Predictive Inverse Kinematics for Redundant Manipulators with Task Scaling and Kinematic Constraints

Marco Faroni, Manuel Beschi, Nicola Pedrocchi, Antonio Visioli

Predictive Inverse Kinematics for Redundant Manipulators with Task Scaling and Kinematic Constraints

Marco Faroni, *Student Member, IEEE,* Manuel Beschi, Nicola Pedrocchi, and Antonio Visioli, *Senior Member, IEEE*

***Abstract*—The paper presents a fast online predictive method to solve the task-priority differential inverse kinematics of redundant manipulators under kinematic constraints. It implements a task-scaling technique to preserve the desired geometrical task, when the trajectory is infeasible for the robot capabilities. Simulation results demonstrate the effectiveness of the methodology.**

***Index Terms*—Inverse kinematics, trajectory scaling, redundant robots, task priority, model predictive control, motion planning.**

I. Introduction

The inverse kinematics (IK) of robot manipulators permits to determine the joint variables associated with a given task (e.g., in the Cartesian space). Generally, the solution of this problem is not trivial, since it involves the inversion of multivariable nonlinear functions. Moreover, when the robot is kinematically redundant, the problem becomes even more complex: the system is underdetermined and it admits infinite solutions. Among them, the IK algorithm should choose the one that tends to satisfy secondary tasks or, similarly, optimize a given performance index (e.g., the manipulability). Typically, the IK is solved at velocity or acceleration level. As a matter of fact, at differential level, it is possible to exploit the linear transformation between joint and task velocities/accelerations given by the Jacobian. Then, the corresponding joint configuration is calculated by forward integration. The same occurs for redundant manipulators. In this case, the joint variables are obtained by means of a generalized inverse (usually the Moore-Penrose pseudoinverse) of the Jacobian, which is in this case a rectangular matrix with more columns than rows. A secondary task is usually handled by the projection in the null space of the Jacobian, in order not to interfere with the execution of the primary task [1], [2]. This formulation has been generalized for any number of tasks in [3], giving birth to the so-called task-priority framework. A great disadvantage of this method is that it does not take explicitly into account the physical limits of the robot. Thus, the obtained solution may violate such limits, resulting infeasible for the real capabilities of the robot. In the case of redundant manipulators, this issue has often been handled by converting the hard bounds into soft ones in a cost function that tends to keep the joints in the middle of their ranges [2], [4] or by using a weighted pseudo-inverse to limit velocities or accelerations [5]. However, these methods cannot ensure the satisfaction of the bounds, and therefore they do not prevent from having infeasible solutions. Note that, in the practical case, this would lead to the saturation of the actuators, with consequent deformation of the original task or even the stop of its execution.

Recent works cope with this by addressing the inverse kinematics as a constrained optimization problem [6]. In general, the task is converted into a Least Square Problem (LSP) and the solution is searched inside an admissible set. Secondary tasks are handled similarly, by successive LSP, wherein the admissible set is limited by the solutions of the tasks with higher priority. A general formulation following this paradigm is presented in [7], where the priority framework is extended to inequality tasks as well. A great advantage of the LSP formulation is that it can explicitly include constraints in a Quadratic Program (QP), which can be solved efficiently by numerical QP solvers (e.g., [8]–[10]). In this way, the solution (if it exists) will surely satisfy the hard bounds of the manipulator. However, the use of such approach requires particular attention on the way the constraints are posed. In particular, if joint configuration, velocity and acceleration limits are considered, viability of the states must be ensured. The viability of a set of states implies that, given a state within the viable set, all the future states will belong to that set [11]. This is fundamental to ensure that constraints will be satisfied in the future. This problem and issues related to its discrete-time implementation have been discussed in details in [12].

The level of satisfaction of the tasks (included the primary one) also strongly depends on how the task is defined with respect to the constraints. In other words, a too-demanding task would rapidly lead to the saturation of the joint commands, with consequent considerable deformation of the original task. Note that the primary task imposed to a manipulator typically consists in following a predefined path in the Cartesian space. Therefore, a significant deformation of such task is often unacceptable and should be avoided by the IK algorithm by any means. A widespread strategy to partially overcome this issue consists in the time scaling of the trajectory. This allows the time dilation of the task, in order to fulfill both the geometrical task and the joint bounds. Several task-scaling techniques have been proposed in the literature. First examples appeared in [13], [14], in order to cope with torque boundaries of the actuators in a dynamic-based control of a manipulator. Another approach was proposed in [15], [16], where the problem was reformulated in terms of nonlinear filtering of the joint trajectories, based on kinematic and/or dynamic constraints. As regards pure kinematic control, an important work is represented by [17], where a scaling factor is introduced in the computation of the IK so that the joint commands can be scaled within the boundaries. Nonetheless, if the robot is redundant, this solution might be non-optimal, as other velocities/accelerations (among the infinite possible solutions) could allow a better or even full satisfaction of the task. An improved method was proposed in [18], wherein an efficient exploration of the null space permits to find the solution ensuring the best scaling factor. This problem is equivalent to a QP where the task is expressed as an equality constraint, which forces the executed task to follow the same direction of the desired one, whereas the scaling factor is optimized in the cost function. The same algorithm was developed at acceleration level in [19], in order to recover continuity on the velocity and to include acceleration limits. However, the introduction of acceleration bounds does not always guarantee the existence of a feasible solution. In this case, the equality constraint has to be converted again into an LSP, allowing the modification of the primary task. Therefore, the obtained solution will firstly minimize the deformation with respect to the original geometrical path, and secondly will optimize the scaling factor.

The methods illustrated up to this point are all termed as local, since they only take into account information available at the current iteration. This means that the solution is optimal with respect to the current configuration of the manipulator, but not with respect to the global execution of the task. As a matter of fact, the iterative local resolution of the inverse kinematics might move the robot to a disadvantageous configuration with respect to the global task, as shown in [20]. Similarly, the local optimization of the task scaling might lead to worse results if compared to an optimization involving the future evolution of the task, the constraints and the kinematic variables. Anyhow, the use of global methods for the redundancy resolution or the scaling problem (i.e., the modification of the timing

M. Faroni and A. Visioli are with the Dipartimento di Ingegneria Meccanica e Industriale, Università degli Studi di Brescia, Brescia, 25123 Brescia, Italy (e-mail: m.faroni003@unibs.it; antonio.visioli@unibs.it).

M. Beschi and N. Pedrocchi are with Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato, Consiglio Nazionale delle Ricerche, 20133 Milan, Italy (e-mail: manuel.beschi@stiima.cnr.it; nicola.pedrocchi@stiima.cnr.it).

law) is strongly limited by the heavy computational load, which makes them unsuitable for online implementation.

To combine the strengths of local and global algorithms, this paper proposes a unified method to solve the inverse kinematics and the task scaling problem of a redundant manipulator by means of a model predictive approach. Model Predictive Control (MPC) is becoming more and more popular in robotic manipulation and mechatronics, thanks to the growing computational power of modern processors [21]. In this context, contributions usually address low-level control problems of actuators for trajectory tracking [22]–[24]. Few works also apply predictive techniques to high-level problems such as point-to-point trajectory generation [25] or motion planning for mobile manipulators [26].

