Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 50-1 (2017) 1937–1943

Efficient Generation of Fast Trajectories for Gantry Cranes with Constraints

Martin Böck* Andreas Stöger* Andreas Kugi*

* *Automation and Control Institute, TU Wien Gußhausstraße 27–29, 1040 Vienna, Austria (e-mail: {boeck,kugi}@acin.tuwien.ac.at, andreas.stoeger1@gmx.at).*

Abstract: Time is a crucial factor in the transport business. Besides the duration of the transport itself, also the loading and unloading of the goods is expected to be done as fast as possible to save valuable time and money. Amongst others, this holds true for container ships where the containers are usually loaded and unloaded by means of ship-to-shore gantry cranes. This scenario exemplarily motivates the investigations in this paper. Different methods are proposed for generating a trajectory for the load based on a geometric path connecting the loading and unloading position. As the path is usually just known right before the task has to begin, special emphasis lies on a fast calculation of the trajectory. The overall goal is to traverse the geometric path as fast as possible under the consideration of constraints for the gantry crane system. The optimal solution is calculated and serves as a reference for comparison reasons as the required computing time is usually rather large. Therefore, tailored methods for a fast calculation of the trajectory are developed. All the different approaches are evaluated and compared by means of representative test paths.

Keywords: Constraints, efficient algorithms, gantry crane, minimum-time control, optimal trajectory, trajectory planning.

1. INTRODUCTION

Time-optimal movements are required in many fields of engineering and for many different tasks. Plenty of solutions for finding such movements for generic dynamical systems as well as specific applications such as robots and vehicles have been developed, see, e.g., Faulwasser et al. (2011); Keerthi and Gilbert (1987); Kondak and Hommel (2001); Laumond (1998); LaValle (2006). The different approaches can be roughly divided into two classes. On the one hand, besides system constraints, the time-optimal trajectory only has to fulfill certain initial and terminal conditions, see, e.g., Knierim and Sawodny (2012); Van den Broeck et al. (2011). This means that the geometric form of the trajectory to be optimized is free. On the other hand, a geometric curve is given along which the time-optimal movement has to take place, see, e.g., Bobrow et al. (1985); Constantinescu and Croft (2000); Verscheure et al. (2009). In the following, such a curve without temporal information is also referred to as path. Essentially, the solution of the problem requires to find a time parameterization turning the path into a trajectory. In general, the two approaches can also be mixed in the form of a trajectory with geometrically predetermined sections and parts where the geometry is free, see, e.g., Böck et al. (2016).

In this paper, a task belonging to the second class is considered for a two-dimensional gantry crane. It is required to find a trajectory for moving the load as fast as possible from an initial configuration (position, velocity) to a terminal configuration along a predetermined, obstacle-free path. A path parameter describes the position along the path. It has to be determined as a function of time for solving the task. Apart from the application considered in this paper, the task is relevant in many other fields, as, e.g., robotics and UAVs. The design of a controller for tracking the obtained trajectory in the presence of, e.g., disturbances does not lie within the focus of this work. Nevertheless, a feedforward control for trajectory tracking in the nominal case can be readily obtained and used, e.g., within a two-degrees-of-freedom control structure, cf. Åström and Murray (2008); Egretzberger et al. (2012).

Many different approaches exist in literature for calculating time-optimal trajectories for gantry cranes. Auernig and Troger (1987) consider straight paths for which the analytical solution is obtained based on Pontryagin's maximum principle. In Van Loock et al. (2011), the trajectory is represented with B-splines and the flatness property of the system is utilized. For a constant cable length, satisfaction of certain constraints is ensured by using the convex hull property of B-spline curves. A bisection method is employed for finding the minimum time. In each iteration of the bisection method, a linear feasibility problem has to be solved. Similarly, Chen et al. (2016) also restrict their investigations to a constant cable length and parameterize the trajectory of the load with B-spline curves. A movement of the load in three-dimensional space is considered in Raczy and Jacob (1999). Constraints on various system variables are taken into account, amongst others for the length of the cable and its first derivative with respect to time. For computing the time-optimal trajectory along a given path, a multi-stage iterative strategy is proposed. Roughly speaking, only one constrained variable is considered and an appropriate trajectory is found. Subsequently, the remaining constraints are checked and it is determined where the trajectory has to be corrected. This procedure is repeated until all constraints are satisfied.

The solution strategies proposed in this paper do not rely on a constant cable length. Constraints for several different system variables are taken into account. For the developed solution methods, the load does not have to necessarily reside in equilibrium positions at the start and end of the trajectory. Furthermore, special emphasis is placed on the real-time feasibility of the proposed algorithms, which is often neglected in literature. From a practical point of view, this means that the time required for computing the trajectory has to be much smaller than the time needed for actually traversing it. Amongst others, this challenge is tackled by using a tailored representation of the path parameter with piecewise polynomials.

Section 2 gives a more detailed presentation of the gantry crane system and the considered task of trajectory generation. A method for calculating the optimal solution is outlined in Section 3. The main results are presented in Section 4 in the form of efficient (in terms of computing time) algorithms for obtaining fast trajectories. All the different methods are evaluated and compared in Section 5.

2. PROBLEM FORMULATION

All subsequent investigations in this paper are tailored to a gantry crane moving a load in two-dimensional space. This system, its mathematical model, and the corresponding constrained variables are presented in Section 2.1. The task of finding a trajectory for transferring the load as fast as possible is described in Section 2.2.

2.1 Gantry Crane

A sketch of the gantry crane is depicted in Fig. 1. The position of the load in the two-dimensional space is given by $\boldsymbol{p}_L = [x_L \ y_L]^{\mathrm{T}}$. The load is hanging on a cable which can be spooled on the trolley. It is assumed that the cable together with the load can be modeled as a mathematical pendulum. The length of the cable from the trolley to the load is denoted by $s_H < 0$. By using this convention, larger values of s_H entail larger values of y_L for practically relevant angular displacements θ of the cable. The cable is pivoted at a fixed height of $y_T = 48\,\mathrm{m}$ and a horizontal distance x_T describing the position of the trolley. All the dimensions are chosen to resemble those of a container crane for loading and unloading of cargo ships.

