2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Discrete-Time Velocity Control of Redundant Robots with Acceleration/Torque Optimization Properties

Fabrizio Flacco Alessandro De Luca

Abstract— **The paper addresses the following problem for redundant robots. Given a second-order inverse differential scheme that realizes instantaneously a desired task acceleration and has some specified properties in terms of joint acceleration or torque, define a discrete-time joint velocity command that shares the same characteristics under suitable hypotheses. The goal is to obtain simpler implementations of possibly complex robot control laws that *i)* can be directly interfaced to the low-level servo loops of a robot, *ii)* require less task information and on-line computations, *iii)* are still provably good with respect to some target performance. The method is illustrated by considering the conversion into discrete-time velocity commands of control schemes for redundant robots that minimize the (possibly, weighted) norm of joint acceleration or joint torque, or that add null-space damping to overcome floating motion of the robot joints. Numerical results are presented for the kinematic control of a 7R KUKA LWR.**

I. Introduction

The most common way to control motion of robot manipulators still relies on a two-level architecture, with a direct layer of joint-level servo-actuators imposing motor currents, and thus joint torques to the robot, and a high-level, user-defined control program sending desired position/velocity references to the servos, based on proprioceptive sensing (encoders, joint torque sensors), exteroceptive sensing (e.g., visual feedback), and input from the task/trajectory planner.

In this framework, *kinematic control* schemes are designed without consideration of robot dynamics, assuming that reference velocity commands are accurately reproduced by the robot, thanks to the presence of fast and well-tuned low-level loops, and that the speed and acceleration involved in the motion task are moderate.

It is textbook matter that *dynamic control* can outperform conventional kinematic controllers, both in speed and accuracy [1]. For this to happen, we require a relatively complete and precise dynamic model of the robot and of its actuating devices, a so-called *open* architecture that allows to specify joint torques (or motor currents) as user commands, as well as advanced control laws.

It has been shown [2], [3] that one can get around a closed control architecture and still be able to generate velocity references at the user level, so as to apply desired torque commands to the robot. However, this torque transformer requires a good knowledge of the structure and parameters of the low-level control loops, an information typically unavailable for a generic end-user.

On the other hand, the latest generation of research-oriented manipulators allows a *torque-controlled* behavior [4]–[6]. In order to take full advantage of this possibility, a dynamic model of the robot is needed anyway (possibly including also joint/transmission compliance [7]). Moreover, the whole task and control treatment should be moved up to a second-order differential level (acceleration or torques) [8]. This complicates the on-line specification of sensor-based behaviors, since designing an acceleration command to accomplish a desired task is harder than doing the same with a velocity command. Therefore, the use of velocity commands at the task level that rely on fast control sampling rates for performance is still an appealing approach.

Our motivation is mainly a practical one. Velocity control laws are simpler to implement, require the least amount of data (e.g., no need of the time derivative of the Jacobian) and measurements, and can easily accommodate additional constraints, such as the presence of joint range limits and command saturation in redundant robots [9]. Conversely, first-order kinematic laws do not share the smoothness and dynamic optimality properties of acceleration or torque control laws (nor their levels of variety).

The purpose of this paper is to show how to define a kinematic control scheme at the velocity level[1] that inherits as much as possible the properties and performance of a given second-order control scheme. We will consider the following working assumptions:

1) Low-level servo loops are present that guarantee ideal execution of any joint velocity command. The reference model of the controlled robot is thus a pure integration between joint velocity commands and measured positions.
2) The high-level kinematic controller generates joint velocity references in discrete time, with sufficiently high sampling rate.
3) Task or Cartesian desired commands are available to the robot on line, but only up to the velocity level. Only local optimization schemes are considered.
4) Measurement in the joint space is limited to position, as provided by encoders.
5) The robot is kinematically redundant with respect to the given task.

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Roma, Italy ({fflacco,deluca}@diag.uniroma1.it). This work is supported by the European Commission, within the FP7 ICT-287513 SAPHARI project (www.saphari.eu).

[1]In the rest of the paper, we assume that velocity references can be directly fed to the low-level controller of the robot. If instead a position reference is required by the robot control interface, a one step discrete-time integration of the velocity command can be used.

The last assumption is not strictly needed, but enlarges considerably the scope and interest of the present analysis. Some early results on the equivalence of velocity and acceleration redundancy resolution schemes in continuous time can be found in [10], while the emphasis is given here to the discrete-time implementation, as well as to the possible inclusion of robot dynamic model information.

After summarizing redundancy control schemes and discretization issues in Sect. II, we present our main analytical results in Sects. III and IV, namely the discrete-time velocity commands that are equivalent to a joint acceleration solution of minimum norm, to an optimal joint torque solution, and to the addition of null-space damping terms for stabilizing joint floating motion. Section V reports numerical results for a 7R KUKA LWR executing end-effector position tasks.

