles. Recent techniques solve the planning problem by means of nonlinear programming techniques [14]–[16] Finally, few other works formulated the constrained planning problem in the Cartesian space by considering differential constraints on the robot Cartesian pose [17] or on a quadrotor pose [18].

The aforementioned planning methods have computational times in the order of 10^{-2} s or higher. Their real-time use is limited to applications with moderate sampling rates, or more commonly, for the generation of trajectories just before the motion execution. Many applications require, however, faster planning times. For example, modern robotic systems require prompt reactions to environmental changes and human behaviors: if a trajectory needs to be replanned while the system is moving, for example, in order to avoid collisions with other machines or human beings, the reaction time must be very short and "almost real-time" techniques cannot be used in such a context. The problem can be handled by generating on-the-fly a new collision-free path and a new timing law, so as to stave off the dangerous situation in minimum time. The available time

Manuscript received June 3, 2020; revised January 20, 2021; accepted February 26, 2021. Date of publication March 3, 2021; date of current version February 17, 2022. Recommended by Technical Editor Y. Liu and Senior Editor H. Gao. *(Corresponding author: Corrado Guarino Lo Bianco.)*

Corrado Guarino Lo Bianco is with the Dipartimento di Ingegneria dell'Informazione, University of Parma, 43121 Parma, Italy (e-mail: guarino@ce.unipr.it).

Marco Faroni is with the Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato, National Research Council, 20133 Milan, Italy (e-mail: m.faroni003@unibs.it).

Manuel Beschi and Antonio Visioli are with the Dipartimento di Ingegneria Meccanica e Industriale, University of Brescia, 25121 Brescia, Italy (e-mail: manuel.beschi@stiima.cnr.it; antonio.visioli@ing.unibs.it).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TMECH.2021.3063627.

Digital Object Identifier 10.1109/TMECH.2021.3063627

does not allow preliminary checks so that the resulting trajectory may be unfeasible with respect to the kinematics or the dynamics constraints, and consequently, collisions may occur because the path tracking is lost. Applications like this motivated the study of the so-called *online trajectory scaling problem*. Trajectory scaling consists in modifying the timing law while the trajectory is being executed, in order to prevent saturation of the robot physical limits. The technique proposed in this article belongs to this class of algorithms.

A. Related Works

The trajectory scaling problem aims to scale the assigned timing law to preserve the trajectory feasibility and to guarantee a good path tracking. The problem was originally addressed in [19] for a manipulator subject to speed and torque constraints. The problem is highly nonlinear, and according to real-time requirements, it must be solved with computational times dictated by the sampling rate of the system. The number of solving techniques proposed in the literature is quite limited.

The large majority of existing methods uses the path-velocity decomposition concept (rare exceptions can be found in [20] and [21]). These methods can be grouped in two families. The first one acts at control level: a path-following controller computes the robot joint commands based on the actual state from the robot. Differently from the path-tracking controller, these techniques aims to minimize the deviation with respect to the desired path by using the longitudinal velocity as an additional manipulated variable. Feedback controllers have been proposed, for example, in [22] and [23]. Model predictive control techniques are now becoming popular; they are also referred to as model predictive path-following control (MPFC) [24]–[27]. Most MPFC strategies generate in real time a minimum-time timing law for the system [24], [25]; few of them can be used also to scale a preassigned timing law, like in the case of the problem at hand [26], [27]. The solution is found by means of nonlinear programming algorithms that generate optimal feedback command signals.

The second family of methods modifies the assigned timing law at the planning level, by avoiding any interactions with the control system. They can be used with any control structures since they preserve the feasibility by acting on the reference signal through the knowledge of the system model. Also regarding this class of methods, both non look-ahead [28]–[30] and look-ahead methods [31], [32] have been proposed.

Control- and planning-based approaches have advantages and as well as disadvantages. As control-based approaches directly act at the control level, they can also partially compensate undesired behaviors caused by parametric uncertainties. On the other hand, they require *ad hoc* controllers (a peculiarity that is generally undesired by developers of robotic systems, who prefer solutions in which the motion control system and the planning system are kept disjoint). Vice versa, the lack of an actual feedback in planning-based approaches can cause constraint violations because of parametric uncertainties and unforeseen control actions. This is generally avoided by slightly down scaling the admissible bounds.

The technique proposed in this paper is based on a widespread planning-based approach, also named Trajectory Scaling System (TSS) [29], [33]. TSSs scale trajectories by first converting a given set of constraints, assigned in the joint space or in the Cartesian space, into equivalent bounds on velocity, acceleration, and when possible, jerk of the longitudinal timing law. If the equivalent limits are satisfied, the original constraints are fulfilled as well. In other words, the original multidimensional constrained problems is converted into a scalar one. The trajectory feasibility is achieved by scaling the timing law with the aid of nonlinear filters like the ones proposed in [33] and [34]. The filters, which are the core of the system, are designed so as to generate feasible output signals starting from references that may be potentially unfeasible. Input and output signals coincide as long as the former ones are feasible, otherwise the filter generates an output that satisfies the constraints and is as close as possible to the reference. The original signal is rejoined in minimum time as soon as it meets the feasibility conditions again. The earliest TSSs were able to manage velocity, acceleration, and torque constraints in the joint space [33]. The scaling strategy was later improved in [29] by introducing bounds on joint jerks and torque derivatives.

The main drawback of all the TSSs proposed up to now [29], [33] comes from the stringent assumption that path and timing law were provided to the scaling system in real time. As a consequence, the scaling system is not aware of the future trend of reference signals. If such condition applies, the feasibility of the generated references cannot be guaranteed with certainty, despite heuristic techniques have been proposed in order to mitigate the problem [35], [36].

B. Contribution

In this article, the scaling approach presented in [29] is revised in order to guarantee the convergence toward a feasible solution. Differently from the TSSs proposed in previous papers, this new technique inspects the assigned path for a finite look-ahead horizon. The preinspection mechanism is obtained by revising the strategy appeared in [34] and ensures that the resulting scaled trajectory is feasible in any circumstances.

The main advantage of the approach is that, differently from the other known look-ahead strategies, it manages bounds concerning the high-order dynamics of the system. More precisely, it simultaneously handles bounds on joint speeds, accelerations, jerks, torques, and torque derivatives. This is an important step ahead with respect to other approaches proposed in the literature that manage bounds on the low-order dynamics. For example, [31] addresses position, velocity, and acceleration limits, while [27] considers velocity and torque limits. This real-time achievement is possible because of the low computational burden of the TSS.

The approach has been tested in simulation and through experiments involving a Universal Robots UR10 manipulator. Comparisons with the earlier versions of the TSS proposed in [29] and [33] have shown that the new strategy returns significantly better results. Furthermore, this article also shows that the TSS and the MPFC return comparable solutions, but

the shorter evaluation times of the TSS allow for managing an increased number of constraints. The method effectiveness and the usefulness of the additional bounds on torque derivatives have been demonstrated in the supplementary material.

II. Preliminary Considerations

Robotic manipulators are subject to kinematics and dynamics constraints so that their trajectories should always be planned by considering the presence of limits on joint velocities $\dot{\mathbf{q}}(t) \in \mathbb{R}^N$, accelerations $\ddot{\mathbf{q}}(t) \in \mathbb{R}^N$, and torques $\boldsymbol{\tau}(t) \in \mathbb{R}^N$, where N indicates the number of joints. When possible, also jerks $\dddot{\mathbf{q}}(t) \in \mathbb{R}^N$ and torque derivatives $\dot{\boldsymbol{\tau}}(t) \in \mathbb{R}^N$ should be bounded in order to generate smoother motions. Analytically, the following equations must be satisfied:

$$\underline{\dot{\mathbf{q}}} \leq \dot{\mathbf{q}}(t) \leq \overline{\dot{\mathbf{q}}} \tag{1}$$

$$\underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}}(t) \leq \overline{\ddot{\mathbf{q}}} \tag{2}$$

$$\underline{\dddot{\mathbf{q}}} \leq \dddot{\mathbf{q}}(t) \leq \overline{\dddot{\mathbf{q}}} \tag{3}$$

$$\underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau}(t) \leq \overline{\boldsymbol{\tau}} \tag{4}$$

$$\underline{\dot{\boldsymbol{\tau}}} \leq \dot{\boldsymbol{\tau}}(t) \leq \overline{\dot{\boldsymbol{\tau}}} \tag{5}$$

where $\underline{\dot{\mathbf{q}}}, \underline{\ddot{\mathbf{q}}}, \underline{\dddot{\mathbf{q}}}, \underline{\boldsymbol{\tau}}$, and $\underline{\dot{\boldsymbol{\tau}}} \in (\mathbb{R}^-)^N$ and $\overline{\dot{\mathbf{q}}}, \overline{\ddot{\mathbf{q}}}, \overline{\dddot{\mathbf{q}}}, \overline{\boldsymbol{\tau}}$, and $\overline{\dot{\boldsymbol{\tau}}} \in (\mathbb{R}^+)^N$ are lower and upper bounds on velocities, accelerations, jerks, torques, and torque derivatives, respectively. Limits are typically assumed constant, but they may actually change depending on the operating conditions. In this article, bounds can be indifferently assumed constant or variable.