It is assumed that the actuated degrees of freedom x_T and s_H are equipped with velocity controllers for $v_T = \frac{\mathrm{d}x_T}{\mathrm{d}t} = \dot{x}_T$ and $v_H = \dot{s}_H$. Usually, these controllers perform well which allows to regard them as ideal. Hence, the (usually

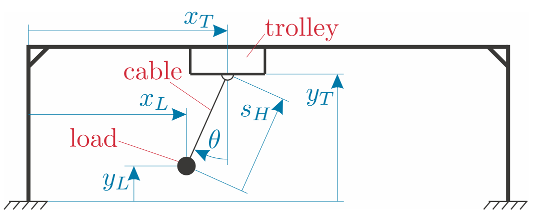

Fig. 1. Gantry crane.

changing) mass of the load as well as the friction in the actuated degrees of freedom are not relevant anymore for the mathematical model of the overall system. In particular with regard to these subordinate controllers, the velocities of the trolley v_T and the winch v_H as well as the corresponding accelerations $a_T = \dot{v}_T$ and $a_H = \dot{v}_H$ are subject to box constraints

$$v_{T,\min} \le v_T \le v_{T,\max} \qquad v_{H,\min} \le v_H \le v_{H,\max} \tag{1a}$$

$$a_{T,\min} \le a_T \le a_{T,\max} \qquad a_{H,\min} \le a_H \le a_{H,\max}. \tag{1b}$$

Similarly, the angle θ has to fulfill

$$\theta_{\min} \le \theta \le \theta_{\max} \tag{1c}$$

which can be motivated with the determination of θ by image processing and the load has to stay within the field of view of the camera.

As will be described in detail in Section 2.2, a trajectory $\boldsymbol{p}_L(t)$ has to be determined. It is a well-known fact that the gantry crane with the velocity controllers constitutes a flat system, see, e.g., Fliess et al. (1995). The position of the load $\boldsymbol{p}_L$ is a flat output which implies that all system variables can be parameterized in terms of $\boldsymbol{p}_L$ and a finite number of its derivatives with respect to time. Hence, such parameterizations are available for the constrained quantities in (1) as well in the form

$$[v_H \ a_H \ v_T \ a_T \ \theta]^{\mathrm{T}} = \boldsymbol{\psi}\left(\boldsymbol{p}_L, \dot{\boldsymbol{p}}_L, \ddot{\boldsymbol{p}}_L, \boldsymbol{p}_L^{(3)}, \boldsymbol{p}_L^{(4)}\right) \tag{2}$$

which will be useful for solving the trajectory generation task. In particular with regard to the subordinate velocity controllers for the trolley and the winch, it is advantageous if v_H, a_H, v_T, and a_T are continuous functions of time. Hence, it is required that the obtained trajectory for $\boldsymbol{p}_L$ is four times continuously differentiable ($\mathcal{C}^4$).

2.2 Task Specification

At a given start position $\boldsymbol{p}_{\text{Start}}$, the load (which may be a container) is picked up by the crane and has to be moved to a predetermined end position $\boldsymbol{p}_{\text{End}}$ where it is unloaded, see Fig. 2. For obtaining a suitable trajectory connecting $\boldsymbol{p}_{\text{Start}}$ and $\boldsymbol{p}_{\text{End}}$, a two-step procedure is used. Firstly, a suitable path has to be found which leads from $\boldsymbol{p}_{\text{Start}}$ to $\boldsymbol{p}_{\text{End}}$ and avoids obstacles. Secondly, a time parameterization of the path has to be determined. According to Section 2.1, the resulting trajectory for the load has to be element of $\mathcal{C}^4$ which means that the same has to hold for the path and the time parameterization.

For finding the path, waypoints $\boldsymbol{p}_{w,i}$, $i = 0, 1, \ldots, l$ are supposed to be given in such a way that they have enough distance to potential obstacles such as other containers on the cargo ship. For brevity, the start and end point are also included as the first and last waypoint $\boldsymbol{p}_{w,0} = \boldsymbol{p}_{\text{Start}}$ and

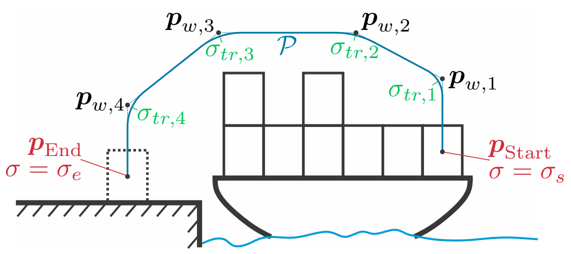

Fig. 2. Path from the start to the end position with $l = 5$.

$\boldsymbol{p}_{w,l} = \boldsymbol{p}_{\mathrm{End}}$, resp. A continuously differentiable path $\mathcal{P}_1$ is generated by sequentially connecting the waypoints with straight lines and inserting circle arcs at the corners. Subsequently, $\mathcal{P}_1$ is replicated with a four times continuously differentiable B-spline curve $\boldsymbol{\xi}(\sigma)$ with the path parameter $\sigma \in \mathcal{I} = [\sigma_s, \sigma_e]$ where σ_s and $\sigma_e > \sigma_s$ correspond to the start and end point. Hence, the final path results in $\mathcal{P} = \left\{ \boldsymbol{p}_L \in \mathbb{R}^2 \middle| \boldsymbol{p}_L = \boldsymbol{\xi}(\sigma), \sigma \in \mathcal{I} \right\}$. For convenience, it is assumed that $\mathcal{P}$ is (at least approximately) parameterized by the arc length.

The task to be solved is given by finding a time parameterization $\sigma(t) \in \mathcal{C}^4 : [0, T] \to [\sigma_s, \sigma_e]$ for the path parameter with T denoting the overall time needed for the trajectory, subsequently also referred to as the trajectory time. To this end, it is required that

(1) $\sigma(0) = \sigma_s$ and $\sigma(T) = \sigma_e$,
(2) the velocity at the start and end of the trajectory can be specified in the form $\dot{\sigma}(0) = \dot{\sigma}_s$ and $\dot{\sigma}(T) = \dot{\sigma}_e$,
(3) the trajectory time T is as small as possible,
(4) the constraints (1) are fulfilled, and
(5) the computing time T_{CPU} for the corresponding calculations is as small as possible, particularly $T_{\mathrm{CPU}} \ll T$.