In general, MPC is based on the optimization of a cost function over a finite predictive horizon, in order to get an optimal set of control actions for the considered time interval. Then, the first control action is applied to the system and the procedure is repeated iteratively, in the spirit of receding horizon control. One great advantage of MPC techniques is the efficient handling of constraints. Moreover, the predictive approach permits to take into account the evolution of the system in the successive time instants as well. For this reason, such approach seems to be a good candidate for overcoming the problems of local scaling and punctual resolution of the IK in a redundant manipulator. Following this principle, [27] exploited Pontryagin's Minimum Principle to set up a nonlinear optimization problem over a finite horizon. Nonetheless, kinematic bounds are not taken into account and this represents a considerable limitation of the algorithm. As a matter of fact, their inclusion would lead to a much heavier optimization problem, increasing significantly the computational time. In [28], [29], nonlinear MPC was applied to the operational-space inverse dynamics problem of a humanoid robot with bounded torques. The algorithm exploits differential dynamic programming to set up a nonlinear problem and the use of a fast nonlinear solver and powerful hardware permit to reach a computational time of 50 ms. In [20], a fast predictive approach to redundancy resolution of robot manipulators was proposed: the task was considered as a linearized constraint and an ad-hoc continuous-time formulation of the predictive equations was derived to lighten the computational burden of the resulting QP. Although the final QP is an approximation of the original nonlinear problem, it allows the computational time to be of the same order of magnitude of local methods (in the order of 1 ms), but still conferring the advantages of a predictive strategy. However, when the robot is required to perform demanding trajectories that often lead to joint saturations, this strategy might provoke a big deformation of the original task. In particular, if infeasible motions are required to the robot, this approach is not able to adapt the desired trajectory to the real capabilities of the robot.

For this reason, in this paper we propose a predictive scaling technique to preserve the geometrical task when the desired motion is not realizable by the robot. In this sense, the proposed method confers to the IK technique the advantages of both a predictive redundancy resolution and a predictive scaling. Then, a Closed Loop Inverse Kinematics (CLIK) scheme is implemented [30], in order to recover from position errors (given by numerical integration or out-of-path initial configurations). Moreover, a viability analysis is conducted and included in the online QP as a linear constraint. In this way, the robot states are always ensured to stay within a feasible region. This, along with considering the task in the cost function (and not as a hard constraint), always allows the feasibility of the QP.

Compared to [20], the proposed technique permits to obtain smaller deformation of the geometrical path, especially when the desired task is demanding with respect to the robot capabilities. Furthermore, compared to local scaling algorithms, it is possible to obtain smaller deformations of the desired tasks and higher mean values of the scaling factor as well.

II. Preliminaries

A. Inverse kinematics as an optimization problem

Consider an n-degree-of-freedom manipulator and an m-dimensional task x with $m \leq n$, defined with respect to a curvilinear abscissa σ, corresponding to the nominal trajectory time. In order to perform the specified task, the following equation must hold for all values of σ:

$$x(\sigma) = f(q(\sigma)) \tag{1}$$

where $x \in \mathbb{R}^m$ is the task position vector, $q \in \mathbb{R}^n$ is the joint configuration vector, and f is the forward kinematics of the robot. Differentiating with respect to σ gives:

$$x'(\sigma) = J(q)\, q'(\sigma) \tag{2}$$

where $x' = dx/d\sigma$, $q' = dq/d\sigma$, and $J = \partial f/\partial q$ is the $m \times n$ Jacobian matrix. Since σ corresponds to the nominal desired time, then x' represents the desired task velocity, and q' corresponds to the joint velocity. The IK problem consists in finding the joint velocities q' that satisfies the linear system (2). This can be reformulated as an LSP problem such that:

$$q' = \underset{q' \in \mathscr{S}}{\operatorname{argmin}} \frac{1}{2} \| J q' - x' \|^2 \tag{3}$$

where $\mathscr{S}$ is an admissible set of joint velocities, which can include the kinematic boundaries of the manipulator. Robot limits at the current iteration are typically linearized around the current state of the system, as described in [31]. In this way, $\mathscr{S}$ results in a set of linear constraints and (3) therefore results to be a QP.

When the robot is redundant with respect to the first task, a secondary task $E\, q' = g$ can be specified. The new optimization problem is:

$$q' = \underset{q' \in \mathscr{S}_2}{\operatorname{argmin}} \frac{1}{2} \| E q' - g \|^2; \quad \mathscr{S}_2 = \left\{ \underset{q' \in \mathscr{S}}{\operatorname{argmin}} \frac{1}{2} \| J q' - x' \|^2 \right\}. \tag{4}$$

This approach can be repeated iteratively for a general number of tasks [7]. The prime goal of an efficient IK algorithm is the maximal satisfaction of the primary task, and this depends on the solution of (3). In particular, if the kinematic limits specified in $\mathscr{S}$ lead to a constrained solution of the LSP, a spatial deformation of the task arises. To reduce this phenomenon, a modification of the timing law might be performed. As proposed in [18], at each time instant, (2) can be modified as:

$$s x'(\sigma) = J(q)\, \dot{q}(t) \tag{5}$$

where $s = d\sigma/dt$ can be seen as a scaling factor that deforms the velocity profile but without modifying the path. Since only downscaling is considered, the value of s has to be between 0 and 1. When no scaling occurs, s is equal to 1, whereas s smaller than 1 slows down the original trajectory. The desired task velocity x' is calculated over the parameter $\sigma = \int s\, dt$. Since in the discrete-time implementation of the algorithm, s is calculated as a scalar value at each time instant, the parameter σ is computed at each cycle as $\sigma(k) = \sigma(k-1) + T s(k-1)$, with $\sigma(0) = 0$, $s(0) = 1$ and T is the sampling period of the controller.

In this way, the primary objective of the IK problem becomes the satisfaction of the geometrical path of the desired task. Secondly, the scaling factor will be optimized, in order to maximize the fulfillment of the original primary task. Finally, the secondary task will be accomplished as well as possible. This hierarchy of QPs can be formalized and extended to a generic number of secondary tasks in

terms of a lexicographic multi-objective quadratic problem [10]. In lexicographic optimization, an ordered sequence of cost functions is optimized with priority determined by the order of the sequence. By defining an ordered sequence $\{w_1, w_2, w_3, \ldots, w_h\}$, $h \in \mathbb{N}$, where:

$$w_1 := \frac{1}{2}\|J\dot{q} - sx'\|^2, \tag{6}$$

$$w_2 := \frac{1}{2}(1-s)^2, \tag{7}$$

$$w_i := \frac{1}{2}\|E_i\dot{q} - g_i\|^2 \quad i = 3\ldots h, \tag{8}$$

the terms w_i, $i = 3,\ldots,h$ represent generic secondary tasks with decreasing priority. The scaling problem can be formalized as follows:

$$\begin{aligned} &\operatorname*{lex\,min}_{\dot{q},s} \{w_1, w_2, w_3, \ldots, w_h\} \\ &\text{subject to } \dot{q} \in \mathscr{S}, s \in [0,1]. \end{aligned} \tag{9}$$

This approach is equivalent to the one developed in [18]. In that case, the higher-priority task could be substituted by the equality constraint $J\dot{q} = sx'$, as acceleration limits were not considered.

In [19] the algorithm was formulated at acceleration level as well, in order to recover continuity on the joint velocities and to include acceleration bounds. However, in this way, feasibility is not ensured anymore, as explained in the following section.

B. Definition of the constraints and existence of the solution

he kinematic variables of the robot are typically required to respect the joint configuration and velocity limits, that is:

$$Q_{\min} \le q \le Q_{\max} \tag{10}$$

$$\dot{Q}_{\min} \le \dot{q} \le \dot{Q}_{\max} \tag{11}$$

where $Q_{\min}, Q_{\max} \in \mathbb{R}^n$ are the minimum and maximum joint configurations respectively, while $\dot{Q}_{\min}, \dot{Q}_{\max} \in \mathbb{R}^n$ are the minimum and maximum joint velocities respectively. Moreover, if acceleration limits are taken into account, it must be:

$$\ddot{Q}_{\min} \le \ddot{q} \le \ddot{Q}_{\max} \tag{12}$$

where $\ddot{Q}_{\min}, \ddot{Q}_{\max} \in \mathbb{R}^n$ are the minimum and maximum joint accelerations respectively.