A trajectory $\mathbf{q}(t)$ in the configuration space can be defined by specifying the joint paths through a function $\mathbf{f}(s) \in \mathbb{R}^N$, where $s \in [0, s_f]$ is the curvilinear coordinate and s_f is the path length. In this article, the first derivative of the path with respect to the curvilinear coordinate is defined as follows: $\mathbf{f}'(s) := (d\mathbf{f})/(ds)$. The third path derivative, i.e., $\mathbf{f}'''(s) := (d^3\mathbf{f})/(ds^3)$, is assumed piecewise continuous so that $\mathbf{f}(s)$, $\mathbf{f}'(s)$, and $\mathbf{f}''(s)$ are continuous functions.

Path $\mathbf{f}(s)$ can be converted into a trajectory by specifying a timing law $s(t)$ for the curvilinear coordinate so that joint references can be expressed as follows:

$$\mathbf{q}(t) := \mathbf{f}[s(t)] \, . \tag{6}$$

Given such premises, and according to the differentiation chain rule, first, second, and third time derivatives of trajectories can be, respectively, represented as follows:

$$\dot{\mathbf{q}}(s, \dot{s}) = \mathbf{f}'(s)\dot{s} \tag{7}$$

$$\ddot{\mathbf{q}}(s, \dot{s}, \ddot{s}) = \mathbf{f}''(s)\dot{s}^2 + \mathbf{f}'(s)\ddot{s} \tag{8}$$

$$\dddot{\mathbf{q}}(s, \dot{s}, \ddot{s}, \dddot{s}) = \mathbf{f}'''(s)\dot{s}^3 + 3\mathbf{f}''(s)\dot{s}\ddot{s} + \mathbf{f}'(s)\dddot{s} \, . \tag{9}$$

Joint torques can be evaluated through the inverse dynamics function as follows:

$$\boldsymbol{\tau} = \mathbf{D}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{v}(\mathbf{q}, \dot{\mathbf{q}}) \tag{10}$$

where $\mathbf{D}(\mathbf{q})$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the matrix of Coriolis and centripetal contributions, $\mathbf{g}(\mathbf{q})$ is the gravity vector, and $\mathbf{v}(\mathbf{q}, \dot{\mathbf{q}})$ is the vector of the friction terms. Bearing in mind (6)–(8), (10) can be rewritten as follows:

$$\boldsymbol{\tau}(s, \dot{s}, \ddot{s}) = \mathbf{a}_1(s)\ddot{s} + \mathbf{a}_2(s, \dot{s}) \tag{11}$$

where

$$\mathbf{a}_1(s) := \widetilde{\mathbf{D}}(s)\mathbf{f}'(s) \tag{12}$$

$$\mathbf{a}_2(s, \dot{s}) := \widetilde{\mathbf{D}}(s)\mathbf{f}''(s)\dot{s}^2 + \widetilde{\mathbf{c}}(s, \dot{s})\dot{s} + \widetilde{\mathbf{g}}(s) + \widetilde{\mathbf{v}}(s, \dot{s}). \tag{13}$$

Analogously, by differentiating (10) and by considering (6)–(9), the torque derivative can be represented as follows:

$$\dot{\boldsymbol{\tau}}(s, \dot{s}, \ddot{s}, \dddot{s}) = \mathbf{a}_1(s)\dddot{s} + \mathbf{a}_3(s, \dot{s}, \ddot{s}) \tag{14}$$

where $\mathbf{a}_1(s)$ is given by (12) and

$$\begin{aligned}\mathbf{a}_3(s, \dot{s}, \ddot{s}) := {} & \dot{\widetilde{\mathbf{D}}}(s, \dot{s})[\mathbf{f}''(s)\dot{s}^2 + \mathbf{f}'(s)\ddot{s}] \\ & + \widetilde{\mathbf{D}}(s)[\mathbf{f}'''(s)\dot{s}^3 + 3\mathbf{f}''(s)\dot{s}\ddot{s}] \\ & + \widetilde{\mathbf{d}}(s, \dot{s})\dot{s} + 2\widetilde{\mathbf{c}}(s, \dot{s})\ddot{s} + \widetilde{\mathbf{l}}(s, \dot{s})\dot{s} \\ & + \widetilde{\mathbf{e}}(s, \dot{s})\ddot{s}\end{aligned} \tag{15}$$

where $\widetilde{\mathbf{d}}(s, \dot{s})\dot{s} + 2\widetilde{\mathbf{c}}(s, \dot{s})\ddot{s}$ comes from the differentiation of the Coriolis and centripetal terms, and $\widetilde{\mathbf{l}}(s, \dot{s})\dot{s} + \widetilde{\mathbf{e}}(s, \dot{s})\ddot{s}$ comes from the differentiation of the gravity and friction terms.

As shown in [29], (7)–(9), (11), and (14) make it possible to convert constraints (1)–(5) into the following equivalent bounds on $\dot{s}$, $\ddot{s}$, and $\dddot{s}$:

$$R^-(s) \leq \dot{s} \leq R^+(s) \tag{16}$$

$$S^-(s, \dot{s}) \leq \ddot{s} \leq S^+(s, \dot{s}) \tag{17}$$

$$U^-(s, \dot{s}, \ddot{s}) \leq \dddot{s} \leq U^+(s, \dot{s}, \ddot{s}) \, . \tag{18}$$

Practically, the dimension of the feasibility problem reduces, since it only requires synthesizing an appropriate scalar trajectory $s(t)$. The equivalent bounds can be evaluated as follows (see also [29]):

$$R^-(s) := \max_{k=1,\ldots,N}\{\zeta_k\}, \quad R^+(s) := \min_{k=1,\ldots,N}\{\eta_k\} \tag{19}$$

$$S^-(s) := \max_{k=1,\ldots,N}\{\beta_k, \mu_k\}, \quad S^+(s) := \min_{k=1,\ldots,N}\{\alpha_k, \lambda_k\} \tag{20}$$

$$U^-(s) := \max_{k=1,\ldots,N}\{\delta_k, \sigma_k\}, \quad U^+(s) := \min_{k=1,\ldots,N}\{\gamma_k, \rho_k\}. \tag{21}$$

The analytical expressions of the terms in (19)–(21) can be found in Table I. Bounds η_k and ζ_k come from (1), λ_k and μ_k come from (2), and ρ_k and σ_k come from (3). Analogously, (4) leads to α_k and β_k, and (5) leads to γ_k and δ_k. It is worth mentioning that, according to Table I, conditions $\eta_k > 0$ and $\zeta_k < 0$ are always satisfied so that $\forall s \in [0, s_f]$, the following expression holds with certainty:

$$R^-(s) < 0 < R^+(s) \, .$$

The same assertion is generally not true for the acceleration and the jerk bounds. For the planning problem considered in this article, the feasible set will be further limited in order to

TABLE I
EXPRESSIONS USED FOR THE EVALUATION OF THE EQUIVALENT LONGITUDINAL BOUNDS

	$f'_k > 0$	$f'_k < 0$	$f'_k = 0$
ρ_k	$[\dddot{\overline{q}}_k - f'''_k \dot{s}^3 - 3f''_k \dot{s}\ddot{s}]/f'_k$	$[\ddot{\underline{q}}_k - f'''_k \dot{s}^3 - 3f''_k \dot{s}\ddot{s}]/f'_k$	∞
σ_k	$[\ddot{\underline{q}}_k - f'''_k \dot{s}^3 - 3f''_k \dot{s}\ddot{s}]/f'_k$	$[\ddot{\overline{q}}_k - f'''_k \dot{s}^3 - 3f''_k \dot{s}\ddot{s}]/f'_k$	$-\infty$
λ_k	$[\ddot{\overline{q}}_k - f''_k \dot{s}^2]/f'_k$	$[\ddot{\underline{q}}_k - f''_k \dot{s}^2]/f'_k$	∞
μ_k	$[\ddot{\underline{q}}_k - f''_k \dot{s}^2]/f'_k$	$[\ddot{\overline{q}}_k - f''_k \dot{s}^2]/f'_k$	$-\infty$
η_k	$\dot{\overline{q}}_k/f'_k$	$\dot{\underline{q}}_k/f'_k$	∞
ζ_k	$\dot{\underline{q}}_k/f'_k$	$\dot{\overline{q}}_k/f'_k$	$-\infty$
	$a_{1k} > 0$	$a_{1k} < 0$	$a_{1k} = 0$
γ_k	$[\dot{\overline{\tau}}_k - a_{3k}(\dot{s}, \ddot{s})]/a_{1k}$	$[\dot{\underline{\tau}}_k - a_{3k}(\dot{s}, \ddot{s})]/a_{1k}$	∞
δ_k	$[\dot{\underline{\tau}}_k - a_{3k}(\dot{s}, \ddot{s})]/a_{1k}$	$[\dot{\overline{\tau}}_k - a_{3k}(\dot{s}, \ddot{s})]/a_{1k}$	$-\infty$
α_k	$[\overline{\tau}_k - a_{2k}(\dot{s})]/a_{1k}$	$[\underline{\tau}_k - a_{2k}(\dot{s})]/a_{1k}$	∞
β_k	$[\underline{\tau}_k - a_{2k}(\dot{s})]/a_{1k}$	$[\overline{\tau}_k - a_{2k}(\dot{s})]/a_{1k}$	$-\infty$

Dependence on s has been dropped for conciseness.

