

Recursive Log-barrier Method for On-line Time-optimal Robot Path Tracking

Diederik Verscheure*, Moritz Diehl[†], Joris De Schutter*, and Jan Swevers*

Abstract—This paper focuses on time-optimal robot path tracking and develops an approximate, log-barrier batch solution method to rapidly solve discretized, convexly reformulated path tracking problems. Based on this batch solution method, which results in smooth actuator torques, a recursive variant is derived for on-line path tracking. The results and trade-offs in calculation time, delay and path duration are compared for the batch and recursive variant of the log-barrier method as well as for an exact solution method, by means of an experimental test case of a robot carrying out a writing task, in which the path data is generated on-line by human demonstration.

I. INTRODUCTION

TIME-OPTIMAL path tracking, which involves planning of robot motions along prescribed geometric paths, is of significant importance for industrial applications, such as robotic welding, gluing and painting, as well as for applications such as programming by human demonstration, and target interception and capturing, but is a complex task, due to the nonlinear, coupled robot dynamics [1].

Methods for time-optimal path tracking subject to actuator constraints have been proposed in [2]–[6] and most of them exploit that motion along a predefined path can be described by a single path coordinate s and its time derivative \dot{s} [2], [3]. However, these methods are all *off-line batch* methods and are generally considered to be intractable for on-line use [7].

In contrast, this paper focuses on *on-line time-optimal path tracking*. Starting from a discretized convex reformulation of time-optimal path tracking problems presented in [6], this paper devises a log-barrier batch solution method, which allows to rapidly obtain an *approximate solution* with smooth actuator torques. Subsequently, a recursive variant of the method is deduced for the case whereby the path data is generated *on-line*, enabling on-line near time-optimal path tracking. By means of an experimental test case, the results and trade-offs in calculation time, delay and path duration are compared for the batch and recursive variant of the log-barrier method as well as for the exact, time-optimal solution method presented in [6]. The test case considers the example of a robotic manipulator performing a writing task, whereby the path data is generated on-line by human demonstration.

The outline of this paper is as follows. Section II discusses the basic time-optimal path tracking problem and summarizes the discretized convex reformulation derived in [6].

D. Verscheure, J. De Schutter and J. Swevers are with the Division PMA, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, e-mail: <firstname>.<lastname>@mech.kuleuven.be.

M. Diehl is with the Division SCD, Department of Electrical Engineering (ESAT), Katholieke Universiteit Leuven, Belgium, e-mail: moritz.diehl@esat.kuleuven.be.

Subsequently, Section III presents a fast, approximate log-barrier batch method to solve this nonlinear optimization problem, while Section IV derives a recursive variant of the batch method for on-line path tracking. Section V applies the batch and recursive variant of the log-barrier method, as well as the exact, time-optimal solution method presented in [6] to the example of a six-DOF KUKA 361 industrial robotic manipulator carrying out a writing task. Finally, conclusions and future work are given in Section VI.

II. PROBLEM FORMULATION

This section introduces the problem formulation and briefly recapitulates the discretized convex reformulation of time-optimal path tracking problems, derived in [6]. Consider an n -DOF robotic manipulator with equations of motion

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_s(\mathbf{q})\text{sgn}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}), \quad (1)$$

where \mathbf{q} , $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\boldsymbol{\tau} \in \mathcal{R}^n$ are the joint angles, their time-derivatives and the actuator torques respectively, and where $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, $\mathbf{F}_s(\mathbf{q})$ and $\mathbf{G}(\mathbf{q})$ represent a mass matrix, a matrix accounting for Coriolis and centrifugal effects, a matrix of Coulomb friction torques and a vector accounting for gravity and other joint angle dependent torques respectively.

Consider a path $\mathbf{q}(s)$, given in joint space coordinates, as a function of a scalar path coordinate $s \in [0, S]$. The path coordinate determines the spatial geometry of the path, whereas the trajectory's time dependency follows from the relation $s(t)$ between the path coordinate s and time t . Since this paper considers *time-optimal* path tracking, it is assumed that $\dot{s}(t) \geq 0$ everywhere and $\dot{s}(t) > 0$ *almost* everywhere.

For the given path, the joint velocities and accelerations can be rewritten using the chain rule as

$$\dot{\mathbf{q}}(s) = \mathbf{q}'(s)\dot{s}, \quad (2)$$

$$\ddot{\mathbf{q}}(s) = \mathbf{q}'(s)\ddot{s} + \mathbf{q}''(s)\dot{s}^2, \quad (3)$$

where $\dot{s} = \frac{ds}{dt}$, $\ddot{s} = \frac{d^2s}{dt^2}$, $\mathbf{q}'(s) = \frac{\partial \mathbf{q}(s)}{\partial s}$ and $\mathbf{q}''(s) = \frac{\partial^2 \mathbf{q}(s)}{\partial s^2}$.

Substituting $\dot{\mathbf{q}}(s)$ and $\ddot{\mathbf{q}}(s)$ based on equations (2)–(3), the equations of motion (1) can be rewritten as [2]

$$\boldsymbol{\tau}(s) = \mathbf{m}(s)\ddot{s} + \mathbf{c}(s)\dot{s}^2 + \mathbf{g}(s), \quad (4)$$

where

$$\mathbf{m}(s) = \mathbf{M}(\mathbf{q}(s))\mathbf{q}'(s), \quad (5)$$

$$\mathbf{c}(s) = \mathbf{M}(\mathbf{q}(s))\mathbf{q}''(s) + \mathbf{C}(\mathbf{q}(s), \mathbf{q}'(s))\mathbf{q}'(s), \quad (6)$$

$$\mathbf{g}(s) = \mathbf{F}_s(\mathbf{q}(s))\text{sgn}(\mathbf{q}'(s)) + \mathbf{G}(\mathbf{q}(s)), \quad (7)$$

and where $\text{sgn}(\dot{\mathbf{q}}(s))$ is replaced by $\text{sgn}(\mathbf{q}'(s))$ using equation (2) and the assumption that $\dot{s} > 0$ almost everywhere.

