Faster Motion on Cartesian Paths Exploiting Robot Redundancy at the Acceleration Level

Khaled Al Khudir and Alessandro De Luca

***Abstract*—The problem of minimizing the transfer time along a given Cartesian path for redundant robots can be approached in two steps, by separating the generation of a joint path associated to the Cartesian path from the exact minimization of motion time under kinematic/dynamic bounds along the obtained parametrized joint path. In this framework, multiple sub-optimal solutions can be found, depending on how redundancy is locally resolved in the joint space within the first step. We propose a solution method that works at the acceleration level, by using weighted pseudoinversion, optimizing an inertia-related criterion, and including null-space damping. Several numerical results obtained on different robot systems demonstrate consistently good behaviors and definitely faster motion times in comparison with related methods proposed in the literature. The motion time obtained with our method is reasonably close to the global time-optimal solution along same Cartesian path. Experimental results on a KUKA LWR IV are also reported, showing the tracking control performance on the executed motions.**

***Index Terms*—Optimization and Optimal Control, Motion Control, Trajectory Planning, Redundant Robots, Dynamics.**

I. Introduction

FOLLOWING a prescribed geometric path with an end-effector tool is one of the most common tasks that robot manipulators perform in industrial applications. The path only determines the task geometry in the Cartesian space, leaving the velocity motion profile along the path unspecified. In such cases, it is often desirable to traverse the path in the least possible time while not violating actuator limits.

Several algorithms have been proposed for the time-optimal path following problem under dynamic constraints, starting with the seminal works [1], [2], refined later in [3], [4]. The original idea was to work in the phase plane defined by the path parameter and its first time derivative. In [5], the problem has been formulated in terms of convex optimal control, taking advantage of general numerical algorithms. More recently, an efficient and stable algorithmic tool, called TOPP (Time-Optimal Path Parameterization), has been implemented in [6], solving the problem under dynamic as well as kinematic constraints. All these results apply both to Cartesian and joint paths, but in the first case it is implicitly assumed that the robot has as many joints as strictly needed for moving along the desired Cartesian path (non-redundancy).

A manipulator is kinematically redundant for a given task if the number n of its degrees of freedom (viz. joints) is larger than the dimension m of the task, or $n > m$. A general and efficient global solution to the time-optimal control problem along Cartesian paths in kinematically redundant robots has not been found yet. In [7], the problem was addressed by Pontryagin's maximum principle, looking for regular trajectories in an extended state space. An approach based on decomposition into non-redundant and redundant joints was introduced in [8]. These techniques are able to generate optimal solutions mainly for the case of degree of redundancy $n-m=1$. A non-convex numerical method with null-space augmentation is proposed in [9].

A different way to tackle the problem is to separate it into two steps: first, a specific joint path is generated from the assigned Cartesian path, typically by local (or semi-global) inverse differential methods; then, motion time is exactly minimized under the given kinematic/dynamic bounds along the unique joint path thus found. This approach was pioneered in [10], and later in [11], providing satisfactory results. In their first step, robot redundancy was locally exploited using first-order differential inverse kinematic solutions that, e.g., increase robot manipulability along the tangent direction to the Cartesian path. Indeed, there are infinite ways to generate a path in the joint space within the first step of this procedure. The challenge is to obtain paths along which the optimal selection of the timing law (say, by the TOPP algorithm) achieves the fastest possible motion transfer. For this, a number of additional dynamic issues, such as those considered in [12] and [13], should be conveniently incorporated in the differential inversion of the Cartesian path.

In this paper, we address the time-optimal trajectory planning along a Cartesian path for a kinematically redundant robot with a two-step procedure. We propose to generate a sequence of joint configurations by means of a second-order differential inverse kinematics scheme, using weighted pseudoinversion, optimizing locally an inertia-related criterion, and including judiciously a damping action in the null space of the task. The obtained configurations are then interpolated with a parameterized path in the joint space, and an exact minimum time solution is computed using the TOPP algorithm. In case the initial robot configuration is not assigned a priori, we include also a kinematic optimization scheme to find the best initial joint configuration corresponding to the starting point of the Cartesian path.

The paper is organized as follows. The formulation of the time-optimal planning problem on a parametrized joint path and its basic solution algorithm are reviewed in Sec. II. Section III presents the core of the method, moving ideas that exploit robot redundancy from the first- to the second-order differential level. Section IV reports comparative numerical

Manuscript received: February 24, 2018; revised May 22, 2018; accepted June 18, 2018. This paper was recommended for publication by Editor P. Rocco upon evaluation of the Associate Editor and Reviewers' comments.

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Roma, Italy (e-mail: alkhudir@diag.uniroma1.it; deluca@diag.uniroma1.it).

Digital Object Identifier (DOI): see top of this page.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2853806, IEEE Robotics and Automation Letters

results for a 3R planar arm performing two-dimensional tasks. We provide also a procedure in order to evaluate the distance between solutions obtained with a two-step approach and the global time-optimal motion computed along the same Cartesian path, using an alternative nonlinear programming method. Experimental results are reported in Sec. V for a KUKA LWR IV robot executing positional tasks (also shown in the accompanying video). Conclusions and future work are summarized in Sec. VI.

II. Time-Optimal Planning on a Geometric Path

We briefly review the basic formulation of the time-optimal planning on a parametrized joint path as in [1], [6].

The dynamic model of a rigid robot with n degrees of freedom is given by

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{S}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ and $\boldsymbol{\tau} \in \mathbb{R}^n$ denote joint configurations and torques, respectively, $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{S}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ accounts for the centrifugal and Coriolis terms (factorized using the Christoffel symbols), and $\boldsymbol{g}(\boldsymbol{q})$ represents the gravitational torques. Matrix $\boldsymbol{S}$ depends linearly on the velocity $\dot{\boldsymbol{q}}$. Assume that robot motion in the joint space is constrained to a given path that is continuously parametrized by a scalar s as a (non-decreasing) function of time t, or

$$\boldsymbol{q} = \boldsymbol{q}(s), \quad s \in [0, s_{\text{end}}], \qquad s = s(t), \quad t \in [0, t_{\text{end}}]. \tag{2}$$