The desired load trajectory results as $\boldsymbol{p}_L(t) = \boldsymbol{\xi}(\sigma(t))$.

From an economic point of view, the most important requirement is given by the third item. However, usually the computation of the trajectory has to be done right before it is traversed. Therefore, the time needed for completing the overall task is given by $T_{\mathrm{CPU}} + T$ motivating the fifth item. Naturally, the third and fifth item are always contradictory. Hence, the method for solving the task presented in Section 4 accepts a certain compromise between these two requirements in order to calculate a fast trajectory within a reasonable computing time.

3. OPTIMAL SOLUTION

For an assessment of the solution method in Section 4 and its achieved trajectory time, one way for calculating the optimal solution (i.e., the actual minimum time trajectory) regardless of the required computing time is shown in this section. To this end, it is proposed to solve the optimal control problem

$$\min_{\bar{\boldsymbol{\sigma}}(\cdot), u(\cdot), T} \quad T \tag{3a}$$

$$\text{s.t.} \quad \dot{\bar{\boldsymbol{\sigma}}} = [\bar{\sigma}_2 \ \bar{\sigma}_3 \ \bar{\sigma}_4 \ \bar{\sigma}_5 \ u]^{\mathrm{T}} \tag{3b}$$

$$\bar{\boldsymbol{\sigma}}(0) = [\sigma_s \ \dot{\sigma}_s \ \mathbf{0}]^{\mathrm{T}}, \ \bar{\boldsymbol{\sigma}}(T) = [\sigma_e \ \dot{\sigma}_e \ \mathbf{0}]^{\mathrm{T}} \tag{3c}$$

$$\boldsymbol{h}(\bar{\boldsymbol{\sigma}}(t)) \leq \mathbf{0} \quad \forall t \in [0, T] \tag{3d}$$

with the state $\bar{\boldsymbol{\sigma}} = \left[\sigma \ \dot{\sigma} \ \ddot{\sigma} \ \sigma^{(3)} \ \sigma^{(4)}\right]^{\mathrm{T}}$ and the artificial input $u = \sigma^{(5)}$. The integrator chain (3b) ensures that the optimal time parameterization is element of $\mathcal{C}^4$. The constraints (1) are respected via (3d) with the function

$$\boldsymbol{h}(\bar{\boldsymbol{\sigma}}(t)) = \begin{bmatrix} \boldsymbol{\psi}_{\min} - \boldsymbol{\psi}\left(\boldsymbol{p}_L, \dot{\boldsymbol{p}}_L, \ddot{\boldsymbol{p}}_L, \boldsymbol{p}_L^{(3)}, \boldsymbol{p}_L^{(4)}\right) \\ \boldsymbol{\psi}\left(\boldsymbol{p}_L, \dot{\boldsymbol{p}}_L, \ddot{\boldsymbol{p}}_L, \boldsymbol{p}_L^{(3)}, \boldsymbol{p}_L^{(4)}\right) - \boldsymbol{\psi}_{\max} \end{bmatrix} \tag{4}$$

and $\boldsymbol{\psi}_{\min/\max} = [v_{H,\dagger} \ a_{H,\dagger} \ v_{T,\dagger} \ a_{T,\dagger} \ \theta_\dagger]^{\mathrm{T}}_{\dagger=\min/\max}$. Furthermore, $\boldsymbol{p}_L = \boldsymbol{\xi}(\sigma(t))$, $\dot{\boldsymbol{p}}_L = \frac{\partial \boldsymbol{\xi}}{\partial \sigma}(\sigma(t))\dot{\sigma}(t)$, $\ddot{\boldsymbol{p}}_L = \frac{\partial^2 \boldsymbol{\xi}}{\partial \sigma^2}(\sigma(t))(\dot{\sigma}(t))^2 + \frac{\partial \boldsymbol{\xi}}{\partial \sigma}(\sigma(t))\ddot{\sigma}(t)$, and analogous expressions for $\boldsymbol{p}_L^{(3)}$ and $\boldsymbol{p}_L^{(4)}$ have to be inserted into $\boldsymbol{\psi}$ in (4).

For the results in Section 5, the solution of (3) is obtained by discretizing the horizon $[0, T]$ and using the values of $\bar{\boldsymbol{\sigma}}$ and u at the discrete points as optimization variables. An additional approximation of (3b) by means of the trapezoidal rule and evaluation of (3d) at the discrete points results in a finite-dimensional static optimization problem which is solved with SNOPT, see Gill et al. (2002). The obtained solution is regarded as the optimal solution.

4. EFFICIENT SOLUTION

Section 5 shows that the calculation of the optimal time parameterization requires a considerable computing time T_{CPU}. Hence, the goal of this section is to develop methods for achieving $T_{\mathrm{CPU}} \ll T$ at the expense of a larger trajectory time. However, the sum of $T_{\mathrm{CPU}} + T$ is still much smaller than for the method from Section 3 which means that the overall task can be completed in less time.

The basic idea is to restrict the shape of the function $\sigma(t)$ using a parameterization with piecewise polynomials being presented in Section 4.1. This reduces the number of degrees of freedom which is advantageously utilized in Section 4.2 for the formulation of a tailored optimization problem together with an efficient solution algorithm in order to decrease the required computing time.

4.1 Path Parameter Representation

The parameterization of $\sigma(t)$ is chosen in such a way that the possible movement along the path is not restricted much in view of the task to be solved. To this end, the overall path defined by $\boldsymbol{\xi}(\sigma)$ is split up into l segments. The ith segment, $i = 1, 2, \ldots, l$ comprises the interval $[\sigma_{tr,i-1}, \sigma_{tr,i}]$ of the path parameter with $\sigma_{tr,0} = \sigma_s$ and $\sigma_{tr,l} = \sigma_e$, cf. Fig. 2. The transition points $\sigma_{tr,i}$, $i = 1, 2, \ldots, l-1$ are determined such that $\boldsymbol{\xi}(\sigma_{tr,i})$ is as close as possible to the middle of the circle arc (forming $\mathcal{P}_1$) at waypoint $\boldsymbol{p}_{w,i}$.