II. Preliminaries

A. Redundancy resolution

Let $\boldsymbol{q} \in \mathbb{R}^n$ be the generalized (joint) coordinates of a n-dof robot and $\boldsymbol{x} \in \mathbb{R}^m$ be the variables describing a generic m-dimensional task, with $m \leq n$. The task kinematics is given by the direct map $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q})$. The first-order task kinematics is

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{1}$$

where $\boldsymbol{J} = \partial \boldsymbol{f}/\partial \boldsymbol{q}$ is the $m \times n$ task Jacobian matrix.

At a given configuration $\boldsymbol{q}$, all inverse velocity solutions to (1) can be expressed as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} + \boldsymbol{P}\dot{\boldsymbol{q}}_N, \tag{2}$$

where $\boldsymbol{J}^{\#}$ is the (Moore-Penrose) unique pseudoinverse of the task Jacobian [11], $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J}$ is the $n \times n$ orthogonal projector in the Jacobian null space, and $\dot{\boldsymbol{q}}_N \in \mathbb{R}^n$ is a generic joint velocity that can be used for auxiliary tasks. The pseudoinverse term $\boldsymbol{J}^{\#}\dot{\boldsymbol{x}}$ in (2) is the joint velocity of minimum norm among all those minimizing the norm of the task velocity error $\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}$. When $\boldsymbol{J}$ has full row rank, then $\dot{\boldsymbol{e}} = \boldsymbol{0}$ for any joint velocity in the form (2).

Differentiating (1), the second-order task kinematics is

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{3}$$

Note that matrix $\dot{\boldsymbol{J}}$ needs both $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ for its evaluation. At a given robot state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, all inverse acceleration solutions to (3) are given by

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) + \boldsymbol{P}\ddot{\boldsymbol{q}}_N. \tag{4}$$

being $\ddot{\boldsymbol{q}}_N \in \mathbb{R}^n$ a generic joint acceleration. Assuming full rank for $\boldsymbol{J}$, the first two terms provide the (exact) solution to (3) with minimum joint acceleration norm.

Let the dynamics of a fully actuated robot in free motion be compactly described by[2]

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{5}$$

[2]For the sake of simplicity, external forces and friction effects are not considered. Moreover, the underactuated case can be treated similarly.

with positive definite inertia matrix $\boldsymbol{M}$, Coriolis, centrifugal, and gravitational terms $\boldsymbol{n}$, and input torque $\boldsymbol{\tau} \in \mathbb{R}^n$. Based on the model (5), several local optimization-based control schemes have been proposed at the torque level [12]–[15] for executing a desired task acceleration $\ddot{\boldsymbol{x}}$. At a given robot state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, the solution with minimum torque norm of [12] is given by

$$\boldsymbol{\tau} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\right)^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{n}\right). \tag{6}$$

If $\boldsymbol{J}$ has full rank, $\left(\boldsymbol{J}\boldsymbol{M}^{-1}\right)^{\#} = \boldsymbol{M}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{M}^{-2}\boldsymbol{J}^T)^{-1}$.

The above schemes (2), (4), and (6), as well as all other local methods for optimal use of robot redundancy existing in the literature, can be obtained from a general Quadratic Programming (QP) formulation (see, e.g., [10] or [16]).

B. Discrete-time implementation

Assume that a user-defined control law is implemented in discrete time, with control samples applied uniformly over time at $t = t_k = kT$, and with a sufficiently small sampling time T (say, in the order of one or few msec). Denote by $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$ the sample of a generic vector or matrix variable $\boldsymbol{u}$. Let the robot be driven by joint velocity commands $\dot{\boldsymbol{q}}_k$.

In the common practice, a desired task acceleration $\ddot{\boldsymbol{x}}_k$ is approximated as

$$\ddot{\boldsymbol{x}}_k \simeq \frac{\dot{\boldsymbol{x}}_k - \dot{\boldsymbol{x}}_{k-1}}{T}, \tag{7}$$

so that the desired task information needed is only at the velocity level. The time-derivative of the task Jacobian $\boldsymbol{J}(\boldsymbol{q})$ at $t = t_k$ can be approximated as

$$\dot{\boldsymbol{J}}_k \simeq \frac{\boldsymbol{J}_k - \boldsymbol{J}_{k-1}}{T}, \tag{8}$$

where $\boldsymbol{J}_k = \boldsymbol{J}(\boldsymbol{q}_k)$. Only the current and previous joint position measurements $\boldsymbol{q}_k$ and $\boldsymbol{q}_{k-1}$ are used. Finally, wherever a joint velocity $\dot{\boldsymbol{q}}$ appears in the continuous-time version of a second-order control law, we need to evaluate this quantity as

$$\dot{\boldsymbol{q}} \simeq \dot{\boldsymbol{q}}_{k-1} \simeq \frac{\boldsymbol{q}_k - \boldsymbol{q}_{k-1}}{T}, \tag{9}$$

namely using either the (last) velocity command computed at the previous sampling instant, or the current and previous joint position measurements. The latter uses the *forward differences* formula to approximate time differentiation.

