Evaluation of Efficiently Generating Fast Robot Trajectories Under Geometric and System Constraints

Evaluation of Efficiently Generating Fast Robot Trajectories Under Geometric and System Constraints ⋆

Martin Böck * **Manuel Plainer** * **Andreas Kugi** *

* *Automation and Control Institute, TU Wien*
Gußhausstraße 27–29, 1040 Vienna, Austria
(e-mail: {boeck,kugi}@acin.tuwien.ac.at, manuel.plainer@gmx.at).

Abstract: This paper investigates and compares some approaches to trajectory generation for an articulated robot with six degrees of freedom. The trajectory to be planned consists of geometrically predetermined segments and phases where the geometry is free. The overall goal is to minimize the time needed for traversing the whole trajectory from a specified start to a terminal configuration. Constraints for the joint angles, velocities, accelerations, and jerks as well as for the joint torques of the robot are taken into account. The optimal solution of this task is calculated as reference. However, this requires considerable computing time. In view of this fact, different parameterizations are investigated. The resulting trajectories require slightly more time for traversing than the optimal one, but they can be calculated more efficiently. Based on four different exemplary trajectories, the proposed approaches are tested and compared on an industrial robot.

Keywords: Constraints, inverse kinematic problem, minimum-time control, robot dynamics, robot kinematics, robotic manipulators, trajectory planning.

1. INTRODUCTION

Trajectory generation is an important task for robot operation. It is understood here to comprise two subtasks. Firstly, a geometric curve for a specific point of the robot has to be found according to the overall motion specification. Subsequently, such a geometric curve without temporal information is referred to as path. Secondly, a time parameterization has to be determined turning the path into a trajectory. Trajectory generation has often been treated in literature and plenty of solutions are available. Many different types of robots have been considered and various tasks with offline as well as online trajectory generation have been investigated. For surveys on trajectory generation for robots, see, e.g., Siciliano and Khatib (2008); Laumond (1998); LaValle (2006); Latombe (1991); Biagiotti and Melchiorri (2008).

In this paper, we focus on trajectory generation for an articulated robot with six degrees of freedom (DOFs). The time-dependent trajectory to be found describes the motion of the tool center point (TCP) of the robot. It is supposed to connect two points of rest with prescribed position and orientation and take into account constraints. The considered trajectories consist of so-called point-to-point (PTP) and continuous path (CP) segments and are subsequently referred to as PTPCP trajectories. The PTP segments are geometrically not predetermined, i.e., the geometry of the curve connecting the boundary points is left as a DOF. On the contrary, the CP segments are specified as paths. The geometry of the PTP segments and the time parameterization of the whole trajectory have to be determined such that the PTPCP trajectory is traversed in a time-optimal or near time-optimal fashion. For the PTP segments, the boundary conditions typically concern position and orientation as well as the corresponding derivatives with respect to time to, e.g., connect to the CP parts.

Many of the existing approaches for solving such a task of trajectory generation can be divided into two classes, see, e.g., Pardo-Castellote and Cannon Jr. (1996). On the one hand, a so-called decoupled approach can be utilized, see, e.g., Bobrow et al. (1985). This approach consists of determining the geometry of the path first. Afterwards, the time parameterization of the path parameter is found. On the other hand, the so-called coupled approach can be used, see, e.g., Shiller and Dubowsky (1991). The latter is also employed in this paper. The geometry of the PTP parts and the time parameterization of the whole trajectory are found at once. While this obviously results in an increased computational complexity compared to the decoupled approach, the time-optimal solution can actually be found.

The aim of this paper is to evaluate and compare different methods for solving the described task of trajectory generation with respect to different aspects. To this end, four exemplary PTPCP trajectories are considered. Amongst others, the computing time and reliability are investigated in more detail. In particular, they can be important for online planning or replanning. For the sake of validation, the calculated trajectories are also tested at an industrial robot.

⋆ The authors are grateful to STIWA Automation GmbH for financial and technical support.

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586

On the one hand, an optimal control problem (OCP) is formulated for obtaining the time-optimal trajectory as reference satisfying all boundary conditions and constraints. This OCP is solved with a ready-to-use software package. On the other hand, certain parameterizations are used beforehand for transforming the problem of trajectory generation into a static optimization problem. Similar approaches are pursued, e.g., in Messner et al. (2013); Verscheure et al. (2009).

Concerning the constraints, not only such for the motor torques and joint angles are considered but also for the first, second, and third time-derivatives of the joint angles (joint velocity, acceleration, and jerk). Existing algorithms do often not account for all these constraints at once, see, e.g., Shin and McKay (1985). The incorporation of the constraints for the torques and the jerk enables to fully utilize the capabilities of the robot for fast movements and vibrations are reduced, cf. Messner et al. (2013).

For the static optimization problem, the analytical gradients and Jacobian matrices are calculated. The strict incorporation of model knowledge, e.g., the forward and inverse kinematics and the dynamics of the considered six DOFs robot, into the optimization has the positive effect that no approximation, e.g., via finite differences, is necessary.

Further details about the considered robot and the specific task of trajectory generation are given in Section 2. The different methods for the generation of fast PTPCP trajectories are presented in Section 3. These methods form the basis for the evaluation carried out in Section 4.

2. CONSIDERED ROBOT AND TASK

The investigations presented in this paper are based on the robot Stäubli TX60L shown in Fig. 1. This robot has six DOFs given by the joint angles q_i, $i = 1, \ldots, 6$ which are collected in the vector $\boldsymbol{q} \in \mathbb{R}^6$. The robot can be equipped with an end effector with the reference point given by the TCP.

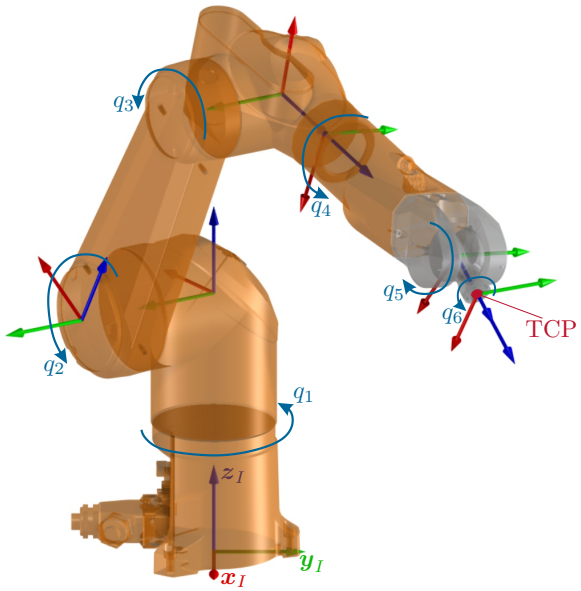

Fig. 1. Six axes articulated robot Stäubli TX60L.

The forward kinematics, i.e., the calculation of the position and orientation of the TCP in the inertial Cartesian coordinate system with basis vectors $(\boldsymbol{x}_I, \boldsymbol{y}_I, \boldsymbol{z}_I)$ depending on the values of the joint angles $\boldsymbol{q}$, can be calculated in a straightforward way. Therefore, it is omitted here for brevity. The resulting position of the TCP is denoted as ${}_I\boldsymbol{r}_{\mathrm{TCP}}$ and $\boldsymbol{R}_{\mathrm{TCP}}$ constitutes the rotation matrix describing the orientation of the end effector with respect to the inertial frame.

The corresponding inverse kinematics are presented in Section 2.1. The mathematical model of the robot used in the remainder of this paper is shortly outlined in Section 2.2 and Section 2.3 provides a detailed description of the considered task of trajectory generation.