Based on this definition, the path in each segment resembles a straight line and sections of the adjacent circle arcs. If just a straight line was present, the time-optimal movement would roughly consist of an acceleration phase, a phase with maximum velocity, and a deceleration phase. Based on this idea, a corresponding piecewise polynomial representing the path parameter is defined for each segment. By combining all segments, the overall piecewise polynomial $\sigma_p(t)$ results. The ith segment according to Fig. 3 with duration T_i consists of nine subsections $j = 1, 2, \ldots, 9$. They have durations $\Delta T_{i,j}$ with $\sum_{j=1}^{9} \Delta T_{i,j} = T_i$ which allow to define the points

$$T_{i,j} = \begin{cases} 0 & i=1, \quad j=1 \\ \sum_{m=1}^{j-1} \Delta T_{1,m} & i=1, \quad 2 \leq j \leq 10 \\ T_{i-1,10} + \sum_{m=1}^{j-1} \Delta T_{i,m} & 2 \leq i \leq l, \ 1 \leq j \leq 10 \end{cases} \tag{5}$$

with $T_{i,10} = T_{i+1,1}$, $i = 1, 2, \ldots, l-1$ and $T_{l,10} = T = \sum_{i=1}^{l} T_i$, see Fig. 3. At the beginning of each subsection, σ_p, $\dot{\sigma}_p$, and $\ddot{\sigma}_p$ have initial values $\sigma_{p,i,j}$, $\dot{\sigma}_{p,i,j}$, and $\ddot{\sigma}_{p,i,j}$, resp. For the subsections $j = 1, 5, 9$, the velocity along the path is constant resulting in

$$\sigma_p(t) = \sigma_{p,i,j} + \dot{\sigma}_{p,i,j}(t - T_{i,j}) \tag{6a}$$

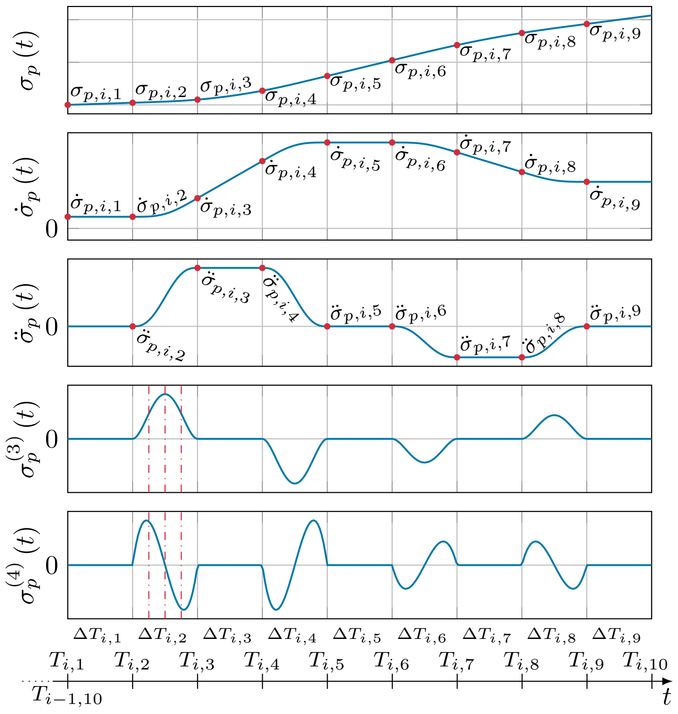

Fig. 3. Piecewise polynomial σ_p in the ith segment.

$$\dot\sigma_p(t) = \dot\sigma_{p,i,j} \tag{6b}$$
$$\ddot\sigma_p(t) = 0. \tag{6c}$$

The subsections $j = 3, 7$ are characterized by a constant acceleration in the form

$$\sigma_p(t) = \sigma_{p,i,j} + \dot\sigma_{p,i,j}(t - T_{i,j}) + \ddot\sigma_{p,i,j}\frac{(t - T_{i,j})^2}{2} \tag{7a}$$
$$\dot\sigma_p(t) = \dot\sigma_{p,i,j} + \ddot\sigma_{p,i,j}(t - T_{i,j}) \tag{7b}$$
$$\ddot\sigma_p(t) = \ddot\sigma_{p,i,j}. \tag{7c}$$

In order to enable that σ_p is four times continuously differentiable, the subsections $j = 2, 4, 6, 8$ comprise polynomials of the form

$$\sigma_p(t) = (\ddot\sigma_{p,i,j+1} - \ddot\sigma_{p,i,j})\left[\frac{\tilde t^5}{2\Delta T_{i,j}^3} - \frac{\tilde t^6}{2\Delta T_{i,j}^4} + \frac{\tilde t^7}{7\Delta T_{i,j}^5}\right] + \sigma_{p,i,j} + \dot\sigma_{p,i,j}\tilde t + \ddot\sigma_{p,i,j}\frac{\tilde t^2}{2} \tag{8a}$$

$$\ddot\sigma_p(t) = (\ddot\sigma_{p,i,j+1} - \ddot\sigma_{p,i,j})\left[\frac{10\tilde t^3}{\Delta T_{i,j}^3} - \frac{15\tilde t^4}{\Delta T_{i,j}^4} + \frac{6\tilde t^5}{\Delta T_{i,j}^5}\right] + \ddot\sigma_{p,i,j} \tag{8b}$$

with $\ddot\sigma_p$ still being two times continuously differentiable, $\tilde t = t - T_{i,j}$, and $\dot\sigma_p(t)$ omitted for brevity. For all functions (6)–(8), it holds that $t \in (T_{i,j}, T_{i,j+1}]$ except for $i = j = 1$ for which $T_{1,1}$ is included in the interval. Amongst others, the requirement of an (approximate) natural parameterization of $\mathcal{P}$ in Section 2.2 stems from the demand that the chosen representation of the path parameter is reasonable and well interpretable. For example, if $\left\|\frac{\partial \boldsymbol{\xi}}{\partial \sigma}\right\| \neq 1$ and not constant, then a constant value for $\dot\sigma$ does not entail a constant velocity along the path.

