A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis

Article *in* IEEE Transactions on Robotics · June 2018

DOI: 10.1109/TRO.2018.2819195

2 authors, including:

Hung Pham
Nanyang Technological University
12 PUBLICATIONS **43** CITATIONS

SEE PROFILE

A New Approach to Time-Optimal Path Parameterization based on Reachability Analysis

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*—Time-Optimal Path Parameterization (TOPP) is a well-studied problem in robotics and has a wide range of applications. There are two main families of methods to address TOPP: Numerical Integration (NI) and Convex Optimization (CO). NI-based methods are fast but difficult to implement and suffer from robustness issues, while CO-based approaches are more robust but at the same time significantly slower. Here we propose a new approach to TOPP based on Reachability Analysis (RA). The key insight is to recursively compute reachable and controllable sets at discretized positions on the path by solving small Linear Programs (LPs). The resulting algorithm is faster than NI-based methods and as robust as CO-based ones (100% success rate), as confirmed by extensive numerical evaluations. Moreover, the proposed approach offers unique additional benefits: Admissible Velocity Propagation and robustness to parametric uncertainty can be derived from it in a simple and natural way.**

I. Introduction

Time-Optimal Path Parameterization (TOPP) is the problem of finding the fastest way to traverse a path in the configuration space of a robot system while respecting the system constraints [1]. This classical problem has a wide range of applications in robotics. In many industrial processes such as cutting, welding, or machining, the robot paths are predefined, and optimal productivity implies tracking those paths at the highest possible speed while respecting the process and robot constraints. From a conceptual viewpoint, TOPP has been used extensively as subroutine to kinodynamic motion planning algorithms [2, 3]. Because of its practical and theoretical importance, TOPP has received considerable attention since its inception in the 1980's, see [4] for a recent review.

Existing approaches to TOPP

There are two main families of methods to TOPP, based respectively on Numerical Integration (NI) and Convex Optimization (CO). Each approach has its strengths and weaknesses.

The NI-based approach was initiated by [1], and further improved and extended by many researchers, see [4] for a review. NI-based algorithms are based on Pontryagin's Maximum Principle, which states that the time-optimal path parameterization consists of alternatively maximally accelerating and decelerating segments. The key advantage of this approach is that the optimal controls can be explicitly computed (and not searched for as in the CO approach) at each path position, resulting in extremely fast implementations. However, this requires finding the switch points between accelerating and decelerating segments, which constitutes a major implementation difficulty as well as the main cause of failure [5, 6, 7, 4]. Another notable implementation difficulty is the direct velocity bounds [8]. The formulation of the present paper naturally removes those two difficulties.

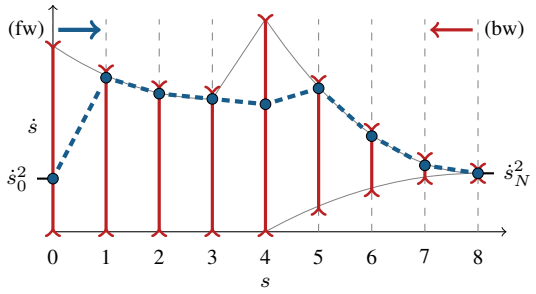

Fig. 1. Time-Optimal Path Parameterization by Reachability Analysis (TOPP-RA) computes the optimal parameterization in two passes. In the first pass (backward), starting from the last grid point N, the algorithm computes controllable sets (red intervals) recursively. In the second pass (forward), starting now from grid point 0, the algorithm greedily selects the highest controls such that resulting velocities remain inside the respective controllable sets.

The CO-based approach was initiated by [9] and further extended in [10]. This approach formulates TOPP as a single large convex optimization program, whose optimization variables are the accelerations and squared velocities at discretized positions along the path. The main advantages of this approach are: (i) it is simple to implement and quite robust, as one can use off-the-shelf convex optimization packages; (ii) other convex objectives than traversal time can be considered. On the downside, the optimization program to solve is huge – the number of variables and constraint inequalities scale with the discretization grid size – resulting in implementations that are one order of magnitude slower than NI-based methods [4]. This makes CO-based methods inappropriate for online motion planning or as subroutine to kinodynamic motion planners [3].

Proposed new approach based on Reachability Analysis

In this paper, we propose a new approach to TOPP based on Reachability Analysis (RA), a standard tool from control theory. The key insight is: given an interval of velocities $\mathbb{I}_s$ at some position s on the path, the *reachable* set $\mathbb{I}_{s+\delta}$ (the set of all velocities at the next path position that can be reached from $\mathbb{I}_s$ following admissible controls) and the *controllable* set $\mathbb{I}_{s-\delta}$ (the set of all velocities at the previous path position such that

there exists an admissible control leading to a velocity in $\mathbb{I}_s$) can be computed quickly and robustly by solving a few *small* Linear Programs (LPs). By recursively computing controllable sets at discretized positions on the path, one can then extract the time-optimal parameterization in time $O(mN)$, where m is the number of constraint inequalities and N the discretization grid size, see Fig. 1 for an illustration.