A. Optimal control problem formulation

Similarly as in [2], [3], the time-optimal path tracking problem subject to lower and upper bounds $\underline{\tau}(s)$ and $\overline{\tau}(s)$ on the torques, can be expressed as

$$\min_{T, s(\cdot), \tau(\cdot)} T = \int_0^T 1 dt, \quad (8)$$

$$\text{s.t. } \tau(t) = \mathbf{m}(s(t))\dot{s}(t) + \mathbf{c}(s(t))\dot{s}(t)^2 + \mathbf{g}(s(t)), \quad (9)$$

$$s(0) = 0 \text{ and } s(T) = S, \quad (10)$$

$$\dot{s}(0) = \dot{s}_0 \text{ and } \dot{s}(T) = \dot{s}_T, \quad (11)$$

$$\dot{s}(t) \geq 0, \quad (12)$$

$$\underline{\tau}(s(t)) \leq \tau(t) \leq \overline{\tau}(s(t)), \quad (13)$$

$$\text{for } t \in [0, T],$$

where T is the path duration, and \dot{s}_0 and \dot{s}_T are the initial and final velocity along the path, usually chosen equal to 0.

B. Discretized convex optimization problem formulation

As shown in [6], based on a nonlinear change of variables $a(s) = \ddot{s}$ and $b(s) = \dot{s}^2$, and a change of integration variable in objective function (8)

$$T = \int_0^T 1 dt = \int_0^S \frac{1}{\dot{s}} ds, \quad (14)$$

problem (8)-(13) can be reformulated as a *convex* optimal control problem in which time t does not appear anymore, while s acts as a pseudo-time, $b(s)$ as the differential state, $a(s)$ as the control input and $\tau(s)$ as algebraic state variable.

Using direct transcription [8], which involves discretizing the path coordinate s on $K + 1$ grid points $s^0 = 0 \leq s^k \leq S = s^K$ and modeling the functions $b(s)$, $a(s)$ and $\tau_i(s)$ by a finite number of variables b^k , a^k , τ_i^k , this convex optimal control problem can be discretized to obtain a large sparse optimization problem. Similar as in [6], the functions $b(s)$, $a(s)$ and $\tau_i(s)$ are assumed to be piecewise linear, piecewise constant, and piecewise nonlinear respectively, while the variables $b^k = b(s^k)$ are assigned on the grid points s^k , and $a^k = a(s^{k+1/2})$ and $\tau_i^k = \tau_i(s^{k+1/2})$ are assigned in between the grid points s^k , where $s^{k+1/2} = (s^k + s^{k+1})/2$. Based on this transcription scheme, the following large scale nonlinear optimization problem is obtained [6]:

$$\min_{\mathbf{a}, \mathbf{b}, \boldsymbol{\tau}} \sum_{k=0}^{K-1} \frac{2\Delta s^k}{\sqrt{b^{k+1}} + \sqrt{b^k}}, \quad (15)$$

$$\text{s.t. } \boldsymbol{\tau}^k = \mathbf{m}(s^{k+1/2})a^k$$

$$+ \mathbf{c}(s^{k+1/2})\frac{(b^k + b^{k+1})}{2} + \mathbf{g}(s^{k+1/2}), \quad (16)$$

$$b^0 = \dot{s}_0^2 \text{ and } b^K = \dot{s}_T^2, \quad (17)$$

$$(b^{k+1} - b^k) = 2a^k \Delta s^k, \quad (18)$$

$$b^k \geq 0 \text{ and } b^K \geq 0, \quad (19)$$

$$\underline{\tau}(s^{k+1/2}) \leq \tau^k \leq \overline{\tau}(s^{k+1/2}), \quad (20)$$

$$\text{for } k = 0 \dots K - 1,$$

where objective function (15) is obtained by analytically calculating (14) as a sum of integrals over $[s^k, s^{k+1}]$ for

$k = 0 \dots K - 1$, $\Delta s^k = s^{k+1} - s^k$, and \mathbf{a} , \mathbf{b} and $\boldsymbol{\tau}$ denote the K - and $(K + 1)$ -dimensional vectors and the $(n \times K)$ -dimensional matrix containing a^k , b^k and τ^k respectively.

III. LOG-BARRIER BATCH SOLUTION METHOD

By reformulating problem (15)-(20) as a second-order cone program (SOCP) [9] and applying an SOCP solver to the resulting problem as in [6], problem (15)-(20) can be solved efficiently and up to a high numerical accuracy. However, in practice, it is rarely necessary to determine control inputs up to a very high accuracy and in such a way that active bound constraints are satisfied up to equality [10]. Based on this consideration, a fast log-barrier based interior point method is devised in [10], to obtain an approximate solution to *quadratic* optimal control problems. This section extends the method in [10] to the nonlinear *non-quadratic* time-optimal path tracking problem (15)-(20).