avoid backward movements by replacing (16) with the following expression:

$$0 \leq \dot{s} \leq R^+(s)\ . \tag{22}$$

III. TRAJECTORY SCALING SYSTEM

In the working scenario considered in [29], paths $\mathbf{f}(s)$ and timing laws $s(t)$ were provided to the system in real time and without any preliminary inspections, thus possibly leading to unfeasibility issues. In order to avoid tracking problems, they were handled at run time by the TSS in order to make them compatible with (1)–(5). In particular, the TSS was conceived so as to properly scale down $s(t)$ to preserve the feasibility, while maintaining an accurate tracking of $\mathbf{f}(s)$. Such target was achieved by means of the third-order nonlinear filter (TONF) proposed in [34], which is able to generate a new feasible reference, as close as possible to the nominal one, starting from a signal that is possibly unfeasible with respect to (16)–(18). If the input signal newly becomes feasible, the filter output reestablishes the tracking condition in minimum time and without overshoots. The mentioned properties apply when bounds on velocity, acceleration, and jerk are constant. Unfortunately, as shown by the expressions in Table I, the problem considered here admits bounds that depend on s, $\dot{s}$, and $\ddot{s}$: depending on the system state, alternative bounds are possible for the same value of s. Owing to the bounds variability, scaling techniques that are not aware of the upcoming references cannot preserve the feasibility with certainty. Indeed, as shown in [8], [9], [35], and [36], the system may enter regions from which it will certainly evolve toward unfeasible configurations, or even worse, S^+ and U^+ may become smaller than S^- and U^-, respectively, so that the space of the feasible solutions may result empty (see, for example, [37, Fig. 5]). The knowledge of the upcoming reference signals can be used to avoid such situations. In [35] and [36], issues deriving from variable bounds were managed by means of heuristic techniques exploiting the knowledge of the past system evolution. In this article, heuristic strategies are replaced by a deterministic approach, which assumes that an analytic representation of $\mathbf{f}(s)$ is provided at the beginning of the motion. Conversely, the *a priori* knowledge of $s(t)$ is not required so that it may also be changed at run time. It is worth mentioning that the manipulator starts moving immediately after $\mathbf{f}(s)$ is assigned, i.e., the problem is completely handled in real time through an algorithm whose computational burden is compatible with the sampling period.

The main difference between the heuristic techniques proposed earlier and the new deterministic one is that the latter always guarantees the generation of a scaled feasible trajectory, independently from the working conditions.

A. Existence of a Feasible Solution to the Trajectory Scaling Problem

The following definitions and propositions are instrumental for the comprehension of the scaling algorithm proposed in Section III-B.

Definition 1: The *admissible region* is the set of points $\mathcal{S}(s, \dot{s}, \ddot{s})$ in the $(s, \dot{s}, \ddot{s})$-space for which the following conditions simultaneously hold:

$$S^-(s, \dot{s}) \leq\ 0 \leq S^+(s, \dot{s}) \tag{23}$$

$$U^-(s, \dot{s}, \ddot{s}) \leq\ 0 \leq U^+(s, \dot{s}, \ddot{s})\,. \tag{24}$$

Practically, $\mathcal{S}(s, \dot{s}, \ddot{s})$ is the volume in the $(s, \dot{s}, \ddot{s})$-space individuated by (17), (18), and (22).

The definition of the admissible region here proposed is slightly different from the one typically used for the management of similar problems. For example, in [9], for an acceleration-bounded planning problem, the inequality to be satisfied was $S^-(s, \dot{s}) \leq S^+(s, \dot{s})$. Four different reasons motivated the alternative definition used in this article. First, the TSS generates feasible trajectories in real time, and to this purpose, the use of $\mathcal{S}(s, \dot{s}, \ddot{s})$ simplifies its convergence toward a solution since it allows a larger number of possible combinations. Second, the nonlinear filter, which is the core of the TSS, currently requires that $R^-, S^-, U^- \in \mathbb{R}^-$ and $R^+, S^+, U^+ \in \mathbb{R}^+$, even if such limit could be dropped by means of techniques derived from [37]. Third, the proposed trajectory scaling algorithm explicitly requires that $\ddot{s} = 0$ must belong to the feasible set. Finally, as already pointed out, upper and lower bounds on acceleration and jerk may invert so that it is always better to maintain a reasonable gap between them.

In Section III-B, it will be shown that the TSS exploits trajectories in which $\ddot{s} = 0$. To this purpose, let us introduce the concept of the *restricted admissible region.*

Definition 2: Assume $\ddot{s} = 0$. The *restricted admissible region* is the set of points $\overline{\mathcal{S}}(s, \dot{s})$ in the $(s, \dot{s})$-space for which the following conditions:

$$S^-(s, \dot{s}) \leq\ 0 \leq S^+(s, \dot{s}) \tag{25}$$

$$U^-(s, \dot{s}, 0) \leq\ 0 \leq U^+(s, \dot{s}, 0) \tag{26}$$

simultaneously hold.

Definition 3: $\overline{\mathcal{S}}(s, \dot{s})$ is *inferiorly connected* if $\widetilde{v}(s) > 0$ $\forall s \in (0, s_f)$, where $\widetilde{v}(s)$ is defined as follows:

$$\widetilde{v}(s) := \min\left\{R^+(s), \dot{\widetilde{s}}(s)\right\} \tag{27}$$

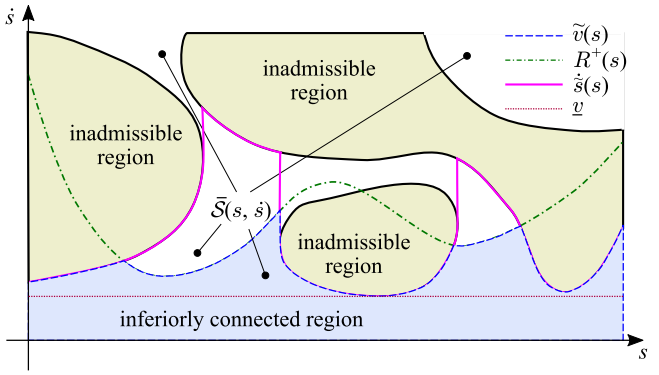

Fig. 1. Restricted admissible regions of $\overline{\mathcal{S}}(s,\dot{s})$ are obtained by subtracting the inadmissible regions from the $(s,\dot{s})$-space. The inferiorly connected subset is the light blue area superiorly limited by $R^+(s)$ (the green dash-dotted line) or by $\dot{\widetilde{s}}(s)$ (the magenta solid line). The dashed blue line highlights the upper bound $\widetilde{v}(s)$ of the inferiorly connected subset, while the dotted brown line represents $\underline{v}$.

and where $\dot{\widetilde{s}}(s)$ is the maximum value of $\dot{s}$ for which pair $\{s,[0,\dot{s}]\}$ belongs to $\overline{\mathcal{S}}(s,\dot{s})$.

Definition 3 can be immediately understood with the aid of Fig. 1, which shows a possible shape for $\overline{\mathcal{S}}(s,\dot{s})$ and for the inferiorly connected region. As already proven in [8], admissible regions may show very complex shapes with unfeasible islands, tunnels, and disjointed sections. This is even more true for the problem considered in this article, since additional constraints on jerk have been added.

The scaling technique proposed in this article requires inferiorly connected admissible regions. Fortunately, the following proposition applies.

Proposition 1: $\overline{\mathcal{S}}(s,\dot{s})$ is inferiorly connected when bounds on velocity, acceleration, and jerk are defined according to (19)–(21) and the system dynamics is given by (11) and (14).

Proof: For the sake of conciseness, the proof will assume that $f'_k(s) > 0$ and $a_{1\,k}(s) > 0$, but, with a similar reasoning, the results can be extended to the other cases. The proposition can be proved by demonstrating that all the points in the $(s,\dot{s})$-space which admit values of $\dot{s}$ sufficiently close to 0, certainly belong to $\overline{\mathcal{S}}(s,\dot{s})$. In turn, this implies that $\overline{\mathcal{S}}(s,\dot{s})$ is inferiorly connected. For all the equations reported in the following, it will be assumed that $k = 1,2,\ldots,N$ and s is any points belonging to $[0,s_f]$.