To check the significance of the choices (7)–(9), consider the standard task controller in the absence of redundancy (and with the robot in a nonsingular configuration) that realizes a desired acceleration $\ddot{\boldsymbol{x}}$ by imposing the joint acceleration

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}\right). \tag{10}$$

In discrete time, using the above approximations, one obtains

$$\begin{aligned} \ddot{\boldsymbol{q}}_k &= \boldsymbol{J}^{-1}(\boldsymbol{q}_k)\left(\ddot{\boldsymbol{x}}_k - \dot{\boldsymbol{J}}(\boldsymbol{q}_k)\dot{\boldsymbol{q}}_k\right) \\ &\simeq \boldsymbol{J}_k^{-1}\left(\frac{\dot{\boldsymbol{x}}_k - \dot{\boldsymbol{x}}_{k-1}}{T} - \frac{\boldsymbol{J}_k - \boldsymbol{J}_{k-1}}{T}\dot{\boldsymbol{q}}_{k-1}\right) \\ &= \frac{1}{T}\left(\boldsymbol{J}_k^{-1}\dot{\boldsymbol{x}}_k - \dot{\boldsymbol{q}}_{k-1}\right), \end{aligned} \tag{11}$$

where we used the identity $\boldsymbol{J}_{k-1}\dot{\boldsymbol{q}}_{k-1} = \dot{\boldsymbol{x}}_{k-1}$, due to our working assumption of perfect joint velocity execution. Using (11) and *backward differences* to approximate integration over time, the joint velocity command at t_k is obtained as

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + \ddot{\boldsymbol{q}}_k T = \boldsymbol{J}_k^{-1}\dot{\boldsymbol{x}}_k, \tag{12}$$

which is exactly what we would have obtained from the discretization of a first-order task controller. This shows that the chosen approximation steps provide a consistent evaluation in discrete time, and we will use them next with confidence also for dealing with redundancy.

III. Velocity Control for Acceleration Optimization

A. Minimum acceleration norm

In the following, we will assume for simplicity that the Jacobian $\boldsymbol{J}$ has full rank at the current configuration. At the acceleration level, we have in continuous time

$$\begin{aligned}\ddot{\boldsymbol{q}}^* &= \arg\min_{\ddot{\boldsymbol{q}}\in\mathbb{R}^n} \frac{1}{2}\left\|\ddot{\boldsymbol{q}}\right\|^2 \quad \text{s.t.} \;\; \boldsymbol{J}\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} \\ &= \boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right).\end{aligned} \tag{13}$$

The exact evaluation of the minimum acceleration norm solution (13) at discrete instants of time $t = t_k$ is

$$\ddot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\left(\ddot{\boldsymbol{x}}_k - \dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k\right), \tag{14}$$

where the current value $\dot{\boldsymbol{q}}_k$ is used twice, as such and within the evaluation of the term $\dot{\boldsymbol{J}}_k$.

In the operative conditions of velocity control, applying to (14) the discrete-time approximations (7–9) as in Sec. II-B, and using backward differences to integrate acceleration by one step, yields

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\dot{\boldsymbol{x}}_k + \boldsymbol{P}_k\,\dot{\boldsymbol{q}}_{k-1}, \tag{15}$$

where $\boldsymbol{P}_k = \boldsymbol{I} - \boldsymbol{J}_k^{\#}\boldsymbol{J}_k$. It is immediately recognized that equation (15) is the discrete-time evaluation of the general first-order solution (2), with $\dot{\boldsymbol{q}}_{N,k} = \dot{\boldsymbol{q}}_{k-1}$. It is easy to verify that $\dot{\boldsymbol{q}}_k$ in (15) solves also

$$\dot{\boldsymbol{q}}_k = \arg\min_{\dot{\boldsymbol{q}}\in\mathbb{R}^n} \frac{1}{2}\left\|\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{k-1}\right\|^2 \quad \text{s.t.} \;\; \boldsymbol{J}_k\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}_k. \tag{16}$$

Indeed, locally minimizing the joint acceleration (in norm) is equivalent to minimizing the instantaneous variation of joint velocity. In discrete time, this means in turn taking the joint velocity solution nearest to the previous value $\dot{\boldsymbol{q}}_{k-1}$. The above confirms this intuitive fact.

Therefore, the velocity control law (15) shares (approximately) the optimal properties of an acceleration-level solution, but it is simpler than (14). In fact, its actual implementation is

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + \boldsymbol{J}_k^{\#}\left(\dot{\boldsymbol{x}}_k - \boldsymbol{J}_k\dot{\boldsymbol{q}}_{k-1}\right), \tag{17}$$

without the need to build the projection matrix.