A. Log-barrier optimization problem formulation

To simplify the log-barrier problem formulation, a^k and τ^k are first eliminated from problem (15)-(20), while b^0 and b^K are replaced by \dot{s}_0^2 and \dot{s}_T^2 , resulting in

$$\min_{\mathbf{b}} \sum_{k=0}^{K-1} \frac{2\Delta s^k}{\sqrt{b^{k+1}} + \sqrt{b^k}}, \quad (21)$$

$$\text{s.t. } b^k \geq 0, \text{ for } k = 1 \dots K - 1, \quad (22)$$

$$\underline{\tau}_i(s^{k+1/2}) \leq f_{c,i}^k(b^k, b^{k+1}) \leq \overline{\tau}_i(s^{k+1/2}), \quad (23)$$

$$\text{for } k = 0 \dots K - 1 \text{ and for } i = 1 \dots n,$$

where the variable vector \mathbf{b} is reduced to a $(K - 1)$ -dimensional vector containing b^k for $k = 1 \dots K - 1$ and

$$f_{c,i}^k(b^k, b^{k+1}) = m_i(s^{k+1/2})\frac{(b^{k+1} - b^k)}{2\Delta s^k} + c_i(s^{k+1/2}) \\ \times \frac{(b^k + b^{k+1})}{2} + g_i(s^{k+1/2}). \quad (24)$$

Subsequently, similar as in [10], problem (21)-(23) is approximated by an *unconstrained* optimization problem, by augmenting objective function (21) with log-barrier terms associated with constraints (23), and omitting constraints (22) and (23) as

$$\min_{\mathbf{b}} \sum_{k=0}^{K-1} f_b^k(b^k, b^{k+1}), \quad (25)$$

where

$$f_b^k(b^k, b^{k+1}) = \frac{2\Delta s^k}{\sqrt{b^{k+1}} + \sqrt{b^k}} + \frac{-\kappa}{2nK} \\ \times \sum_{i=1}^n \log \left[\left(\overline{\tau}_i(s^{k+1/2}) - f_{c,i}^k(b^k, b^{k+1}) \right) \right. \\ \left. \times \left(-\underline{\tau}_i(s^{k+1/2}) + f_{c,i}^k(b^k, b^{k+1}) \right) \right]. \quad (26)$$

Here, the tuning parameter κ , called the log-barrier parameter, is divided by the number of inequality constraints to define problem (25) in a grid-independent way, while the log-barrier terms for constraints (22) are omitted, since the

terms $\frac{2\Delta s^{k-1}}{\sqrt{b^k + \sqrt{b^{k-1}}}}$ and $\frac{2\Delta s^k}{\sqrt{b^{k+1} + \sqrt{b^k}}}$ in objective function (25) already act implicitly as barriers for $b^k \geq 0$.

B. Log-barrier batch solution method

Typical interior point methods determine a solution to problem (21)-(23), by repeatedly solving problem (25) using Newton's method, for progressively decreasing values of κ . However, similar as in [10], this paper solves problem (25) using a single fixed κ instead, to rapidly obtain an *approximate* solution $\mathbf{b}^{(\text{opt})}$ to problem (21)-(23).

Because of the convexity of problem (21)-(23), the globally optimal solution $\mathbf{b}^{(\text{opt})}$ to problem (25), for a given value of κ , is found by solving the first-order optimality conditions

$$\mathbf{r}(\mathbf{b}) = \mathbf{0}, \quad (27)$$

where element k of $\mathbf{r}(\mathbf{b})$ is given by

$$r_k(\mathbf{b}) = \frac{\partial f_b^{k-1}(b^{k-1}, b^k)}{\partial b^k} + \frac{\partial f_b^k(b^k, b^{k+1})}{\partial b^k}. \quad (28)$$

Starting from an initial guess $\mathbf{b}^{(0)}$, the solution to equations (27) is determined by successively linearizing equations (27) around the current iterate $\mathbf{b}^{(j)}$ and determining a search direction $\Delta \mathbf{b}^{(j)}$ as the solution to the linear system,

$$\mathbf{r}(\mathbf{b}^{(j)}) + \left. \frac{\partial \mathbf{r}(\mathbf{b})}{\partial \mathbf{b}} \right|_{\mathbf{b}=\mathbf{b}^{(j)}} \Delta \mathbf{b}^{(j)} = \mathbf{0}, \quad (29)$$

where element kl of the Hessian matrix $\frac{\partial \mathbf{r}(\mathbf{b})}{\partial \mathbf{b}}$ is given by

$$\frac{\partial r_k(\mathbf{b})}{\partial b^l} = \frac{\partial^2 f_b^{k-1}(b^{k-1}, b^k)}{\partial b^k \partial b^l} + \frac{\partial^2 f_b^k(b^k, b^{k+1})}{\partial b^k \partial b^l}. \quad (30)$$

Since $r_k(\mathbf{b})$ depends only on b^{k-1} , b^k and b^{k+1} , the Hessian matrix $\frac{\partial \mathbf{r}(\mathbf{b})}{\partial \mathbf{b}}$ is tri-diagonal. This tri-diagonal structure can be exploited to reduce the computational cost of solving (29) to $O(K)$ instead of $O(K^3)$ [10].

Subsequently, the updated iterate is calculated as $\mathbf{b}^{(j+1)} = \mathbf{b}^{(j)} + t\Delta \mathbf{b}^{(j)}$, where t is determined using a back-tracking line search [9] and such that constraints (23) are satisfied [9]. As soon as $\|\mathbf{r}(\mathbf{b}^{(j+1)})\| \leq \epsilon_{\text{abs}}$, where ϵ_{abs} is a suitably chosen threshold, the iterations are stopped and the iterate $\mathbf{b}^{(j+1)}$ is considered to be the solution to problem (25).