The overall degrees of freedom of $\sigma_p(t)$ for all segments and subsections are given by the durations $\Delta T_{i,j}$ as well as $\sigma_{p,i,j}$, $\dot\sigma_{p,i,j}$, $i = 1, 2, \ldots, l$, $j = 1, 2, \ldots, 9$ and $\ddot\sigma_{p,i,j}$ with $j = 2, 3, \ldots, 9$ as $\ddot\sigma_{p,i,1}$ does not occur. Naturally, they cannot be chosen freely but a number of conditions have to be fulfilled. Firstly, continuity within the segments has to be ensured resulting in

$$\sigma_{p,i,j} = \sigma_p(T_{i,j}) \text{ and } \dot\sigma_{p,i,j} = \dot\sigma_p(T_{i,j}) \tag{9a}$$

for $j = 2, 3, \ldots, 9$. Similarly, six conditions follow for the acceleration as

$$\ddot\sigma_{p,i,2} = \ddot\sigma_{p,i,5} = \ddot\sigma_{p,i,6} = \ddot\sigma_{p,i,9} = 0 \tag{9b}$$
$$\ddot\sigma_{p,i,4} = \ddot\sigma_{p,i,3} \qquad \ddot\sigma_{p,i,8} = \ddot\sigma_{p,i,7} \tag{9c}$$

with $\ddot\sigma_{p,i,3}$ and $\ddot\sigma_{p,i,7}$ remaining free. Secondly, continuity between the segments is ensured with

$$\sigma_{p,i+1,1} = \sigma_p(T_{i,10}) \text{ and } \dot\sigma_{p,i+1,1} = \dot\sigma_{p,i,9} \tag{10}$$

for $i = 1, 2, \ldots, l-1$. Such a condition is not necessary for $\ddot\sigma_p$ as it is zero at the beginning and end of each segment by (6c). Thirdly, the overall initial and terminal conditions have to be fulfilled requiring

$$\sigma_p(0) = \sigma_s,\ \dot\sigma_p(0) = \dot\sigma_s,\ \sigma_p(T) = \sigma_e,\ \dot\sigma_p(T) = \dot\sigma_e. \tag{11}$$

Fourthly, by demanding

$$\sigma_p(T_{i,10}) = \sigma_{tr,i}, \quad i = 1, 2, \ldots, l-1, \tag{12}$$

the segments are aligned with the path. The conditions (9)–(12) effectively reduce the number of degrees of freedom for $\sigma_p(t)$ to $10l - 1$.

These actual degrees of freedom are chosen as

$$\boldsymbol{Y} = [\Delta T_{1,1}\ \Delta T_{1,2}\ \cdots\ \Delta T_{1,9}\ \dot\sigma_{tr,1}\ \Delta T_{2,1}\ \cdots\ \dot\sigma_{tr,2}\ \cdots$$
$$\cdots\ \Delta T_{l-1,9}\ \dot\sigma_{tr,l-1}\ \Delta T_{l,1}\ \cdots\ \Delta T_{l,9}]^{\mathrm{T}} \in \mathbb{R}^{10l-1} \tag{13}$$

which will be motivated later. The quantities $\dot\sigma_{tr,i} := \dot\sigma_{p,i,9}$, $i = 1, 2, \ldots, l-1$ are the transition velocities between the segments. For fully determining $\sigma_p(t)$, relations

$$\sigma_{p,i,j} = f_{0j}(\sigma_{tr,i-1}, \dot\sigma_{tr,i-1}, \sigma_{tr,i}, \dot\sigma_{tr,i}, \Delta T_{i,1:9}) \tag{14a}$$
$$\dot\sigma_{p,i,j} = f_{1j}(\sigma_{tr,i-1}, \dot\sigma_{tr,i-1}, \sigma_{tr,i}, \dot\sigma_{tr,i}, \Delta T_{i,1:9}) \tag{14b}$$
$$\ddot\sigma_{p,i,j} = f_{2j}(\sigma_{tr,i-1}, \dot\sigma_{tr,i-1}, \sigma_{tr,i}, \dot\sigma_{tr,i}, \Delta T_{i,1:9}) \tag{14c}$$

exist where $\Delta T_{i,1:9}$ is a substitute for all $\Delta T_{i,j}$, $j = 1, 2, \ldots, 9$ and $\dot\sigma_{tr,0} = \dot\sigma_s$, $\dot\sigma_{tr,l} = \dot\sigma_e$.

4.2 Static Optimization Problem and Efficient Solution

The variables $\boldsymbol{Y}$ are used for the formulation of a tailored static optimization problem which aims at minimizing the trajectory time. To this end, two possible approaches are proposed. Firstly, the whole trajectory, i.e., all segments, can be optimized at once. Secondly, a finite number of segments in the form of a horizon $l_e - l_s + 1 = \Delta l \geq 1$ can be considered. The trajectory is just optimized for the l_sth up to the l_eth segment. In the following, only the second approach is described as the first one is included for $\Delta l = l$ and $l_s = 1$.

For finding the optimal trajectory, the constraints (1) have to be respected. This is implemented with barrier functions. Hence, in order to be able to apply a numeric optimization algorithm, a feasible trajectory acting as start solution is required. This start solution is proposed to be a trajectory with a small velocity $v_f \ll \min(|v_{T,\min}|, v_{T,\max}, |v_{H,\min}|, v_{H,\max})$. Based on intuition, v_f (being small enough) always exists such that all constraints (1) are fulfilled. Hence, for the variables $\boldsymbol{Y}$,

$$\dot\sigma_{tr,i} = v_f \qquad i = 1, 2, \ldots, l-1 \tag{15a}$$
$$\Delta T_{1,1} = \Delta T_{l,9} = 0 \tag{15b}$$

$$\Delta T_{1,j} = \frac{4}{3}\frac{\sigma_{tr,1}-\sigma_{tr,0}}{9}\frac{2}{\dot{\sigma}_{tr,0}+\dot{\sigma}_{tr,1}} \qquad j=2,3,4 \quad (15c)$$

$$\Delta T_{l,j} = \frac{4}{3}\frac{\sigma_{tr,l}-\sigma_{tr,l-1}}{9}\frac{2}{\dot{\sigma}_{tr,l-1}+\dot{\sigma}_{tr,l}} \qquad j=6,7,8 \quad (15d)$$

$$\Delta T_{i,j} = \left(\sigma_{tr,i}-\sigma_{tr,i-1}\right)/\left(9v_f\right) \quad (15e)$$

is set as a start solution, where (15e) holds for all i, j being not covered by (15b)–(15d). Essentially, (15e) sets the durations of the subsections according to v_f while (15c), (15d) account for the start and end velocities by averaging. During these velocity transitions, e.g., the acceleration limits may be violated which has to be checked separately. If necessary, corrections have to be made, e.g., by suitably adjusting the durations of the subsections.