Note that the above reasoning for conversion to velocity control can be applied similarly also in the case of l multiple tasks, without or with priority [17]. Assume that the multi-task case has been solved at the velocity level by any known methodology, say [18], and denote this solution as $\dot{\boldsymbol{q}}_{l,k}$, while the projector in the null space of all l tasks will be $\boldsymbol{P}_{l,k}$. Then, according to (15), the velocity command that minimizes the joint acceleration norm is obtained as

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{l,k} + \boldsymbol{P}_{l,k}\,\dot{\boldsymbol{q}}_{k-1}. \tag{18}$$

B. Dealing with floating null-space motion

When a redundant robot is controlled using minimum norm commands at the second or higher differential level (i.e., from acceleration upwards), one can typically observe the joints starting to float over time on the self-motion manifold. This drift phenomenon has a simple explanation.

When the robot is being controlled at the velocity level, any joint velocity contribution in the null space of the task Jacobian at the current configuration will be set to zero, because of the minimum norm property. A simple case to analyze is when the generic joint i is not useful for performing the task, i.e., the i-th column of the task Jacobian is zero. In the minimum joint velocity norm solution, the i-th component of $\dot{\boldsymbol{q}}$ is automatically zero, and joint i will not move. On the other hand, when the robot is controlled at the acceleration level, joint accelerations in the null space will be set to zero in the minimum norm solution. In the simple case above, the i-th component of $\ddot{\boldsymbol{q}}$ will be zero, and if joint i was not at rest it will drift at constant velocity, producing a floating motion in the null space of the Jacobian.

To remove this undesired behavior, the simplest solution is to introduce damping on the null-space motion [10]. This can be illustrated well at the acceleration level, since the choice

$$\ddot{\boldsymbol{q}} = -k_d\,\dot{\boldsymbol{q}}, \qquad k_d \geq 0, \tag{19}$$

in the unconstrained case, i.e., when no task is assigned, will bring the robot to a rest configuration, at an exponential rate depending on the positive damping factor k_d (which can be in principle arbitrarily large, albeit only in continuous time). Therefore, the preferred choice in (4) will be $\ddot{\boldsymbol{q}}_N = -k_d\,\dot{\boldsymbol{q}}$, and we obtain in continuous time

$$\begin{aligned}\ddot{\boldsymbol{q}}^* &= \arg\min_{\ddot{\boldsymbol{q}}\in\mathbb{R}^n} \frac{1}{2}\left\|\ddot{\boldsymbol{q}} + k_d\dot{\boldsymbol{q}}\right\|^2 \quad \text{s.t.} \;\; \boldsymbol{J}\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} \\ &= \boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - k_d\boldsymbol{P}\dot{\boldsymbol{q}}.\end{aligned} \tag{20}$$

Proceeding as before, it is easy to see that the conversion of (20) into a discrete-time velocity control law, with sampling time $T > 0$, provides

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\dot{\boldsymbol{x}}_k + \lambda\boldsymbol{P}_k\dot{\boldsymbol{q}}_{k-1}, \qquad \lambda = 1 - k_d T. \tag{21}$$

It can be recognized that the scalar $\lambda \leq 1$ in (21) acts as a *forgetting factor* on the previous joint velocity command —a typical tool for enforcing stability in recursive least squares sequences of sampled data [19].

A particular but significant situation is considered next for a stability analysis of the choice of scalar λ. Suppose that the robot has accomplished its original task, that the joints are still in motion, and that no specific task is required any longer. Reset the time counter ($k = 0$), and let the initial joint position and velocity be $\boldsymbol{q}_0$ and $\dot{\boldsymbol{q}}_0 \neq 0$, respectively. Since there is no task from now, we set $\boldsymbol{J}_k = \boldsymbol{0}$, $\forall k \geq 0$, and so $\boldsymbol{P}_k = \boldsymbol{I}$ (all joint motions are in the task null space). Equation (21) becomes

$$\dot{\boldsymbol{q}}_k = \lambda \dot{\boldsymbol{q}}_{k-1}. \tag{22}$$

For $\lambda = 1$, the joint velocity will never change, and the joint acceleration norm is clearly minimized. However, the sequence of position samples will diverge, $\|\boldsymbol{q}_k\| \to \infty$ (in practice, the joints move up to their range limits).

For $|\lambda| < 1$, equation (22) is a contraction mapping and the robot configuration will converge to

$$\boldsymbol{q}_\infty = \boldsymbol{q}_0 + \frac{T\dot{\boldsymbol{q}}_0}{1-\lambda}, \tag{23}$$

In particular, for $\lambda = 0$, the robot stops at the next step, clearly at the cost of a very high joint acceleration.