C. Initialization of the log-barrier batch solution method

Due to the elimination of a^k and τ^k , it is not straightforward to find an initial guess $\mathbf{b}^{(0)}$ which satisfies constraints (23). A simple and inexpensive initialization approach consists of choosing $\mathbf{b}^{(0)}$ as the function values of a parabola evaluated at the grid points s^k , with values \dot{s}_0^2 and \dot{s}_T^2 at $s = 0$ and $s = S$, and with a parameter \bar{b} . In other words,

$$(b^k)^{(0)} = -\bar{b} (s^k/S)^2 + (\dot{s}_T^2 - \dot{s}_0^2 + \bar{b}) (s^k/S) + \dot{s}_0^2. \quad (31)$$

Starting from some positive or negative value, \bar{b} can be progressively decreased, until $\mathbf{b}^{(0)}$ satisfies constraints (23). This approach is, although not guaranteed to work in general, successful in many cases, and *guaranteed* to work in the frequently encountered case where both \dot{s}_0^2 and \dot{s}_T^2 are equal to 0. In this case, \bar{b} can be chosen as some positive value, which is progressively decreased in absolute value until constraints (23) are satisfied.

D. Influence of the log-barrier parameter

The choice of the log-barrier parameter κ affects the quality of the approximate solution, as well as the calculation time of the log-barrier method. To demonstrate the effect of κ on the quality and optimality of the solution, the example of the elbow manipulator, originally introduced in [3], is considered and solved for five logarithmically spaced values of κ , namely 0.03, 0.12, 0.43, 1.60 and 5.94. The top part of Fig. 1 shows the evolution of the pseudo-velocity $\sqrt{b(s)} = \dot{s}$, while the bottom part shows the evolution of the torques τ in dashed lines for the different values of κ . Fig. 1 also shows the time-optimal solution in full lines, calculated using the exact SOCP method in [6]. For small κ , the log-barrier based solution is close to the time-optimal solution, whereas for increasing κ , the pseudo-velocity \dot{s} , as well as the torques τ , become increasingly smoother.

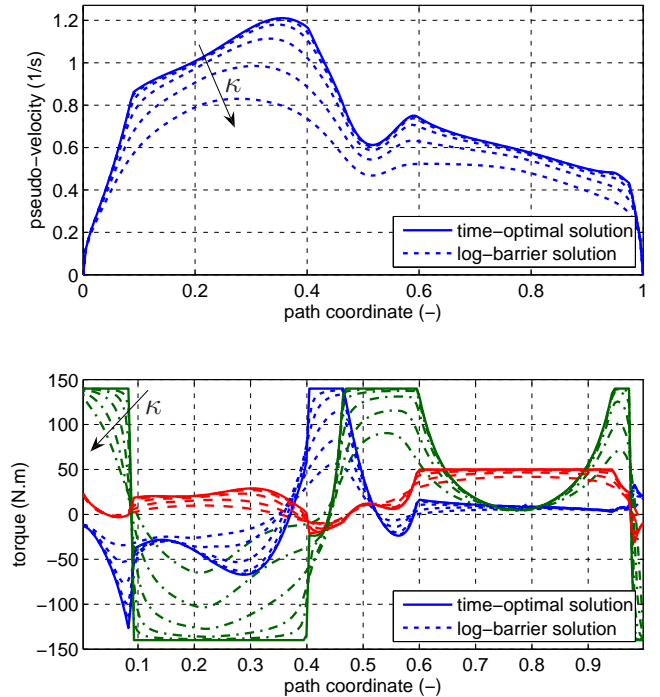


Fig. 1. The pseudo velocity \dot{s} (top) and the torques τ (bottom) as a function of the path coordinate s .

Fig. 2 shows the influence of κ on the optimality of the solution, represented by the path duration, normalized by the optimal path duration obtained using the SOCP method. The five values of κ between 0.03 and 5.94, corresponding to the solution represented by the dashed lines in Fig. 1, are also indicated. For small κ , for example $\kappa \approx 0.03$, the torques differ only slightly from the time-optimal solution (Fig. 1, bottom), while the path duration is nearly equal to the optimal path duration. For larger κ , for example $\kappa \approx 0.12$, the torques are already clearly different from the time-optimal solution (Fig. 1, bottom), although they only result in a 1.1% increase in path duration. For even larger κ , for example $\kappa > 1$ the increase in path duration is more rapid and grows beyond 10% of the optimal path duration.

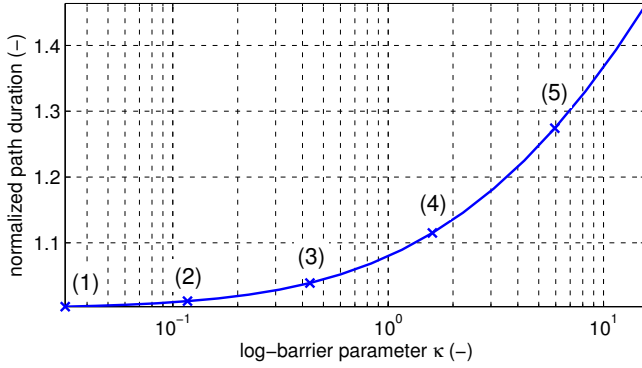


Fig. 2. The path duration as a function of the log-barrier parameter κ .