As compared to NI-based methods, the proposed approach has therefore a better time complexity (actual computation time is similar for problem instances with few constraints, and becomes significantly faster for instances with > 22 constraints). More importantly, the proposed method is much easier to implement and has a success rate of 100%, while state-of-the-art NI-based implementations (e.g. [4]) comprise thousands of lines of code and still report failures on hard problem instances. As compared to CO-based methods, the proposed approach enjoys the same level of robustness and of ease-of-implementation while being significantly faster.

Besides the gains in implementation robustness and performance, viewing the classical TOPP problem from the proposed new perspective yields the following additional benefits:

- constraints for redundantly-actuated systems are handled natively – in particular, there is no need to perform the "polytopic projection" step as in [10, 11];
- Admissible Velocity Propagation [3], a recent concept for kinodynamic motion planning, can be derived "for free", as a side product;
- robustness to parametric uncertainties, e.g. uncertain coefficients of friction or uncertain inertia matrices, can be obtained in a natural way.

Organization of the paper

The rest of the paper is organized as follows. Section II formulates the TOPP problem in a general setting. Section III applies Reachability Analysis to the path-projected dynamics. Section IV shows how Reachability Analysis allows extracting the time-optimal path parameterization. Section V presents extensive experimental results to demonstrate the gains in robustness and performance permitted by the new approach. Section VI discusses the additional benefits mentioned previously: Admissible Velocity Propagation and robustness to parametric uncertainty. Finally, Section VII offers some concluding remarks and directions for future research.

II. Problem formulation

A. Generalized constraints

Consider a n-dof robot system, whose configuration is denoted by a n dimensional vector $\mathbf{q} \in \mathbb{R}^n$. A *geometric path* $\mathcal{P}$ in the configuration space is represented as a function $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$. We assume that $\mathbf{q}(s)$ is piece-wise $\mathcal{C}^2$-continuous. A *time parameterization* is a piece-wise $\mathcal{C}^2$, increasing scalar function $s : [0, T] \to [0, s_{\text{end}}]$, from which a *trajectory* is recovered as $\mathbf{q}(s(t))_{t\in[0,T]}$.

In this paper, we consider *generalized second-order constraints* of the following form [10, 11]

$$\mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{f}(\mathbf{q}) \in \mathscr{C}(\mathbf{q}), \text{ where} \tag{1}$$

- $\mathbf{A}, \mathbf{B}, \mathbf{f}$ are continuous mappings from $\mathbb{R}^n$ to $\mathbb{R}^{m\times n}, \mathbb{R}^{n\times m\times n}$ and $\mathbb{R}^m$ respectively;
- $\mathscr{C}(\mathbf{q})$ is a convex polytope.

Implementation remark 1. The above form is the most general one in the TOPP literature to date, and can account for many types of kinodynamic constraints, including velocity and acceleration bounds, joint torque bounds for fully- or redundantly-actuated robots [11], contact stability under Coulomb friction model [10, 12, 13], etc.

Consider for instance the torque bounds on a fully-actuated manipulator

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \tag{2}$$

$$\tau_i^{\min} \leq \tau_i(t) \leq \tau_i^{\max}, \ \forall i \in [1, \dots, n], \ t \in [0, T] \tag{3}$$

This can be rewritten in the form of (1) with $\mathbf{A} := \mathbf{M}$, $\mathbf{B} := \mathbf{C}$, $\mathbf{f} := \mathbf{g}$ and

$$\mathscr{C}(\mathbf{q}) := [\tau_1^{\min}, \tau_1^{\max}] \times \cdots \times [\tau_n^{\min}, \tau_n^{\max}],$$

which is clearly convex.

For redundantly-actuated manipulators, it was shown that the TOPP problem can also formulated in the form of (1) with

$$\mathscr{C}(\mathbf{q}) := \mathbf{S}^\top \left([\tau_1^{\min}, \tau_1^{\max}] \times \cdots \times [\tau_n^{\min}, \tau_n^{\max}] \right),$$

where $\mathbf{S}$ is a linear transformation [11], which implies that the so-defined $\mathscr{C}(\mathbf{q})$ is a convex polytope.

In legged robots, the TOPP problem under the constraint of contact-stability where the friction cones are linearized was shown to be reducible to the form of (1) with $\mathscr{C}(\mathbf{q})$ being also a *convex polytope* [10, 11, 12].

If the friction cones are not linearized, then $\mathscr{C}(\mathbf{q})$ is still convex, but not polytopic. The developments in the present paper that concern reachable and controllable sets (Section III) are still valid in the convex, non-polytopic case. The developments on time-optimality (Section IV) is however only applicable to the polytopic case ■

Finally, we also consider first-order constraints of the form

$$\mathbf{A}^v(\mathbf{q})\dot{\mathbf{q}} + \mathbf{f}^v(\mathbf{q}) \in \mathscr{C}^v(\mathbf{q}), \tag{4}$$

where the coefficients are matrices of appropriate sizes and $\mathscr{C}^v(\mathbf{q})$ is a convex set. Direct velocity bounds and momentum bounds are examples of first-order constraints.