2.1 Inverse Kinematics

For the considered robot, a closed-form solution of the inverse kinematics is available since the axes of the consecutive revolute joints with angles q_4, q_5, and q_6 intersect at a single point, see Siciliano et al. (2009). Despite this nice property, ambiguities (e.g., the elbow joint might point upward or downward for the same position of the TCP) and singular configurations (e.g., the axes of the joints with angles q_4 and q_6 are collinear) still have to be appropriately treated. Without going into further details, the result for the inverse kinematics can be stated as

$$\boldsymbol{q} = \boldsymbol{\Phi}\left({}_I\boldsymbol{r}_{\mathrm{TCP}}, \boldsymbol{R}_{\mathrm{TCP}}\right). \tag{1}$$

In view of the ambiguities, one particular configuration of the robot is chosen in the following for obtaining (1).

Besides (1), it is further necessary for the upcoming investigations to determine the joint velocities, accelerations, and jerks from those belonging to the end effector. To this end, the vector

$$\dot{\boldsymbol{z}} = \left[{}_I\boldsymbol{v}_{\mathrm{TCP}}{}^{\mathrm{T}} \;\; {}_I\boldsymbol{\omega}_{\mathrm{TCP}}{}^{\mathrm{T}}\right]^{\mathrm{T}} \tag{2}$$

containing the velocity and angular velocity of the end effector in the inertial Cartesian coordinate system is introduced, see, e.g., Siciliano et al. (2009); Corke (2013). As $\dot{\boldsymbol{z}}$ is linear in $\dot{\boldsymbol{q}}$ and under a slight abuse of notation it can be written as

$$\dot{\boldsymbol{z}} = \underbrace{\frac{\partial \dot{\boldsymbol{z}}}{\partial \dot{\boldsymbol{q}}}}_{\overline{\boldsymbol{J}}(\boldsymbol{q})} \dot{\boldsymbol{q}} \tag{3}$$

with the manipulator Jacobian $\overline{\boldsymbol{J}}$. Hence, at nonsingular configurations it holds that

$$\dot{\boldsymbol{q}} = \underbrace{\overline{\boldsymbol{J}}^{-1}(\boldsymbol{q})}_{\boldsymbol{J}(\boldsymbol{q})} \dot{\boldsymbol{z}}. \tag{4a}$$

Similarly, the quantities $\ddot{\boldsymbol{q}}$ and $\boldsymbol{q}^{(3)}$ can be calculated as

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\left(\ddot{\boldsymbol{z}} - \dot{\overline{\boldsymbol{J}}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}\right) \tag{4b}$$

$$\boldsymbol{q}^{(3)} = \boldsymbol{J}(\boldsymbol{q})\left(\boldsymbol{z}^{(3)} - \ddot{\overline{\boldsymbol{J}}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} - 2\dot{\overline{\boldsymbol{J}}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\ddot{\boldsymbol{q}}\right). \tag{4c}$$

2.2 Mathematical Model of the Robot

For the Stäubli TX60L, the equations of motion are derived by means of the projection equation (see Bremer (2008)) yielding

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586

$$\boldsymbol{\Theta}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \boldsymbol{D}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{B}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}. \quad (5)$$

The vector $\boldsymbol{\tau} \in \mathbb{R}^6$ contains the joint torques, $\boldsymbol{D}$ constitutes the inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}$ is the vector of centripetal and Coriolis terms, and $\boldsymbol{B}$ and $\boldsymbol{g}$ comprise the friction and gravitational terms, resp. Naturally, the joint angles, their derivatives, and the torques of the Stäubli TX60L are subject to physical constraints

$$\boldsymbol{q}_{\min} \le \boldsymbol{q} \le \boldsymbol{q}_{\max} \qquad \dot{\boldsymbol{q}}_{\min} \le \dot{\boldsymbol{q}} \le \dot{\boldsymbol{q}}_{\max} \quad (6a)$$

$$\ddot{\boldsymbol{q}}_{\min} \le \ddot{\boldsymbol{q}} \le \ddot{\boldsymbol{q}}_{\max} \qquad \boldsymbol{\tau}_{\min} \le \boldsymbol{\tau} \le \boldsymbol{\tau}_{\max} \quad (6b)$$

$$\boldsymbol{q}^{(3)}_{\min} \le \boldsymbol{q}^{(3)} \le \boldsymbol{q}^{(3)}_{\max}. \quad (6c)$$

For the sake of brevity, the equations of motion and the constraints are not explicitly stated. The limits for $\boldsymbol{q}$ directly result from the construction of the robot, whereas the constraints for $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$, and $\boldsymbol{q}^{(3)}$ may come from different other reasons, e.g., safety considerations or the avoidance of vibrations. The limits for the torques arise from the construction and the electrical properties of the motors. Their explicit incorporation is necessary for fully utilizing the dynamics of the robot which is in particular important in view of the considered task described in Section 2.3.

2.3 Task Specification

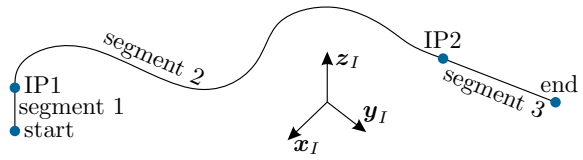

Fig. 2. Considered PTPCP trajectory.

The overall task considered for the robot Stäubli TX60L is given by determining a PTPCP trajectory which is to be traversed in minimum time. This trajectory is planned in the three-dimensional workspace of the robot. Exemplarily, we consider a PTPCP trajectory consisting of three segments, see Fig. 2. The first and third segment are straight CP parts whereas the second segment is given as a PTP trajectory. The points where these segments are connected are referred to as intermediate points with abbreviations IP1 and IP2, cf. Fig. 2. The overall PTPCP trajectory starts and ends at points of rest with predefined positions $\boldsymbol{r}_{\mathrm{S}}$, $\boldsymbol{r}_{\mathrm{E}}$ and orientations of the end effector. The orientations are given by the rotation matrices $\boldsymbol{R}_{\mathrm{TCP,S}}$ and $\boldsymbol{R}_{\mathrm{TCP,E}}$, resp. The corresponding values of the joint angles following from (1) are denoted by $\boldsymbol{q}_{\mathrm{S}}$ and $\boldsymbol{q}_{\mathrm{E}}$. Similarly, at IP1 and IP2 the positions $\boldsymbol{r}_{\mathrm{IP1}}$, $\boldsymbol{r}_{\mathrm{IP2}}$ of the end effector are also predefined. In addition, it is assumed that the orientations of the end effector at IP1 and IP2 are fixed as well and again given by $\boldsymbol{R}_{\mathrm{TCP,S}}$ and $\boldsymbol{R}_{\mathrm{TCP,E}}$. The main reason for this measure is that the corresponding joint angles denoted by $\boldsymbol{q}_{\mathrm{IP1}}$ and $\boldsymbol{q}_{\mathrm{IP2}}$ according to (1) are fully determined which can be beneficial, e.g., with regard to replanning or collision avoidance. In the first and third segment, the position of the end effector has to follow a straight line. These lines can be defined in the parameterized form

$${}_I\boldsymbol{r}_{\mathrm{TCP}} = \begin{cases} \boldsymbol{\eta}_1(\sigma_1) = \boldsymbol{r}_{\mathrm{S}} + \sigma_1 \Delta\boldsymbol{r}_1 & \text{segment 1} \\ \boldsymbol{\eta}_3(\sigma_3) = \boldsymbol{r}_{\mathrm{IP2}} + \sigma_3 \Delta\boldsymbol{r}_3 & \text{segment 3} \end{cases} \quad (7)$$

with the path parameters σ_1 and σ_3 depending on the time t. Furthermore, the path parameters are supposed to be contained in the interval $[0,1]$ which is fulfilled for $\Delta\boldsymbol{r}_1 = \boldsymbol{r}_{\mathrm{IP1}} - \boldsymbol{r}_{\mathrm{S}}$ and $\Delta\boldsymbol{r}_3 = \boldsymbol{r}_{\mathrm{E}} - \boldsymbol{r}_{\mathrm{IP2}}$. By inserting the forward kinematics into (7) and eliminating the path parameters, one obtains an implicit formulation of the path segment $i \in \{1,3\}$ in terms of the joint angles

$$\mathbb{R}^2 \ni \boldsymbol{\xi}_i(\boldsymbol{q}) = \boldsymbol{0}. \quad (8)$$