To illustrate the influence of κ on the calculation time, the log-barrier method is applied to the example of a six-DOF KUKA 361 industrial manipulator performing a complex writing task, which is introduced in [6]. The solution is calculated for different values of κ , which are logarithmically spaced between 0.03 and 320. Fig 3 shows the calculation time as a function of κ . For small κ , the log-barrier batch method calculates a solution which is close to the exact, time-optimal solution, and requires about 0.77 s to converge, whereas for increasing κ , the calculation time reduces spectacularly to 0.05 s and stabilizes around this value.

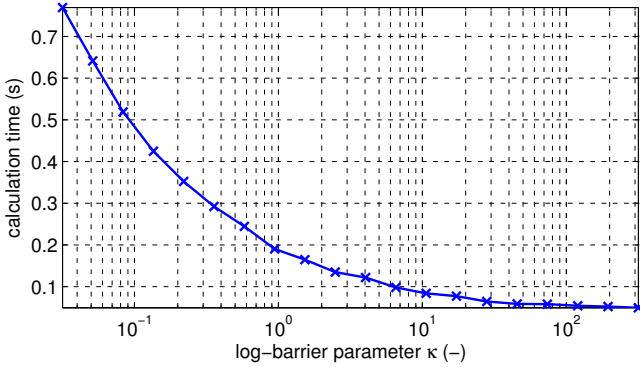


Fig. 3. The calculation time as a function of the log-barrier parameter κ .

These examples illustrate that, although the use of a single fixed κ results in an approximate solution, this solution can be obtained in considerably less calculation time. Furthermore, compared to the exact, time-optimal solution obtained using the SOCP method in [6], which is usually too aggressive for implementation on a real robotic manipulator without incorporation of additional smoothness constraints or penalty terms in the objective function [6], the solution obtained using the log-barrier method is much more practicable, and results in only a moderate increase in path duration. To rapidly obtain a solution with smooth actuator torques, values of κ around or larger than ten per cent of the optimal path duration are typically a good choice¹.

¹As explained in [9], [10], κ is an *upper bound* for the suboptimality of the solution, that is, the difference between the obtained path duration and the optimal path duration.

IV. LOG-BARRIER BASED RECURSIVE SOLUTION METHOD

The batch method presented in Section III allows to obtain a solution very fast. However, it requires all path data to be available in order to determine a solution. In applications such as target interception and capturing, where the path data is generated on-line based on incoming sensor information, it may be desirable to optimize the path tracking based on the already available path data, and have the robotic manipulator track the optimized part of the path as soon as possible. This is illustrated schematically in Fig. 4.

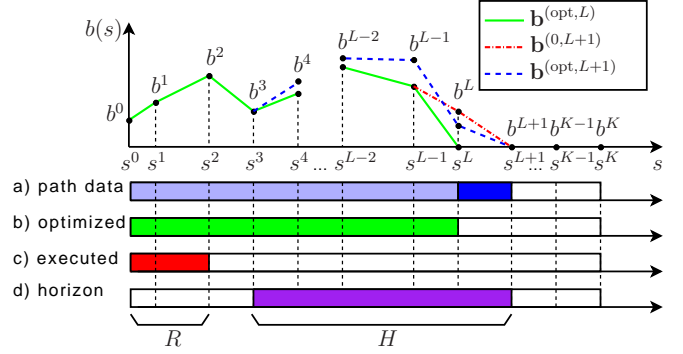


Fig. 4. The data, optimization and execution flow.

As soon as new path data comes in (Fig. 4, a), the path tracking optimization can be updated, while the robotic manipulator continues to execute the optimized part of the path (Fig. 4, b). Obviously, the path tracking optimization should not adjust the executed part of the path (Fig. 4, c). Compared to the previous solution, the addition of new path data has a noticeable impact on the new solution only over a certain horizon (Fig. 4, d). Specifically, the new solution tends to differ considerably from the previous solution near the *end* of the path, whereas closer to the start of the path, the new solution closely resembles the previous solution.

This section derives a recursive variant of the solution method in Section III, which takes into account these aspects and which allows to plan and execute the path tracking with almost zero delay. In addition, the initialization of \mathbf{b} is incremental, simple and guaranteed to work, while the selection of the horizon over which the optimization is updated, is carried out automatically.

A. Recursive log-barrier solution method

Consider a solution $\mathbf{b}^{(\text{opt},L)}$ to the approximate path tracking problem (25) for $s \in [0, s^L]$ with initial and terminal constraint $b^0 = \dot{s}_0^2$ and $b^L = 0$ respectively. Since the end of the path is not known beforehand, the terminal constraint $b^L = 0$ is imposed to ensure that a safe termination of the path is always possible, and Δs^k is chosen equal to 1 for $k \geq 0$. Starting from $\mathbf{b}^{(\text{opt},L)}$, the goal is to determine a solution $\mathbf{b}^{(\text{opt},L+1)}$ for $s \in [0, s^{L+1}]$ with initial and terminal constraint $b^0 = \dot{s}_0^2$ and $b^{L+1} = 0$ respectively.

To initialize $\mathbf{b}^{(0,L+1)}$, the previous solution $\mathbf{b}^{(\text{opt},L)}$ and some reasonable initial guess for $(b^L)^{(0,L+1)}$ can be used.