However, to ensure that each joint can stop within its configuration bound without exceeding the maximum acceleration $\ddot{Q}_{\max,i}$, the following viability condition must be imposed (see [32], [33]):

$$\Psi(q_i, \dot{q}_i) \le 0 \tag{13}$$

where, for $i = 1\ldots n$ (i.e., for all joints),

$$\Psi(q_i,\dot{q}_i) := \begin{cases} q_i + \frac{\dot{q}_i^2}{2\ddot{Q}_{\max,i}} - Q_{\max,i} & \text{if } \dot{q}_i > 0 \\ q_i - Q_{\max,i} & \text{otherwise} \end{cases}$$

and likewise for the lower bound. Issues in the discrete-time implementation of such constraints are well detailed in [12].

Note that the use of a scaling factor as in [18] always guarantees the existence of a solution to the task $J\dot{q} = sx'$ (at worst $\dot{q} = 0$ and $s = 0$), since acceleration bounds were not considered. Therefore, assuming J is full rank, the task could be implemented as an equality constraint in the QP problem, ensuring the full satisfaction of the geometrical task. Nonetheless, there is no control on the acceleration values and this means the algorithm even admits discontinuities between successive solutions. When also acceleration limits are taken into account, the presence of a scaling factor does not ensure the existence of a solution that satisfies both the kinematics limits and the geometrical task. Therefore, a LSP solution would be needed again, with consequent possible deformation of the desired path.

Note that bounded accelerations play an important role in the suitability of planning algorithms. As a matter of facts, acceleration limits are often exploited in kinematic control to approximate torque limits (which typically represent the physical dynamic limits of the actuators). Generally, approximated (and conservative) constant limits for the acceleration are imposed, based on some estimation rules (e.g., the one proposed in [12]), as in [17], [19], [32].

III. Method

Purely local methods are based on the evaluation of the current configuration of the system. In this section, we reformulate the IK problem in a model-predictive-like framework, in order to take into account also the evolution of the variables along the predictive horizon. The basic idea of the method consists in formulating the resolution of the predictive IK problem as a constrained QP, which has to be solved at each time step. Then, in the spirit of receding horizon control, the first input of the sequence is applied to the system and the procedure is repeated until the end of the trajectory.

A. Formulation of the predictive equations

A linear model of the joint-space kinematic variables for an an n-degree-of-freedom manipulator can be derived by considering each joint as a r-th order integrator and, then, by coupling all the joints in state-space representation. Then, a linear predictive model can be easily derived by forward solving the resulting difference equation, as in classic linear MPC. Due to small sampling periods typical of robotic systems, sufficiently long horizons with uniform sampling periods would lead to a very large number of control variables involved in the predictive model [34]. In order to reduce the complexity of the predictive model, the complexity reduction strategy proposed in [20] has been applied. Such technique uses just a small number of prediction and control time instants along the horizons, instead of using all the sampling times. In particular, given an ordered set of prediction time instants $\{\tau_i\}$, $i = 1,\ldots,p$, and an ordered set of control time instants $\{\bar{t}_j\}$, $j = 1,\ldots,c$, the predictive horizon (in terms of seconds) corresponds to τ_p, and the control horizon (in terms of seconds) corresponds to $\bar{t}_c$. The order r of the integrators determines whether the control input is velocity ($r = 1$), acceleration ($r = 2$), and so on. Then, the predictive equations at the current time instant t_0 can be written as:

$$q^{(d)}_{(\tau)} = L_d z_0 + F_d u_{(\bar{t})} \qquad d = 0,\ldots,r-1 \tag{14}$$

where $q^{(d)}_{(\tau)} \in \mathbb{R}^{np\times 1}$ is the prediction of the d-th derivatives of the joint configuration at times $t_0 + \tau_i$, z_0 is the state vector at time t_0, being

$$z := [q_1,\ \dot{q}_1, \ldots q_1^{(r-1)}, \ldots, q_n,\ \dot{q}_n, \ldots q_n^{(r-1)}]^T \in \mathbb{R}^{n(r-1)\times 1}, \tag{15}$$

$u_{(\bar{t})} \in \mathbb{R}^{nc\times 1}$ is the input function (i.e., the r-th derivative of the joint configuration) at times $t_0 + \bar{t}_j$, and $L_d \in \mathbb{R}^{np\times nr}$ and $F_d \in \mathbb{R}^{np\times nc}$ are predictive matrices derived as in [20].

B. Choice of the predictive and control time instants

The predictive model developed in the previous section is based on the a-priori placement of the prediction and control time-instants τ_i and $\bar{t}_j$ along the prediction and the control horizons respectively. Although prediction and control time-instants can be chosen to be different, hereafter they have been considered to be equal without lose of generality. Since the reference signal (i.e., the Cartesian task) is, in general, non-repetitive, it is not possible to determine an optimal placement of τ_i and $\bar{t}_j$. For this reason we use a simple empiric tuning rule based on the following guidelines:

- the control and prediction time-instants have the same distribution (i.e., $\tau_i = \bar{t}_i$);
- the first time-instant is equal to the sampling period of the system (i.e., $\tau_1 = \bar{t}_1 = T$);
- the initial time-instants should be closer, while the successive ones can be sparser.

Thus, given a certain number of time-instants $c = p$ and the length of the horizon τ_p in terms seconds, we set up a parabolic distribution of τ_i and $\bar{t}_i$ such that:

$$\tau_i = \bar{t}_i = \frac{\tau_p - T}{(p-1)^2} i^2 - 2\frac{\tau_p - T}{(p-1)^2} i + \left(\frac{\tau_p - T}{(p-1)^2} + T \right) \tag{16}$$

for $i = 1, \ldots, p$. In this way, the same predictive and control horizons can be reached with a reduced number of variables involved in the predictive model.

C. Satisfaction of the tasks

The satisfaction of the primary task along the predictive horizon can be expressed by imposing (5) at each time-instant of the horizon. This can be approximated by only considering the predictive time-instants τ_i and expressed as the minimization of the following cost function:

$$\frac{1}{2} \| J^*(q_{(\tau)})\, \dot{q}_{(\tau)} - X'_{(\tau)} s_{(\tau)} \|^2 \tag{17}$$

where:

$$\begin{aligned}
J^*(q_{(\tau)}) &= \text{blkdiag}\Big(J\big(q(t_0+\tau_1)\big), \ldots, J\big(q(t_0+\tau_p)\big) \Big); \\
\dot{q}_{(\tau)} &= \big[\dot{q}(t_0+\tau_1)^T, \ldots, \dot{q}(t_0+\tau_p)^T \big]^T; \\
s_{(\tau)} &= \big[s(t_0+\tau_1), \ldots, s(t_0+\tau_p) \big]^T; \\
X'_{(\tau)} &= \text{blkdiag}\big(x'(\sigma(t_0)+\tau_1), \ldots, x'(\sigma(t_0)+\tau_p) \big);
\end{aligned}$$

The cost function (17) is nonlinear in the control action $u_{(\bar{t})}$, since the future joint configuration $q_{(\tau)}$ needed in the calculation of the Jacobians depends on $u_{(\bar{t})}$. However, at each step, the prediction of the joint configuration can be calculated by means of (14) (with $d = 0$). Then, in the successive step, this prediction can be used to compute an estimation of the future Jacobians [35]. Thus, by considering such approximated Jacobians, (17) is now quadratic in $\dot{q}_{(\tau)}$. By substituting (14) with $d = 1$, (17) is quadratic also in the control action $u_{(\bar{t})}$. A quadratic cost function w_1 can therefore be defined as:

$$w_1 := \frac{1}{2} \| J^*(\hat{q}) L_1 z_0 + J^*(\hat{q}) F_1 u_{(\bar{t})} - X'_{(\tau)} s_{(\tau)} \|^2 \tag{18}$$

where $\hat{q}$ is the prediction of the joint configuration calculated at the previous time step by means of (14) (with $d = 0$).