For an optimization horizon Δl starting at l_s, just the corresponding elements Y_i, $i = 10(l_s-1)+1,\dots,10l_e-1$ are adapted and the other ones are set according to (15). Naturally, $\dot{\sigma}_{tr,i} \geq 0$ and $\Delta T_{i,j} \geq 0$ and therefore $\boldsymbol{Y} \geq \boldsymbol{0}$ has to hold. By setting $Y_i = X^2_{i-10(l_s-1)}$, $i = 10(l_s-1)+1,\dots,10l_e-1$ with X_i being the components of the vector of optimization variables $\boldsymbol{X} \in \mathbb{R}^{10\Delta l-1}$, this can be respected easily which is one advantage of the choice (13).

For considering the constraints (1), (4) is normalized by dividing its components h_i and h_{i+5} by $\psi_{\max,i}-\psi_{\min,i}$, $i = 1,2,3,4,5$ resulting in $\boldsymbol{h}_n(\bar{\boldsymbol{\sigma}}(t))$. Subsequently, the constraints are taken into account with barrier functions

$$f_b(\bar{\boldsymbol{\sigma}}(t)) = -\sum_{k=1}^{10}\lambda_k \log(-h_{n,k}(\bar{\boldsymbol{\sigma}}(t))) + \lambda_p\left(\sigma_p^{(4)}(t)\right)^2 \quad (16)$$

with the weighting factors $\lambda_k > 0$ and $\lambda_p > 0$. Naturally, $\bar{\boldsymbol{\sigma}}$ is now meant to contain σ_p and its derivatives. In (16), an additional penalty term for $\sigma_p^{(4)}$ is contained. It basically acts as a regularization term, especially for preventing the durations of subsections $2,4,6,8$ from being optimized to zero which would cause a jump in $\ddot{\sigma}_p$. As a result, also, e.g., a_H could jump which is unfavorable. Based on (14), $\sigma_p(t)$ according to (6)–(8) and its derivatives can be expressed as functions of $\boldsymbol{X}$ and t. By inserting these quantities into f_b, a function $F_b(\boldsymbol{X},t)$ can be obtained which allows to formulate the constrained minimization of the trajectory time as the unconstrained problem

$$\min_{\boldsymbol{X}} \int_{T_{l_s,1}}^{T_{l_e,10}} \left(1+F_b(\boldsymbol{X},t)\right)\mathrm{d}t. \quad (17)$$

Naturally, (17) needs to be suitably discretized which is done with four points $T_{i,j}+k\frac{\Delta T_{i,j}}{4}$, $k = 0,1,2,3$ per subsection j in the ith segment. By collecting these points in ascending order for all subsections $j = 1,2,\dots,9$ in the segments $i = l_s, l_s+1,\dots,l_e$ in the horizon and appending $T_{l_e,10}$, the vector $\boldsymbol{d} \in \mathbb{R}^{N_d}$ with $N_d = 36\Delta l+1$ is obtained. In view of the barrier functions contained in the cost functional of (17), four points per subsection are chosen such that all minimum and maximum values of $\sigma_p^{(3)}$ and $\sigma_p^{(4)}$ are (approximately) captured, see Fig. 3 where these points are marked with vertical dash-dot lines for the subsection $\Delta T_{i,2}$. The discretization of (17) is undertaken with the lower sum resulting in the unconstrained static minimization problem

$$\min_{\boldsymbol{X}} \quad T_{l_e,10} - T_{l_s,1} + \sum_{k=1}^{N_d-1}\left(d_{k+1}-d_k\right)F_b\left(\boldsymbol{X},d_k\right). \quad (18)$$

For the solution of (18), it is proposed to use the conjugate gradient (CG) method with utilizing the formula of Hestenes-Stiefel, the underlying line search based on the Armijo rule, and performing periodic restarts by setting the search direction to the steepest descent direction, see, e.g., Nocedal and Wright (2006). Regarding the CG method, a further advantage of the choice (13) emerges. Due to the local nature of (14), the calculation of the gradient of the cost function in (18) is possible in an efficient and numerically favorable way.

Based on the CG method, the optimal trajectory can be calculated for the whole path, i.e., all segments, at once by setting $\Delta l = l$ and $l_s = 1$. Although this solution strategy already significantly reduces the computing time compared to Section 3, an even more efficient strategy can be deduced. As already indicated above, a fixed number of $\Delta l < l$ segments are considered and the optimization can be performed in a receding horizon fashion. Initially, (18) is solved for $l_s = 1$ (i.e., the first horizon) based on the start solution (15). Afterward, $l_s = 2$ is set (the horizon is shifted forward) and (18) is solved again with the quantities for the first segment retaining their optimal values found previously. Then this optimization is repeated for $l_s = 3$ and segments one and two are left unchanged. These steps are iterated until $l_s = l$ is reached. If $l_e > l$ occurs, $l_e = l$ is set. In this way, the whole trajectory is gradually optimized. For all $l_s \geq 2$, a feasible start solution is also readily available by combining the optimal solution of the previous horizon on the last $\Delta l - 1$ segments with the start solution (15) for the l_eth segment. This is feasible as the transition velocity $\dot{\sigma}_{tr,l_e}$ at the end of each horizon is not optimized which entails that the optimal solutions are connected to the start solution (15). The fact that $\dot{\sigma}_{tr,l_e}$ is not optimized also brings about that $\Delta l = 1$ is only reasonable if $l = 1$. Otherwise, the variables $\dot{\sigma}_{tr,i}$, $i = 1,2,\dots,l-1$ would retain their conservative values from (15) resulting in a very slow trajectory.

The receding horizon strategy has several major advantages. Depending on the value of Δl, the number of optimization variables is significantly smaller compared to optimizing the whole trajectory at once which reduces the required computing time. Furthermore, the computing time is better predictable as the number of optimization variables is constant and not depending on the length of the path (i.e., the number of segments). In view of saving valuable time, the crane should start its movement immediately after the loading and unloading position and the corresponding path are known. However, due to the inevitable calculation of the trajectory, there is always a certain lag. In this regard, the receding horizon strategy also proves valuable as the crane movement can start right after the optimization for the first horizon Δl is completed. The second optimization for $l_s = 2$ can be done when the crane is already moving. Amongst others in this regard, it is crucial that at least one feasible solution is available for each horizon. However, based on the above description of obtaining feasible start solutions for each horizon, this is readily fulfilled. These solutions can also be used for the unlikely event that the iterations of the CG method do not converge.