However, as the addition of a new point on the path may result in large differences between $\mathbf{b}^{(\text{opt},L)}$ and $\mathbf{b}^{(\text{opt},L+1)}$ near the end of the path, the last elements $\mathbf{b}^{(\text{opt},L)}$ are usually *not* good initial guesses for the last elements of $\mathbf{b}^{(\text{opt},L+1)}$. Therefore, solving for $\mathbf{b}^{(\text{opt},L+1)}$ using the method in Section III will result in a large number of iterations, which are *costly*, because for large L , the L -dimensional iterates $\mathbf{b}^{(j,L+1)}$ consists of a large number of variables. Hence, instead of solving for $\mathbf{b}^{(\text{opt},L+1)}$ directly, a series of optimization problems with an increasing horizon H is solved, which require fewer and fewer, but costlier and costlier iterations.

First, the horizon H is chosen small, for example $H = 1$. Subsequently, the first $L - H$ elements of $\mathbf{b}^{(0,L+1)}$ are assumed to be fixed, and a solution for the last H elements of $\mathbf{b}^{(\text{opt},L+1)}$ is determined using the method in Section III, resulting in a *large number* – due to the quality of the initial guess – of *cheap* – due to the small problem size – iterations. The last H elements of this partial solution are then used, as a new and *better* initial guess for a new optimization problem with an increased horizon pH , where for example $p = 3$, which can then be solved with costlier, but fewer – due to the quality of the new initial guess – iterations. This approach can be repeated until the horizon is expanded to L , or, when the robotic manipulator has already executed R points on the path, until the horizon is expanded to $L - R$. Since the addition of a new point on the path has a smaller effect further away from the end of the path, however, it is not always necessary to expand H up to L or $L - R$. Namely, as soon as the current solution $\mathbf{b}^{(\text{opt},L+1)}$ over a horizon H satisfies the termination criterion over a horizon $G > H$, for example²

$$\left\| \begin{pmatrix} r_{(L+1-G)}(\mathbf{b}^{(\text{opt},L+1)}) \\ \vdots \\ r_{(L)}(\mathbf{b}^{(\text{opt},L+1)}) \end{pmatrix} \right\|_{\infty} < \epsilon_{\text{abs}}, \quad (32)$$

it is sufficiently close to the solution over the entire horizon L . Here, $\|\cdot\|_{\infty}$ is the ∞ -norm or the maximum absolute value over the elements of the vector. In this way, the horizon can be automatically selected and limited in size. The recursive algorithm described above is given schematically by Algorithm 1 and illustrated in Fig. 4.

B. Initialization of the recursive log-barrier solution method

When a point is added to the path, a reasonable initial guess $(b^L)^{(0,L+1)}$ is required, while the other elements of $\mathbf{b}^{(0,L+1)}$ are initialized to the previous solution $\mathbf{b}^{(\text{opt},L)}$. Since $\mathbf{b}^{(\text{opt},L)}$ is the solution to a smaller problem, constraints (23) are guaranteed to be satisfied for $k = 0, \dots, L - 2$. Moreover, since $\mathbf{b}^{(\text{opt},L)}$ is the solution for the approximate path tracking problem (25) for $s \in [0, s^L]$, in combination with the terminal constraint $b^L = 0$, constraint (23) for $k =$

²In this paper, the ∞ -norm $\|\cdot\|_{\infty}$ is adopted as it is independent of the size of the residual vector $\mathbf{r}(\mathbf{b})$ and it guarantees that all residuals are sufficiently small.

Algorithm 1 The recursive algorithm.

initialize:

$L = 2, p = 3$

$(b^{L-1})^{(0,L)} = 1$

while $L \leq K$ **do**

 decrease $(b^{L-1})^{(0,L)}$ until $\mathbf{b}^{(0,L)}$ is feasible

$H = 1$

repeat

 determine $\mathbf{b}^{(\text{opt},L)} = \begin{pmatrix} (b^1)^{(0,L)} \\ \vdots \\ (b^{L-H-1})^{(0,L)} \\ (b^{L-H})^{(\text{opt},L)} \\ \vdots \\ (b^{L-1})^{(\text{opt},L)} \end{pmatrix}$

if $H = L - R$ **then**

break

else

$H = \min(L - R, pH)$

end if

until $\left\| \begin{pmatrix} r_{(L-H)}(\mathbf{b}^{(\text{opt},L)}) \\ \vdots \\ r_{(L-1)}(\mathbf{b}^{(\text{opt},L)}) \end{pmatrix} \right\|_1 < \epsilon_{\text{abs}}$

$\mathbf{b}^{(0,L+1)} = \begin{pmatrix} \mathbf{b}^{(\text{opt},L)} \\ (b^{L-1})^{(\text{opt},L)}/2 \end{pmatrix}$

 get next point on the path

 calculate $\mathbf{m}(s^{L+1/2})$, $\mathbf{c}(s^{L+1/2})$ and $\mathbf{g}(s^{L+1/2})$

$L = L + 1$

end while

$L - 1$ is guaranteed to be satisfied with *strict inequalities*³. Hence, it is always possible to initialize $(b^L)^{(0,L+1)}$ to a very small, but non-zero value such that (23) is satisfied for $k = L - 1$. Furthermore, in combination with the new terminal constraint $b^{L+1} = 0$, $(b^L)^{(0,L+1)}$ can be chosen small enough to also satisfy constraint (23) for $k = L$. Thus, provided that a feasible solution to the previous problem for $s \in [0, s^L]$ is available, a feasible initialization to the new problem for $s \in [0, s^{L+1}]$ can always be found. Namely, it suffices to initialize the last element of $\mathbf{b}^{(0,L+1)}$ to some reasonable value, for example $(b^{L-1})^{(\text{opt},L)}/2$, which is progressively decreased, until constraints (23) are satisfied for $k = 0, \dots, L$.