Differentiating (2) once and twice with respect to time yields

$$\dot{\boldsymbol{q}} = \boldsymbol{q}'\dot{s}, \qquad \ddot{\boldsymbol{q}} = \boldsymbol{q}'\ddot{s} + \boldsymbol{q}''\dot{s}^2, \tag{3}$$

where a dot $\dot{(\,)}$ and a prime $(\,)'$ denote differentiation with respect to time t and to parameter s. The robot is subject to bounds on the joint torques

$$\boldsymbol{\tau}^{\min} \le \boldsymbol{\tau}(t) \le \boldsymbol{\tau}^{\max}, \qquad \forall t \in [0, t_{\text{end}}], \tag{4}$$

where $\boldsymbol{\tau}^{\max}$ and $\boldsymbol{\tau}^{\min}$ (usually, equal to $-\boldsymbol{\tau}^{\max}$) are constant vectors, and inequalities are to be intended component-wise. Substituting (2) and (3) into (1) and (4), and rearranging the terms, leads to

$$\boldsymbol{\tau}^{\min} \le \boldsymbol{a}(s)\ddot{s} + \boldsymbol{b}(s)\dot{s}^2 + \boldsymbol{g}(s) \le \boldsymbol{\tau}^{\max} \tag{5}$$

where $\boldsymbol{a} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'$, $\boldsymbol{b} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'' + \boldsymbol{S}(\boldsymbol{q},\boldsymbol{q}')\boldsymbol{q}'$, and $\boldsymbol{g}$ is again the gravity torque vector (all arguments are evaluated using (2) and (3)). As a result, a trajectory $\boldsymbol{q}(s(t))$ will be feasible if and only if the following bounds on the pseudo-acceleration $\ddot{s}$ are satisfied along the whole path

$$\alpha(s(t),\dot{s}(t)) \le \ddot{s}(t) \le \beta(s(t),\dot{s}(t)), \quad \forall s \in [0, s_{\text{end}}]. \tag{6}$$

For each $(s,\dot{s})$, the upper and lower acceleration bounds in (6) are defined as

$$\alpha(s,\dot{s}) = \max_i \alpha_i(s,\dot{s}) \quad \text{and} \quad \beta(s,\dot{s}) = \min_i \beta_i(s,\dot{s}). \tag{7}$$

The expressions of α_i and β_i depend on the sign of $a_i(s)$. In particular, for $i = 1,\dots,n$:

- if $a_i(s) > 0$, then $\begin{cases} \alpha_i = \frac{\tau_i^{\min} - g_i(s) - b_i(s)\dot{s}^2}{a_i(s)}, \\ \beta_i = \frac{\tau_i^{\max} - g_i(s) - b_i(s)\dot{s}^2}{a_i(s)}; \end{cases}$
- if $a_i(s) < 0$, then $\begin{cases} \alpha_i = \frac{-\tau_i^{\max} + g_i(s) + b_i(s)\dot{s}^2}{|a_i(s)|}, \\ \beta_i = \frac{-\tau_i^{\min} + g_i(s) + b_i(s)\dot{s}^2}{|a_i(s)|}; \end{cases}$
- if $a_i(s) = 0$, then s is a zero-inertia point.

The last case is a dynamic singularity that should be handled separately [3], [4]. From (6), a maximum velocity curve $\text{MVC}_t(s)$ is imposed in the $(s,\dot{s})$ plane, defined by

$$\text{MVC}_t(s) = \begin{cases} \min\{\dot{s} \ge 0 : \alpha(s,\dot{s}) = \beta(s,\dot{s})\}, & \text{if } \alpha(s,0) \le \beta(s,0), \\ 0, & \text{if } \alpha(s,0) > \beta(s,0). \end{cases}$$

Kinematic constraints (e.g., joint velocity limits) can also be considered [14]. Assuming symmetric bounds, we have for the velocity of joint i, $i = 1,\dots,n$:

$$-\dot{q}_i^{\max} \le \dot{q}_i \le \dot{q}_i^{\max} \quad \Rightarrow \quad \dot{s}_i^{\max}(s) = \frac{\dot{q}_i^{\max}}{|q_i'(s)|}. \tag{8}$$

The overall bound on $\dot{s}$ will be

$$\dot{s}_{\max}(s) = \min_i \dot{s}_i^{\max}(s), \quad \forall s \in [0, s_{\text{end}}]. \tag{9}$$

Equation (9) induces another maximum velocity curve, denoted $\text{MVC}_v(s)$. As a result, every feasible timing law $s = s(t)$ must remain below the curve in the phase plane $(s,\dot{s})$ defined by $\text{MVC} = \min\{\text{MVC}_t, \text{MVC}_v\}$.

Based on Pontryagin Maximum Principle, the optimal trajectory in the $(s,\dot{s})$ plane that minimizes the rest-to-rest motion time T is given by a control law of the bang-bang type. The pseudo-acceleration $\ddot{s}$ follows alternatively α or β, while the profile of $\dot{s}$ should always stay below the maximum velocity curve MVC. From (6) and (9), it can be shown that at least one joint is saturated at any time either to its torque bound or to its velocity bound. The optimal timing law $s^*(t)$ and the associated minimum time T^* can be found using TOPP, a complete and robust algorithm presented in [6]. From this, having also $(\dot{s}^*(t), \ddot{s}^*(t))$, the time profiles $\dot{\boldsymbol{q}}(t)$ and $\ddot{\boldsymbol{q}}(t)$ are evaluated using (3) and the needed joint torque is computed algebraically using (1).

III. Exploiting Robot Redundancy

A. First-order schemes

Let a parametrized path in the m-dimensional Cartesian (or task) space be assigned as

$$\boldsymbol{p} = \boldsymbol{p}(s), \qquad s \in [0, s_{\text{end}}], \tag{10}$$

and let $\boldsymbol{p} = \boldsymbol{k}(\boldsymbol{q})$ be task kinematics for the considered manipulator. If a $\boldsymbol{q}_0 \in \mathbb{R}^n$ is assigned as initial configuration ($\boldsymbol{q}(0) = \boldsymbol{q}_0$), it should satisfy $\boldsymbol{k}(\boldsymbol{q}_0) = \boldsymbol{p}(0) = \boldsymbol{p}_0$. Dropping

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2853806, IEEE Robotics and Automation Letters

dependencies, the first-order differential kinematics in the parameter space s is

$$\boldsymbol{p}' = \boldsymbol{J}\boldsymbol{q}', \qquad \boldsymbol{J} = \frac{\partial \boldsymbol{k}}{\partial \boldsymbol{q}}. \tag{11}$$

The simplest first-order differential inverse kinematics is given by

$$\boldsymbol{q}' = \boldsymbol{J}^{\dagger}\boldsymbol{p}', \tag{12}$$

where $\boldsymbol{J}^{\dagger}$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{J}$. A parameterized joint path $\boldsymbol{q} = \boldsymbol{q}(s)$ can be generated by numerical integration of (12), starting from $\boldsymbol{q}_0$ and evaluated over discrete samples of the parameter s, followed by interpolation of the obtained data points in the joint space with a class of smooth functions (e.g., cubic splines).