The specific form of the PTPCP trajectory can be motivated, e.g., by pick and place tasks. Segments 1 and 3 describe the parts of the trajectory where the workpiece is picked up by the robot and laid down at its target position, both with predefined orientation. Nevertheless, the methods presented in the following can be straightforwardly extended to other PTPCP trajectories consisting, e.g., of more than three segments. The overall time needed for the PTPCP trajectory, subsequently also referred to as the trajectory time, is denoted by $T = T_1 + T_2 + T_3$ where T_1, T_2, and T_3 are the periods of time needed for each segment. Summarizing, the considered task of trajectory generation consists of determining

1. a geometry of the PTP part in segment 2 (describing the position of the TCP) as well as the orientation of the end effector,
2. the orientation of the end effector during the CP parts in segments 1 and 3, and
3. a time parameterization of the whole geometric path

such that the trajectory time T is as small as possible under the given constraints (6) and the geometry of the CP parts. The solution of this task is presented in the following section.

3. GENERATION OF FAST PTPCP TRAJECTORIES

This section presents different possibilities for finding fast PTPCP trajectories solving the task specified in Section 2.3. On the one hand, the time-optimal solution (up to numerical accuracy) is determined in Section 3.1. Naturally, the computing time for calculating this solution is rather large, however, it is important for comparison and benchmark purposes. On the other hand, a more efficient calculation of the trajectories in the sense of less computing time is presented in Section 3.2. To this end, certain parameterizations are utilized which slightly restrict the shape of the curve in segment 2 and the time parameterization of the PTPCP trajectory.

3.1 Optimal Solution

For calculating the optimal solution (up to numerical accuracy), the task described in Section 2.3 is formulated as an OCP. This OCP is solved with the software package $\mathcal{PSOPT}$, see Becerra (2010). The main reason for employing this solver is that it explicitly supports OCPs with a finite number of phases, each with a different cost functional, system equations basically in the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$ with state $\boldsymbol{x}$ and input $\boldsymbol{u}$, and constraints. Thus, the three segments of the PTPCP trajectory with different constraints can be directly incorporated. The problem is automatically discretized and solved with a solver for static optimization. It can be chosen between SNOPT, see Gill et al. (2002), or IPOPT, cf. Wächter and Biegler (2006). In the following, exclusively IPOPT is used.

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586

The gradients and Jacobian matrices corresponding to the obtained static optimization problem are determined internally by means of automatic differentiation with the package ADOL-C or by finite differences. Experience shows that the complex mathematical model of the robot and the constraints lead to difficulties when using ADOL-C. Hence, solely the determination of the gradients and Jacobian matrices by means of finite differences is used.

In principle, several different formulations of the OCP are possible, in particular motivated by the jerk constraints (6c). They entail that the torques $\boldsymbol{\tau}$, being the "input" to the robot, cannot be directly considered as inputs $\boldsymbol{u}$ to the dynamical system within the OCP. This is due to the fact that $\boldsymbol{q}^{(3)}$ is dependent on $\dot{\boldsymbol{\tau}}$. Hence, $\dot{\boldsymbol{u}}$ would occur which is not supported by $\mathcal{PSOPT}$. The first possibility for avoiding this problem is given by setting $\dot{\boldsymbol{\tau}} = \boldsymbol{u}$ and extending the robot dynamics with further states to obtain $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ for $\mathcal{PSOPT}$. The second possibility is based on a mathematical model $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ of the form $\boldsymbol{q}^{(3)} = \boldsymbol{u}$ with corresponding state $\boldsymbol{x}$ given by $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$. In the following, the second possibility is pursued.

By using the implicit description (8) of the CP parts, the OCP to be solved with $\mathcal{PSOPT}$ reads as

$$\min_{\boldsymbol{u}(\cdot)} \quad J(\boldsymbol{u}(\cdot)) \tag{9a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}} = \left[\dot{\boldsymbol{q}}^{\mathrm{T}} \ \ddot{\boldsymbol{q}}^{\mathrm{T}} \ \boldsymbol{u}^{\mathrm{T}}\right]^{\mathrm{T}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{9b}$$

$$\text{segments 1–3} \begin{cases} \boldsymbol{q}_{\min} \le \boldsymbol{q}(t) \le \boldsymbol{q}_{\max} \\ \dot{\boldsymbol{q}}_{\min} \le \dot{\boldsymbol{q}}(t) \le \dot{\boldsymbol{q}}_{\max} \\ \ddot{\boldsymbol{q}}_{\min} \le \ddot{\boldsymbol{q}}(t) \le \ddot{\boldsymbol{q}}_{\max} \\ \boldsymbol{q}^{(3)}_{\min} \le \boldsymbol{u}(t) \le \boldsymbol{q}^{(3)}_{\max} \\ \boldsymbol{\tau}_{\min} \le \boldsymbol{\Theta}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t)) \le \boldsymbol{\tau}_{\max} \end{cases} \quad \forall t \in [0, T] \tag{9c}$$

$$\text{segment 1} \begin{cases} \boldsymbol{q}(0) = \boldsymbol{q}_{\mathrm{S}} \quad \boldsymbol{q}(T_1) = \boldsymbol{q}_{\mathrm{IP1}} \\ \dot{\boldsymbol{q}}(0) = \ddot{\boldsymbol{q}}(0) = \boldsymbol{0} \\ \boldsymbol{\xi}_1(\boldsymbol{q}(t)) = \boldsymbol{0} \ \forall t \in [0, T_1] \end{cases} \tag{9d}$$

$$\text{segment 2} \begin{cases} \boldsymbol{q}(T_1 + T_2) = \boldsymbol{q}_{\mathrm{IP2}} \end{cases} \tag{9e}$$

$$\text{segment 3} \begin{cases} \boldsymbol{q}(T) = \boldsymbol{q}_{\mathrm{E}} \quad \dot{\boldsymbol{q}}(T) = \ddot{\boldsymbol{q}}(T) = \boldsymbol{0} \\ \boldsymbol{\xi}_3(\boldsymbol{q}(t)) = \boldsymbol{0} \ \forall t \in [T_1 + T_2, T] \end{cases} \tag{9f}$$

with the cost functional

$$J(\boldsymbol{u}(\cdot)) = T_1 + T_2 + T_3 = T. \tag{10}$$

3.2 Efficient Calculation of Fast PTPCP Trajectories

This section aims at developing a more efficient (in terms of less computing time) way of calculating the desired PTPCP trajectories. To this end, the following restriction is made. The optimal solution with $\mathcal{PSOPT}$ according to Section 3.1 of the task described in Section 2.3 shows that the change of orientation of the end effector in the first and third segment is negligible, cf. Fig. 4 in Section 4. Therefore, in view of a reduction of computing time, these DOFs are removed and the orientations in the first and third segment are assumed to be fixed with the rotation matrices $\boldsymbol{R}_{\mathrm{TCP,S}}$ and $\boldsymbol{R}_{\mathrm{TCP,E}}$, resp.