V. TEST CASE: WRITING TASK

To illustrate the practicality of the batch and the recursive variant of the log-barrier method discussed in Sections III and IV, they are applied to the example of a six-DOF KUKA 361 industrial manipulator performing a writing task (Fig. 5), whereby the path data is generated on-line by a human demonstrator using a Wacom tablet. Using the recursive algorithm explained in Section IV, the path tracking is optimized on-line and the robotic manipulator attempts to follow the human demonstrator as fast as possible.

³Since $\kappa > 0$ is chosen as a fixed value, equality of one of the constraints (23) for the solution to problem (25) would imply an infinite optimal value of function (25).

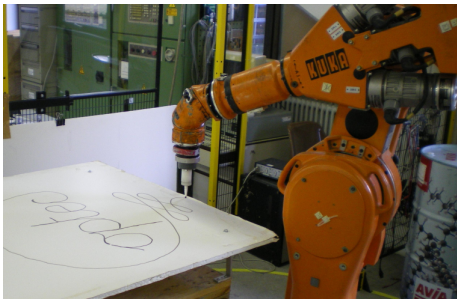


Fig. 5. A KUKA 361 industrial manipulator performing a writing task.

The path tracking is illustrated for the case whereby the demonstrator writes “Optec”⁴ on the Wacom tablet. As soon as the demonstrator starts writing, the path tracking optimization starts, while the robotic manipulator moves to the corresponding position and starts tracking the demonstrator. After writing “Op”, the demonstrator briefly pauses the writing, allowing the manipulator to catch up. Subsequently, the demonstrator resumes the writing, while the robotic manipulator tracks the human again. A video of the demonstration and execution has been made available online [11].

The path tracking solution obtained using the recursive log-barrier method is compared to the solution obtained using the batch variant, and to the exact, time-optimal solution determined using the SOCP method in [6]. The pseudo-velocity $\dot{s} = \sqrt{b(s)}$ obtained using the different methods is shown in the top part of Fig. 6, while the path time as a function of the path coordinate is shown in the bottom part of Fig. 6. Although in the implementation of the recursive method $\Delta s^k = 1$ for $k \geq 0$, the path coordinate is normalized in Fig. 6 to compare the different methods. Because the recursive method is not allowed to update the part of the path that has already been executed by the manipulator, it does not yield the same solution as the batch variant of the method. When the human demonstrator is far ahead of the manipulator, the solution produced by the recursive method resembles the batch solution closely, for example for $s = 0.05 \dots 0.3$ (Fig. 6, top), because there is a *long planning horizon*. When the manipulator is close to the human demonstrator, the manipulator moves with a low, almost constant velocity along the path. This can be seen in the top part of Fig. 6 around $s = 0.3$, which is where the human slows down, and eventually pauses at around $s = 0.35$. The closer the manipulator is to the human on the path, the shorter the planning horizon, and the more conservative the path execution, because the velocity at the *end* of the path, which by definition coincides with the *position of the human* on the path, is imposed to be zero.

As shown in the bottom part of Fig. 6, the SOCP method realizes a shorter path duration than the log-barrier methods. In turn, the log-barrier batch method realizes a shorter path duration than the recursive variant, because the latter

is limited by the speed of the human demonstrator. For example, around $s = 0.3$, where the human demonstrator slows down, the path time of the recursive method increases considerably compared to that of the batch method. Furthermore, when the human demonstrator stops, around $s = 0.35$, the path tracking optimization does not receive new path data, such that the path time simply increases. When the human demonstrator starts moving and speeds up again, the path times of the recursive and batch variant again progress similarly.

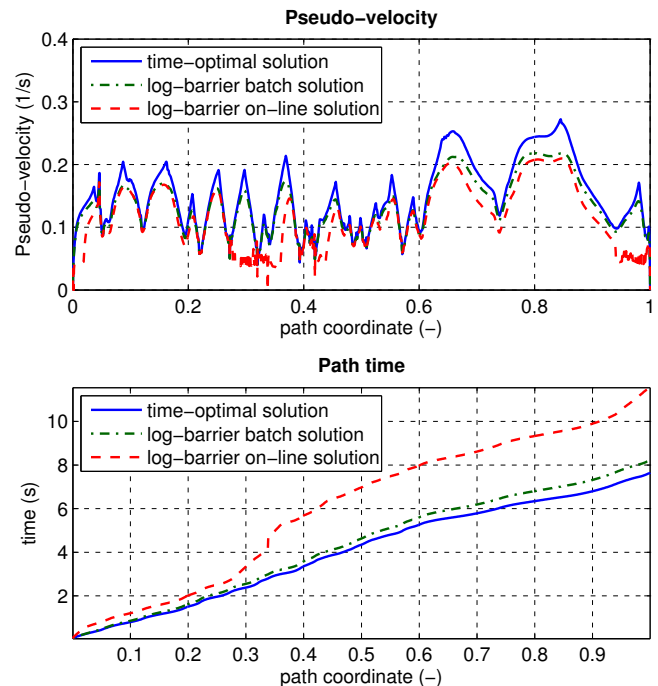


Fig. 6. The pseudo velocity \dot{s} (top) and the path time t (bottom) as a function of the path coordinate s .