To recover from position errors, a CLIK strategy can be implemented as primary controller of a cascade architecture where the MPC is the secondary controller. In particular, the desired reference velocity $X'_{(\tau)}$ can be modified by adding a corrective term, that is,

$$\begin{aligned}
X'_{(\tau)} = \text{blkdiag}\big(& x'(\sigma(t_0)+\tau_1) + K e_c, x'(\sigma(t_0)+\tau_2) + K e_c, \ldots \\
& \ldots, x'(\sigma(t_0)+\tau_p) + K e_c \big);
\end{aligned} \tag{19}$$

where $K \in \mathbb{R}^{m \times m}$ is a positive-definite diagonal matrix, and $e_c \in \mathbb{R}^m$ represents the current Cartesian position error, between the desired and the current positions. When the task orientation is expressed in terms of Euler angles and the analytical Jacobian is used in (18), the Cartesian error can be simply defined as $e_c := x_d - x$, where x_d is the desired Cartesian position at the current time-instant, and x is the current Cartesian position. However, when angular velocities are used in the representation of the desired task velocity x', the Cartesian orientation error should be defined as the mutual orientation, in terms of axis-angle representation, between the desired and the current end-effector frames [36]. Further details about different orientation representations in the CLIK strategy can be found in [37].

Note that the CLIK controller bandwidth is set to be much smaller than the MPC controller one. This implies that the secondary loop can be reasonably approximated as a unitary gain. Thus, the stability of the primary loop can be guaranteed by imposing that K belongs to a region that depends on the sampling period [38].

The primary task (18) can now be included in a priority framework with other secondary tasks, by following a hierarchical approach [7]. The resulting hierarchical QP should (in order of priority): i) optimize the primary task (18); ii) limit the scaling; iii) optimize the lower-priority tasks. Considering a generic secondary task in the form $E\, u_{(\bar{t})} = g$, this can be formalized as a lexicographic multi-objective quadratic problem, similarly to (9):

$$\begin{aligned}
& \underset{u_{(\bar{t})}, s_{(\tau)}}{\text{lex min}} \, \{ w_1, w_2, w_3 \} \\
& \text{subject to } u_{(\bar{t})} \in \mathscr{S}, s_i \in [0,1]
\end{aligned} \tag{20}$$

where $w_2 := \frac{1}{2} \| 1 - s_{(\tau)} \|^2$, $w_3 := \frac{1}{2} \| E\, u_{(\bar{t})} - g \|^2$, and w_1 is defined in (18). The number of optimization variables involved in (20) is given by $nc + p$. In order to efficiently handle prioritized problems in the form of (20), fast dedicated QP solvers for hierarchical task have been recently developed, for instance, in [10], while in [39] a technique to solve hierarchical tasks in a single enlarged QP was proposed. Alternatively, a weighted approach can be adopted, in order to solve a single multi-objective QP with $nc + p$ variables, in which the tasks have different weights, according to their priority. This can be written as:

$$\begin{aligned}
\begin{bmatrix} u_{(\bar{t})} \\ s_{(\tau)} \end{bmatrix} = \underset{u \in \mathscr{S}, s_i \in [0,1]}{\text{argmin}} \; & \lambda_1 \| J^*(\hat{q})\, L_1 z_0 + J^*(\hat{q})\, F_1 u_{(\bar{t})} - X'_{(\tau)} s_{(\tau)} \|^2 \\
& + \lambda_2 \| 1 - s_{(\tau)} \|^2 + \lambda_3 \| E\, u_{(\bar{t})} - g \|^2
\end{aligned} \tag{21}$$

with $\lambda_1 \gg \lambda_2 \gg \lambda_3$.

Examples of secondary tasks can be, for instance,

- joint range availability: $E = F_0$; $g = (Q_{\max} + Q_{\min})/2 - L_0 z_0$;
- minimum norm of the control action: $E = I_{nc}$; $g = 0$.

D. Analysis and design of the constraints

This section aims at translating the kinematic limits of the joints into linear constraints, in order to include them in the QP formulation. Note that, by means of the predictive equations (14), the bounds will be imposed along the whole control horizon. In this way, the optimal control action calculated at each step will take into account the presence of the constraints for the future time instants as well.

As introduced in Section II-B, the IK resolution should consider the boundaries of the kinematic variables, such as configuration, velocity and acceleration. Thus, by choosing the order r of the resolution algorithm, and by using (14), the following set of linear inequalities results:

$$\begin{aligned}
Q^{(d)}_{\min} - L_d z_0 \leq \; & F_d u_{(\bar{t})} \leq Q^{(d)}_{\max} - L_d z_0 \qquad \forall\, d = 0, \ldots, r-1 \\
Q^{(r)}_{\min} \leq \; & u_{(\bar{t})} \leq Q^{(r)}_{\max}
\end{aligned} \tag{22}$$

The linear inequalities (22) permit to take into account the joint boundaries up to the rth derivative of the configuration. However, the IK should consider the viability set condition (13) as well, as mentioned in Section II-B. Unfortunately, (13) is nonlinear in the joint configuration and velocity and cannot be linearized as in the local resolution. In order to approximate it with a linear constraint a further analysis is required. First of all, it is important to point out that shrinking the state space $(q, \dot{q})$ to a more conservative admissible set does not guarantee the satisfaction of the constraint in the future.

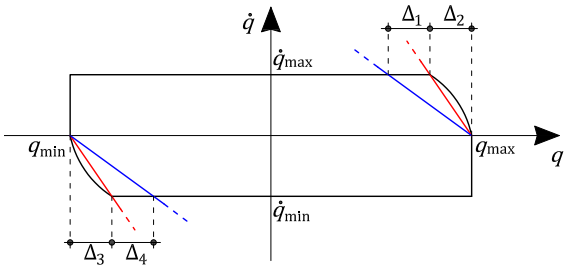


Fig. 1: Admissible velocity area. Black: area given by the constraints (10)–(13); Red: infeasible linearization of the viability condition; Blue: feasible linearization (23), (24) of the viability condition. Note that Proposition 1 yields to the graphical result $\Delta_1 = \Delta_2$ and $\Delta_3 = \Delta_4$ (if the boundaries are symmetric all the quantities are equal).

For instance, for every joint, consider the linear approximation of the constraint (13) given by the line passing through the points $(Q_{\max,i}, 0)$ and $(Q_{\max,i} + \frac{\dot{q}_i^2}{2\dot{Q}_{\min,i}}, \dot{Q}_{\max,i})$ (red line in Figure 1). When the constraint is activated, the state is forced to follow the trajectory given by the curve $\dot{q}_i = 2\frac{\ddot{Q}_{\min,i}}{\dot{Q}_{\max,i}} q_i - 2\frac{\ddot{Q}_{\min,i}}{\dot{Q}_{\max,i}} Q_{\max,i}$. The corresponding acceleration is given by $\ddot{q}_i = d\dot{q}_i/dt = 2\frac{\ddot{Q}_{\min,i}}{\dot{Q}_{\max,i}} \dot{q}_i$, with $\dot{q}_i \in [0, \dot{Q}_{\max,i}]$. Thus,

$$\max |\ddot{q}_i| = \max_{\dot{q}_i \in [0, \dot{Q}_{\max,i}]} \left| 2\frac{\ddot{Q}_{\min,i}}{\dot{Q}_{\max,i}} \dot{q}_i \right| = 2|\ddot{Q}_{\min,i}|.$$

This means that the acceleration needed to move from the current value of $\dot{q}_i$ to zero without violating the linear constraint is twice the maximum acceleration from which the constraint was constructed. Following a similar reasoning, it is possible to obtain the following proposition.