This analysis can be backtracked so as to set actual boundaries for $k_d \geq 0$ in the continuous-time acceleration solution (20), taking into account the need of its discretization to a velocity control law. In fact, for $k_d \in (0, 1/T)$, an uniformly stable behavior is obtained. For $k_d = 1/T$, the acceleration-level law leads to a first-order control with minimum velocity norm. Convergence is obtained also for $k_d \in (1/T, 2/T)$, but pseudo-oscillating in nature, while for $k_d = 2/T$ the self-motion will be persistently oscillating. Finally, for $k_d > 2/T$ there will be a divergent, oscillatory floating motion in the null space of the task.

IV. Extension to Torque Optimization

Moving the procedure for conversion into a discrete-time velocity control from an acceleration-level scheme to a torque-level scheme is rather straightforward. Two analytical examples of interest are presented, which illustrate also the different complexity resulting from slightly different local optimization formulations that deal with robot dynamics. Again, we assume that the Jacobian $\boldsymbol{J}$ has full rank at the current configuration.

A. Minimum of a weighted norm of the torque

Following [13], and later [15], a convenient robot behavior is obtained when minimizing the joint torques in terms of a norm weighted by the *squared inverse of the inertia* matrix:

$$\begin{aligned}\boldsymbol{\tau}^* \;=\; & \arg\min_{\boldsymbol{\tau}\in\mathbb{R}^n} \frac{1}{2}\boldsymbol{\tau}^T \boldsymbol{M}^{-2}\boldsymbol{\tau} \\ & \text{s.t. } \boldsymbol{J}\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}, \quad \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{n} = \boldsymbol{\tau}.\end{aligned} \tag{24}$$

The joint motion remains bounded even for longer motion tasks, as opposed to the case of the (unweighted) minimum torque norm solution [12].

The optimal solution to (24) is

$$\boldsymbol{\tau}^* = \boldsymbol{M}\boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{n}\right). \tag{25}$$

Associated to this torque, there is a unique joint acceleration that takes the form

$$\ddot{\boldsymbol{q}}^* = \boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - \boldsymbol{P}\boldsymbol{M}^{-1}\boldsymbol{n}. \tag{26}$$

Conversion to the equivalent discrete-time velocity control is easily obtained when starting from (26). Define first the notation $\boldsymbol{n}_{k|k-1} = \boldsymbol{n}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_{k-1})$, where all quadratic velocity terms are evaluated using the previous joint velocity sample[3]. Proceeding as in Sect. III, we obtain

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\dot{\boldsymbol{x}}_k + \boldsymbol{P}_k\left(\dot{\boldsymbol{q}}_{k-1} - T\boldsymbol{M}_k^{-1}\boldsymbol{n}_{k|k-1}\right). \tag{27}$$

Note that all needed dynamic information is contained in the last extra term, which is also scaled by the sampling time T.

B. Dynamically consistent redundancy resolution

The other considered torque-level method for redundant robots is the dynamically consistent approach of [14], which is based on a task-oriented decomposition of the joint torques. However, also in this case the problem can be formulated as a QP where torque differences are minimized with respect to a desired reference $\boldsymbol{\tau}_N$ and using a norm weighted by the simple *inverse of the inertia* matrix:

$$\begin{aligned}\boldsymbol{\tau}^* \;=\; & \arg\min_{\boldsymbol{\tau}\in\mathbb{R}^n} \frac{1}{2}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_N\right)^T \boldsymbol{M}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_N\right) \\ & \text{s.t. } \boldsymbol{J}\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}, \quad \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{n} = \boldsymbol{\tau}\end{aligned} \tag{28}$$

The optimal solution to (28) is

$$\begin{aligned}\boldsymbol{\tau}^* \;=\; & \boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{n}\right) \\ & + \left(\boldsymbol{I} - \boldsymbol{J}^T(\boldsymbol{J}^T)^{\#}_{\boldsymbol{M}}\right)\boldsymbol{\tau}_N\end{aligned} \tag{29}$$

where the inertia-weighted pseudoinverse of the Jacobian transpose is

$$(\boldsymbol{J}^T)^{\#}_{\boldsymbol{M}} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}\boldsymbol{J}\boldsymbol{M}^{-1}. \tag{30}$$

The unique joint acceleration associated to (29),

$$\ddot{\boldsymbol{q}}^* = \boldsymbol{J}^{\#}_{\boldsymbol{M}}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - \boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{M}^{-1}\left(\boldsymbol{n} - \boldsymbol{\tau}_N\right), \tag{31}$$

serves again as a basis for deriving the discrete-time velocity control. We obtain

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}^{\#}_{\boldsymbol{M},k}\dot{\boldsymbol{x}}_k + \boldsymbol{P}_{\boldsymbol{M},k}\left(\dot{\boldsymbol{q}}_{k-1} - T\boldsymbol{M}_k^{-1}\left(\boldsymbol{n}_{k|k-1} - \boldsymbol{\tau}_{N,k}\right)\right) \tag{32}$$