As shown in Fig. 7, the torques obtained using the log-barrier methods are generally much smoother than the torques obtained using the exact solution method. However, when the human demonstrator writes very slowly, for example between $s = 0.27 \dots 0.34$, new points are added to the path very slowly and one by one. As a result, during this phase, the robot repeatedly moves forward only one point at a time, and waits for the next point on the path to become available. The path tracking optimization causes these incremental motions to be executed in a near time-optimal fashion, thereby resulting in a behavior of the torques which is somewhat comparable to bang-bang behavior, although considerably less aggressive.

For the considered test case, the duration of the human demonstration is equal to 9.373 s. When solving the problem ($K = 1436$) on an Intel Pentium M CPU running at 1.73 GHz, the SOCP method takes 4.900 s, the batch log-barrier method 0.086 s and the recursive log-barrier method 0.937 s. Although the batch variant is much faster than the recursive variant, it can only be applied after all path data is collected. As a result, there is a considerable *delay*

⁴Acronym for “Optimization in Engineering Center”. More information can be found on <http://homes.esat.kuleuven.be/~optec/>.

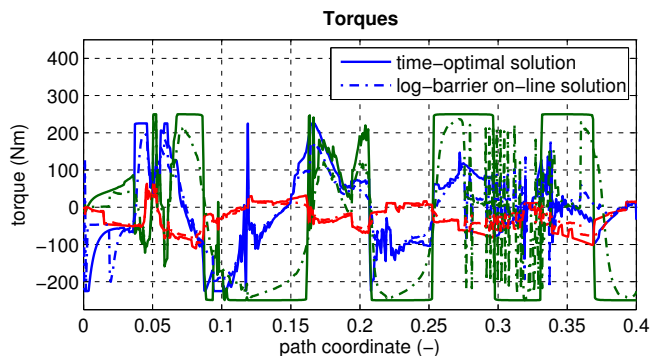


Fig. 7. The torques for the first three axes (blue: axis 1, green: axis 2, red: axis 3) as a function of the path coordinate s .

between the start of the human demonstration and the *end* of the path execution, which amounts to the sum of the *demonstration duration*, the *calculation time* and the *path duration* (Fig. 6, bottom), namely $9.373 + 0.086 + 8.004 = 17.463$ s. Conversely, for the recursive method, the delay between the start of the human demonstration and the start of the path execution is negligible, with an average calculation time per additional point on the path of $6 \cdot 10^{-4}$ s, such that the time between the *start* of the human demonstration and the *end* of the path execution amounts only to the path duration, namely 11.548 s (Fig. 6, bottom).

VI. CONCLUSIONS AND FUTURE WORK

Starting from a discretized convex reformulation of time-optimal path tracking problems, which is presented in [6], this paper presents a log-barrier batch solution method, which allows to rapidly obtain an *approximate solution* with smooth actuator torques. Compared to the exact solution obtained using the SOCP method in [6], the solution obtained using the log-barrier method is much more practicable, and results in only a modest increase in path duration. Using the log-barrier parameter, the smoothness of the torques can be easily adjusted, while increasing the smoothness allows to greatly reduce the calculation time.

To enable on-line path tracking, a recursive variant of the log-barrier batch method is deduced, which takes into account that the addition of new path data has an effect on the previous solution only over a limited horizon.

Experimental results show that the log-barrier batch method is much faster than the SOCP method presented in [6], while the recursive method is faster than the SOCP method, but slower than the batch method. The main advantage of the recursive method with respect to the log-barrier batch and SOCP method is that the path execution can start right away, thus considerably reducing the time between the *start* of the path generation and the *end* of the path execution.

Future work will focus on extending the log-barrier methods to deal with varying paths and to take into account uncertainty on the parameters of the equations of motion of the robotic manipulator. Furthermore, the zero-velocity constraint at the end of the path will be replaced by a less conservative velocity constraint, based on an estimate

of the speed of the human demonstrator, to improve the performance of the on-line path tracking.

ACKNOWLEDGMENT

The authors gratefully acknowledge Prof. Stephen Boyd of Stanford University for his encouragement, assistance and advice in implementing the method in [10] for the time-optimal path tracking problem. The authors gratefully acknowledge the financial support by K.U.Leuven's Concerted Research Action GOA/05/10, K.U.Leuven's CoE EF/05/006 Optimization in Engineering Center (OPTEC), and the Belgian Program on Interuniversity Poles of Attraction IAP VI/4 DYSCO (Dynamic Systems, Control and Optimization) initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture.

REFERENCES

- [1] Z. Shiller, "Interactive time optimal robot motion planning and work-cell layout design," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, Scottsdale, AZ, 1989, pp. 964–969.
- [2] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.
- [3] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 2, pp. 115–123, 1987.
- [4] Z. Shiller, "Time-energy optimal control of articulated systems with geometric path constraints," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, 1994, pp. 2680–2685.
- [5] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000.
- [6] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: a convex optimization approach," *IEEE Transactions on Automatic Control*, 2008, accepted.
- [7] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42–52, 2003.
- [8] L. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Computers and Chemical Engineering*, vol. 8, pp. 243–248, 1984.
- [9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. [Online]. Available: <http://www.ee.ucla.edu/~vandenbe/cvxbook.html>
- [10] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," in *IFAC World Congress*, Seoul, Korea, 2008, pp. 6974–6997.
- [11] D. Verscheure, "On-line time-optimal path tracking for robots," 2008. [Online]. Available: <http://homes.esat.kuleuven.be/~optec/software/onlinetimeopt/>