Proposition 1: Consider the upper viability condition (13) and its linear approximation in the form $\dot{q}_i \le \alpha q_i + \beta$. Imposing the passage of the curve through the point $(Q_{\max_i}, 0)$ of the $(q_i, \dot{q}_i)$ plane, then the acceleration does not exceed the value $\ddot{Q}_{\max,i}$ if

$$\alpha = -\frac{|\rho_{\min,i}|}{\dot{Q}_{\max,i}} \; ; \quad \beta = \frac{|\rho_{\min,i}|}{\dot{Q}_{\max,i}} Q_{\max,i} \tag{23}$$

where $|\rho_{\min,i}| \le |\ddot{Q}_{\min,i}|$. Likewise, for the lower viability condition and its linear approximation $\dot{q}_i \ge \gamma q_i + \delta$ it results

$$\gamma = \frac{|\rho_{\max,i}|}{\dot{Q}_{\min,i}} \; ; \quad \delta = -\frac{|\rho_{\max,i}|}{\dot{Q}_{\min,i}} Q_{\min,i} \tag{24}$$

where $|\rho_{\max,i}| \le |\ddot{Q}_{\max,i}|$.

Proof: Let the curve $\dot{q}_i \le \alpha q_i + \beta$ be the approximation of the viability condition (13). Imposing the passage through the point $(Q_{\max,i}, 0)$ yields $\alpha\, Q_{\max,i} + \beta = 0 \Rightarrow \alpha = -\beta/Q_{\max,i}$. Therefore, $\dot{q}_i = (-\beta/\dot{Q}_{\max,i})\, q_i + \beta$. The acceleration is given by $\ddot{q}_i = d\dot{q}_i/dt = (-\beta/\dot{Q}_{\max,i})\, \dot{q}_i$, with $\dot{q}_i \in [0, \dot{Q}_{\max,i}]$. Thus,

$$\max |\ddot{q}_i| = \max_{\dot{q}_i \in [0, \dot{Q}_{\max,i}]} \left| \frac{-\beta}{Q_{\max,i}} \dot{q}_i \right| = \frac{\dot{Q}_{\max,i}}{Q_{\max,i}} |-\beta|.$$

Assuming β is negative and imposing $\max|\ddot{q}_i| \le |\ddot{Q}_{\min,i}|$ gives $\beta \le (\ddot{Q}_{\min,i} Q_{\max,i})/\dot{Q}_{\min,i} \Rightarrow \alpha \ge (\ddot{Q}_{\min,i})/\dot{Q}_{\min,i}$ or equivalently $\alpha = -|\rho_{\min,i}|/\dot{Q}_{\max,i}$ and $\beta = (|\rho_{\min,i}|/\dot{Q}_{\max,i})\, Q_{\max,i}$, where $|\rho_{\min,i}| \le |\ddot{Q}_{\min,i}|$. ■

Once the constraint related to the viability set has been wisely linearized, its inclusion in the predictive framework is straightforward. As a matter of fact, it has to be:

$$\begin{cases} \dot{q}_i(t) \le \alpha_i q_i(t) + \beta_i \\ \dot{q}_i(t) \ge \gamma_i q_i(t) + \delta_i \end{cases} \quad \forall\, t \in \bar{t}_c \quad \forall\, i = 1, \ldots, n. \tag{25}$$

Reminding the predictive equation (14), the constraints become:

$$\begin{cases} (F_2 - \bar{\alpha} F_1)\, u_{(\bar{t})} \le (\bar{\alpha} L_1 - L_2)\, z_0 + \bar{\beta} \\ (F_2 - \bar{\gamma} F_1)\, u_{(\bar{t})} \ge (\bar{\gamma} L_1 - L_2)\, z_0 + \bar{\delta} \end{cases} \tag{26}$$

where $\bar{\alpha} := \mathrm{diag}\big([\alpha_1, \ldots, \alpha_n], \ldots, [\alpha_1, \ldots, \alpha_n]\big) \in \mathbb{R}^{nc \times nc}$, $\bar{\beta} := \big[[\beta_1, \ldots, \beta_n], \ldots, [\beta_1, \ldots, \beta_n]\big]^T \in \mathbb{R}^{nc}$ and likewise for $\bar{\gamma}$ and $\bar{\delta}$.

IV. Numerical Results

This section aims at testing the proposed method via simulation on a 9-dof redundant manipulator. The robot is composed of a 7-dof KUKA iiwa14 manipulator, mounted on an actuated base, which consists in a 2-dof planar arm hanging on the ceiling. The manipulator is equipped with a 0.2m-long tool, aligned with the rotation axis of the last joint. The configuration and velocity limits are given by:

$$\begin{aligned} Q_{\max} &= -Q_{\min} \\ &= [\, 3.00,\ 3.00,\ 2.96,\ 2.09,\ 2.96,\ 2.09,\ 2.96,\ 2.09,\ 3.05\,]^T \ \text{[rad]} \\ \dot{Q}_{\max} &= -\dot{Q}_{\min} \\ &= [\, 1.48,\ 1.48,\ 1.48,\ 1.48,\ 1.74,\ 1.30,\ 2.26,\ 2.35,\ 2.35\,]^T \ \text{[rad/s]} \end{aligned}$$

The acceleration bounds are chosen equal to ± 12 rad/s^2 for all joints. The method is implemented with a sampling period $T = 1$ ms.

Regarding the task, consider the two following reference paths:

- Path $X1$: a circle in the plane $x = 0.35$ m with center $P_1(0.35, -0.3, 1.3)$ m and radius $R = 0.25$ m. The circle has to be performed twice to complete the task;
- Path $X2$: a sinusoidal curve $z = 1.2 + 0.2\sin(4\pi - 8\pi x)$ in the plane $y = 0.1$ m, from the initial point $P_2(0.5, 1, 1.2)$ m to the final point $P_3(0, 0.1, 1.2)$ m.

For each of these reference paths, ten random paths are generated by perturbing the path parameters by adding a random number between ± 0.05 m (with uniform distribution).

The position of the end-effector should perform each path by following the nominal longitudinal timing law

$$\eta(\sigma) = \frac{6}{T_{\text{tot}}^5}\sigma^5 - \frac{15}{T_{\text{tot}}^4}\sigma^4 + \frac{10}{T_{\text{tot}}^3}\sigma^3 \tag{27}$$

where T_{tot} is the desired total duration of the nominal task. The following values of T_{tot} are considered: $T_{\text{tot}} = 0.5, 1, 2, 4$ s for Path $X1$, and $T_{\text{tot}} = 0.8, 1.2, 1.6, 2$ s for Path $X2$. The longitudinal velocity profile of the end-effector can be obtained by differentiating (27).

As secondary task, the robot elbow has to be kept as high as possible. This is expressed by defining a required velocity to the z-coordinate of the origin of the elbow frame in the form $\dot{x}_2 = |d\eta/d\sigma|$. From a practical perspective, such behavior could be useful to reduce as much as possible the occupation of the space below the robot.

The following algorithms are compared:

- the local method (9) at acceleration level [19];
- the proposed predictive method, with number of control and predictive time instants $c = p = 3$ and predictive horizon $\tau_p = 100$ ms (namely, $\{\tau_i\} = \{1, 26, 100\}$ ms);
- the proposed predictive method, with $c = p = 5$ and predictive horizon $\tau_p = 100$ ms (namely, $\{\tau_i\} = \{1, 7, 26, 57, 100\}$ ms).

Referring to Section III-C, simulations related to Path $X1$ were performed by implementing the hierarchical approach, whereas simulations related to Path $X2$ were performed by implementing the weighted approach, with $\lambda_1 = 10^6$, $\lambda_2 = 10^3$, $\lambda_3 = 10$. The proportional gain related to the CLIK strategy is $K = 200I$, where I is the identity matrix. Simulations were performed in Matlab.

The evaluated parameters are the deformation with respect to the desired path in terms of Euclidean distance with respect to the closest

TABLE I: Numerical comparison between the local method and the proposed predictive method ($p = 3$ and $p = 5$).