B. Projecting the constraints on the path

Differentiating successively $\mathbf{q}(s)$, one has

$$\dot{\mathbf{q}} = \mathbf{q}'\dot{s}, \quad \ddot{\mathbf{q}} = \mathbf{q}''\dot{s}^2 + \mathbf{q}'\ddot{s}, \tag{5}$$

where $\square'$ denotes differentiation with respect to the path parameter s. From now on, we shall refer to $s, \dot{s}, \ddot{s}$ as the position, velocity and acceleration respectively.

Substituting (5) to Equation (1), one transforms second-order constraints on the system dynamics into constraints on $s, \dot{s}, \ddot{s}$ as follows

$$\mathbf{a}(s)\ddot{s} + \mathbf{b}(s)\dot{s}^2 + \mathbf{c}(s) \in \mathscr{C}(s), \text{ where} \tag{6}$$

$$\begin{aligned} \mathbf{a}(s) &:= \mathbf{A}(\mathbf{q}(s))\mathbf{q}'(s), \\ \mathbf{b}(s) &:= \mathbf{A}(\mathbf{q}(s))\mathbf{q}''(s) + \mathbf{q}'(s)^\top \mathbf{B}(\mathbf{q}(s))\mathbf{q}'(s), \\ \mathbf{c}(s) &:= \mathbf{f}(\mathbf{q}(s)), \\ \mathscr{C}(s) &:= \mathscr{C}(\mathbf{q}(s)). \end{aligned} \tag{7}$$

Similarly, first-order constraints are transformed into

$$\mathbf{a}^v(s)\dot{s} + \mathbf{b}^v(s) \in \mathscr{C}^v(s), \text{ where} \tag{8}$$

$$\begin{aligned} \mathbf{a}^v(s) &:= \mathbf{A}^v(\mathbf{q}(s))\mathbf{q}'(s), \\ \mathbf{b}^v(s) &:= \mathbf{f}^v(\mathbf{q}(s)), \\ \mathscr{C}^v(s) &:= \mathscr{C}^v(\mathbf{q}(s)). \end{aligned} \tag{9}$$

C. Path discretization

As in the Convex Optimization approach, we divide the interval $[0, s_{\text{end}}]$ into N segments and $N+1$ grid points

$$0 =: s_0, s_1 \dots s_{N-1}, s_N := s_{\text{end}}.$$

Denote by u_i the constant path acceleration over the interval $[s_i, s_{i+1}]$ and by x_i the squared velocity $\dot{s}_i^2$ at s_i. Using simple algebraic manipulations, one can show that the following relation holds

$$x_{i+1} = x_i + 2\Delta_i u_i, \quad i = 0 \dots N-1, \tag{10}$$

where $\Delta_i := s_{i+1} - s_i$. In the sequel we refer to s_i as the i-stage, u_i and x_i as respectively the control and state at the i-stage. Any sequence $x_0, u_0, \dots, x_{N-1}, u_{N-1}, x_N$ that satisfies the linear relation (10) is referred to as a path parameterization.

A parameterization is *admissible* if it satisfies the constraints at every points in $[0, s_{\text{end}}]$. One possible way to bring this requirement into the discrete setting is through a *collocation* discretization scheme: for each position s_i, one evaluates the continuous constraints and requires the control and state u_i, x_i to verify

$$\mathbf{a}_i u_i + \mathbf{b}_i x_i + \mathbf{c}_i \in \mathscr{C}_i, \tag{11}$$

where $\mathbf{a}_i := \mathbf{a}(s_i)$, etc.

Since the constraints are enforced only at a finite number of points, the actual continuous constraints might not be respected everywhere along $[0, s_{\text{end}}]$ [1]. Therefore, it is important to bound the constraint satisfaction error. We show in Appendix C that the collocation scheme has an error of order $O(\Delta_i)$. Appendix C also presents a first-order interpolation discretization scheme, which has an error of order $O(\Delta_i^2)$ but which involves more variables and inequality constraints than the collocation scheme.

[1] This limitation is however not specific to the proposed approach as both the NI and CO approaches require discretization at some stages of the algorithm.

III. Reachability Analysis of the path-projected dynamics

The key to our analysis is that the "path-projected dynamics" (10), (11) is a *discrete-time linear system* with *linear control-state inequality constraints*. This observation immediately allows us to take advantage of the set-membership control problems studied in the Model Predictive Control (MPC) literature [14, 15, 16]. Note however that the objective function here (traversal time) is convex but non quadratic [9].