By denoting with $df_k(s)$ the elements of vector $\widetilde{\mathbf{D}}(s)\mathbf{f}''(s)$ and by using the same notation for the elements of $\widetilde{\mathbf{c}}(s,\dot{s})$, $\widetilde{\mathbf{g}}(s)$, and $\widetilde{\mathbf{v}}(s,\dot{s})$, each component of (13) can be written as follows:

$$a_{2\,k}(s,\dot{s}) = df_k(s)\dot{s}^2 + c_k(s,\dot{s})\dot{s} + g_k(s) + v_k(s,\dot{s})\ . \quad (28)$$

Since $df_k(s)$ and $c_k(s,\dot{s})$ are always finite, a positive constant $\mathcal{K}_k$ can be found such that the following equation applies $\forall s \in [0,s_f]$ and for sufficiently small values of $\dot{s} > 0$:

$$\left|df_k(s)\dot{s}^2 + c_k(s,\dot{s})\dot{s}\right| \le \mathcal{K}_k\dot{s}\ . \quad (29)$$

From Table I, and from (28) and (29), it follows that

$$\begin{aligned}\alpha_k(s,\dot{s})\,a_{1\,k}(s) &= \overline{\tau}_k - a_{2\,k}(s,\dot{s})\\ &\ge \overline{\tau}_k - g_k(s) - v_k(s,\dot{s}) - \mathcal{K}_k\dot{s} \quad (30)\end{aligned}$$

$$\begin{aligned}\beta_k(s,\dot{s})\,a_{1\,k}(s) &= \underline{\tau}_k - a_{2\,k}(s,\dot{s})\\ &\le \underline{\tau}_k - g_k(s) - v_k(s,\dot{s}) + \mathcal{K}_k\dot{s}. \quad (31)\end{aligned}$$

Since the manipulator motors are always selected so as to generate torques which surely compensate both gravity and friction effects, the following expression holds:

$$\underline{\tau}_k < g_k(s) + v_k(s,\dot{s}) < \overline{\tau}_k$$

and consequently, the following definitions apply:

$$\overline{\sigma}(s,\dot{s}) := \overline{\tau}_k - g_k(s) - v_k(s,\dot{s}) > 0 \quad (32)$$

$$\underline{\sigma}(s,\dot{s}) := \underline{\tau}_k - g_k(s) - v_k(s,\dot{s}) < 0\ . \quad (33)$$

By substituting (32) and (33) into (30) and (31), respectively, and by choosing sufficiently small values for $\dot{s}$, the following conditions are satisfied:

$$\alpha_k(s,\dot{s})a_{1\,k}(s) \ge \overline{\sigma}_k(s,\dot{s}) - \mathcal{K}_k\dot{s} > 0$$

$$\beta_k(s,\dot{s})a_{1\,k}(s) \le \underline{\sigma}_k(s,\dot{s}) + \mathcal{K}_k\dot{s} < 0\ .$$

Since $a_{1\,k}(s)$ was assumed positive, then $\alpha_k(s,\dot{s}) > 0$ and $\beta_k(s,\dot{s}) < 0$. In the same way, from Table I, it immediately descends that for sufficiently small values of $\dot{s}$, conditions $\lambda_k > 0$ and $\mu_k < 0$ are satisfied. As a consequence, by virtue of (20), (25) always holds for sufficiently small values of $\dot{s}$.

By recalling that $\ddot{s} = 0$ and since all functions that appear in (15) are finite $\forall s \in [0,s_f]$, it can be asserted that, for $\dot{s} \to 0$, there exists $\hat{\mathcal{K}} > 0$ such that

$$|a_{3\,k}(s,\dot{s},0)| < \hat{\mathcal{K}}\dot{s} \quad (34)$$

Table I makes it possible to write

$$\gamma_k(s,\dot{s},0)a_{1\,k}(s) = \dot{\overline{\tau}}_k - a_{3\,k}(s,\dot{s},0) \ge \dot{\overline{\tau}}_k - \hat{\mathcal{K}}\dot{s} \quad (35)$$

$$\delta_k(s,\dot{s},0)a_{1\,k}(s) = \dot{\underline{\tau}}_k - a_{3\,k}(s,\dot{s},0) \le \dot{\overline{\tau}}_k + \hat{\mathcal{K}}\dot{s} \quad (36)$$

which immediately leads to $\gamma_k(s,\dot{s},0) > 0$ and $\delta_k(s,\dot{s},0) < 0$ for $\dot{s} \to 0$. Since Table I also shows that when $\dot{s} \to 0$, $\rho_k > 0$ and $\sigma_k < 0$, (21) implies that (26) is satisfied. ■

Remark 1: Practically, Proposition 1 asserts that, if motor torques are sufficient for compensating gravity and friction terms, $\overline{\mathcal{S}}(s,\dot{s})$ is always inferiorly connected. This in turn implies, as shown in next subection, that a solution for the scaling problem can be found.

Proposition 2: By assuming that $\underline{v}$ is the minimum value of $\tilde{v}(s)$ for $s \in [0,s_f]$, i.e.

$$\underline{v} := \min_{s\in[0,s_f]}\{\tilde{v}(s)\} \quad (37)$$

any trajectories admitting a constant speed $\dot{s}$ is certainly feasible $\forall s \in [0,s_f]$ if the following condition holds:

$$0 < \dot{s} \le \underline{v}.$$

Proof: The following inequality necessarily holds $\forall s \in [0,s_f]$: $\dot{s} \le \underline{v} \le \tilde{v}(s) \le R^+(s)$ so that (16) is satisfied. In the same way, $\forall s \in [0,s_f]$, it can be asserted that $\dot{s} \le \underline{v} \in \overline{\mathcal{S}}(s,\dot{s})$ so that (25) and (26) apply. In turn, this implies that (it is worth recalling that $\dot{s}$ is constant so that $\ddot{s} = \dddot{s} = 0$) acceleration and jerk satisfy (17) and (18), respectively, so that the trajectory is feasible. ■

Algorithm 1: Solution of the Trajectory Scaling Problem.

Data: $\mathbf{q}_{lim} := [\underline{q}, \dot{\underline{q}}, \ddot{\underline{q}}, \dddot{\underline{q}}, \overline{q}, \dot{\overline{q}}, \ddot{\overline{q}}, \dddot{\overline{q}}, \underline{\tau}, \overline{\tau}, \dot{\underline{\tau}}, \dot{\overline{\tau}}]^T$, assigned bounds; $\mathbf{s}_i := [s_i\ \dot{s}_i\ \ddot{s}_i]^T$, current filter state; $\mathbf{s}_i^* := [s_i^*, \dot{s}_i^*, \ddot{s}_i^*]^T$, reference signal for the curvilinear coordinate.

Result: $\mathbf{s}_k$, scaled trajectory.

```
do
    (s̃_{i+1}, F_1) ← TONF_Update(s_i, s_i^*, f(s), q_lim);
    F_2 ← SolveESF(s̃_{i+1}, v_l, q_lim);
    if !F_1 | !F_2 then
        s̃_{i+1} ← SingleStepESF(s_i, v_l, q_lim);
    s_{i+1} ← s̃_{i+1};
    F_1, F_2 ← 0
while s_i < s_f;
```

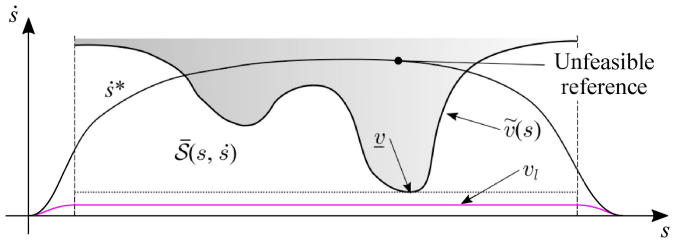

Fig. 2. Magenta curve represents a feasible, but inefficient, solution to the scaling problem.

Definition 4: The escape speed function (ESF) is the twice differentiable velocity function $\dot{s}(t)$, which is found by solving the following optimization problem:

$$\min\{t_f\} \tag{38}$$

subject to the acceleration and the jerk constraints, i.e., to (17) and (18), and to the following boundary conditions:

$$\dot{s}(0) = \dot{s}_0\,, \quad \ddot{s}(0) = \ddot{s}_0\,, \quad \dot{s}(t_f) = v_l\,, \quad \ddot{s}(t_f) = 0$$

where t_f is the total traveling time. The exact structure of the function $\dot{s}(t)$ has not been specified, since it is not relevant for the discussion. Potentially, any twice differentiable function can be used. For such reason, the problem optimizer has not been specified in (38).

The ESF is a minimum-time trajectory starting from a generic initial velocity, $\dot{s}_0$, and acceleration, $\ddot{s}_0$, and ending with final speed v_l and null acceleration. As the ESF is twice differentiable, the associated jerk is, at least, piecewise continuous. It is important to point out that, depending on $\dot{s}_0$, on $\ddot{s}_0$, and on constraints (17) and (18), the solution set of the optimization problem may be empty.

B. Look-Ahead Trajectory Scaling Algorithm

The trajectory scaling problem always admits the straightforward solution shown in Fig. 2. It is found by first imposing $v_l < \underline{v}$, in order to guarantee that the central part of the velocity function is feasible, being contained in $\overline{\mathcal{S}}(s, \dot{s})$. For the initial and final transients, the expressions in Table I can be used to

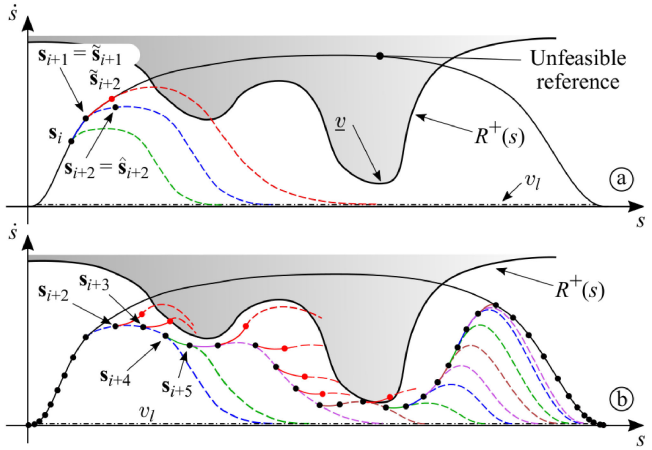

Fig. 3. Typical trajectory generated by means of Algorithm 1. (a) At time $i+1$, the TONF proposes evolving toward $\tilde{\mathbf{s}}_{i+2}$ (solid red line), but the subsequent transient obtained by solving the ESF is unfeasible (dashed red line): the TONF transient is discarded and the system evolves toward $\hat{\mathbf{s}}_{i+2}$. (b) Complete transient: Unfeasible combinations TONF + ESF (red curves) are always discarded in favor of feasible transients. The final scaled trajectory is highlighted by means of black dots.

individuate the maximum values that $\ddot{s}(t)$ and $\dddot{s}(t)$ should assume so as to satisfy (16)–(18). Roughly speaking, the feasibility is achieved by means of sufficiently smooth initial and final transients.