1) Projected Gradient at the Velocity level (PGV): In order to include an auxiliary optimization in the redundancy resolution step, the basic scheme (12) is modified as

$$\boldsymbol{q}' = \boldsymbol{J}^{\dagger}\boldsymbol{p}' - \delta(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})\nabla_{\boldsymbol{q}} h, \tag{13}$$

where $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ is the orthogonal projection matrix in the null space of $\boldsymbol{J}$, $\nabla_{\boldsymbol{q}} h$ is the gradient of a configuration-dependent objective function $h(\boldsymbol{q})$ to be minimized, and the scalar $\delta \geq 0$ is a suitable stepsize.

In order to improve the acceleration/deceleration capabilities of the robot end-effector along the specified Cartesian path, the following objective function was used in [10]

$$h(\boldsymbol{q}) = \boldsymbol{t}^T (\boldsymbol{M}\boldsymbol{J}^{\dagger})^T \boldsymbol{M}\boldsymbol{J}^{\dagger}\boldsymbol{t}, \tag{14}$$

with $\boldsymbol{t} \in \mathbb{R}^m$ being a unit vector along the tangent direction to the path. This will help in locally generating a joint path along which the robot will expose a reduced inertial load in the task space (minimum distance to the surface of the dynamic manipulability ellipsoid defined in [15]).

2) Weighted Pseudoinverse at the Velocity level (WPV): As an alternative to adopting a null-space projection scheme, we could use a weighted pseudoinverse at the velocity level

$$\boldsymbol{q}' = \boldsymbol{J}_{\boldsymbol{Q}}^{\dagger}\boldsymbol{p}'. \tag{15}$$

Assuming full rank for Jacobian $\boldsymbol{J}$, a regular weighted pseudoinverse in (15) takes the form

$$\boldsymbol{J}_{\boldsymbol{Q}}^{\dagger} = \boldsymbol{Q}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{Q}^{-1}\boldsymbol{J}^T)^{-1}. \tag{16}$$

Considering symmetric bounds on the joint torques, the use of a symmetric matrix $\boldsymbol{Q} = (\boldsymbol{L}^{-1}\boldsymbol{M})^T\boldsymbol{L}^{-1}\boldsymbol{M} > 0$, with a diagonal scaling matrix $\boldsymbol{L} = \mathrm{diag}\{\tau_1^{\max}, \ldots, \tau_n^{\max}\}$, was proposed in [11]. Following the pseudo-velocity (15) will limit the motion of those joints that have larger inertia-to-maximum torque ratios. A scalar parameter $\gamma \geq 0$ can be added to the weighting matrix $\boldsymbol{Q}$ as

$$\boldsymbol{Q}_{\gamma} = \exp(\gamma \ln \boldsymbol{Q}), \tag{17}$$

where $\exp(.)$ and $\ln(.)$ compute the matrix exponential and the principal matrix logarithm, respectively. For $\gamma = 0$, the simple psoudinverse solution (12) is used, while for $\gamma = 1$ the weighted pseudoinverse (16) is obtained.

B. Second-order scheme

Instead of using first-order differential inverse kinematics solutions as in (13) and (15), in the first step of our minimum-time planning problem we propose to exploit redundancy at the second-order (pseudo-acceleration) level. Differentiating (11) w.r.t. the parameter s gives

$$\boldsymbol{p}'' = \boldsymbol{J}\boldsymbol{q}'' + \boldsymbol{J}'\boldsymbol{q}', \qquad \boldsymbol{J}' = \frac{d\boldsymbol{J}}{ds}. \tag{18}$$

Using the weighting matrix in (17), the second-order differential inverse kinematics can be written as a weighted pseudoinversion with a null-space term as

$$\boldsymbol{q}'' = \boldsymbol{J}_{\boldsymbol{Q}_{\gamma}}^{\dagger}\left(\boldsymbol{p}'' - \boldsymbol{J}'\boldsymbol{q}'\right) + \left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{Q}_{\gamma}}^{\dagger}\boldsymbol{J}\right)\boldsymbol{q}_0'', \tag{19}$$

where $\boldsymbol{q}_0'' \in \mathbb{R}^n$ is a preferred pseudo-acceleration vector in the joint space. We will label this solution as *ACC*.

To determine the preferred pseudo-acceleration $\boldsymbol{q}_0''$, similar techniques as those introduced in [10] and [16] will be used. For the general case of different bounds on the joint torque components, it is useful to use a normalization. Assume that $\dot{\boldsymbol{q}} = \boldsymbol{0}$, so that velocity-dependent terms vanish, and that gravitational terms are neglected in (1). Using $\boldsymbol{Q}_{\gamma}$ as weighting matrix in the pseudoinverse, the normalized joint torques $\tilde{\boldsymbol{\tau}}$ in the time domain can be written as

$$\tilde{\boldsymbol{\tau}} = \boldsymbol{L}^{-1}\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{L}^{-1}\boldsymbol{M}\boldsymbol{J}_{\boldsymbol{Q}_{\gamma}}^{\dagger}\ddot{\boldsymbol{p}}. \tag{20}$$

Consequently, the associated dynamic manipulability ellipsoid in the Cartesian space will be

$$\ddot{\boldsymbol{p}}^T {\boldsymbol{J}^{\dagger}}_{\boldsymbol{Q}_{\gamma}}^T \boldsymbol{Q}\, \boldsymbol{J}_{\boldsymbol{Q}_{\gamma}}^{\dagger}\ddot{\boldsymbol{p}} \leq 1. \tag{21}$$

To improve the acceleration/deceleration capabilities of the robot end effector along the Cartesian path, it is useful to minimize the quantity

$$f(\boldsymbol{q}) = \boldsymbol{t}^T {\boldsymbol{J}^{\dagger}}_{\boldsymbol{Q}_{\gamma}}^T \boldsymbol{Q}\, \boldsymbol{J}_{\boldsymbol{Q}_{\gamma}}^{\dagger}\boldsymbol{t}, \tag{22}$$

with $\boldsymbol{t} \in \mathbb{R}^m$ defined as in (14). The preferred vector $\boldsymbol{q}_0''$ in (19) is then chosen as

$$\boldsymbol{q}_0'' = -\delta_1 \nabla_{\boldsymbol{q}} f - \boldsymbol{D}\boldsymbol{q}', \tag{23}$$

where $\delta_1 \geq 0$ is a scalar gain and $\boldsymbol{D}$ is a $n \times n$ diagonal, positive semi-definite matrix. The second term in (23) is a damping term on the pseudo-velocity, which guarantees that bounded displacements are generated in the joint space. This property is similar to the known effect of null-space damping in the time domain, needed for stabilizing and smoothing joint trajectories when redundancy is resolved at the acceleration level, see, e.g. [17].