For all strategies, the CG method can be executed until some kind of optimality condition is satisfied. However,

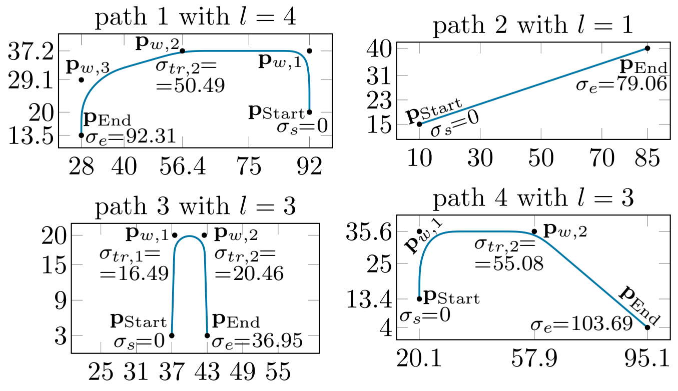

Fig. 4. Test paths plotted in the two-dimensional workspace of the crane (all quantities in m).

the corresponding computing time is hard to predict as the required number of iterations is unknown and varying. At this point, the utilization of barrier functions proves valuable as they ensure that the solutions in all iterations of the CG method are feasible. Hence, the CG method can be stopped at any number of iterations, e.g., to satisfy an upper limit for the computing time. Naturally, then a further trade-off regarding optimality has to be made. However, experience shows that the convergence close to the optimal value is often rather slow which means that the trade-off may not be significant.

5. RESULTS

This section aims at evaluating the methods from Sections 3 and 4. To this end, they are implemented in C-code and executed with MATLAB R2015b. All calculations are done on a 64 bit Windows 10 computer with an Intel Core i7-4700HQ CPU with four cores/eight threads but where no use of parallel computing features is made.

For obtaining the optimal solution according to Section 3, the optimization horizon is discretized with 400 points. The efficient methods from Section 4 use $\lambda_p = 0.01$, $\lambda_1 = \lambda_3 = \lambda_6 = \lambda_8 = 0.001$, $\lambda_2 = \lambda_5 = \lambda_7 = \lambda_{10} = 0.01$, and $\lambda_4 = \lambda_9 = 0.1$. The maximum number of iterations for the CG method is set to 600 for calculating the whole trajectory at once and 300 for the receding horizon solution. For the start solution, $v_f = 0.3\,\text{m}\,\text{s}^{-1}$ is chosen. The limits of the gantry crane are given by $v_{T,\max} = -v_{T,\min} = 4\,\text{m}\,\text{s}^{-1}$, $v_{H,\max} = -v_{H,\min} = 3\,\text{m}\,\text{s}^{-1}$, $a_{H,\max} = -a_{H,\min} = 0.75\,\text{m}\,\text{s}^{-2}$, $a_{T,\max} = -a_{T,\min} = 0.67\,\text{m}\,\text{s}^{-2}$, and $\theta_{\max} = -\theta_{\min} = 3°$.

For the evaluations, four representative paths according to Fig. 4 are considered. While path 1 roughly resembles the scenario depicted in Fig. 2, path 2 is of a rather simple form. Path 3 has a tricky shape due to the narrow curves. In contrast to paths 1–3 with $\dot{\sigma}_s = 0$, $\dot{\sigma}_s = 1.5\,\text{m}\,\text{s}^{-1}$ is given for path 4. For all four paths, $\dot{\sigma}_e = 0$ holds.

The trajectory times and the required computing times of the optimal solution for all four paths are listed in Table 1. Especially for path 1, the required computing time is of the same order as the obtained trajectory time. This underlines that the method of Section 3 is not suitable for a fast calculation of the trajectories right before they are executed. On the contrary, this does not

Table 1. Optimal solution and efficient solution for $l_s = 1$ and $\Delta l = l$.

	T in s		T_{CPU} in s	
	opt. sol.	eff. sol.	opt. sol.	eff. sol.
Path 1	34.3	36.9	31.3	1.28
Path 2	28.9	29.8	18.5	0.23
Path 3	22.6	25.0	17.2	1.10
Path 4	33.0	35.4	23.0	0.97

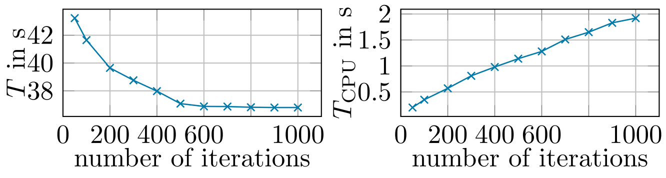

Fig. 5. T and T_{CPU} for the efficient solution with $\Delta l = l$ and $l_s = 1$ over the number of iterations for path 1.

apply to the proposed method from Section 4 for efficiently calculating the trajectories, even when this is done for the whole path at once, i.e., $\Delta l = l$ and $l_s = 1$. The corresponding trajectory and computing times are also shown in Table 1. Although the obtained trajectory times are not much larger than the ones for the optimal solution, the reductions in T_{CPU} are tremendous. The total time $T_{\text{CPU}} + T$ needed for the overall task is roughly 40 % smaller than for the optimal solution.

Apart from path 2 where an optimality condition is satisfied, the CG method always stops by reaching the maximum number of iterations (600). In order to show that this does not significantly influence the achieved trajectory time, Fig. 5 depicts T and T_{CPU} over the number of iterations of the CG method for path 1. Clearly, after approx. 600 iterations, the obtained trajectory time does not become much smaller. This also fortifies the statement at the end of Section 4 that the convergence close to the optimal value is usually rather slow. On the contrary, the required computing time roughly increases linearly with the number of iterations which does not justify further iterations with a rather small decrease of T.