Unfortunately such solution shows two problems: it is evidently inefficient, because v_l may be small, and it totally neglects the user-defined timing law $\dot{s}^*$. The discrete-time approach proposed hereafter has been conceived to overcome both problems.

In the remainder of this section, subscript i identifies the sampling instants, so that:

1) $\mathbf{s}_i^* := [s_i^*, \dot{s}_i^*, \ddot{s}_i^*]^T$ is the reference trajectory at time $t = iT_s$, where T_s is the sampling rate;
2) $\tilde{\mathbf{s}}_i := [\tilde{s}_i, \dot{\tilde{s}}_i, \ddot{\tilde{s}}_i]^T$ is the output of a TONF system [34];
3) $\hat{\mathbf{s}}_i := [\hat{s}_i, \dot{\hat{s}}_i, \ddot{\hat{s}}_i]^T$ indicates the first sample of the ESF solution;
4) $\mathbf{s}_i := [s_i, \dot{s}_i, \ddot{s}_i]^T$ is the scaled trajectory, i.e., the output of Algorithm 1.

The trajectory scaling problem admits multiple solutions. Algorithm 1 proposes one of them, trying to keep it as close as possible to the assigned reference signal. In particular, at each sampling time, Algorithm 1 can only return one of the following two different outputs: $\mathbf{s}_i = \hat{\mathbf{s}}_{i+1}$, evaluated by solving the ESF, or $\mathbf{s}_i = \tilde{\mathbf{s}}_{i+1}$, evaluated with the TONF.

Fig. 3 can help to understand how such selection is made. For simplicity, trajectories in Fig. 3 are considered feasible when they lie below $R^+(s)$. Furthermore, $\underline{v}$ coincides with the minimum of $R^+(s)$. These simplifications have been introduced for presentation reasons, but the real algorithm actually checks that (17), (18), and (22) are simultaneously fulfilled so that the solution feasibility also depends on the acceleration and the jerk bounds.

Assume that at time i, the ESF solution is feasible: it will soon be shown that such condition is always satisfied. Thus, starting

from $\mathbf{s}_i$, it is possible to reach v_l through a feasible trajectory. The resulting composite profile—see the dashed green curve followed by the constant v_l segment in Fig. 3(a)—is a possible alternative solution for the scaling problem. It is faster than the one proposed in Fig. 2, but it is still inefficient.

Alternatively, starting again from $\mathbf{s}_i$, a new output $\widetilde{\mathbf{s}}_{i+1}$ can be obtained by means of the *TONF_Update* function (see line 1 of Algorithm 1). The inputs of such function are given by $\mathbf{s}_i$, $\mathbf{s}_i^*$, $\mathbf{f}(s)$, and $\mathbf{q}_{\text{lim}}$ [i.e., a vector that contains the bounds associated to (1)–(5)]. Flag $F_1 = 1$ indicates that the corresponding transient is feasible. Then, function *SolveESF* evaluates a new solution of the ESF problem starting from $\widetilde{\mathbf{s}}_{i+1}$ (see line 1 of Algorithm 1). If such solution is feasible, *SolveESF* returns $F_2 = 1$. If $F_1 = F_2 = 1$, then the combination TONF+ESF (blue segments in Fig. 3), is feasible and represents a further solution to the scaling problem. Such solution is better than the previous one—it is faster and it allows tracking the reference signal—so that the system output will be assigned as follows: $\mathbf{s}_{i+1} = \widetilde{\mathbf{s}}_{i+1}$ (see line 1 of Algorithm 1). It is important to notice that the solution of the ESF problem at time $i + 1$ (dashed blue line) will be necessarily feasible. This implies that two possible solutions will be potentially available at time $i + 1$: the first one, based on the ESF solution found at time i, is certainly feasible, the second one, based on a new combination TONF+ESF, is more efficient, but can be unfeasible. In the latter case, for example, the red transient in Fig. 3(a), one of the two flags, F_1 or F_2, will be zero: the next state is evaluated by *SingleStepEFS* (see line 1 of Algorithm 1), which returns the first step of the ESF transient found at time i. Practically, the reference signal is abandoned in order to preserve the feasibility.

Fig. 3(b) shows a possible further evolution of the system. In particular, it highlights that the system lingers on the dotted blue trajectory until a better alternative is found. At time $i + 4$, the sequence TONF + ESF returns a feasible combination so that $\mathbf{s}_{i+5}$ coincides with the TONF output. The system evolves according to such scheme until it safely reaches s_f.

Remark 2: As shown in Fig. 3(b), under normal operating conditions, the velocity v_l is never reached, since Algorithm 1 naturally generates profiles that are as close as possible to s^*. For such reason, the actual algorithm has been implemented by imposing $v_l = 0$. Such choice, which makes unnecessary to evaluate the actual value of $\underline{v}$, may be risky only if $\underline{v}$ is very close to 0 and if the sampling rate is too wide, since the system could actually stop. This is clearly a degenerate situation since, $\underline{v}$ close to 0 implies that the acceleration and jerk bounds almost overlap despite the system is moving very slowly. This situation only occurs if the motor torques cannot compensate the friction and the gravity terms.

Remark 3: The ESF can be solved with the aid of a nonlinear programming algorithm but, alternatively, very similar results can be obtained by means of the second-order nonlinear filter proposed in [33], i.e., the precursor of the TONF. The advantage of such filter is that it has negligible computational times, a very important characteristic for an online technique. The filter, which generates minimum-time position trajectories subject to bounds on velocity and acceleration, has been conversely used in this article for the generation of velocity transients admitting bounds on acceleration and jerk. Such result is achieved by assigning an initial state $\dot{s}(0) = \dot{s}_0$ and $\ddot{s}(0) = \ddot{s}_0$ to the filter, and by subsequently imposing an input reference signal equal to $\dot{s}(t) = v_l$ and $\ddot{s}(t) = 0$. It is worth remarking that, due to the variability of bounds (20) and (21), the resulting transients may be unfeasible. This is not a problem since, according to Algorithm 1, the information provided by *SolveESF* is a flag that simply points out if a feasible solution toward v_l has been found or not. If such solution is not available, the system evolves along the feasible trajectory found during the previous iteration.

Remark 4: A similar technique was proposed in [37] for a scaling problem considering bounded velocities, accelerations, and torques and a piecewise continuous longitudinal acceleration. It was based on a finite horizon preinspection of the trajectory. It is interesting to notice that the reasoning used in this article to prove the existence of a solution also applies to the strategy used in [37]. Such result can be achieved by expanding the inspection horizon used in that work. In particular, since in [37], the acceleration can be discontinuous, the ESF can be obtained by imposing $\ddot{s}_{i+1}$ equal to S_i^- until v_l is almost reached. The "safe" speed profile, i.e., v_l, is reached through a final step admitting acceleration $\ddot{s}_{i+1} = (v_l - \dot{s}_i)/T$. Conversely, in this article, accelerations must be continuous so that the ESF must be found by solving, at each sampling time, a continuous-acceleration velocity planning problem.

IV. Results and Discussion

The TSS has been validated by means of simulations and experimental tests. Sections IV-A describes the setup of the tests, and Sections IV-B and IV-C show simulation and experimental results, respectively.

A. Setup

The tests consider a Universal Robots UR10, version 3.5, manipulator, a six-link anthropomorphic system. The manipulator constraints have been obtained by slightly downgrading the ones suggested by the robot company. The following limits have been used for velocities, accelerations, and torques:

$$\dot{\mathbf{q}} = -\underline{\dot{\mathbf{q}}} = [2\ 2\ 3\ 3\ 3\ 3]^T \text{rad s}^{-1}$$

$$\ddot{\mathbf{q}} = -\underline{\ddot{\mathbf{q}}} = [5\ 5\ 10\ 10\ 10\ 10]^T \text{rad s}^{-2}$$

$$\overline{\boldsymbol{\tau}} = -\underline{\boldsymbol{\tau}} = [200\ 200\ 100\ 50\ 50\ 50]^T \text{Nm}.$$

The following bounds have been added when the scaling method makes it possible:

$$\dddot{\mathbf{q}} = -\underline{\dddot{\mathbf{q}}} = [50\ 100\ 100\ 100\ 100\ 100]^T \text{rad s}^{-3} \quad (39)$$

$$\dot{\boldsymbol{\tau}} = -\underline{\dot{\boldsymbol{\tau}}} = [700\ 700\ 700\ 50\ 20\ 10]^T \text{Nm s}^{-1}. \quad (40)$$

The outputs of the scaling methods are the reference signals for the robot controller, which is based on a cascade architecture: its outer loop is a position controller with sampling period equal to 8×10^{-3} s, while the inner loop is a velocity controller with a sampling period equal to 10^{-3} s. The outer position loop, together with the scaling algorithms, runs in ROS (Robot

Operating System) Kinetic on an external PC (operating system Ubuntu 16.04) communicating with the robot via a TCP connection. The position controller is given by a proportional gain plus a feedforward velocity term derived from the reference signal. In the real setup, the inner loop is built-in in the robot control system so that its actual structure is unknown. In the simulations, the inner loop was implemented as a proportional-integral controller with inverse dynamics decoupling.