A. Admissible states and controls

We first need some definitions. Denote the i-stage set of *admissible* control-state pairs by

$$\Omega_i := \{(u, x) \mid \mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathscr{C}_i\}.$$

One can see Ω_i as the projection of $\mathscr{C}_i$ on the $(\ddot{s}, \dot{s}^2)$ plane [10]. Since $\mathscr{C}_i$ is a polytope, Ω_i is a *polygon*. Algorithmically, the projection can be obtained by e.g. the recursive expansion algorithm [17].

Next, the i-stage set of *admissible states* is the projection of Ω_i on the second axis

$$\mathcal{X}_i := \{x \mid \exists u : (u, x) \in \Omega_i\}.$$

The i-stage set of *admissible controls* given a state x is

$$\mathcal{U}_i(x) := \{u \mid (u, x) \in \Omega_i\}.$$

Note that, since Ω_i is convex, both $\mathcal{X}_i$ and $\mathcal{U}_i(x)$ are *intervals*.

Classic terminologies in the TOPP literature (e.g. Maximum Velocity Curve, α and β acceleration fields, etc.) can be conveniently expressed using these definitions. See the first part of Appendix A for more details.

Implementation remark 2. For redundantly-actuated manipulators and contact-stability of legged robots, both NI-based and CO-based methods must compute Ω_i at each discretized position i along the path, which is costly. Our proposed approach avoids performing this 2D projection: instead, it will only require a few 1D projections per discretization step. Furthermore, each of these 1D projections amounts to a pair of LPs and can therefore be performed extremely quickly ■

B. Reachable sets

The key notion in Reachability Analysis is that of i-stage reachable set.

Definition 1 (i-stage reachable set)**.** Consider a set of starting states $\mathbb{I}_0$. The *i-stage reachable set* $\mathcal{L}_i(\mathbb{I}_0)$ is the set of states $x \in \mathcal{X}_i$ such that there exist a state $x_0 \in \mathbb{I}_0$ and a sequence of admissible controls $u_0, \dots, u_{i-1}$ that steers the system from x_0 to x ■

To compute the i-stage reachable set, one needs the following intermediate representation.

Definition 2 (Reach set)**.** Consider a set of states $\mathbb{I}$. The *reach set* $\mathcal{R}_i(\mathbb{I})$ is the set of states x such that there exist a state

$\tilde{x} \in \mathbb{I}$ and an admissible control $u \in \mathcal{U}_i(\tilde{x})$ that steers the system from $\tilde{x}$ to x, i.e.

$$x = \tilde{x} + 2\Delta_i u \ \blacksquare$$

Implementation remark 3. Let us note $\Omega_i(\mathbb{I}) := \{(u, \tilde{x}) \in \Omega_i \mid \tilde{x} \in \mathbb{I}\}$. If $\mathbb{I}$ is convex, then $\Omega_i(\mathbb{I})$ is convex as the intersection of two convex sets. Next, $\mathcal{R}_i(\mathbb{I})$ can be seen as the projection of $\Omega_i(\mathbb{I})$ onto a line. Thus, $\mathcal{R}_i(\mathbb{I})$ is an interval, hence defined by its lower and upper bounds (x^-, x^+), which can be computed as follows

$$x^- := \min_{(u,\tilde{x})\in\Omega_i(\mathbb{I})} \tilde{x} + 2\Delta_i u,$$

$$x^+ := \max_{(u,\tilde{x})\in\Omega_i(\mathbb{I})} \tilde{x} + 2\Delta_i u.$$

Since $\Omega_i(\mathbb{I})$ is a polygon, the above equations constitute two LPs. Note finally that there is no need to compute explicitly $\Omega_i(\mathbb{I})$, since one can write directly

$$x^+ := \max_{(u,\tilde{x})\in\mathbb{R}^2} \tilde{x} + 2\Delta_i u,$$

$$\text{subject to: } \mathbf{a}_i u + \mathbf{b}_i \tilde{x} + \mathbf{c}_i \in \mathscr{C}_i \ \text{ and } \tilde{x} \in \mathbb{I},$$

and similarly for x^- $\blacksquare$

The i-stage reachable set can now be recursively computed by

$$\begin{aligned} \mathcal{L}_0(\mathbb{I}_0) &= \mathbb{I}_0 \cap \mathcal{X}_0, \\ \mathcal{L}_i(\mathbb{I}_0) &= \mathcal{R}_{i-1}(\mathcal{L}_{i-1}(\mathbb{I}_0)) \cap \mathcal{X}_i. \end{aligned} \tag{12}$$

Implementation remark 4. If $\mathbb{I}_0$ is an interval, then by recursion and by application of Implementation remark 3, all the $\mathcal{L}_i$ are intervals. Each step of the recursion requires solving four LPs: two for computing $\mathcal{R}_{i-1}(\mathcal{L}_{i-1}(\mathbb{I}_0))$ and two for computing $\mathcal{X}_i$. Therefore, $\mathcal{L}_i$ can be computed by solving $4i$ LPs $\blacksquare$

The i-stage reachable set may be empty, which implies that the system can not evolve without violating constraints: the path is not time-parameterizable. One can also note that

$$\mathcal{L}_i(\mathbb{I}_0) = \emptyset \implies \forall j \geq i,\ \mathcal{L}_j(\mathbb{I}_0) = \emptyset.$$

C. Controllable sets

Controllability is the dual notion of reachability, as made clear by the following definitions.