In the present framework, the choice of both δ_1 and $\boldsymbol{D}$ turns out to be critical in determining the total length of the generated joint path, and thus indirectly also the achievable minimum time associated to the path. Intuitively, a too small damping matrix $\boldsymbol{D}$ (or no damping at all) will lead to a potential drift or wandering of the joint path associated to the original Cartesian path. Conversely, if the damping action is too strong, joint reconfigurations intended to optimize the objective function (22) will be penalized. Similarly, the choice of δ_1 should balance the length of the path generated in the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2853806, IEEE Robotics and Automation Letters

joint space vs. the efficacy in the auxiliary optimization of $f(\boldsymbol{q})$. For handling this trade off, the following bounds are imposed to δ_1:

$$0 \leq \delta_1 \leq \min\left\{ \delta_2 \frac{\|\boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}(\boldsymbol{p}'' - \boldsymbol{J}'\boldsymbol{q}')\|}{\|(\boldsymbol{I} - \boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}\boldsymbol{J})\nabla_{\boldsymbol{q}} f\|}, \delta_{1,\max} \right\}, \quad (24)$$

with $\delta_2 \in [0, 1]$. Finally, in order to avoid numerical drifts during calculations, a stabilizing PD term on the (spacial) task error can be added to (19), obtaining

$$\boldsymbol{q}'' = \boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}\big(\boldsymbol{p}'' - \boldsymbol{J}'\boldsymbol{q}' + \boldsymbol{K}_d\boldsymbol{e}' + \boldsymbol{K}_p\boldsymbol{e}\big) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}\boldsymbol{J})\,\boldsymbol{q}''_0, \quad (25)$$

where $\boldsymbol{K}_p > 0$ and $\boldsymbol{K}_d > 0$ are diagonal gain matrices, $\boldsymbol{e}(s) = \boldsymbol{p}(s) - \boldsymbol{k}(\boldsymbol{q}(s))$ and $\boldsymbol{e}'(s) = \boldsymbol{p}'(s) - \boldsymbol{J}(\boldsymbol{q}(s))\boldsymbol{q}'(s)$.

With the second-order *ACC* method, a parameterized path $\boldsymbol{q} = \boldsymbol{q}(s)$ in the joint space will be generated by double numerical integration of (25), used together with (23) and (22). The second step of the minimum-time planning procedure is identical to that of first-order schemes, e.g., *PGV* or *WPV*.

C. Finding an initial configuration

To start a first- or a second-order redundancy resolution scheme, either a consistent initial configuration $\boldsymbol{q}(0) = \boldsymbol{q}_0$ is assigned, or it should be determined so as to match the end-effector path at the start, i.e., $\boldsymbol{k}(\boldsymbol{q}(0)) = \boldsymbol{p}(0) = \boldsymbol{p}_0$. Indeed, being the robot redundant, there is an infinite number of such initial robot configurations. To find the most efficient $\boldsymbol{q}_0$ for our motion task, a preliminary kinematic control scheme is used in the *time domain*, similar to (25) in the *space domain*,

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}(-\dot{\boldsymbol{J}}\dot{\boldsymbol{q}} - \boldsymbol{K}_d\dot{\boldsymbol{p}} + \boldsymbol{K}_p\boldsymbol{e}_0) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}\boldsymbol{J})\ddot{\boldsymbol{q}}_0, \quad (26)$$

with $\ddot{\boldsymbol{q}}_0 = -\delta_1\nabla_{\boldsymbol{q}} f - \boldsymbol{D}\dot{\boldsymbol{q}}$, $f(\boldsymbol{q})$ computed as in (22), $\boldsymbol{e}_0 = \boldsymbol{p}_0 - \boldsymbol{k}(\boldsymbol{q})$, and $\dot{\boldsymbol{p}} = \boldsymbol{J}\dot{\boldsymbol{q}}$. In this preliminary phase, the robot may start from any configuration, but typically one still corresponding to the initial point of the Cartesian path (i.e., with $\|\boldsymbol{e}_0\| = 0$). Equation (26) is then integrated forward until $\|\dot{\boldsymbol{q}}\| < \epsilon$, a specific threshold set, e.g., to 10^{-3}. The robot joints will move mainly according to the null-space projection term in (26), while the first term is used to keep the robot end-effector in the initial Cartesian position.

For the second-order scheme (25), we need in addition a suitable initial pseudo-velocity $\boldsymbol{q}'(0) = \boldsymbol{q}'_0$. Using the available $\boldsymbol{q}_0$, this can be computed as

$$\boldsymbol{q}'(0) = \boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}(\boldsymbol{q}_0)\boldsymbol{p}'(0) - \delta\left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}_{\boldsymbol{Q}_\gamma}(\boldsymbol{q}_0)\boldsymbol{J}(\boldsymbol{q}_0)\right)\nabla_{\boldsymbol{q}} f(\boldsymbol{q}_0). \quad (27)$$

The overall computational scheme for our proposed method is shown in Fig. 1.

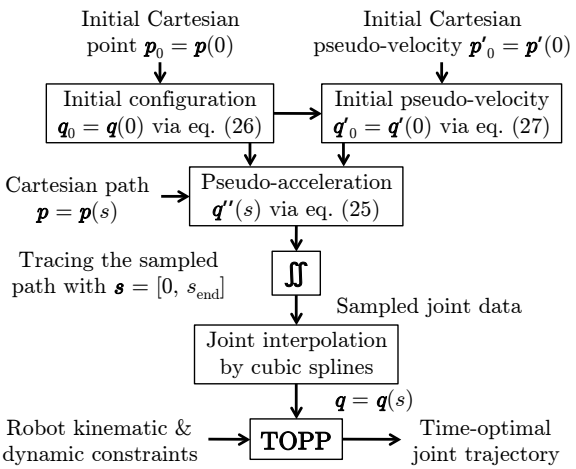


Fig. 1: The overall scheme for the proposed *ACC* solution.

IV. Simulation results for a 3R planar arm

The proposed solution *ACC* in (19) and the methods *WPV* in (15) and *PGV* in (13) have been implemented in MATLAB for a thorough comparison of results via simulation. For this, we considered a 3R planar arm ($n = 3$) with links of equal length $l = 0.5$ m, uniformly distributed mass $m_l = 1$ kg, and moment of inertia $I_l = m_l l^2/12$. The end-effector position ($m = 2$) should follow a path on the horizontal plane, so that the degree of redundancy is $n - m = 1$. Torque and velocity limits have been set respectively to ± 20 Nm and ± 10 rad/s, for all three joints.