The remaining DOFs for the optimization are given by the choice of $\sigma_1(t)$ and $\sigma_3(t)$ in segments 1 and 3 as well as the joint angles $\boldsymbol{q}(t)$ in segment 2. These time-dependent quantities are represented by polynomials of fifth order (P5s) or cubic spline functions (CSFs). Naturally, this restricts the shape of the respective quantities. Therefore, the obtained solutions are merely optimal in the sense of the chosen parameterization. Nevertheless, these parameterizations reduce the number of DOFs which enables to reach the overall goal of formulating a static optimization problem which can be efficiently solved. Note that henceforth a generic quantity $\chi(t)$ being represented with either a P5 or a CSF is written in the form $\hat{\chi}(\boldsymbol{p}, t)$. For the ease of notation, the corresponding coefficients of all subsequently stated P5s or CSFs are denoted as $\boldsymbol{p}$.

The coefficients of a time-dependent P5 are uniquely determined with three boundary conditions for the derivatives of order 0–2 at each end of a given interval of time. For the path parameter σ_1 in the first segment, being represented as $\hat{\sigma}_1(\boldsymbol{p}, t)$, these boundary conditions are given by

$$\sigma_1(0) = 0 \qquad \sigma_1(T_1) = 1 \tag{11a}$$

$$\dot{\sigma}_1(0) = 0 \qquad \dot{\sigma}_1(T_1) = \frac{v_{\mathrm{IP1}}}{\|\Delta \boldsymbol{r}_1\|} \tag{11b}$$

$$\ddot{\sigma}_1(0) = 0 \qquad \ddot{\sigma}_1(T_1) = \frac{a_{\mathrm{IP1}}}{\|\Delta \boldsymbol{r}_1\|} \tag{11c}$$

with $\|\cdot\|$ denoting the Euclidean norm. The conditions for $\dot{\sigma}_1(T_1)$ and $\ddot{\sigma}_1(T_1)$ follow from the first and second time-derivative of (7) in the form

$$\dot{\boldsymbol{\eta}}_1(\dot{\sigma}_1) = \dot{\sigma}_1 \Delta \boldsymbol{r}_1 \qquad \ddot{\boldsymbol{\eta}}_1(\ddot{\sigma}_1) = \ddot{\sigma}_1 \Delta \boldsymbol{r}_1 \tag{12}$$

and noting that the constant directions of the velocity and acceleration vectors are determined by $\Delta \boldsymbol{r}_1$. Therefore, only the magnitudes at the end of segment 1 (at IP1) given by v_{IP1} and a_{IP1} remain as DOFs. In an analogous manner, the boundary conditions for σ_3 follow as

$$\sigma_3(T_1 + T_2) = 0 \qquad \sigma_3(T) = 1 \tag{13a}$$

$$\dot{\sigma}_3(T_1 + T_2) = \frac{v_{\mathrm{IP2}}}{\|\Delta \boldsymbol{r}_3\|} \qquad \dot{\sigma}_3(T) = 0 \tag{13b}$$

$$\ddot{\sigma}_3(T_1 + T_2) = \frac{a_{\mathrm{IP2}}}{\|\Delta \boldsymbol{r}_3\|} \qquad \ddot{\sigma}_3(T) = 0. \tag{13c}$$

In segment 2, the joint angles $\boldsymbol{q}$ are represented by P5s $\hat{\boldsymbol{q}}(\boldsymbol{p}, t)$ as well. The translational and rotational velocities and accelerations at $t = T_1$ (IP1) and $t = T_1 + T_2$ (IP2) are given by

$$\dot{\boldsymbol{z}}(T_1) = \begin{bmatrix} v_{\mathrm{IP1}} \frac{\Delta \boldsymbol{r}_1}{\|\Delta \boldsymbol{r}_1\|} \\ \boldsymbol{0} \end{bmatrix} \qquad \dot{\boldsymbol{z}}(T_1 + T_2) = \begin{bmatrix} v_{\mathrm{IP2}} \frac{\Delta \boldsymbol{r}_3}{\|\Delta \boldsymbol{r}_3\|} \\ \boldsymbol{0} \end{bmatrix} \tag{14a}$$

$$\ddot{\boldsymbol{z}}(T_1) = \begin{bmatrix} a_{\mathrm{IP1}} \frac{\Delta \boldsymbol{r}_1}{\|\Delta \boldsymbol{r}_1\|} \\ \boldsymbol{0} \end{bmatrix} \qquad \ddot{\boldsymbol{z}}(T_1 + T_2) = \begin{bmatrix} a_{\mathrm{IP2}} \frac{\Delta \boldsymbol{r}_3}{\|\Delta \boldsymbol{r}_3\|} \\ \boldsymbol{0} \end{bmatrix}. \tag{14b}$$

With these quantities, the boundary conditions for the P5s in the second segment read as

$$\boldsymbol{q}(T_1) = \boldsymbol{q}_{\mathrm{IP1}} \qquad \boldsymbol{q}(T_{12}) = \boldsymbol{q}_{\mathrm{IP2}} \tag{15a}$$

$$\dot{\boldsymbol{q}}(T_1) = \boldsymbol{J}(\boldsymbol{q}_{\mathrm{IP1}}) \dot{\boldsymbol{z}}(T_1) \qquad \dot{\boldsymbol{q}}(T_{12}) = \boldsymbol{J}(\boldsymbol{q}_{\mathrm{IP2}}) \dot{\boldsymbol{z}}(T_{12}) \tag{15b}$$

$$\ddot{\boldsymbol{q}}(T_1) = \boldsymbol{J}(\boldsymbol{q}_{\mathrm{IP1}}) \left(\ddot{\boldsymbol{z}}(T_1) - \dot{\boldsymbol{J}}(\boldsymbol{q}_{\mathrm{IP1}}, \dot{\boldsymbol{q}}(T_1)) \dot{\boldsymbol{q}}(T_1) \right) \tag{15c}$$

$$\ddot{\boldsymbol{q}}(T_{12}) = \boldsymbol{J}(\boldsymbol{q}_{\mathrm{IP2}}) \left(\ddot{\boldsymbol{z}}(T_{12}) - \dot{\boldsymbol{J}}(\boldsymbol{q}_{\mathrm{IP2}}, \dot{\boldsymbol{q}}(T_{12})) \dot{\boldsymbol{q}}(T_{12}) \right) \tag{15d}$$

with the shortcut $T_{12} = T_1 + T_2$. The DOFs v_{IP1}, a_{IP1}, v_{IP2}, and a_{IP2} need to be fixed in order to uniquely define all P5s from the boundary conditions (11), (13), and (15).