Definition 3 (i-stage controllable set)**.** Consider a set of desired ending states $\mathbb{I}_N$. The *i-stage controllable set* $\mathcal{K}_i(\mathbb{I}_N)$ is the set of states $x \in \mathcal{X}_i$ such that there exist a state $x_N \in \mathbb{I}_N$ and a sequence of admissible controls $u_i, \ldots, u_{N-1}$ that steers the system from x to x_N $\blacksquare$

The dual notion of "reach set" is that of "one-step" set.

Definition 4 (One-step set)**.** Consider a set of states $\mathbb{I}$. The *one-step set* $\mathcal{Q}_i(\mathbb{I})$ is the set of states x such that there exist a state $\tilde{x} \in \mathbb{I}$ and an admissible control $u \in \mathcal{U}_i(x)$ that steers the system from x to $\tilde{x}$, i.e.

$$\tilde{x} = x + 2\Delta_i u \ \blacksquare$$

The i-stage controllable set can now be computed recursively by

$$\begin{aligned} \mathcal{K}_N(\mathbb{I}_N) &= \mathbb{I}_N \cap \mathcal{X}_N, \\ \mathcal{K}_i(\mathbb{I}_N) &= \mathcal{Q}_i(\mathcal{K}_{i+1}(\mathbb{I}_N)). \end{aligned} \tag{13}$$

Implementation remark 5. By similar reasoning as in Implementation remark 4, every one-step set $\mathcal{Q}_i(\mathbb{I})$ is an interval, whose lower and upper bounds (x^-, x^+) are given by the following two LPs

$$x^+ := \max_{(u,x)\in\mathbb{R}^2} x,$$

$$\text{subject to: } \mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathscr{C}_i \ \text{ and } x + 2\Delta_i u \in \mathbb{I},$$

and similarly for x^-. Thus, computing the i-stage controllable set will require solving $2i$ LPs $\blacksquare$

The i-stage controllable set may be empty, in that case, the path is not time-parameterizable. One also has

$$\mathcal{K}_i(\mathbb{I}_N) = \emptyset \implies \forall j \leq i,\ \mathcal{K}_j(\mathbb{I}_0) = \emptyset.$$

IV. TOPP by Reachability Analysis

A. Algorithm

Armed with the notions of reachable and controllable sets, we can now proceed to solving the TOPP problem. The reachability-analysis-based TOPP algorithm (TOPP-RA) is given in Algorithm 1 below and illustrated in Fig. 1.

Algorithm 1: TOPP-RA

Input : Path $\mathcal{P}$, starting and ending velocities $\dot{s}_0, \dot{s}_N$

Output: The optimal parameterization $x_0^*, u_0^*, \ldots, u_{N-1}^*, x_N^*$

`/* Backward pass: compute the controllable sets */`

$\mathcal{K}_N := \{\dot{s}_N^2\}$

for $i \in [N-1 \ldots 0]$ **do**

  $\mathcal{K}_i := \mathcal{Q}_i(\mathcal{K}_{i+1})$

if $\mathcal{K}_0 = \emptyset$ *or* $\dot{s}_0 \notin \mathcal{K}_0$ **then**

  return `Infeasible`

`/* Forward pass: greedily select the controls */`

$x_0^* := \dot{s}_0^2$

for $i \in [0 \ldots N-1]$ **do**

  $u_i^* := \max u$, subject to: $x_i^* + 2\Delta_i u \in \mathcal{K}_{i+1}$ and $(x_i^*, u) \in \Omega_i$

  $x_{i+1}^* := x_i^* + 2\Delta_i u_i^*$

The algorithm proceeds in two passes. The first pass goes backward: it recursively computes the controllable sets $\mathcal{K}_i(\{\dot{s}_N^2\})$ given the desired ending velocity $\dot{s}_N$, as described in Section III-C. If any of the controllable sets is empty or if the starting velocity $\dot{s}_0$ is not contained in the 0-stage controllable set, then clearly the path is not time-parameterizable.

Otherwise, the algorithm proceeds to a second, forward, pass. Here, the optimal states and controls are constructed in a

greedy manner: at each stage i, the highest control u such that the resulting next state belongs to the $i+1$-stage controllable set is selected.

The correctness of the algorithm as well as a more detailed complexity analysis are given in the subsequent sections.