In order to compare the obtained results with a global time-optimal solution on a Cartesian path, we followed the procedure in Fig. 2. We first formulate a point-to-point (PTP) minimum time problem for a given initial configuration $\boldsymbol{q}_0$, a desired final end-effector position $\boldsymbol{p}_f$, zero initial/final joint velocities ($\dot{\boldsymbol{q}}_0 = \dot{\boldsymbol{q}}_f = \boldsymbol{0}$), and robot dynamic and kinematic limits (4) and (8). The resulting nonlinear programming (NLP) problem is solved by a numerical method based on direct collocation [17], yielding a joint trajectory $\boldsymbol{q}^*(t)$ and global minimum time T^*. Next, we associate to this motion the resulting Cartesian path, suitably expressed in a parametrized form $\boldsymbol{p}(s)$. Finally, this will be the input to two-step methods that handle robot redundancy. They will generate solution trajectories $\boldsymbol{q}^*_{\text{method}}(t)$ and associated motion times T^*_{method}, with method = {*ACC*, *WPV*, *PGV*}, and the results can be compared to each other and to the global minimum time solution on the same Cartesian path.

As a matter of fact, this fair procedure is needed since it is still prohibitive in general to address by numerical methods the global minimum time problem for redundant robots along

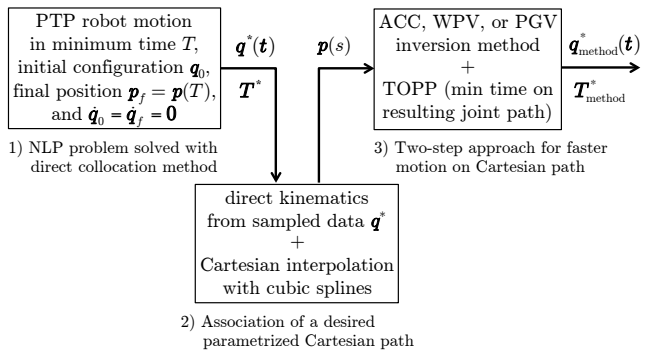


Fig. 2: Comparison procedure between the global time-optimal solution and the solutions obtained on the same Cartesian path with two-step methods for redundant robots.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2853806, IEEE Robotics and Automation Letters

TABLE I: Motion tasks for the 3R planar arm.

	Initial configuration $\boldsymbol{q}_0$ [rad]	Final Cartesian position $\boldsymbol{p}_f$ [m]
Task 1	$(3\pi/8, \pi/4, -\pi/4)^T$	(-0.4,0.8)
Task 2	$(\pi/4, \pi/4, \pi/4)^T$	(-0.4,0.4)
Task 3	$(\pi/4, \pi/4, \pi/4)^T$	(0,-1)

TABLE II: Minimum motion times [s] for the 3R planar arm, using global optimization and two-step solution methods (the best parameters used for each method are indicated).

	Direct collocation	*ACC* $\{\gamma, \delta_2, \delta_{1,\max}, \boldsymbol{D}\}$	*WPV* $\{\gamma\}$	*PGV* $\{\delta\}$
Task 1	0.2499	0.2689 $\{0.655, 1, 210, 2\boldsymbol{I}\}$	0.3357 $\{0.5\}$	0.4394 $\{0\}$
Task 2	0.2699	0.3021 $\{1, 0.8, 3500, 160\boldsymbol{I}\}$	0.3100 $\{1\}$	0.3867 $\{0.1\}$
Task 3	0.4415	0.5332 $\{0, 0.6, 300, 16\boldsymbol{I}\}$	0.5513 $\{0.1\}$	0.5550 $\{1\}$

predefined Cartesian paths (i.e., providing directly $\boldsymbol{p}(s)$ as input to the PTP problem). Indeed, two-step methods can only return longer motion times than the global optimal time. On the other hand, two-step approaches are computationally more efficient and accept any Cartesian path to start with.

We considered three different motion tasks with the boundary conditions specified in Tab. I. The global PTP minimum time optimization problem is solved using the direct collocation method in the trajectory optimization library *OptimTraj* [18]. For illustration, the resulting globally optimal joint velocity and torque profiles for the first motion task are shown in Fig. 3.

In order to apply a two-step solution method in the presence of redundancy, we follow the procedure in Fig. 2: from the sequence of robot configurations in the time-optimal trajectory $\boldsymbol{q}^*(t)$, a corresponding sequence of end-effector positions is computed via the direct kinematics of the 3R robot arm, and then interpolated in the Cartesian space using cubic splines. In the first step, this parametrized path is input to the *ACC*, *PGV*, and *WPV* methods, which generate different joint paths, sampled every $\Delta s = 0.001$ and with $s_{\text{end}} = 1$, that are associated to the same Cartesian path. For better accuracy, all the available joint configurations samples are used within the cubic splines interpolation. In the second (and common) step, the time-optimal motion is obtained on each joint path using TOPP [6]. The tuning of parameters is done separately for each method, so as to achieve the best possible performance for each task. Specific ranges are chosen for each parameter and the best values are searched on a discretized grid by evaluating a very large number of simulations. In the *ACC* solution, the γ and $\boldsymbol{D}$ parameters are more influential than δ_2 and $\delta_{1,max}$. For efficiency, the first two parameters are tuned together, and then kept fixed to tune the latter ones.

Table II reports the comparative results obtained on the three motion tasks of Tab. I, together with the parameters used for each method/task. The proposed second-order solution *ACC* returns the fastest motion time among the three methods, i.e., also the closest to the global optimal solution. The resulting minimum time for the first motion task is only 7.6% longer than the global optimal solution. On the three tasks, the average increase of the motion time for the *ACC*, *WPV*, and *PGV* methods is, respectively, 13.4%, 24.7%, and 48.2% with respect to the global optimal solution. Figure 4 shows the joint velocities and torques obtained using the *ACC* method for the first motion task: at every instant, at least one joint is saturated either to its torque or velocity limit.

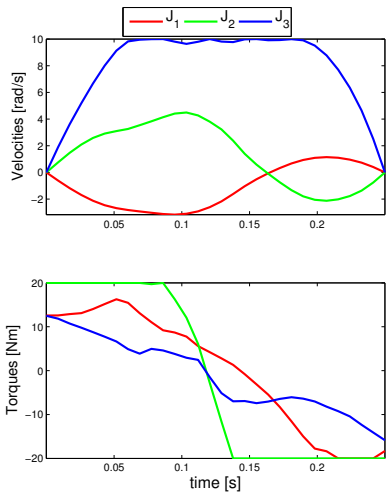


Fig. 3: Global time-optimal joint velocities and torques for the 3R planar arm on the first motion task.