T_{tot} [s]	$e_{\max}$ [m]			e_{mean} [m]			s_{mean}			$z_{\text{elb,mean}}$ [m]		
	Local	$p=3$	$p=5$	Local	$p=3$	$p=5$	Local	$p=3$	$p=5$	Local	$p=3$	$p=5$
Path $X1$												
0.5	$1.17\cdot10^{-1}$	$1.01\cdot10^{-2}$	$4.48\cdot10^{-3}$	$1.33\cdot10^{-2}$	$2.66\cdot10^{-3}$	$2.22\cdot10^{-3}$	0.286	0.101	0.082	1.28	1.59	1.65
1	$3.40\cdot10^{-2}$	$2.14\cdot10^{-3}$	$1.47\cdot10^{-3}$	$2.17\cdot10^{-3}$	$6.93\cdot10^{-4}$	$5.85\cdot10^{-4}$	0.617	0.386	0.343	1.27	1.56	1.57
2	$1.67\cdot10^{-2}$	$2.59\cdot10^{-4}$	$1.10\cdot10^{-4}$	$7.22\cdot10^{-4}$	$2.24\cdot10^{-5}$	$1.97\cdot10^{-5}$	0.920	0.964	0.962	1.33	1.39	1.40
4	$2.56\cdot10^{-5}$	$2.40\cdot10^{-5}$	$2.36\cdot10^{-5}$	$5.59\cdot10^{-6}$	$6.53\cdot10^{-6}$	$6.34\cdot10^{-6}$	0.998	1.000	1.000	1.32	1.51	1.50
Path $X2$												
0.8	$1.52\cdot10^{-1}$	$4.53\cdot10^{-3}$	$1.50\cdot10^{-3}$	$2.51\cdot10^{-2}$	$2.15\cdot10^{-4}$	$1.92\cdot10^{-4}$	0.479	0.556	0.455	1.30	1.39	1.38
1.2	$1.10\cdot10^{-1}$	$1.51\cdot10^{-3}$	$6.12\cdot10^{-4}$	$1.29\cdot10^{-2}$	$5.64\cdot10^{-5}$	$6.02\cdot10^{-5}$	0.578	0.773	0.735	1.30	1.35	1.40
1.6	$3.43\cdot10^{-2}$	$8.88\cdot10^{-4}$	$4.05\cdot10^{-4}$	$1.41\cdot10^{-3}$	$2.20\cdot10^{-5}$	$2.06\cdot10^{-5}$	0.869	0.928	0.909	1.31	1.41	1.37
2.0	$6.38\cdot10^{-4}$	$2.76\cdot10^{-4}$	$4.43\cdot10^{-5}$	$1.29\cdot10^{-5}$	$1.15\cdot10^{-5}$	$1.19\cdot10^{-5}$	0.992	0.982	0.985	1.30	1.40	1.46

point on the desired path, the scaling factor along the task, and the satisfaction of the secondary task. The results are shown in Table I. For each method and for each value of T_{tot}, the table shows the maximum error $e_{\max}$, the mean error e_{mean}, the mean scaling factor s_{mean}, and the average height of the robot elbow $z_{\text{elb,mean}}$, given by the mean of the ten randomly-generated tasks. First of all, note that $e_{\max}$ and e_{mean} decrease for greater values of T_{tot} (i.e., less demanding tasks). As regards the scaling values, as expected, more demanding tasks lead to smaller values of s_{mean}, since a greater scaling of the timing law is needed. Consider now the comparison of the various methods. Firstly, $e_{\max}$ and e_{mean} given by the predictive methods always result to be significantly better than ones given by the corresponding local method. Secondly, the values of s_{mean} given by the predictive methods are usually better than the ones given by the corresponding local method. An exception is represented by the fastest tasks. In such cases the predictive methods tend to perform a heavier scaling of the timing law to preserve the original geometrical path, which is the priority of the method.

Simulations using Gazebo [40] have then been performed to assess the effectiveness of the method in a realistic scenario. Gazebo receives the motor torque signals by a simple cascade P-PI control architecture with inertia matrix decoupling. The inner- and outer-loop bandwidths are respectively equal to 11 Hz and 4 Hz. In particular, few Cartesian tasks performed by the robot are shown in Figure 2. Table II shows the values of $e_{\max}$ and e_{mean} (considering also the controlled system tracking error), and s_{mean} for such tests. In Figure 2a, it is possible to note that, for Path $X1$ and $T_{\text{tot}} = 2$ s, the local method (dashed-dotted blue line) gives a significant deformation of the desired path, while the predictive method (dashed red line) is able to avoid such deformation. This behavior is even more evident when the desired task becomes more demanding: as shown in Figure 2b for $T_{\text{tot}} = 0.5$ s, the local method (dashed-dotted blue line) leads to a distorted path, especially in the final deceleration phase (zoomed window), while the predictive strategy (dashed red line) still gives satisfactory results. The same applies for Path $X2$, as shown in Figure 2c and 2d.

As an illustrative example, the trends of the kinematic variables for Path $X1$, $T_{\text{tot}} = 2$ s, and $p = 3$ are shown in Figure 3. First of all, the kinematic variables of the joint are always within their bounds. Moreover, as shown in the fourth column of the figure, the torque applied to the joints are kept within their limits, thanks to the appropriate choice of the acceleration limits.

Finally, the proposed method has also been tested at jerk level ($r = 3$). In many applications, for example to cope with unmodeled elasticities, it is advisable to reduce also higher-order derivatives to increase smoothness. By using $r = 3$, such soft constraints are taken into account as a part of the cost function. Joint accelerations are shown in Figure 4, which shows smoother trends of such variables, since the algorithm also considers jerk minimization.

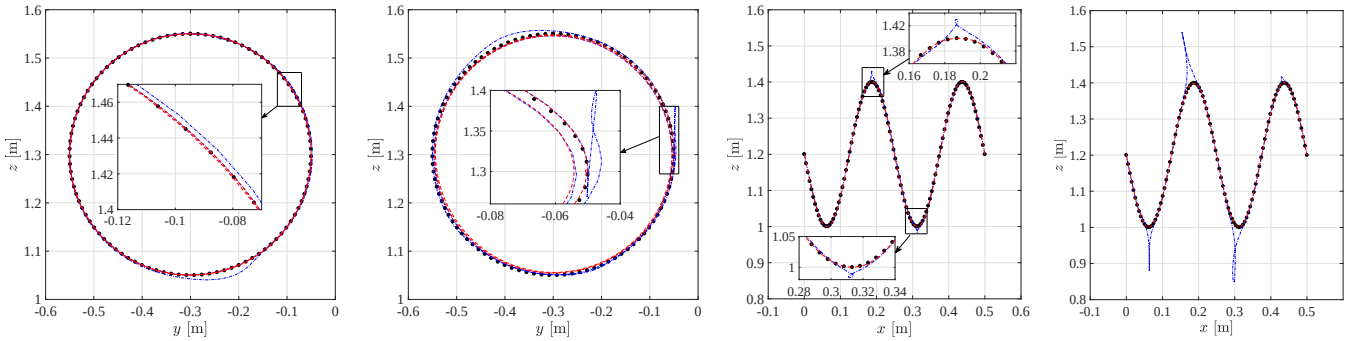
TABLE II: Numerical comparison between the local method and the proposed predictive method in Figure 2.