Note finally that one can construct a "dual version" of TOPP-RA as follows: (i) in a forward pass, recursively compute the i-stage reachable sets, $i \in [0, \dots, N]$; (ii) in a backward pass, greedily select, at stage i, the lowest control such that the resulting previous state belongs to the $i-1$-stage reachable set. However, since the construction of the reachable sets already requires solving $4N$ optimization programs, this dual version is not as efficient as the primal one.

B. Proof of correctness

We first define the maximal controls α and β

$$\alpha_i(x) := \min(\mathcal{U}_i(x)), \quad \beta_i(x) := \max(\mathcal{U}_i(x)).$$

and the maximal transition functions

$$T_i^\alpha(x) := x + 2\Delta_i \alpha_i(x), \quad T_i^\beta(x) := x + 2\Delta_i \beta_i(x).$$

Note that the maximal transition functions are equivalent to the forward integration operation commonly used in NI-based algorithms.

We note the following important property

Proposition 1 (Monotonicity of the maximal transition functions)**.** *For sufficiently small discretization steps Δ_i, one has: for all $x, \tilde{x} \in \mathcal{X}_i$ such that $x \leq \tilde{x}$:*

$$T_i^\beta(x) \leq T_i^\beta(\tilde{x}),$$
$$T_i^\alpha(x) \geq T_i^\alpha(\tilde{x}).$$

Proof. See Appendix B. □

We are now ready to prove the correctness of TOPP-RA. The key idea is that the monotonicity of the maximal transition functions allows greedily selecting the optimal controls.

Theorem 1. *TOPP-RA is correct.*

Proof. (1) We first show that, if TOPP-RA returns `Infeasible`, then the instance is indeed not parameterizable. By contradiction, assume that there exists an admissible parameterization $\dot{s}_0^2 = x_0, u_0, \dots, u_{N-1}, x_N = \dot{s}_N^2$. We now show by induction on i that $\mathcal{K}_i$ contains at least x_i.

Initialization: $\mathcal{K}_N$ contains x_N by construction.

Induction: Assume that $\mathcal{K}_i$ contains x_i. Since the parameterization is admissible, one has $x_i = x_{i-1} + 2\Delta_i u_{i-1}$ and $(u_{i-1}, x_{i-1}) \in \Omega_{i-1}$. By definition of the controllable sets, $x_{i-1} \in \mathcal{K}_{i-1}$.

We have thus shown that none of the $\mathcal{K}_i$ is empty and that $\mathcal{K}_0$ contains at least $x_0 = \dot{s}_0^2$, which implies that TOPP-RA does not return `Infeasible`.

(2) In a similar way, one can show by induction on i that the forward pass produces an admissible parameterization.

(3) We now show that the parameterization produced by the forward pass is optimal. Consider an arbitrary admissible parameterization $\dot{s}_0^2 = x_0, u_0, \dots, u_{N-1}, x_N = \dot{s}_N^2$. We show by induction that for all $i = 0 \dots N$, $x_i^* \geq x_i$.

Initialization: We have $x_0 = \dot{s}_0^2 = x_0^*$, so the assertion is true at $i = 0$.

Induction: Steps 8 and 9 of TOPP-RA can in fact be rewritten as follows

$$x_{i+1}^* := \min\{T_i^\beta(x_i^*), \max(K_{i+1})\}.$$

By the induction hypothesis, one has $x_i^* \geq x_i$. Since $x_i^*, x_i \in \mathcal{K}_i$, Proposition 1 yields

$$T_i^\beta(x_i^*) \geq T_i^\beta(x_i) \geq x_{i+1}.$$

Thus,

$$\min\{T_i^\beta(x_i^*), \max(K_{i+1})\} \geq \min\{x_{i+1}, \max(K_{i+1})\}, \text{ i.e.}$$
$$x_{i+1}^* \geq x_{i+1}.$$

We have thus shown that the parameterization $x_0^*, \dots, x_N^*$ has a higher velocity at every path position than any admissible parameterization, which implies that it is time-optimal. □

C. Complexity analysis

We now perform a complexity analysis of TOPP-RA and compare it with the Numerical Integration and the Convex Optimization approaches. For simplicity, we shall restrict the discussion to the non-redundantly actuated case (the redundantly-actuated case actually brings an additional advantage to TOPP-RA, see Implementation remark 3).

Assume that there are m constraint inequalities and that the path discretization grid size is N. As a large part of the computation time is devoted to solving LPs, we need a good estimate of the practical complexity of this operation. Consider a LP with κ optimization variables and m inequality constraints. Different LP methods (ellipsoidal, simplex, active sets, etc.) have different complexities. For the purpose of this section, we consider the best *practical* complexity, which is realized by the simplex method, in $O(\kappa^2 m)$ [18].