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Hence, the PTPCP trajectory is fully determined and the optimization variables are given by

$$\boldsymbol{X} = \left[T_1 \; T_2 \; T_3 \; v_{\mathrm{IP1}} \; a_{\mathrm{IP1}} \; v_{\mathrm{IP2}} \; a_{\mathrm{IP2}}\right]^{\mathrm{T}}. \tag{16}$$

The relation between $\boldsymbol{X}$ and the coefficients of the polynomials are denoted by $\boldsymbol{\kappa}_i(\boldsymbol{X})$ for the ith segment. By using these relations, the final P5s follow as

$$\hat{\sigma}_1(\boldsymbol{\kappa}_1(\boldsymbol{X}), t), \; \hat{\sigma}_3(\boldsymbol{\kappa}_3(\boldsymbol{X}), t), \text{ and } \hat{\boldsymbol{q}}(\boldsymbol{\kappa}_2(\boldsymbol{X}), t). \tag{17}$$

The second possibility considered for representing the path parameters $\sigma_1(t)$ and $\sigma_3(t)$ as well as the joint angles $\boldsymbol{q}(t)$ is given by CSFs. The basic idea is to add additional DOFs such that the shape of the curve is more flexible compared to the P5s. These additional DOFs are given by points through which the CSF is supposed to pass. In the first and third segment, N_{S1} and N_{S3} evenly distributed additional points collected in the vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_3$ are introduced for describing $\sigma_1(t)$ and $\sigma_3(t)$. In the second segment, for each joint angle N_{S2} evenly distributed additional points are considered. All $6N_{S2}$ points for $\boldsymbol{q}(t)$ are collected in a vector $\boldsymbol{e}_2$. Similar to the P5s, the boundary conditions (11), (13), and (15) as well as the values given by $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, and $\boldsymbol{e}_3$ allow to uniquely determine all occurring CSFs representing $\sigma_1(t)$, $\sigma_3(t)$, and $\boldsymbol{q}(t)$. Hence, the DOFs being the optimization variables are given by

$$\boldsymbol{X} = \left[T_1 \; T_2 \; T_3 \; v_{\mathrm{IP1}} \; a_{\mathrm{IP1}} \; v_{\mathrm{IP2}} \; a_{\mathrm{IP2}} \; \boldsymbol{e}_1^{\mathrm{T}} \; \boldsymbol{e}_2^{\mathrm{T}} \; \boldsymbol{e}_3^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{18}$$

In an abstract form, the relation between $\boldsymbol{X}$ and the coefficients of the CSFs can again be written as $\boldsymbol{\kappa}_i(\boldsymbol{X})$ for the ith segment. For the sake of brevity, the same symbols are used as for P5s. Furthermore, the basic structure of the calculations is the same for P5s and CSFs. Hence, the final representations of $\sigma_1(t)$, $\sigma_3(t)$, and $\boldsymbol{q}(t)$ with CSFs can also be written in the form (17).

For implementation as a static optimization problem, it is further necessary to transform the variable time durations of segments 1–3 into time intervals of constant lengths. To this end, the transformations

$$\text{segment } i: \qquad t = \Lambda_i(\boldsymbol{X}, \nu_i) = \textstyle\sum_{j=1}^{i-1} T_j + T_i \nu_i \tag{19}$$

are utilized with the new independent variables ν_i for the ith segment being contained in the fixed intervals $[0, 1]$.

For the consideration of the constraints (6), $\boldsymbol{q}(t)$ as well as its derivatives have to be explicitly known. In segment 2, this does not require any special calculations. On the contrary, in the first and third segment, the joint angles and its derivatives have to be determined from the inverse kinematics (1) and the equations (4). In combination with the derived parameterizations either given by P5s or CSFs, this results in

$$\boldsymbol{\rho}_i(\boldsymbol{X}, t) = \boldsymbol{\Phi}\left(\boldsymbol{\eta}_i(\hat{\sigma}_i(\boldsymbol{\kappa}_i(\boldsymbol{X}), t)), \boldsymbol{R}_{\mathrm{TCP,S/E}}\right) \tag{20a}$$

for the joint angles $\boldsymbol{q}$ and the expressions

$$\dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t) = \boldsymbol{J}(\boldsymbol{\rho}_i(\boldsymbol{X}, t)) \begin{bmatrix} \dot{\hat{\sigma}}_i(\boldsymbol{\kappa}_i(\boldsymbol{X}), t) \, \Delta \boldsymbol{r}_i \\ \boldsymbol{0} \end{bmatrix} \tag{20b}$$

$$\ddot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t) = \boldsymbol{J}(\boldsymbol{\rho}_i(\boldsymbol{X}, t)) \left(\begin{bmatrix} \ddot{\hat{\sigma}}_i(\boldsymbol{\kappa}_i(\boldsymbol{X}), t) \, \Delta \boldsymbol{r}_i \\ \boldsymbol{0} \end{bmatrix} - \dot{\overline{\boldsymbol{J}}}(\boldsymbol{\rho}_i(\boldsymbol{X}, t), \dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t)) \, \dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t) \right) \tag{20c}$$

$$\boldsymbol{\rho}_i^{(3)}(\boldsymbol{X}, t) = \boldsymbol{J}(\boldsymbol{\rho}_i(\boldsymbol{X}, t)) \left(\begin{bmatrix} \hat{\sigma}_i^{(3)}(\boldsymbol{\kappa}_i(\boldsymbol{X}), t) \, \Delta \boldsymbol{r}_i \\ \boldsymbol{0} \end{bmatrix} - \ddot{\overline{\boldsymbol{J}}}(\boldsymbol{\rho}_i(\boldsymbol{X}, t), \dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t), \ddot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t)) \, \dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t) - 2\dot{\overline{\boldsymbol{J}}}(\boldsymbol{\rho}_i(\boldsymbol{X}, t), \dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t)) \, \ddot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t) \right) \tag{20d}$$

for the corresponding derivatives $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$, and $\boldsymbol{q}^{(3)}$ with $i \in \{1, 3\}$ according to segments 1 and 3, resp. In the first segment, $\boldsymbol{R}_{\mathrm{TCP,S}}$ has to be used whereas in the third segment $\boldsymbol{R}_{\mathrm{TCP,E}}$ has to be inserted into (20a). Demanding the fulfillment of the constraints (6) for every point in time contradicts computational efficiency. Therefore, the constraints (6) are only verified for a finite number of points. In terms of the new independent variables ν_i, these points are evenly distributed in the interval $[0, 1]$ and given by ν_{ij}, $j = 1, \ldots, N_{di}$ for the ith segment.

Based on these preparatory steps, the static optimization problem with the parameterizations given by P5s or CSFs and the cost function $F(\boldsymbol{X}) = T_1 + T_2 + T_3$ follows as

$$\min_{\boldsymbol{X}} \quad F(\boldsymbol{X}) \tag{21a}$$

$$\text{s.t.} \quad \text{segment } i, \; i \in \{1, 3\} \begin{cases} \boldsymbol{q}_{\min} \le \boldsymbol{\rho}_i(\boldsymbol{X}, \Lambda_i(\boldsymbol{X}, \nu_{ij})) \le \boldsymbol{q}_{\max} \\ \dot{\boldsymbol{q}}_{\min} \le \dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, \Lambda_i(\boldsymbol{X}, \nu_{ij})) \le \dot{\boldsymbol{q}}_{\max} \\ \ddot{\boldsymbol{q}}_{\min} \le \ddot{\boldsymbol{\rho}}_i(\boldsymbol{X}, \Lambda_i(\boldsymbol{X}, \nu_{ij})) \le \ddot{\boldsymbol{q}}_{\max} \\ \boldsymbol{q}^{(3)}_{\min} \le \boldsymbol{\rho}_i^{(3)}(\boldsymbol{X}, \Lambda_i(\boldsymbol{X}, \nu_{ij})) \le \boldsymbol{q}^{(3)}_{\max} \\ \boldsymbol{\tau}_{\min} \le \boldsymbol{\Theta}(\boldsymbol{\rho}_i(\boldsymbol{X}, t), \dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t), \\ \qquad \ddot{\boldsymbol{\rho}}_i(\boldsymbol{X}, t))|_{t=\Lambda_i(\boldsymbol{X}, \nu_{ij})} \le \boldsymbol{\tau}_{\max} \\ \qquad\qquad \forall j = 1, \ldots, N_{di} \end{cases} \tag{21b}$$