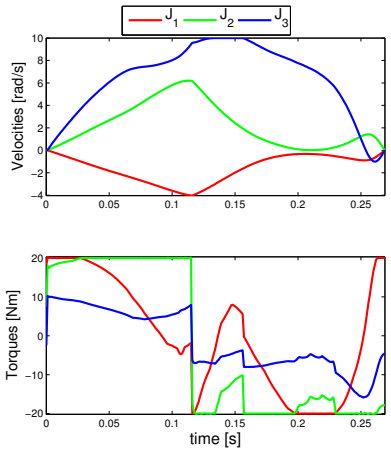


Fig. 4: Joint velocities and torques for the 3R planar arm with the *ACC* solution on the first motion task.

Finally, Figure 5 shows stroboscopic views of the best solutions found for each method on the first motion task, and the associated evolutions in the phase plane $(s, \dot{s})$. Each solution produces in fact a different path in the joint space, which leads also to different maximum velocity curves and associated optimal trajectories. Although the MVC curves of the *PGV* method are higher than those of *WPV*, allowing in principle larger pseudo-velocities and thus a faster motion, this feature is not exploited efficiently and the optimal trajectory remains far from these curves in the intermediate range of s values. Instead, the *ACC* solution leads to the highest MVC curves, and the optimal trajectory is able to cover most of the underlying area. For further comparison, the joint path

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2853806, IEEE Robotics and Automation Letters

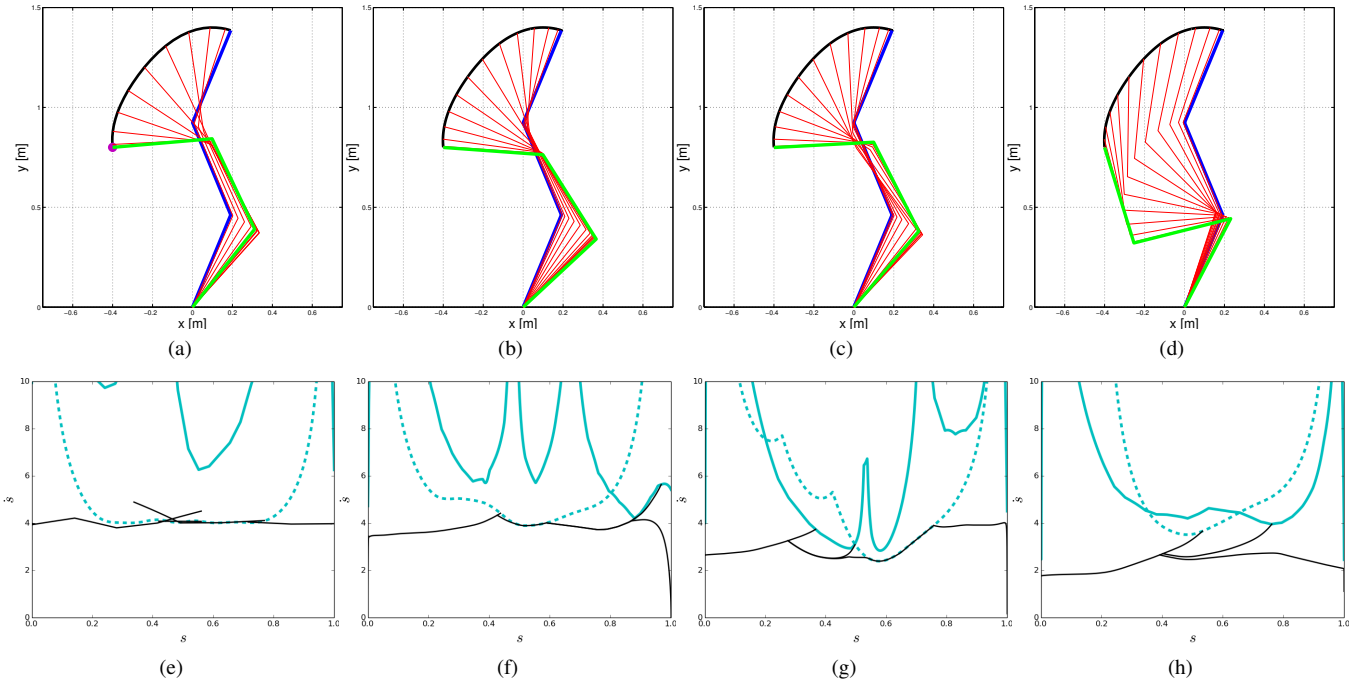


Fig. 5: Stroboscopic motion of the 3R planar arm [top] and optimal trajectories in the phase plane [bottom] using different solutions for the first motion task in Tab. I: (a-e) with direct collocation method; (b-f) with the proposed *ACC*; (c-g) with *WPV*; (d-h) with *PGV*. The initial configuration $\boldsymbol{q}_0$ (in blue) is always the same. The purple point in (a) is the final position $\boldsymbol{p}_f$ used to generate the Cartesian path from the PTP optimization. The resulting Cartesian path $\boldsymbol{p}(s)$ (in black) is used then by all two-step methods, while the obtained final configurations are shown in green. In (e,f,g,h), the cyan lines are the maximum velocity curves MVC_t (solid) and MVC_v (dotted), while the black lines are the obtained time-optimal profiles.

corresponding to the global time-optimal solution (obtained with the direct collocation method) were also fed into the TOPP algorithm. As expected, the resulting minimum time using TOPP is exactly equal to the optimal time obtained from the direct collocation method. The phase-plane plot in Fig. 5(e) clearly shows how the optimal trajectory fully exploits the area below the MVC_v curve.

V. Experimental results for a KUKA LWR IV

As a second case study, we have considered a 7R KUKA LWR lightweight robot and compared the different inverse differential methods using two Cartesian tasks defined for the position of the end-effector flange center ($m = 3$). Since the rotation of joint 7 has no effect on it, the final flange was frozen resulting in only $n = 6$ active joints, with a redundancy degree $n - m = 3$. All computations were done using the dynamic model identified in [19]. The (symmetric) limits on joint velocities and torques are set to:

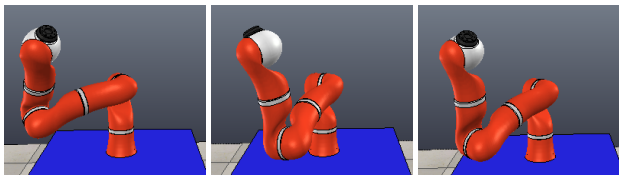
$$\begin{aligned} \dot{\boldsymbol{q}}^{\max} &\rightleftharpoons \pm(1.92\ 1.92\ 2.23\ 2.23\ 3.56\ 3.21)\ \text{[rad/s]}, \\ \boldsymbol{\tau}^{\max} &\rightleftharpoons \pm(176\ 176\ 100\ 100\ 100\ 38)\ \text{[Nm]}. \end{aligned} \tag{28}$$

The first motion task was a linear path of length 0.66 m. To study the influence of the initial configuration $\boldsymbol{q}_0$ on the optimal solution, the *ACC* method was applied starting from the three different configurations given in Tab. III, the first one obtained using (26) and the other two chosen randomly, all associated to the same initial position $\boldsymbol{p}_0 = (-0.4,\ 0.25,\ 0.3)$ [m]. The solution obtained when starting with the configuration (a) provided a reduction of the minimum time by 8.5-10.5%.

