Time-Optimal Path Tracking: Online Scaling with Guarantees

Hung Pham, Quang-Cuong Pham
ATMRI, SC3DP, School of Mechanical and Aerospace Engineering
Nanyang Technological University, Singapore
Email: pham0074@e.ntu.edu.sg, cuong.pham@normalesup.org

Abstract—Given a geometric path, the Time-Optimal Path Tracking problem consists in finding the control strategy to traverse the path time-optimally while regulating tracking errors caused by model inaccuracies and online perturbations. Online Scaling (OS) is a simple yet effective approach to this problem. The overall OS controller is composed of two components: (i) a path controller, which modulates the parameterization of the desired path in an online manner, yielding a reference trajectory; and (ii) a torque controller, which takes the reference trajectory and outputs torque commands for tracking. However, OS has one major drawback: the path controller might not find any feasible reference trajectory that can be tracked by the torque controller because of torque constraints. In turn, this results in degraded tracking performances. Here, we propose a new controller, termed Online Scaling with Guarantees (OSG), which is guaranteed to find a feasible reference trajectory by accounting for all possible future perturbations. Our approach is based on Robust Reachability Analysis. Simulations show that the OSG controller outperforms existing methods.

I. INTRODUCTION

Time-optimal motion planning and control along a pre-defined path are fundamental and practically-important problems in robotics, motivated by many industrial applications, ranging from machining, to cutting, to welding, to painting, etc.

The planning version consists in finding the Time-Optimal Path Parameterization (TOPP) of a path under kinematics and dynamics bounds. The underlying assumption is that the robot is perfectly modeled and that no perturbation arises at execution time. This problem has been extensively studied since the 1980’s [1, 2, 3] for recent reviews.

The control version, which looks for a control strategy to time-optimally track the path while accounting for model inaccuracies and online perturbations, is comparatively less well understood. We refer to this problem as the Time-Optimal Path Tracking problem, or “tracking problem” in short.

A. Approaches to Time-Optimal Path Tracking

The main approach to address the tracking problem was first proposed by Dahl and colleagues in the 1990’s [4, 5]. Suppose that we are given a desired path \( \mathcal{P} := \mathbf{p}(s)_{s \in [0, 1]} \). In Dahl’s approach, termed Online Scaling (OS), the trajectory controller is composed of two sub-controllers: a torque controller and a path controller, see Fig. 1. The path controller acts on the path parameter \( s : t \mapsto s(t) \). At execution time, the path controller has access to the current state \((q, \dot{q})\) of the robot and generates “online” a reference \((q_d, \dot{q}_d, \ddot{q}_d)\) by modulating the path parameter \( s \) (“scaling”), formally

\[
q_d(t) := \mathbf{p}(s(t)), \quad \dot{q}_d(t) := \mathbf{p}'(s(t))\dot{s}(t), \quad \ddot{q}_d(t) := \mathbf{p}''(s(t))\ddot{s}(t), \quad \text{etc.} \tag{1}
\]

The torque controller then takes the reference \((q_d, \dot{q}_d, \ddot{q}_d)\) and generates the robot torques \( \tau \) to drive the current state to the reference state. In OS, one assumes that the torque controller is fixed, usually as a computed-torque controller with fixed Proportional-Derivative (PD) gains. Thus, the problem consists in designing the path controller that can regulate the path tracking errors while minimizing execution time.

Fig. 1. Block diagram of an Online Scaling controller.

There have been a number of developments to OS. In [6], the author proposed to use an observer to estimate the online constraints on the parameterization. In [7] and [8], OS was extended to handle manipulators with elastic joints or are subject to high-order dynamics such as torque rate or jerk.

The main problem of OS is that there is no guarantee for the path controller to find feasible controls at execution time. This issue is recognized in most of the papers devoted to OS. For example, in the original paper [4], the authors proposed to use the nominal control if there is no feasible control for the path controller. In a more recent work [7], the authors asserted that: “since [the path control] bounds are online evaluated [. . . ], it is not possible to guarantee [. . . that] a feasible solution exists [. . . ]”. Yet, employing arbitrary substitute controls when no feasible control exists will either cause emergency stopping or generate large path tracking errors. To make matters worse, this issue of empty control sets is far from rare since, by Pontryagin Maximum Principle, time optimality is associated with controls being saturated almost at every time instant.