$$\text{segment } 2 \begin{cases} \boldsymbol{q}_{\min} \le \hat{\boldsymbol{q}}(\boldsymbol{\kappa}_2(\boldsymbol{X}), \Lambda_2(\boldsymbol{X}, \nu_{2j})) \le \boldsymbol{q}_{\max} \\ \dot{\boldsymbol{q}}_{\min} \le \dot{\hat{\boldsymbol{q}}}(\boldsymbol{\kappa}_2(\boldsymbol{X}), \Lambda_2(\boldsymbol{X}, \nu_{2j})) \le \dot{\boldsymbol{q}}_{\max} \\ \ddot{\boldsymbol{q}}_{\min} \le \ddot{\hat{\boldsymbol{q}}}(\boldsymbol{\kappa}_2(\boldsymbol{X}), \Lambda_2(\boldsymbol{X}, \nu_{2j})) \le \ddot{\boldsymbol{q}}_{\max} \\ \boldsymbol{q}^{(3)}_{\min} \le \hat{\boldsymbol{q}}^{(3)}(\boldsymbol{\kappa}_2(\boldsymbol{X}), \Lambda_2(\boldsymbol{X}, \nu_{2j})) \le \boldsymbol{q}^{(3)}_{\max} \\ \boldsymbol{\tau}_{\min} \le \boldsymbol{\Theta}\left(\hat{\boldsymbol{q}}(\boldsymbol{\kappa}_2(\boldsymbol{X}), t), \dot{\hat{\boldsymbol{q}}}(\boldsymbol{\kappa}_2(\boldsymbol{X}), t), \right. \\ \qquad \left. \ddot{\hat{\boldsymbol{q}}}(\boldsymbol{\kappa}_2(\boldsymbol{X}), t)\right)\Big|_{t=\Lambda_2(\boldsymbol{X}, \nu_{2j})} \le \boldsymbol{\tau}_{\max} \\ \qquad\qquad \forall j = 1, \ldots, N_{d2}. \end{cases} \tag{21c}$$

For solving the static optimization problem (21), the solver `fmincon` from MATLAB is utilized. As it will be shown in Section 4, significant improvements with respect to computing time and reliability can be achieved if the analytical gradients and Jacobian matrices, subsequently collectively referred to as gradients, of the cost function and constraints with respect to the optimization variables are provided to `fmincon`. Most of the necessary calculations for obtaining the gradients are straightforward and basically consist of applying the chain rule of differentiation. For brevity, just some exemplary results for segments 1 and 3 are shortly sketched. One required derivative is given by (function arguments are frequently omitted for brevity)

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{\rho}_i}{\mathrm{d}\boldsymbol{X}} &= \frac{\mathrm{d}\boldsymbol{\Phi}\left(\boldsymbol{\eta}_i(\hat{\sigma}_i(\boldsymbol{\kappa}_i(\boldsymbol{X}), \Lambda_i(\boldsymbol{X}, \nu_{ij}))), \boldsymbol{R}_{\mathrm{TCP,S/E}}\right)}{\mathrm{d}\boldsymbol{X}} \\ &= \frac{\partial \boldsymbol{\Phi}}{\partial {}_I\boldsymbol{r}_{\mathrm{TCP}}} \frac{\partial \boldsymbol{\eta}_i}{\partial \sigma_i} \left(\frac{\partial \hat{\sigma}_i}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{\kappa}_i}{\partial \boldsymbol{X}} + \frac{\partial \hat{\sigma}_i}{\partial t} \frac{\partial \Lambda_i}{\partial \boldsymbol{X}} \right). \end{aligned} \tag{22}$$

The term $\frac{\partial \boldsymbol{\Phi}}{\partial {}_I\boldsymbol{r}_{\mathrm{TCP}}}$ is the most interesting one. Mainly due to ambiguities and the extensive expressions in (1) and (4), a direct differentiation of the inverse kinematics is not practicable. Instead, by differentiating (1) with respect to time for $\boldsymbol{R}_{\mathrm{TCP}} = \text{const.}$ (in segments 1 and 3) one obtains

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

$$\dot{\boldsymbol{q}} = \frac{\partial \boldsymbol{\Phi}}{\partial {}_I\boldsymbol{r}_{\mathrm{TCP}}} {}_I\boldsymbol{v}_{\mathrm{TCP}}. \tag{23}$$

A comparison of this equation with (4a) reveals that $\frac{\partial \boldsymbol{\Phi}}{\partial {}_I\boldsymbol{r}_{\mathrm{TCP}}}$ is given by the first three columns of $\boldsymbol{J}(\boldsymbol{q})$ with $\boldsymbol{\rho}_i(\boldsymbol{X}, \Lambda_i(\boldsymbol{X}, \nu_{ij}))$ being inserted for $\boldsymbol{q}$. In principle, the calculation of

$$\frac{\mathrm{d}\dot{\boldsymbol{\rho}}_i(\boldsymbol{X}, \Lambda_i(\boldsymbol{X}, \nu_{ij}))}{\mathrm{d}\boldsymbol{X}} \tag{24}$$

requires the differentiation of $\boldsymbol{J} = \overline{\boldsymbol{J}}^{-1}$, see (20b). However, in contrast to $\overline{\boldsymbol{J}}$, its inverse is not available in analytic form which prohibits straightforward differentiation. Nevertheless, the required expressions can be obtained from

$$\dot{z}_k = \sum_{j=1}^{6} \overline{J}_{kj}(\boldsymbol{\rho}_i)\, \dot{\rho}_{i,j}, \quad k \in \{1, 2, \ldots, 6\} \tag{25}$$

with $\overline{J}_{kj}$ being the components of $\overline{\boldsymbol{J}}$ in the kth row and jth column and $\dot{\rho}_{i,j}$ being the jth component of $\dot{\boldsymbol{\rho}}_i$, cf. (3). For the case at hand, it holds that

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{\hat{\sigma}}_i(\boldsymbol{\kappa}_i(\boldsymbol{X}), t)\, \Delta \boldsymbol{r}_i \\ \boldsymbol{0} \end{bmatrix}. \tag{26}$$

Hence, by differentiating (25) with respect to $\boldsymbol{X}$ the required gradients are obtained in the form

$$\frac{\mathrm{d}\dot{\boldsymbol{\rho}}_i}{\mathrm{d}\boldsymbol{X}} = \boldsymbol{J}(\boldsymbol{\rho}_i) \left(\frac{\partial \dot{\boldsymbol{z}}}{\partial \boldsymbol{X}} - \begin{bmatrix} \sum_{j=1}^{6} \frac{\partial \overline{J}_{1j}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{\rho}_i}{\partial \boldsymbol{X}} \dot{\rho}_{i,j} \\ \vdots \\ \sum_{j=1}^{6} \frac{\partial \overline{J}_{6j}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{\rho}_i}{\partial \boldsymbol{X}} \dot{\rho}_{i,j} \end{bmatrix} \right) \tag{27}$$

where $\frac{\partial \boldsymbol{\rho}_i}{\partial \boldsymbol{X}}$ is already known from (22). The gradients $\frac{\mathrm{d}\ddot{\boldsymbol{\rho}}_i}{\mathrm{d}\boldsymbol{X}}$ and $\frac{\mathrm{d}\boldsymbol{\rho}_i^{(3)}}{\mathrm{d}\boldsymbol{X}}$ are calculated in a very similar way.

Concerning the implementation, the gradients in analytic form are computed for the parameterization with P5s. Besides the reduction in computing time, which will be shown in Section 4, the main reason for this calculation is that `fmincon` often does not find a solution unless the analytic gradients are provided. For the parameterization with CSFs, this issue occurs more seldom and the implementation of the gradients is considerably more tedious which is why it is not further pursued for the CSFs.