TABLE III: Minimum motion times along a linear path for the KUKA LWR robot using the *ACC* method from three different initial configurations (with V-REP views).

Initial configuration $\boldsymbol{q}_0$ [rad]	T^*_{ACC} [s]
$\boldsymbol{q}_0^{(a)} = (-1.12\ \ 1.80\ \ -0.55\ \ 1.71\ \ 2.43\ \ 0.29)^T$	**0.4743**
$\boldsymbol{q}_0^{(b)} = (-0.34\ \ 1.94\ \ 0.16\ \ 1.71\ \ 1.20\ \ -0.43)^T$	0.5144
$\boldsymbol{q}_0^{(c)} = (-0.72\ \ 1.94\ \ -0.08\ \ 1.73\ \ 1.60\ \ 0.51)^T$	0.5242

(a) (b) (c)



When executing the linear motion task from the initial configuration (a) in Tab. III using the two-step methods, the proposed *ACC* solution leads to the fastest motion time, with an improvement of 22.6% and 31.7% over the *WPV* and *PGV* solutions. We considered a second motion task along an ellipse in the 3D space, with major and minor axes $r_M = 0.2$ and $r_m = 0.1$ [m], starting the robot at rest from the configuration $\boldsymbol{q}_0 = (1.15\ \ -0.54\ \ 0.10\ \ 1.47\ \ -0.30\ \ 0.76)^T$ [rad], which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2853806, IEEE Robotics and Automation Letters

corresponds to $\boldsymbol{p}_0 = (0.2, 0.6, 0.2)$ [m]. Again, the *ACC* solution provided the best result, with a reduction of the motion time by 14.6% over the other two methods. These results are summarized in Tab. IV, which reports also the (best) set of parameters used for each method/task.

TABLE IV: Minimum motion times for the KUKA LWR robot using different two-step solution methods.

Method	Task	T^* [s]
PGV ($\delta = 0.3$)	linear	0.6901
WPV ($\gamma = 1$)	linear	0.6127
ACC ($\gamma = 0.5$, $\delta_2 = 0.4$, $\delta_{1,\max} = 100$, $\boldsymbol{D} = 5\boldsymbol{I}$)	linear	**0.4743**
PGV ($\delta = 0.6$)	ellipse	1.2
WPV ($\gamma = 0.35$)	ellipse	1.2
ACC ($\gamma = 0.5$, $\delta_2 = 0.4$, $\delta_{1,\max} = 100$, $\boldsymbol{D} = 5\boldsymbol{I}$)	ellipse	**1.0245**

In view of the good results obtained with the *ACC* method, the two motion tasks on the linear and the elliptic path were implemented in experiments on the KUKA LWR IV robot using the FRI library [20] in position control mode. Due to residual uncertainty in the robot dynamic model, the *ACC* solution was re-generated in a conservative way, using only 95% of the maximum available nominal torques and joint velocities. The new motion times were 0.496 s for the linear task and 1.081 s for the ellipse task (compare with Tab. IV).

Figure 6 shows the phase plane solution obtained for ellipse task using the *ACC* method. The time-optimal trajectory matches the curve MVC_v along the entire path, following the torque-related bounds specified by $\alpha(s, \dot{s})$ and $\beta(s, \dot{s})$ in (6) just at the beginning and toward the end of the path. In fact, the MVC_v curve is much lower than the MVC_t curve, and the robot reaches its velocity limits very quickly because of its large torque/acceleration capabilities. Figure 7 shows the experimental joint velocities and torques *normalized* with respect their nominal values in (28). The minimum time planned torque (in red) of the second joint saturates at the start, near the middle, and toward the end of the trajectory. In the rest of the trajectory, the second and fourth joint velocities saturate in turn, consistently with the plot in Fig. 6.

During task execution with the KUKA LWR, the torques are measured by the available joint torque sensors. The differences between planned and executed/measured torques in Fig. 7 are due to unmodeled dynamics (motor friction, joint elasticity) neglected in the optimization, measurement noise (encoders and torque sensors), as well as non-idealities of the low-level robot controller. Because of the latter, the joint torques cannot follow perfectly the planned discontinuities of the optimal torques at the switching points.

For the linear task, the Cartesian error norm using different two-step solutions is shown in Fig. 8. The largest peak error is obtained with the *ACC* method, which is also the one with the fastest motion time. On the other hand, the tracking error vanishes as motion comes to an end, whereas some residual error is left with the other two methods. A trade-off between faster motion times and better tracking performance can be

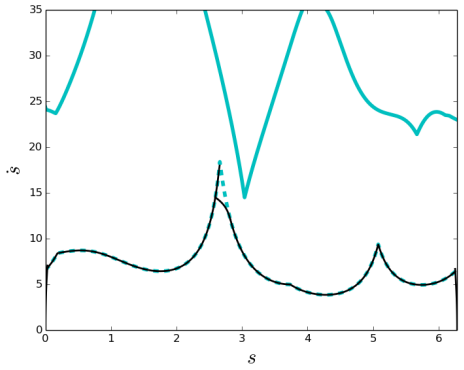


Fig. 6: Optimal phase-plane trajectory for the KUKA LWR robot tracing an ellipse path using the *ACC* method.