The complexity of the control law is indeed increased in this case. For instance, the expression of the weighted projector $\boldsymbol{P}_{\boldsymbol{M},k}$ is

$$\boldsymbol{P}_{\boldsymbol{M},k} = \boldsymbol{I} - \boldsymbol{M}_k^{-1}\boldsymbol{J}_k^T\left(\boldsymbol{J}_k\boldsymbol{M}_k^{-1}\boldsymbol{J}_k^T\right)^{-1}\boldsymbol{J}_k. \tag{33}$$

On the other hand, the inclusion of a velocity damping torque for the stabilization of the robot self-motions can be devised so as to simplify (32). For this, choose the reference torque

[3]This is different from [20], where Coriolis and centrifugal terms were evaluated mixing previous and current velocities.

τ_N so as to oppose the *generalized momentum* $M\dot{q}$ with a suitable damping gain $k_d = 1/T$. In discrete time, we have

$$\tau_{N,k} = -\frac{1}{T} M_k \dot{q}_{k-1}, \tag{34}$$

Using this in (32) yields the simpler form

$$\dot{q}_k = J^{\#}_{M,k}\dot{x}_k - T P_{M,k}\, M_k^{-1}\, n_{k|k-1}. \tag{35}$$

V. Numerical results

The method has been tested in simulations using a kinematic model of the KUKA LWR robot ($n = 7$). A point-to-point motion task is specified for the position $x = f(q)$ of the robot end-effector ($m = 3$), starting from the initial Cartesian point $x_A = (-0.37, 0.3, 1.12)$ [m], associated to the configuration $q_A = q(0) = (0, 45, 45, 45, 0, 0, 0)$ [deg] at time $t = 0$, and reaching the point $x_B = (0.7, 0.15, 0.2)$ [m]. The trajectory is a linear path, with rest-to-rest timing law given by a double-normalized quintic polynomial:

$$\begin{aligned} x_d(t) &= x_A + (x_B - x_A)\, s(\tau), \\ s(\tau) &= 6\tau^5 - 15\tau^4 + 10\tau^3, \quad \tau = \frac{t}{T_{AB}} \in [0,1], \\ \dot{x}_d(t) &= \frac{x_B - x_A}{T_{AB}} \left(30\tau^4 - 60\tau^3 + 30\tau^2\right). \end{aligned} \tag{36}$$

where $T_{AB} = 1$ [s] is the motion time. The desired task velocity is specified using also a Cartesian error feedback to recover linearization errors

$$\dot{x} = \dot{x}_d + k_P\left(x_d - f(q)\right), \tag{37}$$

with the gain parameter set to $k_P = 10$. The simulation ends when the position error $\|x_d - f(q)\| < \epsilon$, with $\epsilon = 1$ [mm]. The sampling time is $T = 1$ [ms].

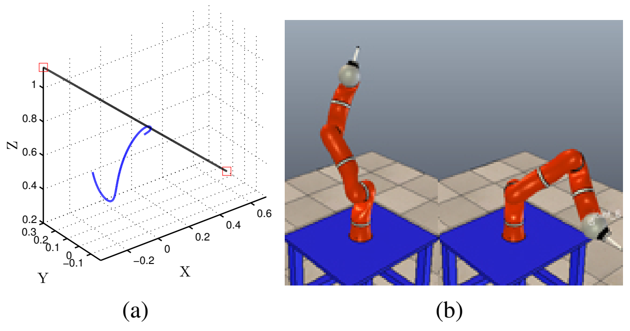


Fig. 1. Velocity control for point-to-point motion with the proposed discrete-time approach and *without* null-space damping ($\lambda = 1$ in eq. (21)): (a) end-effector (black) and elbow (blue) trajectories; (b) initial and final screenshots of the simulation

Figures 1–2 show the results using the discrete-time velocity control (21) with forgetting factor $\lambda = 1$, corresponding to a minimum acceleration norm solution without damping, i.e., with $k_d = 0$ in (20). The Cartesian trajectories of the LWR end-effector and elbow (i.e., the tip of the third robot link) are shown in Fig. 1. Figure 2 reports the evolution of three norms used as indices of performance: joint acceleration, null-space contribution, and joint velocity (the actual command). For validation, the same task has been