Other than Online Scaling, a simpler approach to the tracking problem can be found in [9]. The authors proposed to consider more conservative torque bounds at the planning stage, “reserving” thereby some torque authority for the execution stage. This approach is however clearly sub-optimal.

At the other end of the spectrum, some authors considered the full optimal control problem, whose state is \((\mathbf{q}, \dot{\mathbf{q}}, s, \dot{s})\) and
whose control is \((\tau, \dot{s})\), and applied Nonlinear Model Predictive Control (NMPC) \cite{10, 11}. This approach has two major limitations. First, the time-optimality objective is challenging since it is non-convex in the time domain \cite{12}. Second, while NMPC controllers can account for hard constraints on state and control, ensuring stability, however, is non-trivial \cite{13}. In \cite{10}, to achieve stability, the path tracking NMPC controller requires hand-designed terminal sets.

B. Contribution and organization of the paper

To guarantee that the path controller will always find feasible controls requires a certain level of foresight: one needs to take into account all possible perturbations along the whole path. In this paper, we build on the recent formulation of TOPP by Reachability Analysis \cite{14} to provide such foresight. Specifically, we formulate the feasibility of OS as a minimax optimal control problem, which can be solved efficiently by Conic-Quadratic Programming (CQP) \cite{15}.

The rest of the paper is organized as follows. Section II provides the background on the tracking problem and on OS. Section III presents our approach in detail. Section IV reports experimental results, demonstrating the effectiveness of our approach. Finally, Section V concludes by discussing the advantages and drawbacks of our approach and sketches some directions for future research.

II. BACKGROUND: TIME-OPTIMAL PATH TRACKING AND ONLINE SCALING

A. Time-Optimal Path Tracking problem statement

Consider a \(n\)-dof manipulator, with dynamic equation

\[
\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{q}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{h}(\mathbf{q}) = \tau,
\]

where \(\mathbf{q} \in \mathbb{R}^n\) and \(\tau \in \mathbb{R}^n\) are vectors of joint positions and joint torques; \(\mathbf{M}()\), \(\mathbf{C}()\), \(\mathbf{h}()\) are tensors of appropriate dimensions. The torques are subject to the following constraints

\[
\tau_{\text{min}} \leq \tau \leq \tau_{\text{max}}.
\]

Consider now a smooth geometric path \(\mathcal{P}\), represented by a function \(\mathbf{p}(s), s \in [0, 1] \in \mathbb{R}^n\). A path parameterization is a scalar-valued function \(s(t) : [0, T_{\text{end}}] \mapsto [0, 1]\). One has the relations

\[
\begin{align*}
\mathbf{q}(t) &= \mathbf{p}(s(t)), \\
\dot{\mathbf{q}}(t) &= \mathbf{p}'(s(t))\dot{s}(t), \\
\ddot{\mathbf{q}}(t) &= \mathbf{p}''(s(t))\dot{s}^2(t) + \mathbf{p}'(s(t))\ddot{s}(t).
\end{align*}
\]

One also has the constraint on the terminal velocity

\[
\dot{s}(T_{\text{end}}) \in \mathbb{I}_{\text{end}}.
\]

To enforce zero terminal velocity, we define \(\mathbb{I}_{\text{end}} := \{0\}\).

The tracking error associated with a given parameterization \(s(t) : [0, T_{\text{end}}] \mapsto [0, 1]\) is defined as the difference between the state trajectory and the parameterized path

\[
e(t) := \begin{bmatrix} \mathbf{q}(t) - \mathbf{p}(s(t)) \\ \dot{\mathbf{q}}(t) - \mathbf{p}'(s(t))\dot{s}(t) \end{bmatrix}.
\]

We now define the Time-Optimal Path Tracking problem.

**Definition 1** (Time-Optimal Path Tracking). Design a control policy \(\pi : (\mathbf{q}, \dot{\mathbf{q}}) \mapsto \tau\) such that there exists a parameterization \(s^* : [0, T_{\text{end}}] \mapsto [0, 1]\) and

- the control torques \(\tau\) respect the constraint \(3\);
- the tracking error \(e(t)\) associated with \(s^*\) converges exponentially to zero;
- \(s^*(T_{\text{end}}) \in \mathbb{I}_{\text{end}}\);
- \(T_{\text{end}}\) is minimal.