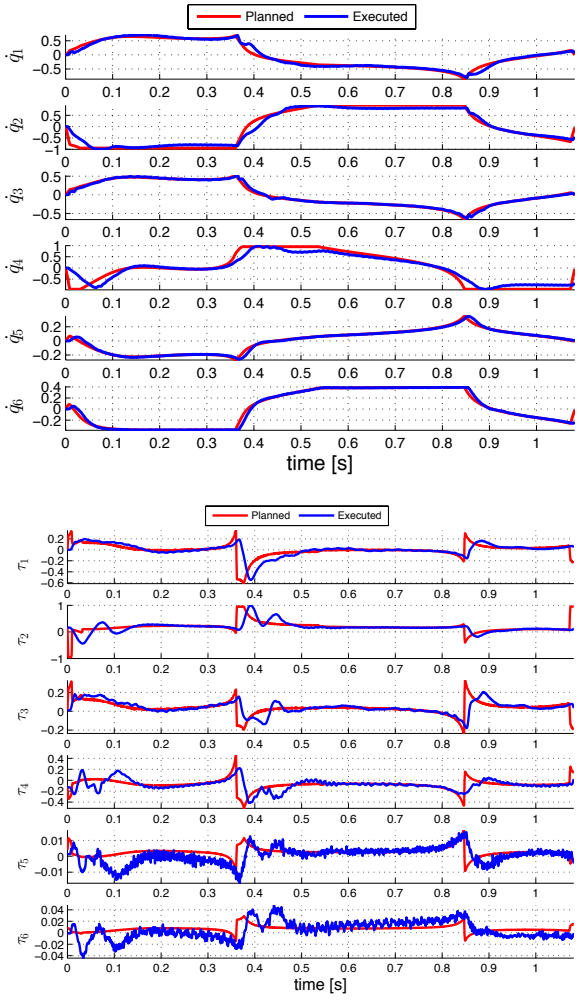


Fig. 7: Normalized joint velocities [top] and normalized torques [bottom] in the minimum time experiment with the KUKA LWR robot on an ellipse path using the *ACC* method.

achieved by downrating the maximum available nominal joint torque in (28). From Fig. 8 and Tab. V, using the *ACC* solution with only 35% of $\boldsymbol{\tau}^{\max}$ returns better optimal time and less mean Cartesian error norm than using the other two solutions with 95% of $\boldsymbol{\tau}^{\max}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2853806, IEEE Robotics and Automation Letters

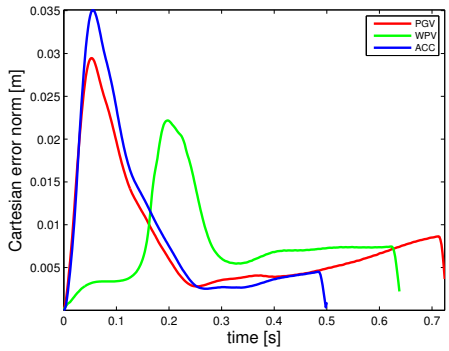


Fig. 8: Cartesian error norm for the KUKA LWR robot tracing a linear path using different two-step solutions with 95% of the maximum available nominal torques and joint velocities.

TABLE V: Mean Cartesian error norm for the KUKA LWR robot tracing a linear path using the *ACC* method with 95% of $\dot{\boldsymbol{q}}^{\max}$ and different $\boldsymbol{\tau}^{\max}$ percentages.

% $\boldsymbol{\tau}^{\max}$	T^*[s]	Mean Cartesian error norm [m]
35	0.556	0.0068
45	0.530	0.0072
55	0.518	0.0075
65	0.510	0.0080
75	0.505	0.0087
85	0.501	0.0094
95	0.496	0.0095

VI. Conclusions

We presented a two-step method that addresses in an approximate but effective way the minimum time control problem for redundant robots moving along a given Cartesian path. In a first step, a local second-order inverse kinematic method was used to map Cartesian paths into joint paths, while in a second step an established minimum time planning algorithm provides the optimal solution under joint velocity and torque bounds. As ingredients in our method, we used weighted pseudoinversion, optimized an inertia-related criterion, and included a damping term in the null-space of the task Jacobian. Working at the second-order level allows obtaining smoother paths while including dynamic issues.

Based on the extensive tests on various paths and for different robots, which are reported only in part here, we have found consistent improvements in the obtained motion times over similar approaches that use first-order inverse solutions at the velocity level. As shown experimentally, the combination of our second-order solution method with the TOPP algorithm leads to reasonable performance in tracking minimum time trajectories. A trade-off between faster motion times and better tracking performance can be achieved by including additional constraints, such as torque rate bounds that eliminate critical discontinuities in the solution [21].

Our two-step second-order method leads to faster motion times, but is still intended currently for off-line planning situations only. Real-time limitations are distributed between both steps, and depend on the length of the original Cartesian path, the path parameter sampling, the number of joints, and the complexity of the used robot dynamics, leading to running times in the order of seconds. On the other hand, finding the accurate global minimum time with a constrained solution trajectory by means of general numerical optimization techniques requires at present minutes to hours of computation. We plan to pursue computationally more efficient implementations of the present method, as well as other semi-global methods that can run in real time, such as model predictive control along Cartesian paths for redundant robots that minimizes the motion time to go.

References

[1] J. E. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

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

[3] J.-J. Slotine and Y. Hyun, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 1, pp. 118–124, 1989.

[4] Z. Shiller, "On singular time-optimal control along specified paths," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 4, pp. 561–566, 1994.

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

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

[7] M. Galicki, "Time-optimal controls of kinematically redundant manipulators with geometric constraints," *IEEE Trans. on Robotics and Automation*, vol. 16, no. 1, pp. 89–93, 2000.

[8] M. Shugen and M. Watanabe, "Time optimal path-tracking control of kinematically redundant manipulators," *JSME Int. J. Ser. C Mechanical Systems, Machine Elements and Manufacturing*, vol. 47, no. 2, pp. 582–590, 2004.

[9] A. Reiter, A. Müller, and H. Gattringer, "Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators," in *Proc. 42nd Ann. Conf. IEEE Industrial Electronics Soc.*, 2016, pp. 6873–6878.

[10] P. Chiacchio, "Exploiting redundancy in minimum-time path following robot control," in *Proc. American Control Conf.*, 1990, pp. 2313–2318.

[11] F. Basile and P. Chiacchio, "A contribution to minimum-time task-space path-following problem for redundant manipulators," *Robotica*, vol. 21, no. 2, pp. 137–142, 2003.

[12] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[13] A. De Luca and L. Ferrajoli, "Exploiting robot redundancy in collision detection and reaction," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 3299–3305.

[14] L. Zlajpah, "On time optimal path control of manipulators with bounded joint velocities and torques," in *Proc. IEEE Int. on Robotics and Automation*, 1996, pp. 1572–1577.

[15] T. Yoshikawa, "Dynamic manipulability of robot manipulators," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1985, pp. 1033–1038.

[16] P. Chiacchio and M. Concilio, "The dynamic manipulability ellipsoid for redundant manipulators," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 95–100.

[17] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[18] M. Kelly. (2016) OptimTraj: Trajectory optimization library for matlab. [Online]. Available: https://github.com/MatthewPeterKelly/OptimTraj

[19] C. Gaz, F. Flacco, and A. De Luca, "Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 1386–1392.

[20] *KUKA.FastResearchInterface 1.0*, KUKA System Technology (KST), D-86165 Augsburg, Germany, 2011, version 2.

[21] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of robotic systems*, vol. 17, no. 5, pp. 233–249, 2000.