4. RESULTS

This section presents some results from the methods of Section 3. To this end, four different PTPCP trajectories are considered in Sections 4.1 and 4.2 which are also experimentally tested at the Stäubli TX60L. In particular, the methods are evaluated with respect to the found trajectory time T and the required computing time T_{CPU}. Some conclusions are drawn in Section 4.3.

In all subsequent figures, horizontal dash-dot-dot lines represent the constraints for certain quantities. The number of points for which the constraints are verified is chosen as $N_{d1} = N_{d2} = N_{d3} = 30$. For the CSFs, $N_{S1} = N_{S2} = N_{S3} = 8$ additional points are used.

All trajectories are calculated on a 64 bit Windows 7 computer equipped with an Intel i7-620M CPU with two cores/four threads at a processor frequency of 2.67 GHz. For `fmincon`, MATLAB R2013b is used. The built-in parallel computing capabilities of `fmincon` are utilized. However, the required start-up time of the parallel pool is not included in T_{CPU}.

4.1 PTPCP Trajectory 1

The start and end points of the first considered PTPCP trajectory and the corresponding positions of the intermediate points can be seen in Fig. 3. The start and end orientations are given by

$$\boldsymbol{R}_{\mathrm{TCP,S}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \boldsymbol{R}_{\mathrm{TCP,E}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{28}$$

As presented in Section 3, four different possibilities are considered for calculating the PTPCP trajectory. Firstly, the optimal solution is calculated with $\mathcal{PSOPT}$. The second and third possibility are given by utilizing a parameterization with P5s and the solver `fmincon` for solving the resulting static optimization problem, either without or with using the analytical gradients. Fourthly, a parameterization with CSFs is used for obtaining the static optimization problem being solved by `fmincon`. The obtained trajectory times T with these four possibilities are listed in Table 1. Furthermore, the required computing time T_{CPU} is displayed for each method. As expected, $\mathcal{PSOPT}$ delivers the best solution with the smallest trajectory time T. On the contrary, T for the parameterization with P5s is considerably larger. For a better analysis of this result, Table 2 shows T_1, T_2, and T_3 needed for traversing the three segments determined by each method. It can be seen that in the first and third segment there is not a large difference in performance of the four methods. In particular, $\mathcal{PSOPT}$ and the CSFs deliver almost identical results.

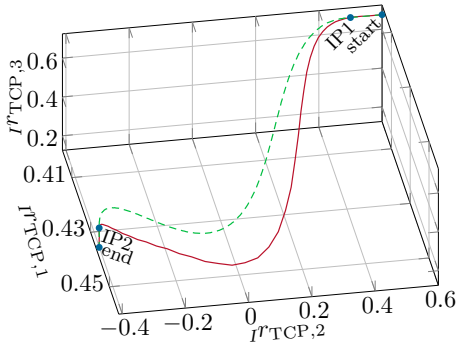

Fig. 3. PTPCP trajectory 1 found by $\mathcal{PSOPT}$ (solid line) and `fmincon` with P5s (dashed) (all quantities in m).

Table 1. Trajectory and computing times for PTPCP trajectory 1.

Method	T in ms	T_{CPU} in s
Optimal solution	741.3	3056.3
Parameterization P5s	837.1	435.8
Par. P5s analytical grad.	837.1	195.8
Parameterization CSFs	775.6	1597.0

Table 2. Traversing times T_1, T_2, and T_3 for each segment of PTPCP trajectory 1.

Method	T_1 in ms	T_2 in ms	T_3 in ms
Optimal solution	153.0	440.6	147.7
Parameterization P5s	160.1	521.9	155.1
Par. P5s analytical grad.	160.1	521.9	155.1
Parameterization CSFs	152.9	474.5	148.2

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586

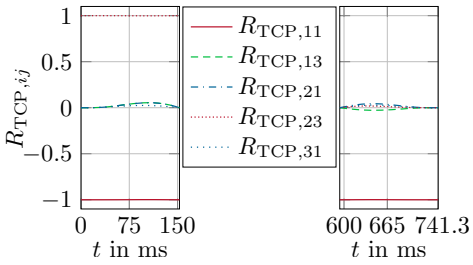

Fig. 4. Some elements of $\boldsymbol{R}_{\mathrm{TCP}}$ for PTPCP trajectory 1 in segments 1 (left) and 3 (right) determined by $\mathcal{PSOPT}$.

The fact that T_1 is slightly smaller for CSFs appears counterintuitive. However, this solution might be better for $\mathcal{PSOPT}$ in view of a smaller time T_2. Table 2 reveals that most of the loss of the parameterization with P5s occurs in the second segment. This result had to be expected as the P5s do not allow much freedom for shaping the geometry of the curve in the PTP segment. This constitutes the main reason for considering the parameterization with CSFs as they introduce additional DOFs for influencing the geometry in a wider range (of course, depending on the number N_{Si} of additional points). Table 2 justifies this approach as the time T_2 is considerably smaller for the CSFs compared to the P5s.

As expected, Table 1 shows that the superior trajectory time of $\mathcal{PSOPT}$ comes at the expense of the (by far) largest computing time. Although T obtained by the parameterization with CSFs is not much worse than the one from $\mathcal{PSOPT}$, the computing time is smaller by almost a factor of two. In this regard, the effect of using the analytical gradients for the parameterization with P5s is clearly visible in Table 1 as well. It reduces the computing time by more than a factor of two.

For illustration, Fig. 3 shows the trajectories of the end effector found by $\mathcal{PSOPT}$ and `fmincon` based on P5s with ${}_I r_{\mathrm{TCP},i}$ denoting the ith component of ${}_I\boldsymbol{r}_{\mathrm{TCP}}$. As expected, the difference in segment 2 is quite large. Figure 4 shows some representative elements of the rotation matrix $\boldsymbol{R}_{\mathrm{TCP}}$ in segments 1 and 3 as determined by $\mathcal{PSOPT}$. Obviously, the orientation of the end effector remains almost constant in these segments. This justifies the assumption of fixed orientations in segments 1 and 3 for the parameterizations with P5s and CSFs. Exemplarily, the joint accelerations $\ddot{\boldsymbol{q}}$ obtained by $\mathcal{PSOPT}$ are displayed in Fig. 5. Clearly, the respective constraints become active several times but are never violated. The corresponding joint torques $\boldsymbol{\tau}$ are shown in Fig. 6.

4.2 PTPCP Trajectories 2–4

Similar to the previous section, all four methods are tested for three further PTPCP trajectories 2–4. These trajectories as determined by $\mathcal{PSOPT}$ and `fmincon` based on P5s are displayed in the workspace of the robot in Fig. 7. The corresponding trajectory and computing times are listed in Table 3.

For PTPCP trajectory 2, `fmincon` does not converge for the parameterization with P5s without using the analytical gradients. If they are provided, then a solution is found without further problems. This clearly underlines the advantage with respect to reliability when using the analytical gradients. For the parameterization with CSFs, a solution is found although the analytical gradients are not provided. The corresponding trajectory time is not much worse than the one of $\mathcal{PSOPT}$ but the benefit in computing time is outstanding.