B. Online Scaling

As introduced in the Introduction, a generic Online Scaling controller consists of two sub-controlers, namely a path controller \(\pi_p\) and the torque controller \(\pi_t\), see Fig. 4.

One possible choice for the torque controller \(\pi_t\) is the classical computed-torque controller \(16\)

\[
\pi_t : (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{e}) \mapsto \tau := \mathbf{M}(\mathbf{q})(\ddot{\mathbf{q}}_d - \mathbf{K}\mathbf{e}) + \mathbf{q}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{h}(\mathbf{q}),
\]

where \(\mathbf{K}\) is a fixed gain matrix. Without torque bounds, the tracking error \(e(t)\) converges exponentially to zero \(16\).

The path controller has an internal state \((s, \dot{s})\). It takes as input the current robot state \((\mathbf{q}, \dot{\mathbf{q}})\) and generates as output a path acceleration \(\ddot{s}\)

\[
\pi_p : (s, \dot{s}, \ddot{s}, \mathbf{q}) \mapsto \ddot{s},
\]

which is used to generate the path parameter \(s\). The reference trajectory \(\mathbf{q}_d\) (and its derivatives) can then be obtained from \(s\) by composition with the path function, cf. \(1\).

The main idea of OS consists in selecting \(\ddot{s}\) so as to satisfy the torque bounds \(3\). Plugging the path controller into the torque controller, one obtains the following constraints

\[
\tau_{\text{min}} \leq \mathbf{F}(s, \mathbf{e})\ddot{s} + \mathbf{G}(s, \dot{s}, \mathbf{e}) \leq \tau_{\text{max}},
\]

where

\[
\begin{align*}
\mathbf{F}(s, \mathbf{e}) &:= \mathbf{M}(\mathbf{q})\mathbf{p}'(s), \\
\mathbf{G}(s, \dot{s}, \mathbf{e}) &:= \mathbf{M}(\mathbf{q})\mathbf{p}''(s)s^2 - \mathbf{M}(\mathbf{q})\mathbf{K}\mathbf{e} + \mathbf{q}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{h}(\mathbf{q}).
\end{align*}
\]

Path accelerations that satisfy inequality \(7\) are said to be feasible. Note that, in the above expressions, \(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}\) can be expressed as a function of \(\mathbf{e}\) and the path using \(5\).

C. Difficulties associated with Online Scaling

As the OS controller tracks the given geometric path, there can be time instants when there is no feasible path acceleration, i.e. no \(\ddot{s}\) that satisfies \(7\). This means that there is no reference joint acceleration \(\ddot{\mathbf{q}}_d\) that will yield control torques \(\tau\) satisfying \(3\), which in turn implies that the tracking error might not converge to zero. In fact, experiments show that, in such cases, the closed-loop robotic system becomes temporarily unstable, see e.g. Fig. 4.

Thus, the main difficulty in OS is to guarantee that one can always find feasible accelerations. Another difficulty is that, even when a path controller \(\pi_p\) can always find feasible
accelerations, it might not be able to achieve the desired terminal velocity. The final difficulty is for \( \pi_T \) to be time-optimal. The next section addresses these difficulties.

III. TIME-OPTIMAL ONLINE SCALING WITH GUARANTEES

A. The discrete optimal robust control problem

We formulate the acceleration selection problem in OS as an optimal control problem for a discrete system subject to uncertain state-control constraints. For that, we first discretize the path into \( N + 1 \) stages

\[
0 := s_0, s_1, \ldots, s_N := 1.
\]

The state \( x_i \) and the control \( u_i \) at stage \( i \) are defined as

\[
x_i := \hat{s}_i^2, \quad u_i := \hat{s}_i.
\]

We thus have the transition function

\[
x_{i+1} = f_i(x_i, u_i) := x_i + 2\Delta_i u_i,
\]

where \( \Delta_i := s_{i+1} - s_i. \) We say that \( u_i \) “steers” \( x_i \) to \( x_{i+1}. \)

The terminal velocity constraint is enforced by requiring \( x_N \) to belong to the terminal set \( X_f := \{ x : \sqrt{x} \in \mathbb{I}_{\text{end}} \}. \)