The path used in the tests, parametrized as function of $s \in [0, 1]$, is expressed as follows:

$$\mathbf{q}_d = \mathbf{q}_{start} + \mathbf{\Omega} \sin(\omega s) \tag{41}$$

where $\mathbf{q}_{\text{start}} = [0, -2, 0, 0, 0, 0.5]^T$ rad, while different configurations were chosen for $\mathbf{\Omega}$ and ω, so as to generate the following two different test cases (all terms are expressed in radians).

1) Case A: $\mathbf{\Omega} = [0.3\ 0.6\ 0.7\ 0.65\ 0.75\ 0.8]^T$, $\omega = 2\pi$.
2) Case B: $\mathbf{\Omega} = -[0.3\ 0.6\ 0.7\ 0.65\ 0.75\ 0.8]^T$, $\omega = 3\pi$.

Nominal timing law $s^*(t)$, i.e., the timing law that is used for the generation of the possibly unfeasible reference, was obtained by considering a bang-zero-bang, piecewise-constant jerk signal so that the corresponding acceleration signal is characterized by continuous trapezoidal profiles. Rest-to-rest transients have been achieved by imposing initial and final velocities and accelerations equal to zero. A proper choice of upper and lower bounds on $\dot{s}^*(t)$, $\ddot{s}^*(t)$, and $\dddot{s}^*(t)$ allows trajectories with different traveling times. Depending on the chosen limits, trajectories $\mathbf{q}_d[s^*(t)]$ may be unfeasible.

B. Simulation Results

A comparison with different scaling methods is proposed. First of all, two versions of the predictive TSS have been implemented. The first one handles bounds on joint velocities, accelerations, and torques and will be referred as acceleration bounded TSS (A-TSS). The second one can also manage jerk and torque derivative constraints so that it is synthetically indicated as jerk bounded TSS (J-TSS). The performance of both TSSs has been compared with the ones obtained with the sole alternative methods proposed in the literature: the nonpredictive jerk bounded TSS (NJ-TSS) proposed in [29], which is the precursor of the novel scaling system and admits a zero look-ahead horizon, and the strategy recently proposed in [26] and [27], which uses an MPFC, and conversely, allows path preinspections. Similarly to [27], the MPFC has been implemented in ACADO [38] with the following settings: the predictive horizon is equal to 50 sampling periods, inputs are parametrized as staircase functions whose intervals are five times larger than the sampling period, and the problem is solved by means of a single-shooting SQP solver with maximum number of iterations equal to one. The problem in [27] minimizes the weighted sum of the squared norm of the path-following error, the deviation from the assigned timing law, and the control efforts (joint torques and second derivative of the timing law). Weights are, respectively, set to 10^8, 10^5, 0.5, and 10^{-7}. The characteristics of all systems are summarized in Table II.

TABLE II
CHARACTERISTICS OF THE TESTED SCALING SYSTEMS

	Vel.	Acc.	Jerk	Torque	Torque deriv.	Pred.
A-TSS	✓	✓		✓		✓
J-TSS	✓	✓	✓	✓	✓	✓
NJ-TSS	✓	✓	✓	✓	✓	
MPFC	✓	✓		✓		✓

The first columns indicate which constraints are considered, the last one points out predictive systems.

TABLE III
PATH TRACKING ERRORS AND SCALING FACTORS, OBTAINED THROUGH SIMULATIONS

(a) Case A

		t_{nom} [s] 5.0	4.0	3.5	3.0	2.5	2.0
$e_{\max}$ [rad]	A-TSS	0.53	1.39	2.71	3.76	3.84	5.65
	J-TSS	0.53	1.39	1.97	2.47	2.46	5.52
	NJ-TSS	12.0	16.4	17.5	18.5	18.8	19.8
	MPFC	0.52	1.78	2.26	2.53	3.14	4.03
e_{mean} [rad]	A-TSS	0.66	2.09	4.07	7.16	8.15	10.7
	J-TSS	0.66	1.93	3.40	4.17	4.85	6.02
	NJ-TSS	4.16	11.4	11.0	11.1	12.4	14.2
	MPFC	0.59	1.92	3.20	5.06	5.67	5.69
$t_{\text{nom}}/t_{\text{sc}}$	A-TSS	0.99	0.99	0.99	0.97	0.84	0.69
	J-TSS	0.99	0.99	0.99	0.92	0.80	0.65
	NJTSS	0.99	0.99	0.93	0.83	0.69	0.56
	MPFC	1.00	1.00	0.98	0.92	0.79	0.60

(b) Case B

		t_{nom} [s] 9.0	7.0	5.5	4.0	2.5
$e_{\max}$ [rad]	A-TSS	0.64	4.21	5.08	7.44	7.80
	J-TSS	0.64	2.46	2.43	2.85	2.95
	NJ-TSS	16.8	17.6	18.7	18.6	19.3
	MPFC	0.48	1.33	1.89	2.77	3.87
e_{mean} [rad]	A-TSS	0.37	2.20	5.57	8.03	9.16
	J-TSS	0.37	1.49	3.04	3.75	4.20
	NJ-TSS	12.8	17.5	18.9	20.1	21.0
	MPFC	0.33	1.10	2.67	5.34	4.19
$t_{\text{nom}}/t_{\text{sc}}$	A-TSS	1.00	0.99	0.99	0.86	0.57
	J-TSS	1.00	0.99	0.99	0.80	0.52
	NJ-TSS	1.00	0.94	0.77	0.59	0.38
	MPFC	1.00	1.00	0.97	0.84	0.47

Values associated to maximum errors ($e_{\max}$) must be multiplied by 10^{-3}, while average errors (e_{mean}) must be multiplied by 10^{-4}.

Comparisons are shown in Table III (a) and (b). They mainly concern path tracking errors, which are defined as follows:

$$e(iT_s) := \min_{s \in [0,1]} \|\mathbf{q}(iT_s) - \mathbf{q}_d(s)\|_2 \tag{42}$$

where $\mathbf{q}(iT_s)$ is the vector of the joint variables sampled at time iT_s. Each column of both tables refers to a different trajectory, individuated through its nominal traveling time (t_{nom}), i.e., the desired transient time. Both tables propose comparisons between the four alternative methods in terms of maximum ($e_{\max}$) and mean (e_{mean}) path tracking errors. The last four rows compare the approaches in terms of traveling times. More precisely, they show the ratios between the traveling times of nominal (t_{nom})

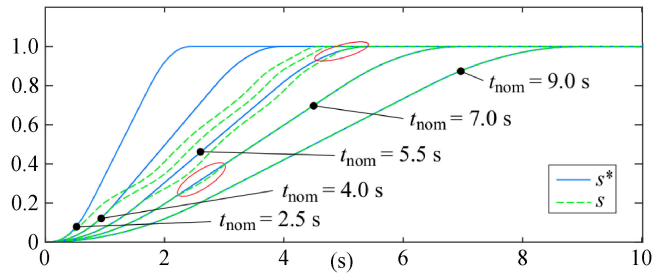

Fig. 4. Comparison between the nominal reference signal $s^*(t)$ and the output of the scaling system, i.e., $s(t)$, for Case B of the torque derivative constrained problem. The red ovals highlight situations in which $s(t)$ initially abandons, and then, rejoins $s^*(t)$. $s^*(t)$ exactly tracks $s(t)$ when this latter is feasible.

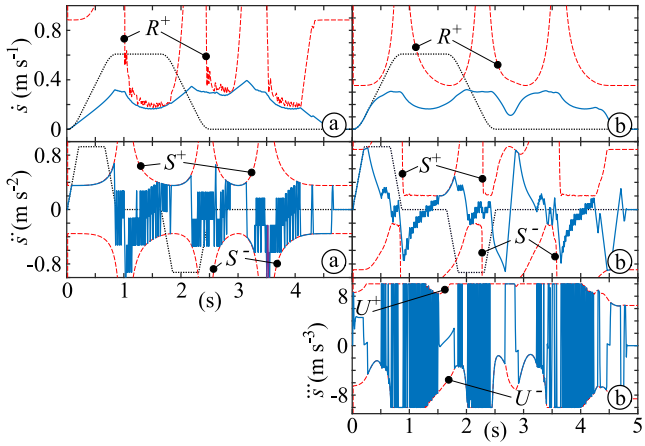

Fig. 5. Scaled longitudinal reference signals (solid blue lines) obtained for Case B and a nominal transient time $t_{nom} = 2.5$ s. (a) A-TSS. (b) J-TSS. Dashed red lines indicate the equivalent longitudinal bounds, while dotted black lines refer to the nominal, unfeasible reference. According to Table I, the equivalent bounds instantly depend on the state of motion so that the A-TSS and the J-TSS limits are different. The J-TSS transients are evidently smoother.