By comparing Table 1 with Table 3, it can be seen that the situation for PTPCP trajectories 1 and 3 is quite similar. However, the extraordinary decrease in computing time when using the analytical gradients for the parameteriza-

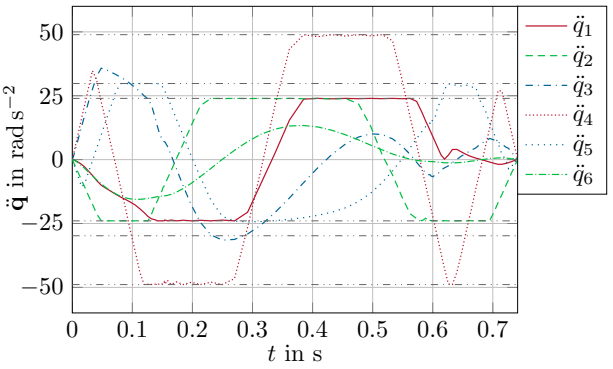

Fig. 5. Joint accelerations $\ddot{\boldsymbol{q}}$ determined by $\mathcal{PSOPT}$ for PTPCP trajectory 1.

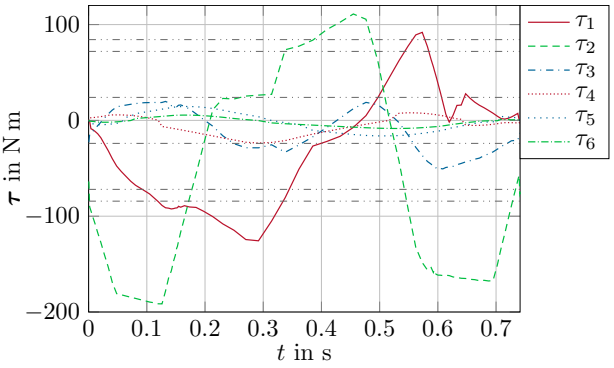

Fig. 6. Joint torques $\boldsymbol{\tau}$ determined by $\mathcal{PSOPT}$ for PTPCP trajectory 1.

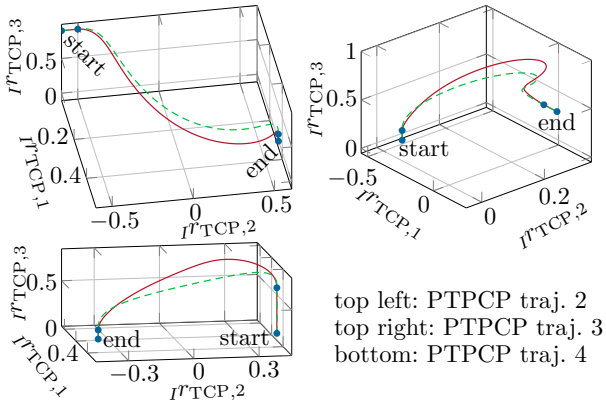

Fig. 7. PTPCP trajectories 2–4 found by $\mathcal{PSOPT}$ (solid) and `fmincon` with P5s (dashed) (all quantities in m).

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586

Table 3. Trajectory and computing times for PTPCP trajectories 2–4.

	Method	T in ms	T_{CPU} in s
traj. 2	Optimal solution	1012.4	11949.1
	Parameterization P5s	not converged	-
	Par. P5s analytical grad.	1157.6	265.1
	Parameterization CSFs	1058.7	1378.3
traj. 3	Optimal solution	1021.4	2311.7
	Parameterization P5s	1136.5	376.4
	Par. P5s analytical grad.	1136.5	47.7
	Parameterization CSFs	1046.7	1122.2
traj. 4	Optimal solution	972.4	3680.0
	Parameterization P5s	1112.7	169.2
	Par. P5s analytical grad.	1112.7	52.5
	Parameterization CSFs	1043.7	3332.5

tion with P5s is remarkable. For PTPCP trajectory 4, the computing times required by $\mathcal{PSOPT}$ and `fmincon` with CSFs are almost identical and by far larger than the ones for `fmincon` with P5s although the latter do not deliver much worse trajectory times.

4.3 Conclusions

Four different possibilities were shown for calculating trajectories for a six DOFs articulated robot. These trajectories connect two points of rest with prescribed position and orientation of the end effector and consist of three segments. For the first and third segment, the end effector is supposed to follow straight lines. In the second segment, the geometry of the curve is left as a DOF. On the one hand, as a benchmark the optimal solution was calculated with $\mathcal{PSOPT}$. On the other hand, two different parameterizations in the form of fifth-order polynomials (P5s) and cubic spline functions (CSFs) were presented for obtaining suboptimal solutions by solving a static optimization problem with `fmincon`. For P5s, also the analytical gradients in due consideration of the kinematics and dynamics of the robot were provided to `fmincon`. All methods were experimentally validated at the industrial robot Stäubli TX60L.

The parameterizations with P5s and CSFs have the advantage that the required computing time is much smaller compared to the optimal solution. If nothing else, this is remarkable in view of the fact that all the functions for `fmincon` are implemented in MATLAB code which in general is considered to be slower in execution than C/C++ code (on which $\mathcal{PSOPT}$ is based). Furthermore, the solver `fmincon` is a universal solver for a variety of static optimization problems. Hence, by implementing the static optimization problem based on the parameterizations with P5s or CSFs in C/C++ and by using a specialized solver, further significant reductions in computing time can be expected.

In addition, it was shown that by providing the analytical gradients, the computing time can be significantly reduced too. Fast computations are especially important for online planning or replanning. To this end, it may be reasonable to refrain from calculating the optimal solution and come back to a suboptimal one. Furthermore, online calculations require increased reliability. If a solution exists, it has to be found by the optimizer. In this regard, it has been shown that the calculation and usage of the analytical gradients is highly beneficial.

REFERENCES

Becerra, V.M. (2010). Solving complex optimal control problems at no cost with $\mathcal{PSOPT}$. In *Proc. IEEE International Symposium on Computer-Aided Control System Design*, 1391–1396. Yokohama, Japan.

Biagiotti, L. and Melchiorri, C. (2008). *Trajectory Planning for Automatic Machines and Robots*. Springer.

Bobrow, J.E., Dubowsky, S., and Gibson, J.S. (1985). Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3), 3–17.

Bremer, H. (2008). *Elastic Multibody Dynamics: A Direct Ritz Approach*, volume 35 of *International Series on Intelligent Systems, Control, and Automation: Science and Engineering*. Springer.

Corke, P. (2013). *Robotics, Vision and Control*, volume 73 of *Springer Tracts in Advanced Robotics*. Springer.

Gill, P.E., Murray, W., and Saunders, M.A. (2002). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4), 979–1006.

Latombe, J.C. (1991). *Robot Motion Planning*. Springer.

Laumond, J.P. (ed.) (1998). *Robot Motion Planning and Control*, volume 229 of *Lecture Notes in Control and Information Sciences*. Springer.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press.

Messner, L., Gattringer, H., and Bremer, H. (2013). Efficient online computation of smooth trajectories along geometric paths for robotic manipulators. In H. Gattringer and J. Gerstmayr (eds.), *Multibody System Dynamics, Robotics and Control*, 17–30. Springer.

Pardo-Castellote, G. and Cannon Jr., R.H. (1996). Proximate time-optimal algorithm for on-line path parameterization and modification. In *Proc. IEEE International Conference on Robotics and Automation*, 1539–1546. Minneapolis, MN, USA.

Shiller, Z. and Dubowsky, S. (1991). On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6), 785–797.

Shin, K.G. and McKay, N.D. (1985). Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, AC-30(6), 531–541.

Siciliano, B. and Khatib, O. (eds.) (2008). *Springer Handbook of Robotics*. Springer.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer.

Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J., and Diehl, M. (2009). Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10), 2318–2327.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57.

Post-print version of the article: M. Böck, M. Plainer, and A. Kugi, "Evaluation of efficiently generating fast robot trajectories under geometric and system constraints", in *Proceedings of the 7th IFAC Symposium on Mechatronic Systems*, Sep. 2016, pp. 395–402. DOI: 10.1016/j.ifacol.2016.10.586