At each stage, we are given a measurement of the tracking error \( e_i. \) Define next

\[
g_i(x_i, u_i, e_i) := \left[ \tau_{\min} - F(s_i, e_i)u_i + G(s_i, \sqrt{x_i}, e_i) \right],
\]

such that

\[
g_i(x_i, u_i, e_i) \leq 0 \iff (s_i, \hat{s}_i, \hat{s}_i, e_i) \text{ satisfies } (7).
\]

At a given stage \( i, \) for a state \( x_i \) and an error \( e_i, \) a control \( u_i \) is said to be feasible if \( g_i(x_i, u_i, e_i) \leq 0. \) Note that there are values of \( (x_i, e_i) \) such that no control is feasible.

A control policy is a sequence of control laws \( \pi := (\mu_0, \ldots, \mu_{N-1}) \), where each control law \( \mu_i \) maps the state and the measured error \( (x_i, e_i) \) to a control \( u_i. \)

A starting state \( x_0 \) is said to be robust feasible if given any realization of the error trajectory \( (e_0, \ldots, e_{N-1}), x_0 \) can be steered to \( X_f \) by a feasible control trajectory \( (u_0, \ldots, u_{N-1}). \) A policy is said to be robust feasible if it steers any robust feasible starting state to \( X_f \) by feasible controls.

Given a robust feasible policy \( \pi, \) the worst-case traversal time from a robust feasible initial state \( x_0 \) is defined as

\[
J_\pi(x_0) := \max_{(e_0, \ldots, e_{N-1})} \left\{ \sum_{i=0}^{N-1} \frac{2\Delta_i}{\sqrt{x_i} + \sqrt{x_{i+1}}} \right\}.
\]

The time-optimal robust feasible policy \( \pi^* \) can be finally defined as

\[
\pi^* := \arg\min_{\pi} J_\pi(x_0).
\]

This paper proposes to find the optimal policy \( \pi^* \) by (i) characterizing the class of robust feasible policies, and (ii) identifying the optimal robust feasible policy.

B. Robust Reachability Analysis

To characterize the class of robust feasible policies, we recursively compute the robust controllable sets at each stage, following and extending the development of [14].

Definition 2 (i-stage robust controllable set). The i-stage robust controllable set \( K_i \) is the set of states \( x_i \) such that, for all possible error trajectory \( (e_i, \ldots, e_{N-1}), \) there exists a feasible control trajectory \( (u_i, \ldots, u_{N-1}) \) that steers \( x_i \) to \( X_f \)

Fig. 2. For any sequence of perturbations, a state (green dot) in the 0-stage robust controllable \( K_0 \) set can always be steered towards the terminal set \( X_f. \)

Definition 3 (Robust one-step set). Given a target set \( \mathbb{I}, \) the i-stage robust one-step set \( Q_i(\mathbb{I}) \) is the set of states \( x_i \) such that, for any realization of the error \( e_i, \) there exists a feasible control \( u_i \) that steers \( x_i \) to \( \mathbb{I}. \)

Proposition 1. The i-stage robust controllable sets, for \( i \in \{0, \ldots, N\}, \) can be computed recursively by

\[
K_N = X_f, \quad K_i = Q_i(K_{i+1})
\]

The proof of this proposition can be found in the most notable references on Set Invariance Theory [17, 18].

The robust feasible policies can now be characterized.

Proposition 2. Assume that \( K_0 \) is non empty. Then, clearly, for all \( i \in \{0, \ldots, N-1\}, K_i \) is non empty. Next, by construction, given a state \( x_i \in K_i \) and a realization of the error \( e_i, \) there exists at least one control \( u_i \) such that

\[
g_i(x_i, u_i, e_i) \leq 0, \quad f_i(x_i, u_i) \in K_{i+1}.
\]

A policy \( \pi \) is robust feasible if and only if it always selects such controls.

The key to robust feasible control is therefore to compute the robust one-step set of Definition 3. This computation is however intractable in general because of the nonlinear coefficients \( F(\cdot), G(\cdot) \) hidden in the uncertain state-control constraints \( g. \) The next section proposes an approximation of the robust one-step set, allowing efficient computations.