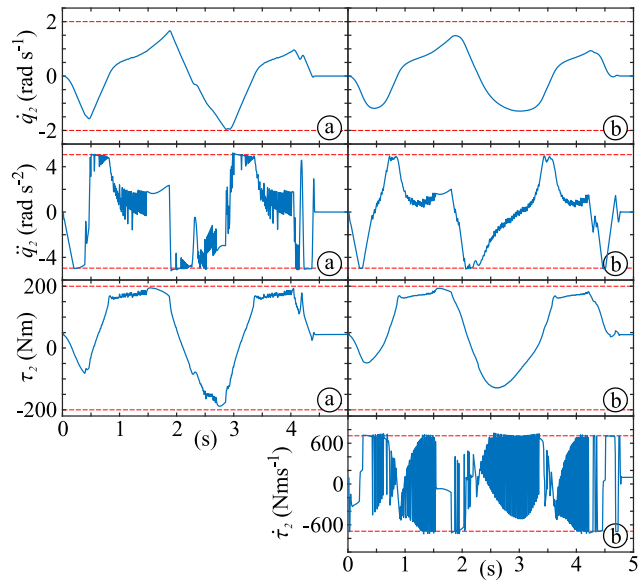

Fig. 6. Reference signals for the second joint, obtained (a) with the A-TSS and (b) with the J-TSS for Case B and a nominal transient time $t_{nom} = 2.5$ s. Assigned bounds are satisfied, but minor violations occur because of the long sample time of the control system. Transients of the J-TSS are evidently smoother.

and scaled (t_{sc}) trajectories: the closer the result to 1, the better it is. As expected, the nonpredictive technique of the NJ-TSS is outperformed by all the other methods: the heuristic strategy used for the avoidance of possible feasibility issues activates also in noncritical situations so that transient times are inevitably longer and the robot controller is solicited by frequent timing law changes. Conversely, a direct comparison between the A-TSS and the MPFC—which adopt, as shown in Table II, the same constraints—reveals that the first generally returns slightly faster transients, at the cost of slightly higher tracking errors. The J-TSS, which additionally bounds jerks and torque derivatives, shows similar scaling factors: despite the higher number of constraints, transient times are generally shorter than the MPFC ones. In conclusion, a deeper analysis of the results in Table III(a) and (b) highlights that differences between the three predictive techniques are minimal in terms of tracking errors and scaling factors, while worse performances have been detected for the NJ-TSS. As shown in Section IV-C, different conclusions can be drawn in terms of computational efficiency.

A relevant property of the scaling technique proposed in this article is represented by its capability to recover delays accumulated to preserve the feasibility. Fig. 4 shows the trends of $s(t)$ and $s^*(t)$ for all the experiments of Case B. More precisely, solid blue lines represent the assigned nominal references, while dashed green lines indicate the corresponding scaled signals provided by the J-TSS. The 9-s-long trajectory does not violate the assigned bounds so that the output and input of the J-TSS coincide, i.e., the trajectory is not scaled. On the contrary, for transients of length 2.5 and 4 s, the feasibility is preserved by scaling $s^*(t)$. The 5.5 and the 7 s transients show the aforementioned behavior: the nominal reference is left for feasibility reasons but, as soon as critical points are passed, the system accelerates to recover the reference signal (see the red ovals).

Figs. 5 and 6 still refer to Case B and to a nominal transient time $t_{nom} = 2.5$ s. Fig. 5 shows the scaled longitudinal timing law and its derivatives, while Fig. 6 shows the resulting reference signals for the second joint of the robot, i.e., the most solicited one. In both figures, (a) indicates the scaled profiles generated by the A-TSS, while (b) indicates the J-TSS profiles. Fig. 5 points out that, as desired, the scaling system prevents the problems highlighted in [35] and [37], so that conditions (23) and (24) are both satisfied and a feasible solution to the scaling problem is found. Both figures also highlight that the J-TSS output signals are evidently much smoother, and consequently, they less stress the system mechanics. Despite minor violations occasionally occur, constraints are generally satisfied. Violations are largely due to the wide sampling period imposed by the communication between PC and controller: faster sampling rates would reduce the amplitude and frequency of the violations.

TABLE IV
MEAN, STANDARD DEVIATION, AND MAXIMUM EVALUATION TIMES

	mean [s]	std [s]	max [s]
A-TSS	$4.28 \cdot 10^{-4}$	$3.92 \cdot 10^{-4}$	$2.62 \cdot 10^{-3}$
J-TSS	$6.63 \cdot 10^{-4}$	$3.11 \cdot 10^{-4}$	$5.17 \cdot 10^{-3}$
NJ-TSS	$7.80 \cdot 10^{-5}$	$9.78 \cdot 10^{-5}$	$2.32 \cdot 10^{-3}$

C. Experimental Results

The implementation of the scaling techniques on the actual control system required to take into account for the evaluation times of the algorithms. In [27], the MPFC showed a computational time close to 0.5 ms for a 3-dof robot. In that paper, acceleration constraints were not considered and the dynamic model of the robot was simplified, so as to neglect gravity terms and to consider a continuous Coulomb friction. Conversely, this article considers a 6-dof robot and accounts for the full dynamics of the system, including its friction discontinuities. The complexity of the optimal control problem associated to the MPFC exponentially grows depending on the number of joints and constraints. Some preliminary tests revealed that, owing to the higher complexity of the problem at hand and to the computational capabilities of the experimental setup, the MPFC computational time was incompatible with the sampling rate of the robot controller. For such reason, the results proposed in the following will only concern the TSS approaches. However, the simulations of Section IV-B have proved that, in terms of tracking errors and ratio $t_{\text{nom}}/t_{\text{sc}}$, the A-TSS and the MPFC algorithms are substantially equivalent so that similar experimental outcomes are expected.

A preinspection technique like the one proposed in Section III-B is exhaustive and it guarantees with certainty the generation of feasible profiles; however, it shows variable evaluation times—as the look-ahead horizon is not constant. Under realistic operating conditions, the look-ahead inspection can be stopped before v_l is reached and a constant time horizon can be used for all the trajectories. For the problem at hand, good performances were achieved for the A-TSS by adopting a time horizon of seven sampling instants, while for the J-TSS, the horizon had to be increased to ten intervals. Such values were obtained by repeating the simulations of Section IV-B for progressively shorter horizons and by simultaneously checking that performances were not affected by the changes.

The evaluation times statistics of the three algorithms are summarized in Table IV. The tests were performed on a Intel i7-6700HQ at 2.60-GHz CPU with a single-core implementation. In all the cases, peak times are compatible with the sampling period of the system (i.e., 8×10^{-3} s). The nonpredictive system shows the lowest computational burden since the look-ahead inspection is missing. The higher computational times of the J-TSS with respect to the A-TSS are partially due to the increased look-ahead horizon but, above all, to the evaluation of the torque derivatives.

The experimental results are summarized in Table V(a) and (b) for Cases A and B, respectively. Due to the model uncertainties, detected errors are clearly larger than simulated ones, but all trajectories have been correctly executed by the robot. Conversely, without the aid of scaling systems, the trajectories relative to $t_{\text{nom}} = 3, 2.5, 2$ s for Case A and the ones relative to $t_{\text{nom}} = 5.5, 4, 2.5$ s for Case B cannot be executed because they are too demanding, and consequently, the manipulator controller enters an emergency status.

TABLE V
MAX AND MEAN PATH TRACKING ERRORS EXPERIMENTALLY MEASURED

(a) Case A

		t_{nom} [s] 5.0	4.0	3.5	3.0	2.5	2.0
$e_{\max}$ [rad]	A-TSS	14.5	24.3	26.1	36.0	32.4	36.0
	J-TSS	14.8	25.4	32.2	33.4	34.2	33.7
	NJ-TSS	13.6	19.8	21.0	19.1	19.9	22.1
e_{mean} [rad]	A-TSS	0.73	1.12	1.39	1.89	1.86	2.21
	J-TSS	0.69	0.65	1.45	1.78	1.92	2.02
	NJ-TSS	0.75	1.25	1.28	1.63	1.72	2.03

(b) Case B

		t_{nom} [s] 9.0	7.0	5.5	4.0	2.5
$e_{\max}$ [rad]	A-TSS	9.72	17.9	25.2	34.2	36.1
	J-TSS	10.9	16.7	32.8	32.0	32.7
	NJ-TSS	12.9	14.1	19.8	20.1	23.3
e_{mean} [rad]	A-TSS	0.47	0.85	1.49	1.86	2.32
	J-TSS	0.53	0.81	1.51	1.67	1.89
	NJ-TSS	0.82	1.01	1.17	1.24	1.56

All values must be multiplied by 10^{-3}.

Another detail must be pointed out. All the three TSSs do not scale the slowest trajectories—see for example $t_{\text{nom}} = 5$ s for Case A and $t_{\text{nom}} = 9$ s for Case B—since they are feasible with respect to the constraints. It can be consequently asserted that measured tracking errors only depend on the performance of the feedback control system. It is interesting to notice that path tracking errors for the fastest trajectories of the test set remain at comparable levels, thus proving the effectiveness of the approach: error increments are only caused by the faster dynamics of the reference signals.