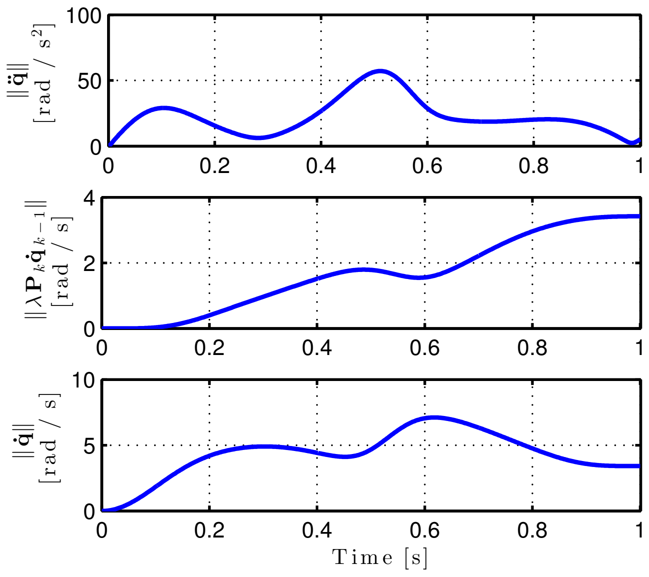


Fig. 2. Numerical results of the proposed discrete-time approach *without* null-space damping ($\lambda = 1$ in eq. (21)): norm of the joint acceleration $\|\ddot{q}_k\|$ (top); norm of the null-space contribution $\|\lambda P_k \dot{q}_{k-1}\|$ (center); norm of the joint velocity $\|\dot{q}_k\|$ (bottom)

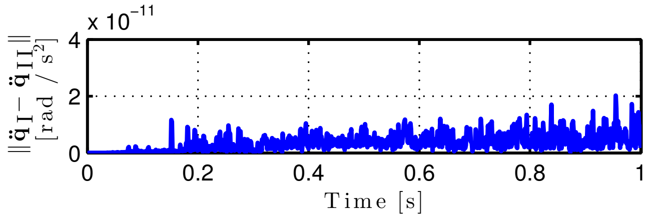


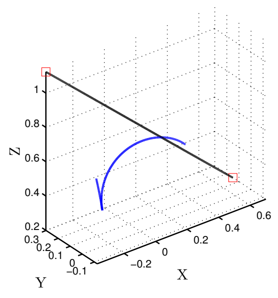
Fig. 3. Norm of the difference between the joint acceleration $\ddot{q}_{\rm I}$ obtained when using the velocity control (21) and the joint acceleration $\ddot{q}_{\rm II}$ obtained with the acceleration control (14) —no damping is used here

performed with the minimum norm joint acceleration command (14). The equivalence between the proposed approach and the acceleration-level control is confirmed by the plot of the differences between the joint accelerations in the two simulations (Fig. 3), which is zero up to numerical rounding.

Figures 4–5 refer to the same point-to-point motion obtained with the discrete-time velocity control (21) and the forgetting factor $\lambda = 0.8$ (damping in eq. (20) is $k_d = 200$). As expected, the peak of the acceleration norm is slightly higher than before (over 60 w.r.t. 56 [rad/s^2]), the positive effect of velocity damping in the null space is quite clear when comparing the corresponding plots at the center of Figs. 2 and 5. The drift of internal motion is removed and the peak value is decreased by a factor of 60 (in face of a reduction of λ by just 20%). The final configuration reached by the manipulator is different than before, compare Fig. 4(b) with Fig. 1(b).

VI. Conclusions

Considering practical operative conditions and common implementation constraints, we have shown how to determine a discrete-time joint velocity command for redundant robots that executes a desired task in a more dynamically oriented



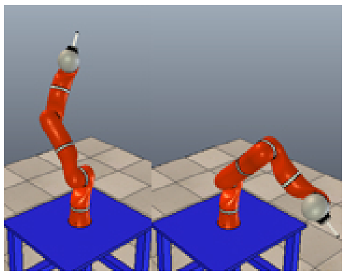


Fig. 4. Velocity control for point-to-point motion with the proposed discrete-time approach and *with* null-space damping ($\lambda = 0.8$ in eq. (21)): (a) end-effector (black) and elbow (blue) trajectories; (b) initial and final screenshots of the simulation