C. Quadratic approximation of \( Q_i(\mathbb{I}) \)

We first make the assumption that the tracking errors are uniformly upper-bounded by a constant \( M \)

\[
\forall i \in \{0, \ldots, N-1\}, \quad e_i \in B(M),
\]

where \( B(M) \) is the ball of radius \( M \) in the Euclidean metric centered at the origin.
We propose computationally tractable approximation \( \hat{Q_i}(\mathbb{I}) \) of the robust one-step set \( Q_i(\mathbb{I}) \), which has two favorable properties. First, it is more conservative, i.e.

\[ \hat{Q_i}(\mathbb{I}) \subseteq Q_i(\mathbb{I}) \]

Second, in the limit of \( M \to 0 \), it is equal to the exact set.

We construct this approximation in two steps. Under the assumption of bounded tracking error, Definition 3 becomes

\[ Q_i(\mathbb{I}) := \{ x : \forall e_i \in B(M), \exists u_i : g(x_i, u_i, e_i) \leq 0, f_i(x_i, u_i) \in \mathbb{I} \} \]

The first step swaps the quantifiers of the tracking error and of the control, leading to a stricter condition on feasible states

\[ \hat{Q_i}(\mathbb{I}) := \{ x : \exists u_i, \forall e_i \in B(M), g(x_i, u_i, e_i) \leq 0, f_i(x_i, u_i) \in \mathbb{I} \} \]

It is easy to see that \( \hat{Q_i}(\mathbb{I}) \subseteq Q_i(\mathbb{I}) \) and for \( M = 0 \), equality holds. This step implies that one requires the existence of a single control \( u_i \) that satisfies all realizations of the errors.

Next, observe that \( \hat{Q_i}(\mathbb{I}) \) can be seen as the projection on the \( x \)-axis of the intersection of two sets

\[ \hat{Q_i}(\mathbb{I}) = \text{Proj}_x(F \cap G), \quad \text{where} \]

\[ F := \{ (x_i, u_i) : f_i(x_i, u_i) \in \mathbb{I} \}, \quad G := \{ (x_i, u_i) : \forall e_i \in B(M), g_i(x_i, u_i, e_i) \leq 0 \}. \]

The second step approximates \( G \) by replacing the nonlinear inequality \( g_i(\cdot) \) by 2n conic-quadratic constraints.

Consider the case \( M = 0 \): setting \( e_i = 0 \) in (8) yields the well-known linear constraints of TOPP [2, 13]

\[ g_i(x_i, u_i, 0) = a_i u_i + b_i x_i + c_i, \quad \text{(12)} \]

where \( a_i, b_i, c_i \) are 2n-dimensional vectors given by

\[ \begin{align*}
    a_i &= \begin{bmatrix} M(p(s_i)) \\ -M(p(s_i)) \end{bmatrix}, \\
    b_i &= \begin{bmatrix} M(p(s_i)) p'(s_i) + p'(s_i)^T C(p(s_i)) p'(s_i) \\ -M(p(s_i)) p'(s_i) + p'(s_i)^T C(p(s_i)) p'(s_i) \end{bmatrix}, \\
    c_i &= \begin{bmatrix} h(p(s_i)) - \tau_{\text{max}} \\ -h(p(s_i)) + \tau_{\text{min}} \end{bmatrix}. 
\end{align*} \]

Next, for small positive \( M \), the first order Taylor approximation of the \( k \)th constraint yields

\[ g_i[k](x_i, u_i, e_i) \approx g_i[x_i, u_i, 0] + \frac{\partial g_i[k]}{\partial e_i} x_i, u_i, 0 e_i, \leq g_i[x_i, u_i, 0] + M \left\| \frac{\partial g_i[k]}{\partial e_i} x_i, u_i, 0 \right\|_2. \]

Notice that we are only interested in approximating \( g_i[k] \) for \( (x_i, u_i) \in G \), which in practice can usually be assumed to be bounded. Hence it is always possible to find an upper-bound of the smooth function \( \left\| \frac{\partial g_i[k]}{\partial e_i} x_i, u_i, 0 \right\|_2 \) as

\[ \left\| \frac{\partial g_i[k]}{\partial e_i} x_i, u_i, 0 \right\|_2 \leq \left\| P_i[k] u_i \right\|_2, \quad \text{(13)} \]

where \( P_i[k] \) is a positive definite matrix. The procedures used to identify \( P_i[k] \) are given in Appendix A.