Tables V(a) and (b) point out that errors produced by the NJ-TSS are less dependent on the nominal time length of the trajectories. In particular, for the fastest transients, errors are more limited than the ones detected for the other two methods, i.e., apparently the nonpredictive technique returns better solutions. The reason of such behavior can be easily explained. The NJ-TSS trajectories are scaled more heavily than the ones generated by the other algorithms [see Table III(a) and (b)] so that the controller is less solicited and tracking errors are, consequently, smaller. For the same reason, the J-TSS generally produces slightly smaller errors than the A-TSS.

The multimedia attachment shows the execution of the two most demanding trajectories for Cases A and B, respectively. In the video, a trajectory generated by the A-TSS is immediately followed by the corresponding J-TSS one. The robot base was deliberately left unfixed with respect to the ground: system vibrations induced by the motion are highlighted through a water-filled bottle. The increased motion smoothness achieved

by bounding torque derivatives is evident: the A-TSS trajectories are only slightly faster [see the values of $t_{\text{nom}}/t_{\text{sc}}$ reported in Table III(a) and (b)], but vibrations produced by the J-TSS are evidently lower.

V. Conclusion

The method proposed in this article modifies the trajectory timing laws to make them feasible with respect to the given kinematics and dynamics constraints, and consequently, to guarantee an accurate path tracking. Compared to the existing strategies, the proposed approach guarantees the feasibility of the trajectory in any circumstances. Differently from alternative strategies, the proposed approach is able to take into account high-order constraints (namely, jerk and torque derivative bounds) in real time thanks to the use of an efficient nonlinear filter. Simulations and experiments have demonstrated that predictive strategies provide better performance than approaches based on causal data. In particular, the proposed algorithm modifies the assigned timing law less than non look-ahead techniques. Moreover, high-order constraints evidently improve the smoothness of the resulting motion.

It is worth noticing that, when tested on a real system, all methods gave comparable path-following errors because of the uncertainties of the system model and the consequent nonideal behavior of the real controller. Choosing either a look-ahead or a non look-ahead technique is, therefore, also a matter of computational capabilities of the available hardware. Limited hardware capabilities would impose using the NJ-TSS, as its evaluation times are, at least, one order of magnitude smaller. Alternatively, the A-TSS or J-TSS should be chosen to guarantee the aforementioned advantages.

References

[1] X. Broquère, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 2808–2813.

[2] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of timeoptimal, jerk-limited trajectories," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 3248–3253.

[3] T. Kröger and F. M. Wahl, "On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 94–111, Feb. 2010.

[4] B. Ezair, T. Tassa, and Z. Shiller, "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *Int. J. Robot. Res.*, vol. 33, no. 6, pp. 898–916, 2014.

[5] D. Sidobre and K. Desormeaux, "Smooth cubic polynomial trajectories for human-robot interactions," *J. Intell. Robot. Syst.*, vol. 95, no. 3, pp. 851–869, 2019.

[6] K. Kant and S. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[7] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," *J. Dyn. Syst. Meas. Control*, vol. 106, no. 1, pp. 102–106, 1984.

[8] K. G. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control*, vol. 30, no. 6, pp. 531–541, Jun. 1985.

[9] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotics manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, 1985.

[10] M. Tarkiainen and Z. Shiller, "Time optimal motions of manipulators with actuator dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1993, vol. 2, pp. 725–730.

[11] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1533–1540, Dec. 2014.

[12] P. Shen, X. Zhang, and Y. Fang, "Complete and time-optimal path-constrained trajectory planning with torque and velocity constraints: Theory and applications," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 2, pp. 735–746, Apr. 2018.

[13] P. Shen, X. Zhang, Y. Fang, and M. Yuan, "Real-time acceleration-continuous path-constrained trajectory planning with built-in tradeoff between cruise and time-optimal motions," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1911–1924, Oct. 2020.

[14] A. K. Singh and K. M. Krishna, "A class of non-linear time scaling functions for smooth time optimal control along specified paths," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2015, pp. 5809–5816.

[15] H. Pham and Q. Pham, "Time-optimal path tracking via reachability analysis," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2018, pp. 3007–3012.

[16] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[17] J. Kim and E. A. Croft, "Online near time-optimal trajectory planning for industrial robots," *Robot. Comput.-Integr. Manuf.*, vol. 58, pp. 158–171, 2019.

[18] X. Zhang, Y. Fang, X. Zhang, P. Shen, J. Jiang, and X. Chen, "Attitude-constrained time-optimal trajectory planning for rotorcrafts: Theory and application to visual servoing," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 4, pp. 1912-1921, Aug. 2020.

[19] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Trans. Robot. Autom.*, vol. 6, no. 5, pp. 554–561, Oct. 1990.

[20] F. Lange and M. Suppa, "Trajectory generation for immediate path-accurate jerk-limited stopping of industrial robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2021–2026.

[21] F. Lange and A. Albu-Schäffer, "Iterative path-accurate trajectory generation for fast sensor-based motion of robot arms," *Adv. Robot.*, vol. 30, no. 21, pp. 1380–1394, 2016.

[22] B. Olofsson and L. Nielsen, "Path-tracking velocity control for robot manipulators with actuator constraints," *Mechatronics*, vol. 45, pp. 82–99, 2017.

[23] Y.-S. Lu and Y.-Y. Lin, "Smooth motion control of rigid robotic manipulators with constraints on high-order kinematic variables," *Mechatronics*, vol. 49, pp. 11–25, 2018.

[24] M. Böck and A. Kugi, "Real-time nonlinear model predictive path-following control of a laboratory tower crane," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 4, pp. 1461–1473, Jul. 2014.

[25] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation," in *Proc. Eur. Control Conf.*, 2016, pp. 477–482.

[26] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 1026–1039, Apr. 2016.

[27] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of nonlinear model predictive path-following control for an industrial robot," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1505–1511, Jul. 2017.

[28] G. Antonelli, S. Chiaverini, and G. Fusco, "A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits," *IEEE Trans Robot Automat.*, vol. 19, no. 1, pp. 162–167, Feb. 2003.

[29] C. Guarino Lo Bianco and O. Gerelli, "Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints," *IEEE Trans. Robot.*, vol. 27, no. 6, pp. 1144–1152, Dec. 2011.

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

[31] M. Faroni, M. Beschi, N. Pedrocchi, and A. Visioli, "Predictive inverse kinematics for redundant manipulators with task scaling and kinematic constraints," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 278–285, Feb. 2019.

[32] M. Faroni, M. Beschi, C. Guarino Lo Bianco, and A.Visioli, "Predictive joint trajectory scaling for manipulators with kinodynamic constraints," *Control Eng. Pract.*, vol. 95, 2020, Art. no. 104264.

[33] C. Guarino Lo Bianco and F. Wahl, "A novel second order filter for the real-time trajectory scaling," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 5813–5818.

[34] C. Guarino Lo Bianco and F. Ghilardelli, "A discrete-time filter for the generation of signals with asymmetric and variable bounds on velocity, acceleration, and jerk," *IEEE Trans. Ind. Electron.*, vol. 61, no. 8, pp. 4115–4125, Aug. 2014.
[35] C. Guarino Lo Bianco and F. Ghilardelli, "Techniques to preserve the stability of a trajectory scaling algorithm," in *Proc. IEEE Int. Conf. Robot. Autom*, 2013, pp. 870–876.
[36] C. Guarino Lo Bianco and F. Ghilardelli, "A scaling algorithm for the generation of jerk-limited trajectories in the operational space," *Robot. Comput.-Integr. Manuf.*, vol. 44, pp. 284–295, 2017.
[37] M. Yuan, Z. Chen, B. Yao, and J. Hu, "An improved online trajectory planner with stability-guaranteed critical test curve algorithm for generalized parametric constraints," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 5, pp. 2459–2469, Oct. 2018.
[38] B. Houska, H. Ferreau, and M. Diehl, "ACADO toolkit—An open source framework for automatic control and dynamic optimization," *Optimal Control Appl. Methods*, vol. 32, pp. 298–312, 2011.

Corrado Guarino Lo Bianco received the graduate degree (with honors) in electronic engineering and the Ph.D. degree in control system engineering from the University of Bologna, Bologna, Italy, in 1989 and 1994, respectively.

He is currently with the Dipartimento di Ingegneria dell'Informazione, University of Parma, Parma, Italy, as an Associate Professor in industrial robotics. His research interests include mobile and industrial robotics.

Marco Faroni received the bachelor's and master's degrees in industrial automation engineering and the Ph.D. degree in mechanical and industrial engineering from the University of Brescia, Brescia, Italy, in 2013, 2015, and 2019, respectively.

He is currently a Researcher with the Italian National Research Council, Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing, Milan, Italy.

Manuel Beschi received the bachelor's and master's degrees from the University of Brescia, Brescia, Italy, in 2008 and 2010, respectively, both in industrial automation engineering and the Ph.D. degree in computer science, engineering and control systems technologies from the Department of Mechanical and Industrial Engineering.

He is currently an Assistant Professor with the University of Brescia.

Antonio Visioli received the Laurea degree in electronic engineering from the University of Parma, Parma, Italy, in 1995, and the Ph.D. degree in applied mechanics from the University of Brescia, Brescia, Italy, in 1995 and 1999, respectively.

He is currently a Full Professor of control systems with the Department of Mechanical and Industrial Engineering, University of Brescia.