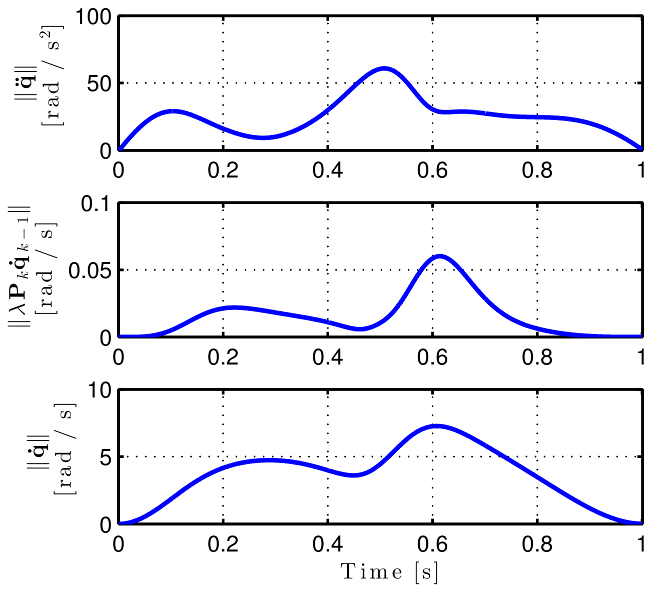


Fig. 5. Numerical results of the proposed discrete-time approach *with* null-space damping ($\lambda = 0.8$ in eq. (21)): norm of the joint acceleration $\|\ddot{\boldsymbol{q}}_k\|$ (top); norm of the null-space contribution $\|\lambda \boldsymbol{P}_k \dot{\boldsymbol{q}}_{k-1}\|$ (center); norm of the joint velocity $\|\dot{\boldsymbol{q}}_k\|$ (bottom)

way. The resulting first-order kinematic control law mimics, at least approximately, the behavior of second-order schemes optimizing suitable acceleration or torque norms, and can be directly interfaced to the low-level servo loops of a closed robot control architecture.

In doing the conversion, we found a nice analogy between the addition of null-space damping in acceleration/torque second-order schemes and the introduction of a forgetting factor in discrete-time velocity control. More in general, all the obtained results can be formally derived from a QP formulation that provides also additional insights on the role of kinematic/dynamic terms in the null space of the task.

Some of the discrete-time velocity schemes have already been implemented in control experiments on the KUKA LWR-IV robot available in our laboratory. We are currently plannning to evaluate the comparative performance of a discrete-time implementation of our SNS (Saturation in the Null Space) method that deals with hard bounds in the joint space for redundant robots [21]. In fact. this method is a good candidate for our purposes, since it has been developed in continuous time both at the velocity and acceleration/torque level, with pros and cons in each case. Comparison of performance has been made so far using a common, sufficiently small sampling time. Indeed, for a fair comparison simpler controllers should be allowed to run at a faster rate than computationally heavier laws.

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modeling, Planning and Control*, 3rd ed. London: Springer, 2008.

[2] O. Khatib, P. Thaulad, T. Yoshikawa, and J. Park, "A torque-position transformer for task control of position controlled robots," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 1729–1734.

[3] T. Yoshikawa and O. Khatib, "Compliant humanoid robot control by the torque transformer," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 3011–3018.

[4] W. Townsend and J. Salisbury, "Mechanical design for whole-arm manipulation," in *Robots and Biological Systems: Towards a New Bionics?*, P. Dario, G. Sandini, and P. Aebischer, Eds. Springer, 1993, pp. 153–164.

[5] G. Hirzinger, A. Albu-Schäffer, M. Hähnle, I. Schaefer, and N. Sporer, "On a new generation of torque controlled light-weight robots," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 3356–3363.

[6] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR Lightweight Robot arm a new reference platform for robotics research and manufacturing," in *Proc. 41st Int. Symp. on Robotics*, 2010, pp. 741–748.

[7] A. De Luca and W. Book, "Robots with flexible elements," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 287–319.

[8] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[9] S. Chiaverini, G. Oriolo, and I. Walker, "Kinematically redundant manipulators," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 245–268.

[10] A. De Luca, G. Oriolo, and B. Siciliano, "Robot redundancy resolution at the acceleration level," *Laboratory Robotics and Automation*, vol. 4, no. 2, pp. 97–106, 1992.

[11] T. L. Boullion and P. L. Odell, *Generalized Inverse Matrices*. Wiley-Interscience, 1971.

[12] J. Hollerbach and K. Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE J. of Robotics and Automation*, vol. 3, no. 4, pp. 308–316, 1987.

[13] K. Kazerounian and Z. Wang, "Global versus local optimization in redundancy resolution of robotic manipulators," *Int. J. of Robotics Research*, vol. 7, no. 5, pp. 3–12, 1988.

[14] O. Khatib, "The operational space framework," *JSME Int. J. Ser. C: Dynamics, Control, Robotics, Design and Manufacturing*, vol. 36, no. 3, pp. 277–287, 1993.

[15] S. Ma, "Local torque optimization of redundant manipulators in torque-based formulation," in *Proc. 20th Int. Conf. on Industrial Electronics, Control and Instrumentation*, vol. 2, 1994, pp. 697–702.

[16] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal, "A unifying framework for robot control with redundant DOFs," *Autonomous Robots*, vol. 24, no. 1, pp. 1–12, 2008.

[17] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

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

[19] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. MIT Press, 1983.

[20] K. Kazerounian and A. Nedungadi, "An alternative method for minimization of the driving forces in redundant manipulators," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1987, pp. 1701–1706.

[21] F. Flacco, A. De Luca, and O. Khatib, "Motion control of redundant robots under joint constraints: Saturation in the null space," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 285–292.