Substituting (13) into (12), we obtain 2n conic-quadratic constraints: for \( k \in [1, \ldots, 2n] \),

\[ [a_i[k], b_i[k], c_i[k]] \begin{bmatrix} u_i \\ x_i \end{bmatrix} + M \left\| P_i[k] u_i \right\|_2 \leq 0. \quad \text{(14)} \]

Letting \( \hat{G} \) be the set of pairs \((x_i, u_i)\) that satisfy (14), we obtain the quadratic approximation of the robust-one step set as

\[ \hat{Q_i}(\mathbb{I}) := \text{Proj}_x(F \cap \hat{G}). \]

We note the following useful fact: suppose \( \mathbb{I} \) is an interval, then \( \hat{Q_i}(\mathbb{I}) \) is also an interval. This can be shown by noting that \( F \) is convex as it is the inverse affine mapping of an interval and \( \hat{G} \) is a quadratic cone, which is also convex. Hence, to compute \( \hat{Q_i}(\mathbb{I}) \) one only needs to solve two Conic-Quadratic optimization programs to obtain the two endpoints.

Finally, we recursively construct quadratic approximations of the robust controllable sets by

\[ \hat{K}_N = X_f, \quad \hat{K}_i = \hat{Q_i}(\hat{K}_{i+1}). \quad \text{(15)} \]

Clearly the sets \( \hat{K}_0, \ldots, \hat{K}_N \) are intervals.

\section{D. The time-optimal robust feasible policy}

Based on the approximate robust controllable sets \( \hat{K}_0, \ldots, \hat{K}_N \), we characterize the class of approximate robust feasible policies \( \hat{\Pi} \) by the same reasoning as in Proposition 2. The following proposition identifies the optimal policy in \( \hat{\Pi} \).

\begin{proposition}

The optimal policy \( \hat{x}^* \in \hat{\Pi} \) selects, at every step \( i \), the highest feasible control \( u_i^* \) such that

\[ g_i(x_i, u_i^*, e_i) \leq 0, \quad f_i(x_i, u_i^*) \in \hat{K}_{i+1}. \quad \text{(16)} \]

\end{proposition}

This result is intuitive: since the controls \( u_i \) are path accelerations, selecting higher accelerations leads to higher path velocities, hence time-optimality. This result relies on the proposition below, whose proof is assumed.

\begin{proposition}

(Monotonicity of the worst-case transition function) Define first the worst-case transition function at stage \( i \) as

\[ T_i : x \mapsto \min_{e_i \in B(M)} \max_{u_i} \left\{ f_i(x, u), g_i(x, u, e_i) \leq 0, f_i(x, u) \in \hat{K}_{i+1} \right\}. \]

Then, for all \( x, x' \in \hat{K}_i \)

\[ x \leq x' \implies T_i(x) \leq T_i(x'). \quad \text{\blacksquare} \]

We can now provide a sketch of proof for Proposition 3.

\textbf{Sketch of proof of Proposition 3} The proof has two steps. First, we show the property (*) that the worst-case traversal time \( J_i(\cdot) \) is monotonically decreasing with \( x_i \in \hat{K}_i \). Second, we show that property (*) implies Proposition 3.

Step 1) We show property (*) by induction on \( i \). From (9), \( J_N(\cdot) \) is zero, hence monotone. Suppose now that \( J_{i+1}(\cdot) \) is
monotonically decreasing. By the principle of optimality [19], one has

\[ J_i(x_i) = \max_{u_i \in \mathcal{U}(M)} \min_{x_{i+1} \in \mathcal{K}(M)} \left\{ \frac{2\Delta_i}{\sqrt{f_i(x_i, u_i)} + J_{i+1}(f_i(x_i, u_i))} \right\}. \]  

(17)

By the monotonicity of \(J_{i+1}(\cdot)\) and Proposition 4

\[ J_i(x_i) = \frac{2\Delta_i}{\sqrt{f_i(x_i, u_i)}} + J_{i+1}(f_i(x_i, u_i)). \]

Again by Proposition 4 \(J_i(\cdot)\) is monotonically decreasing.