For illustration, Fig. 6 shows v_H, a_H, v_T, σ, $\dot{\sigma}$, and θ over time obtained by the optimal and the efficient solution for path 1. The horizontal dash-dot-dot lines represent the corresponding constraints. In principle, the quantities from the optimal and efficient solution look similar. However, the latter is not able to utilize the constraints to the same extent as the optimal approach which results in the larger trajectory time. Amongst others, this is due to the chosen polynomial representation of the path parameter. This representation (and the difference to the optimal solution) can be clearly seen, e.g., at around 18 s in the constant value of $\dot{\sigma}$. The efficient solution meets the constraints well apart from very small time intervals. This also justifies the chosen discretization.

Table 2 serves for illustrating the characteristics of the efficient method in receding horizon fashion according to Section 4. To this end, this method is executed for the paths 1, 3, and 4, each with $\Delta l = 2$ and $\Delta l = 3$ (path 2 is not relevant as $l = 1$). The CG method always terminates by reaching the maximum number of iterations (300), except for the cases where just one segment is optimized. Initially, T_{CPU} for $l_s = 1$ decides when the crane can start

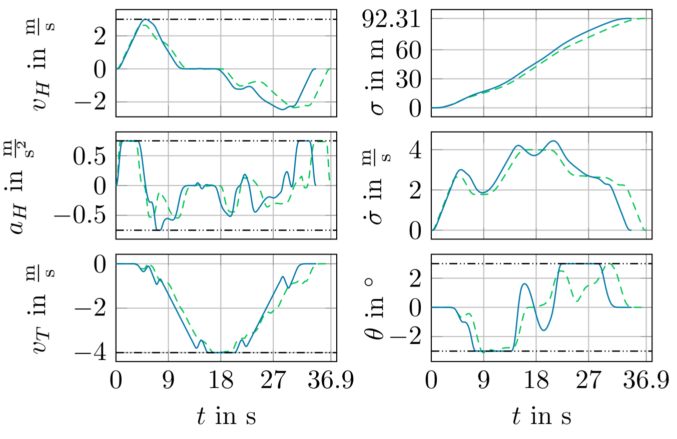

Fig. 6. Comparison of the optimal (solid lines) and the efficient solution (dashed lines) for $\Delta l = l$ and $l_s = 1$ for test path 1.

Table 2. Efficient solution, $\Delta l = 2$ and $\Delta l = 3$.

		T in s	T_{CPU} in s			
			$l_s = 1$	$l_s = 2$	$l_s = 3$	$l_s = 4$
Path 1	$\Delta l = 2$	38.6	0.32	0.43	0.36	0.26
	$\Delta l = 3$	36.8	0.49	0.34	0.27	0.04
Path 3	$\Delta l = 2$	25.4	0.46	0.54	0.19	-
	$\Delta l = 3$	25.3	0.50	0.30	0.07	-
Path 4	$\Delta l = 2$	38.2	0.42	0.57	0.03	-
	$\Delta l = 3$	35.5	0.45	0.35	0.03	-

to traverse the trajectory. This computing time is smaller by approx. 50 % or even more compared to Table 1 based on optimizing the whole trajectory at once. This is mainly due to the fact that only half as much iterations are carried out (300 vs. 600 iterations at most). Despite this fact, the obtained trajectory time for $\Delta l = 3$ is almost identical to the one from optimizing the whole trajectory at once, or even smaller for path 1, cf. Table 1. This comes from the incremental refinement done by the receding horizon approach inducing that most segments are optimized more than once (with the previous solution as starting point for the iterations). Naturally, this effect is weaker for $\Delta l = 2$ which provokes that T is larger than for $\Delta l = 3$. Hence, the choice $\Delta l = 3$ is quite reasonable and the receding horizon approach effectively enables to further reduce the time $T_{\mathrm{CPU}} + T$ required for the overall task.

REFERENCES

Åström, K.J. and Murray, R.M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

Auernig, J.W. and Troger, H. (1987). Time optimal control of overhead cranes with hoisting of the load. *Automatica*, 23(4), 437–447.

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.

Böck, M., Plainer, M., and Kugi, A. (2016). Evaluation of efficiently generating fast robot trajectories under geometric and system constraints. In *Proc. 7th IFAC Symposium on Mechatronic Systems*, 395–402. Loughborough, United Kingdom.

Chen, H., Fang, Y., and Sun, N. (2016). Optimal trajectory planning and tracking control method for overhead cranes. *IET Control Theory & Applications*, 10(6), 692–699.

Constantinescu, D. and Croft, E.A. (2000). Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of Robotic Systems*, 17(5), 233–249.

Egretzberger, M., Graichen, K., and Kugi, A. (2012). Flatness-based MPC and global path planning towards cognition-supported pick-and-place tasks of tower cranes. In H. Irschik, M. Krommer, and A.K. Belyaev (eds.), *Advanced Dynamics and Model-Based Control of Structures and Machines*, 63–71. Springer.

Faulwasser, T., Hagenmeyer, V., and Findeisen, R. (2011). Optimal exact path-following for constrained differentially flat systems. In *Proc. 18th IFAC World Congress*, 9875–9880. Milano, Italy.

Fliess, M., Lévine, J., Martin, P., and Rouchon, P. (1995). Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61(6), 1327–1361.

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.

Keerthi, S.S. and Gilbert, E.G. (1987). Computation of minimum-time feedback control laws for discrete-time systems with state-control constraints. *IEEE Transactions on Automatic Control*, AC-32(5), 432–435.

Knierim, K.L. and Sawodny, O. (2012). Real-time trajectory generation for three-times continuous trajectories. In *Proc. 7th IEEE Conference on Industrial Electronics and Applications*, 1462–1467. Singapore.

Kondak, K. and Hommel, G. (2001). Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms. In *Proc. IEEE International Conference on Robotics & Automation*, 2698–2703. Seoul, Korea.

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.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition.

Raczy, C. and Jacob, G. (1999). Fast and smooth controls for a trolley crane. *Journal of Decision Systems*, 8(3), 367–388.

Van den Broeck, L., Diehl, M., and Swevers, J. (2011). A model predictive control approach for time optimal point-to-point motion control. *Mechatronics*, 21(7), 1203–1212.

Van Loock, W., Pipeleers, G., and Swevers, J. (2011). Optimal input design for flat systems using *B*-splines. In *Proc. American Control Conference*, 3281–3282. San Francisco, CA, USA.

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.