- *TOPP-RA:* The LPs considered here have 2 variables and $m+2$ inequalities. Since one needs to solve $3N$ such LPs, the complexity of TOPP-RA is $O(mN)$.
- *Numerical integration approach:* The dominant component of this approach, in terms of time complexity, is the computation of the Maximum Velocity Curve (MVC). In most TOPP-NI implementations to date, the MVC is computed, at each discretized path position, by solving $O(m^2)$ second-order polynomials [1, 5, 19, 7, 4], which results in an overall complexity of $O(m^2 N)$.
- *Convex optimization approach:* This approach formulates the TOPP problem as a single large convex optimization program with $O(N)$ variables and $O(mN)$ inequality constraints. In the fastest implementation we know of, the author solves the convex optimization problem by solving a sequence of linear programs (SLP) with the same number of variables and inequalities [10]. Thus, the time complexity of this approach is $O(KmN^3)$, where K is the number of SLP iterations.

This analysis shows that TOPP-RA has the best theoretical complexity. The next section experimentally assesses this observation.

V. Experiments

We implements TOPP-RA in Python on a machine running Ubuntu with a Intel i7-4770(8) 3.9GHz CPU and 8Gb RAM. To solve the LPs we use the Python interface of the online active-set solver `qpOASES` [20]. The implementation and test cases are open-source and available at https://github.com/hungpham2511/toppra.

A. Experiment 1: Pure joint velocity and acceleration bounds

In this experiment, we compare TOPP-RA against TOPP-NI – the fastest known implementation of TOPP, which is based on the Numerical Integration approach [4]. For simplicity, we consider pure joint velocity and acceleration bounds – which involve the same difficulty as any other types of kinodynamic constraints, as far as TOPP is concerned.

a) Effect of the number of constraint inequalities: We considered random geometric paths with varying degrees of freedom $n \in [2, 60]$. Each path was generated as follows: we sampled 5 random waypoints and interpolated smooth geometric paths using cubic splines. For each path, velocity and acceleration bounds were also randomly chosen. Each problem instance thus has $m = 2n + 2$ constraint inequalities: $2n$ inequalities corresponding to acceleration bounds (no pruning was applied, contrary to [10]) and 2 inequalities corresponding to velocity bounds (the joint velocity bounds could be immediately pruned into one lower and one upper bound on $\dot{s}$). According to the complexity analysis of Section IV-C, we consider the number of inequalities, rather than the degree of freedom, as independent variable. Finally, the discretization grid size was chosen as $N = 500$.

Fig. 2 shows the time-parameterizations and the resulting trajectories produced by TOPP-RA and TOPP-NI on a given instance $(n = 6, m = 14)$. One can observe that the two algorithms produced virtually identical results, hinting at the correctness of TOPP-RA.

Fig. 3 shows the computation time for TOPP-RA and TOPP-NI, excluding the "extract trajectory" step (which takes much longer in TOPP-NI than in TOPP-RA). The experimental results confirm our theoretical analysis in that the complexity of TOPP-RA is in linear in m while that of TOPP-NI is quadratic in m. In terms of actual computation time, TOPP-RA becomes faster than TOPP-NI as soon as $m \geq 22$. Table I reports the different components of the computation time.

Perhaps even more importantly than mere computation time, TOPP-RA was extremely robust: it maintained 100% success rate over all instances, while TOPP-NI struggled with instances with many inequality constraints ($m \geq 40$), see Fig. 4.

b) Effect of discretization grid size: Grid size (or its inverse, discretization time step) is an important parameter for both TOPP-RA and TOPP-NI as it affects running time, success rate and solution quality (measured by constraint satisfaction error) of both algorithms. Here, we assess the effect of grid size on *success rate* and *solution quality*. Remark that, based on our complexity analysis in Section IV-C, running time depends linearly on grid size in both algorithms.

TABLE I
Breakdown of TOPP-RA and TOPP-NI total computation time to parameterize a path discretized with $N = 500$ grid points, subject to $m = 30$ inequalities.

	Time (ms)	
	TOPP-RA	TOPP-NI
setup	1.0	111.6
solve TOPP	25.0	28.3
backward pass	16.2	
forward pass	8.8	
extract trajectory	2.1	344.4
total	28.1	484.3

In addition to TOPP-RA and TOPP-NI, we considered TOPP-RA-intp. This variant of TOPP-RA employs the first-order interpolation scheme (see Appendix C) to discretize the constraints, instead of the collocation scheme introduced in Section II-C.

We considered different grid sizes $N \in [20, 1000]$. For each grid size, we generated and solved 100 random parameterization instances; each instance consists of a random path with $n = 14$ subject to random kinematic constraints, as in the previous experiment. Fig. 5-**A** shows success rates versus grid sizes. One can observe that TOPP-RA and TOPP-RA-intp maintained 100% success rate across all grid sizes, while TOPP-NI report failures for small grid sizes ($N \leq 500$).