Figure	$e_{\max}$ [m]		e_{mean} [m]		s_{mean}	
	Local	$p=3$	Local	$p=3$	Local	$p=3$
2a	$1.66\cdot10^{-2}$	$8.69\cdot10^{-4}$	$1.11\cdot10^{-3}$	$2.78\cdot10^{-4}$	0.909	0.918
2b	$1.01\cdot10^{-1}$	$7.33\cdot10^{-3}$	$1.30\cdot10^{-2}$	$4.21\cdot10^{-3}$	0.294	0.102
2c	$3.09\cdot10^{-2}$	$1.32\cdot10^{-3}$	$2.79\cdot10^{-3}$	$3.68\cdot10^{-4}$	0.855	0.854
2d	$1.53\cdot10^{-1}$	$5.31\cdot10^{-3}$	$4.39\cdot10^{-2}$	$5.84\cdot10^{-4}$	0.471	0.594

TABLE III: Mean and maximum computational time taken by the local and the predictive ($p = 3$ and $p = 5$) methods for Path $X1$.

Method	$t_{\max}$ [s]	t_{mean} [s]
Local	$8.3\cdot10^{-4}$	$1.2\cdot10^{-4}$
Predictive ($p=3$)	$1.1\cdot10^{-3}$	$2.1\cdot10^{-4}$
Predictive ($p=5$)	$2.3\cdot10^{-3}$	$3.8\cdot10^{-4}$

An important issue to take into account when using an optimization-based algorithm with fast sampling periods is the required computational time. For this reason, the mean and the maximum cycle time required by the algorithm are evaluated. Referring to the above-mentioned simulations, Table III shows the mean and the maximum computational time taken to perform Path $X1$. The simulations were performed by using the parametric active-set QP solver *QpOASES* [8], [9], on a standard laptop mounting a 2.5 GHz Intel Core i5-2520M processor. As expected, the computational time required by the predictive method is generally higher than the one required by the local method, due to the greater number of variables involved in the online optimization. Moreover, the computational time obviously grows with the number of control moves $c = p$. However, it is important to remind that the extension of the IK problem to the predictive framework originally led to a constrained nonlinear problem. The proposed method converts such problem into a QP problem, and the required computational time is still comparable with the one taken by the purely local method, and, thus, suitable for online implementation. Indeed, considering the example with $p = 3$, it is interesting to note that the maximum computational time is in the order of 1 ms. Furthermore, its mean value is around 0.21 ms, that is, less than the double of the local method, although the number of optimization variables is three times bigger. Finally, it is worth stressing that, in order to handle the real time execution of the control algorithm for which a fixed sampling period has to be selected, it is in any case possible to set a soft maximum CPU time, within which the optimization shall stop. From a practical perspective, the use of such setting does not give rise to a significant performance decay, if such maximum time is chosen properly, according to the problem to solve.



(a) Path $X1$, $T_{\text{tot}} = 2$ s (b) Path $X1$, $T_{\text{tot}} = 0.5$ s (c) Path $X2$, $T_{\text{tot}} = 1.6$ s (d) Path $X2$, $T_{\text{tot}} = 0.8$ s

Fig. 2: Dotted black: ideal reference path. Dashed-dotted blue: path obtained by the local method. Dashed red: path obtained by the predictive method with $r = 2$, $p = 3$.

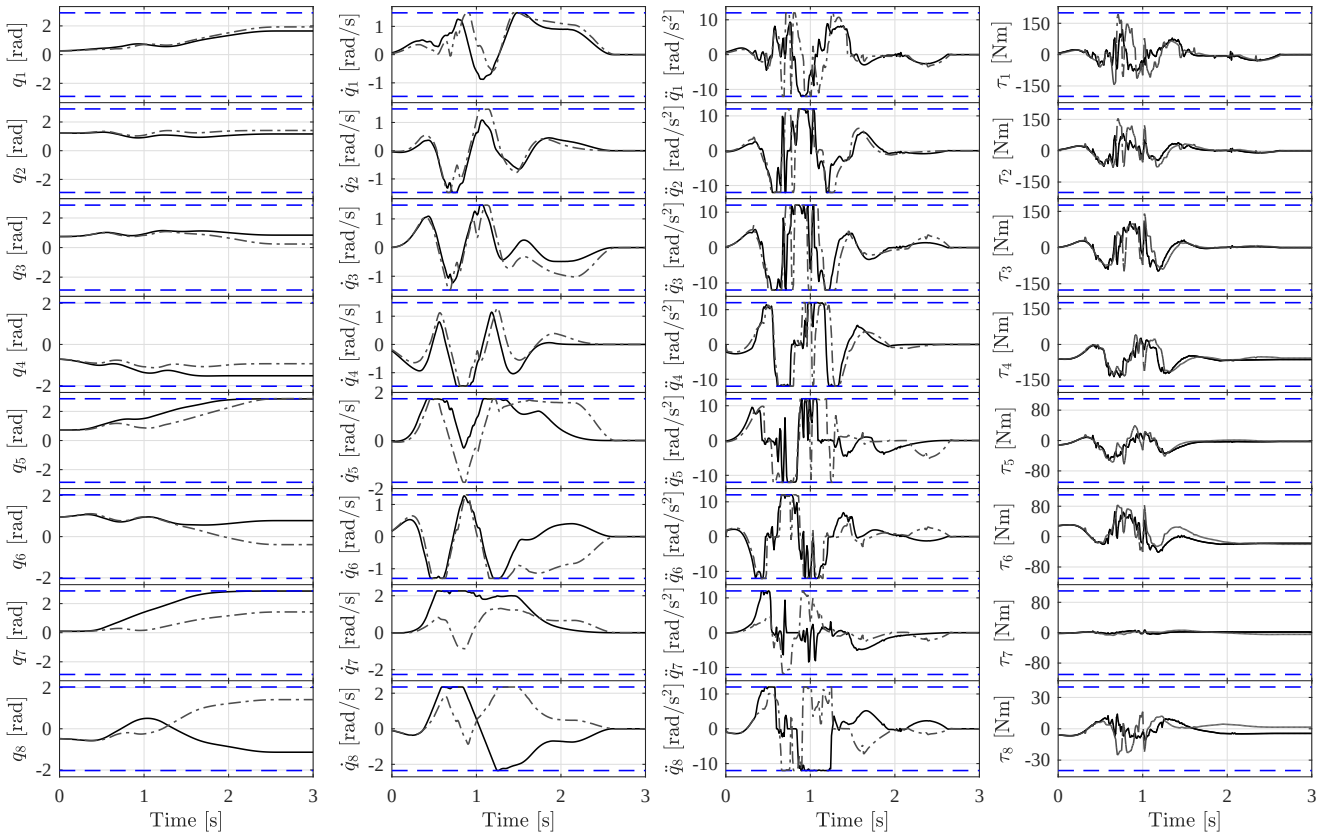


Fig. 3: Solid black: joint configuration, velocity, acceleration and torque obtained by the predictive method with $r = 2$, $p = 3$, Path $X1$, $T_{\text{tot}} = 2$ s. Dashed-dotted gray: joint configuration, velocity, acceleration and torque obtained by the local method, Path $X1$, $T_{\text{tot}} = 2$ s. Dashed blue: joint configuration, velocity, acceleration and torque limits. The ninth joint is omitted for it does not affect the motion.

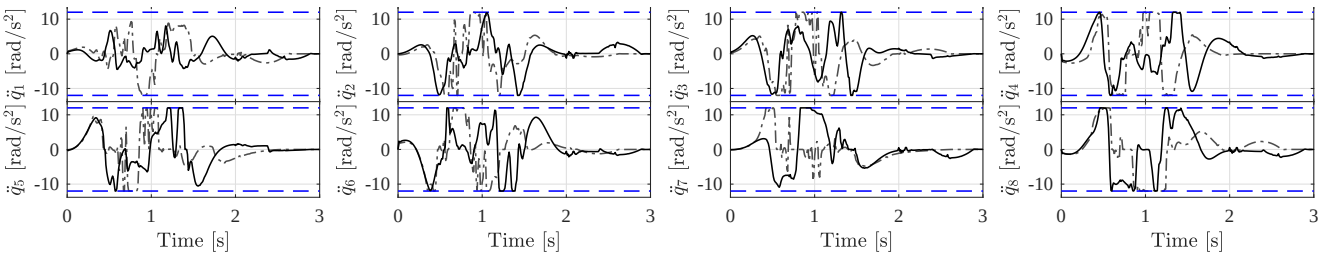


Fig. 4: Solid black: joint acceleration obtained by the predictive method with $r = 3$, $p = 3$, Path $X1$, $T_{\text{tot}} = 2$ s. Dashed-dotted gray: joint acceleration obtained by the local method, Path $X1$, $T_{\text{tot}} = 2$ s. Dashed blue: joint acceleration bounds. The ninth joint is omitted for it does not affect the motion.