Step 2) At a state \(x_i \in \mathcal{K}_i\), the optimal action minimizes the worst-case traversal time, which is given by

\[ \frac{2\Delta_i}{\sqrt{f_i(x_i, u_i)}} + J_{i+1}(f_i(x_i, u_i)). \]

Since both terms are monotonically decreasing, the optimal action is to select the highest possible control. \(\square\)

IV. EXPERIMENTAL RESULTS

We simulated a 6-axis robotic arm and controlled it to track a geometric path with zero terminal velocity. Joint torques were bounded by symmetric bounds

\[ \tau_{\max} = -\tau_{\min} = [120, 280, 280, 120, 80, 80] \text{(Nm)}. \]

The geometric path \(p(s) \in [0, 1]\) is shown in Figure 3. Initially, the robot was at rest and had non-zero initial position error: \(q(0) = \Delta q_{\text{init}} + p(0)\), where \(\|\Delta q_{\text{init}}\| = 0.1 \text{ rad}\). Forward dynamics computations were performed using OpenRAVE and integrations were performed using the dopri5 ODE solver. Torque commands had sample time of 1 ms. At each time step, torque commands were perturbed by noises drawn from Gaussian distributions with zero mean and standard deviations being 5% of the respective torque bounds.

We implemented the Online Scaling with Guarantees controller (OSG) with \(M = 0.1 \text{ rad}\) (to account for \(\Delta q_{\text{init}}\) whose magnitude is 0.1 rad) and \(N = 100\). Computing the robust controllable sets \(\tilde{K}_0, \ldots, \tilde{K}_N\) took 0.29 sec with our Python implementation on a 3.1 Gz laptop running Ubuntu. Solving the conic-quadratic programs accounted for 37% of total time. Note that those computations were performed at planning time. At execution time, the online computation of the commands \(\dot{s}\) and \(\tau\) took 0.9 ms per time step.

We compared the OSG controller with the vanilla OS controller and the Computed Torque Trajectory Tracking (TT) controller (which tracks the trajectory obtained by TOPP) by looking at position error, defined as follows

\[ \|\Delta q(t)\| := ||q(t) - p(s(t)_{\square})||, \]

where \(s_{\square}\) is the parameterization produced by respective OSG, OS or the nominal parameterization obtained by TOPP. Table I reports the maximum position errors of the controllers; it shows that OSG outperformed the others by a large amount.

Qualitatively, the TT controller was observed to be bad at handling the initial position error. As seen in Fig. 4, the resulting position error increased by more than 300%, reaching a maximum of 0.49 rad, before stabilizing. This is because the

restoring torque component \(-M(q)Ke\) could not be realized due to constraints saturating reference torque commands.

The OS controller, on the other hand, was capable of regulating the initial position error. However, at \(s \approx 0.18\), position error rose sharply to 0.87 rad. The cause is precisely the limitation we have noted in the Introduction: the path controller could not find any feasible path acceleration resulting in bad tracking performance. This event can be seen on \(s(t)_{\text{OS}}\) as a sharp spike (the bottom plot of Fig. 4).

The OSG controller did not have any of the above problems. The resulting position error converged quickly to zero. The total tracking duration of the OSG controller, however, was slightly higher than the other controllers, about 3.3% longer than the nominal duration. See Table I for more details.

Finally, we observed that the parameterization generated by the OSG controller \(s(t)_{\text{OSG}}\) differed from the nominal parameterization \(s(t)_{\text{Nominal}}\) only during deceleration, see for instance the interval \(s \in [0.1, 0.18]\). This observation suggests that the OSG controller is more efficient compared to the naive approach of “reserving” torque authority [9] for tracking, which limits both the accelerating and decelerating segments.

V. CONCLUSION

In this paper, we considered the Time-Optimal Path Tracking problem: given a geometric path, find the control strategy to traverse the path time-optimally while regulating tracking errors caused by model inaccuracies and online perturbations. We have introduced the Online Scaling with Guarantees (OSG) controller and showed that the OSG controller outperforms both the vanilla Online Scaling (OS) controller and the classic Computed Torque Trajectory Tracking controller. The OSG controller differs from the vanilla OS controller by the use of the robust controllable sets, which intuitively are the sets of “safe” reference trajectories that can be tracked for all possible random perturbations. By ensuring feasibility, the OSG controller also inherits the stability of the component torque controller. The fundamental tool underlying OSG – Robust Reachability Analysis – is an extension of the technique introduced in [14].