V. Conclusions

A new predictive task-scaling technique has been proposed in this paper. It allows solving the IK of a redundant manipulator taking into account the kinematic limits of the robot. The task-scaling strategy preserves the geometrical task when the desired motion is infeasible for the robot capabilities, dramatically reducing the geometrical error in case of very demanding tasks. Moreover, compared to local techniques, it is possible to obtain much better results in terms of task deformation and trajectory scaling. The use of a single-step linearization of the predicted task around the previous prediction of the joint configuration permits to convert the online optimization into a QP. In this way, the computational complexity is dramatically decreased, making the method suitable for online implementation.

Acknowledgment

The research leading to these results has been partially funded by the European Union H2020-CS2-738039-Eureca: Enhanced Human Robot cooperation in Cabin Assembly tasks.

References

[1] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. MA, USA: Addison-Wesley, 1990.

[2] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. on Systems Man and Cybernetics*, vol. 7, pp. 868–871, 1977.

[3] B. Siciliano and J.-J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. Int. Conf. on Advanced Robotics*, Pisa (Italy), 1991, pp. 1211–1216.

[4] A. Atawnih, D. Papageorgiou, and Z. Doulgeri, "Kinematic control of redundant robots with guaranteed joint limit avoidance," *Robotics and Autonomous Systems*, vol. 79, pp. 122–131, 2016.

[5] 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, pp. 286–292, 1995.

[6] A. Rocchi, E. M. Hoffman, D. G. Caldwell, and N. G. Tsagarakis, "Opensot: a whole-body control library for the compliant humanoid robot Coman," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Seattle (USA), 2015, pp. 6248–6253.

[7] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks," *IEEE Trans. on Robotics*, vol. 27, pp. 785–792, 2011.

[8] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, pp. 327–363, 2014.

[9] H. J. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *Int. J. of Robust and Nonlinear Control*, vol. 18, pp. 816–830, 2008.

[10] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The Int. J. of Robotics Research*, vol. 33, pp. 1006–1028, 2014.

[11] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, pp. 1747–1767, 1999.

[12] A. Del Prete, "Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators," *IEEE Robotics and Automation Letters*, vol. 3, pp. 281–288, 2018.

[13] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," in *Proc. IEEE American Control Conf.*, San Francisco (USA), 1983, pp. 752–756.

[14] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Trans. on Robotics and Automation*, vol. 6, pp. 554–561, 1990.

[15] O. Gerelli and C. Guarino Lo Bianco, "Nonlinear variable structure filter for the online trajectory scaling," *IEEE Trans. on Industrial Electronics*, vol. 56, pp. 3921–3930, 2009.

[16] C. Guarino Lo Bianco and O. Gerelli, "Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints," *IEEE Trans. on Robotics*, vol. 27, pp. 1144–1152, 2011.

[17] 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. on Robotics and Automation*, vol. 19, pp. 162–167, 2003.

[18] 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, pp. 637–654, 2015.

[19] ——, "Motion control of redundant robots under joints constraints: Saturation in the null space," in *Proc. IEEE Int. Conf. on Robotics and Automation*, St. Paul (USA), 2012, pp. 285–292.

[20] M. Faroni, M. Beschi, A. Visioli, and L. Molinari Tosatti, "A predictive approach to redundancy resolution for robot manipulators," in *Proc. IFAC World Congress*, Toulouse (France), 2017.

[21] J. Wilson, M. Charest, and R. Dubay, "Nonlinear model predictive control schemes with application on a 2 link vertical robot manipulator," *J. of Robotics and C.I.M.*, vol. 41, pp. 23–30, 2016.

[22] R. Hedjar and P. Boucher, "Nonlinear Receding-Horizon control of rigid link robot manipulators," *Int. J. of Advanced Robotic Systems*, vol. 2, pp. 15–24, 2005.

[23] P. Poignet and M. Gautier, "Nonlinear model predictive control of a robot manipulator," in *Proc. Int. Workshop on Advanced Motion Control*, Nagoya (Japan), 2000, pp. 401–406.

[24] A. Ferrara, G. P. Incremona, and L. Magni, "A robust MPC/ISM hierarchical multi-loop control scheme for robot manipulators," in *Proc. Annual Conf. on Decision and Control*, Firenze (Italy), 2013, pp. 3560–3565.

[25] M. M. Ghazaei Ardakani, N. Olofsson, A. Robertsson, and R. Johansson, "Real-time trajectory generation using model predictive control," in *Proc. IEEE Int. Conf. on Automation Science and Engineering*, Gothenburg (Sweden), 2015, pp. 942–948.

[26] G. Buizza Avanzini, A. M. Zanchettin, and P. Rocco, "Reactive constrained model predictive control for redundant mobile manipulators," *Intelligent Autonomous Systems 13, Advances in Intelligent Systems and Computing*, pp. 1301–1314, 2016.

[27] C. Schuetz, T. Buschmann, J. Baur, J. Pfaff, and H. Ulbrich, "Predictive online inverse kinematics for redundant manipulators," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Hong Kong (China), 2014, pp. 5056–5061.

[28] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Seattle (USA), 2015, pp. 1168–1175.

[29] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Hamburg (Germany), 2015, pp. 3346–3351.

[30] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *The Int. J. of Robotics Research*, vol. 10, pp. 410–425, 1991.

[31] F.-T. Cheng, T.-H. Chen, and Y.-Y. Sun, "Resolving manipulator redundancy under inequality constraints," *IEEE Trans. on Robotics and Automation*, vol. 10, pp. 65–71, 1994.

[32] M. Scheint, J. Wolff, and M. Buss, "Invariance control in robotic applications: Trajectory supervision and haptic rendering," in *Proc. IEEE American Control Conf.*, Seattle (USA), 2008, pp. 1436–1442.

[33] A. M. Zanchettin and P. Rocco, "Motion planning for robotic manipulators using robust constrained control," *Control Engineering Practice*, vol. 59, pp. 127–136, 2017.

[34] D. Dimitrov, P.-B. Wieber, H. J. Ferreau, and M. Diehl, "On the implementation of model predictive control for on-line walking pattern generation," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Pasadena (USA), 2008, pp. 2685–2690.

[35] L. Tamas, I. Nascu, and R. De Keyser, "The NEPSAC nonlinear predictive controller in a real life experiment," in *Int. Conf. on Intelligent Engineering Systems*, Budapest (Hungary), 2007, pp. 229–234.

[36] B. Siciliano and L. Villani, "Six-degree-of-freedom impedance robot control," in *Proc. IEEE Int. Conf. on Advanced Robotics*, Monterey (CA), 1997, pp. 387–392.

[37] F. Caccavale, B. Siciliano, and L. Villani, "The role of Euler parameters in robot control," *Asian J. of Control*, vol. 1, pp. 25–34, 1999.

[38] L. Roveda, N. Pedrocchi, M. Beschi, and L. Molinari Tosatti, "High-accuracy robotized industrial assembly task control schema with force overshoots avoidance," *Control Engineering Practice*, vol. 71, pp. 142–153, 2018.

[39] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, vol. 40, pp. 17–31, 2016.

[40] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai (Japan), 2004, pp. 2149–2154.