Several matters have been left for future investigations. The most important ones include extending OSG to handle industrial manipulators with position or velocity interfaces as well as other types of constraints such as jerk or torque-rate bounds and evaluating the controller on physical systems.

Acknowledgment

This work was partially supported by grant ATMRI:2014-R6-PHAM (awarded by NTU and the Civil Aviation Authority of Singapore) and by the Medium-Sized Centre funding
scheme (awarded by the National Research Foundation, Prime Minister’s Office, Singapore).

## APPENDIX

### A. Identification of the quadratic bounds

To obtain the quadratic bounds in (13), we sampled $L$ data points $(x^{(j)}_i, u^{(j)}_i) \in \mathcal{G}, j \in [0, \ldots, L]$ and defined for $j \in [0, \ldots, L]$

$$y^{(j)} := \left\| \frac{\partial g}{\partial e_i} \right\|_{x^{(j)}_i, u^{(j)}_i, 0}^2, z^{(j)} := [u^{(j)}, x^{(j)}, 1].$$

Next, we solved the Linear Program (where $W$ is symmetric)

$$\min_{\xi} \xi \quad \text{s.t.} \quad \xi \geq z^{(j)^T} W z^{(j)} - y^{(j)} \geq 0, j \in [1, \ldots, L].$$

Let $W^*$ be the optimal solution, then $P_i[k]$ is given by $(W^*)^{1/2}$. 

---

**Fig. 3.** The swinging motion used in the experiment.

**Fig. 4.** Top: Comparisons of position errors between three controllers: OSG (solid), OS (dashed) and TT (dashed dot). Bottom: Parameterizations (solid) generated online by the controllers and the robust controllable sets (dashed).

---

**REFERENCES**

[1] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, “Time-optimal control of robotic manipulators along specified paths,” *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[2] Q.-C. Pham, “A General, Fast, and Robust Implementation of the Time-Optimal Path Parameterization Algorithm,” *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, dec 2014.

[3] Q.-C. Pham and O. Stasse, “Time-Optimal Path Parameterization for Redundantly Actuated Robots: A Numerical Integration Approach,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 3257–3263, dec 2015.

[4] O. Dahl and L. Nielsen, “Torque-limited path following by online trajectory time scaling,” *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 5, pp. 554–561, 1999.

[5] O. Dahl, “Path-constrained robot control with limited torques-experimental evaluation,” *IEEE transactions on robotics and automation*, vol. 10, no. 5, pp. 658–669, 1994.

[6] H. Arai, K. Tanie, and S. Tachi, “Path tracking control of a manipulator considering torque saturation,” *IEEE Transactions on Industrial Electronics*, vol. 41, no. 1, pp. 25–30, 1994.

[7] C. L. Bianco and O. Gerelli, “Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints,” *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1144–1152, 2011.

[8] O. Gerelli and C. G. I. Bianco, “Real-time Path Tracking Control of Robotic Manipulators with Bounded Torques and Torque-derivatives,” *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, IROS 2008, 2008.

[9] J. Kieffer, A. J. Cahill, and M. R. James, “Robust and accurate time-optimal path-tracking control for robot manipulators,” *IEEE Transactions on Robotics and Automation*, vol. 13, no. 6, pp. 880–890, 1997.

[10] T. Faulwasser and R. Findeisen, “Nonlinear model predictive control for constrained output path following,” *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2016.

[11] T. Faulwasser, T. Weber, P. Zometra, and R. Findeisen, “Implementation of nonlinear model predictive path-following control for an industrial robot,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1505–1511, 2017.

[12] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, “Practical time-optimal trajectory planning for robots: a convex optimization approach,” *IEEE Transactions on Automatic Control*, 2008.

[13] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[14] H. Pham and Q.-C. Pham, “A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis,” *Robotics and Automation*, vol. 30, no. 6, pp. 1533–1540, dec 2014.

[15] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[16] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[17] D. Bertsekas and I. Rhodes, “On the minimax reachability of target sets and target tubes,” *Automatica*, vol. 7, no. 2, pp. 233–247, mar 1971.

[18] F. Blanchini, “Set invariance in control,” *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[19] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, 2.

[20] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010.