Next, to measure the effect of grid size on solution quality, we looked into the *greatest constraint satisfaction errors*. For each instance, we sampled the resulting trajectories at $1\,\mathrm{ms}$ and computed the greatest constraint violation by comparing the sampled joint accelerations and velocities to their respective bounds. Then, we averaged instances with the same grid size to obtain the average error for each $N \in [20, 1000]$. Fig. 5-**B** shows the average greatest constraint satisfaction errors of the three algorithms with respect to grid size. One can observe that TOPP-RA and TOPP-NI have constraint satisfaction errors of the same order of magnitude, while TOPP-RA-intp produces solutions with much higher quality. This result confirms our error analysis of different discretization schemes in Appendix C and demonstrates that the interpolation discretization scheme is better than the collocation scheme whenever solution quality is concerned.

B. Experiment 2: Legged robot in multi-contact

Here we consider the time-parameterization problem for a 50-dof legged robot in multi-contact under joint torque bounds and linearized friction cone constraints.

a) Formulation: We now give a brief description of our formulation, for more details, refer to [10, 11]. Let $\mathbf{w}_i$ denote the net contact wrench (force-torque pair) exerted on the robot

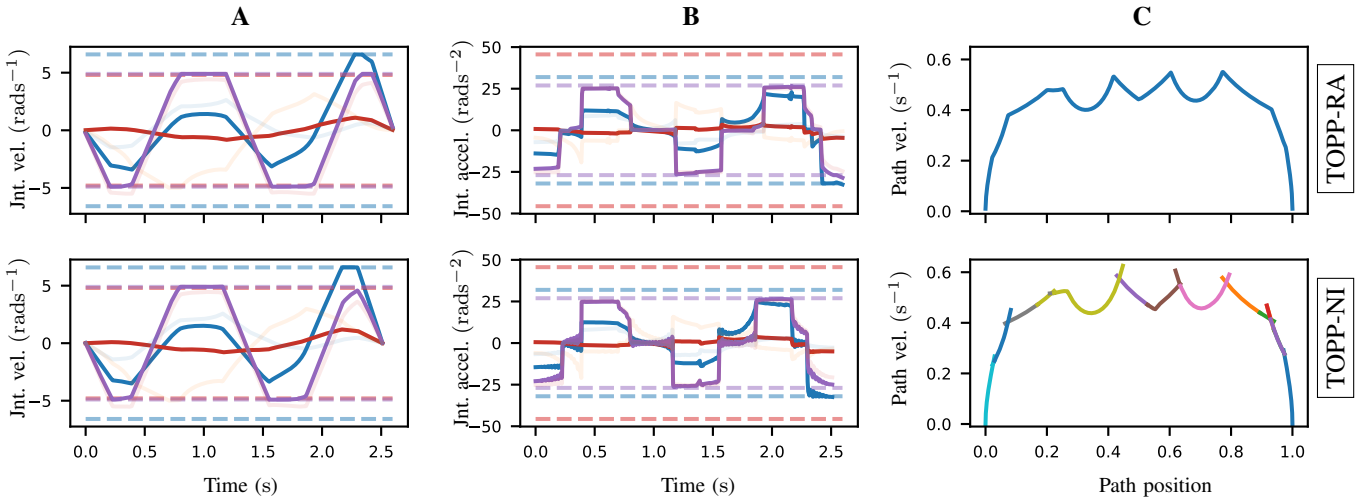

Fig. 2. Time-optimal parameterization of a 6-dof path under velocity and acceleration bounds ($m = 14$ constraint inequalities and $N = 500$ grid points). TOPP-RA and TOPP-NI produce virtually identical results. **(A)**: joint velocities. **(B)**: joint accelerations. **(C)**: velocity profiles in the $(s, \dot{s})$ plane. Note the small chattering in the joint accelerations produced by TOPP-NI, which is an artifact of the integration process. This chattering is absent from the TOPP-RA profiles.

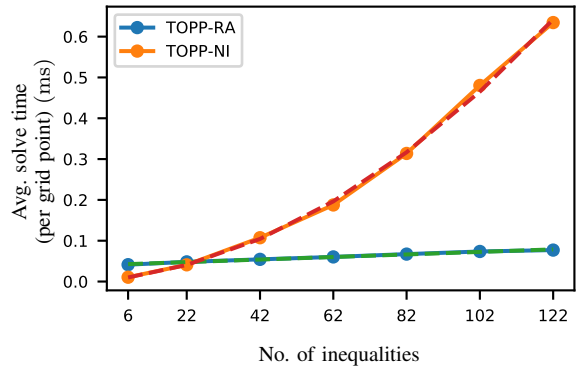

Fig. 3. Computation time of TOPP-RA (solid blue) and TOPP-NI (solid orange), excluding the "extract trajectory" step, as a function of the number of constraint inequalities. Confirming our theoretical complexity analysis, the complexity of TOPP-RA is linear in the number of constraint inequalities m (linear fit in dashed green), while that of TOPP-NI is quadratic in m (quadratic fit in dashed red). In terms of actual computation time, TOPP-RA becomes faster than TOPP-NI as soon as $m \geq 22$.

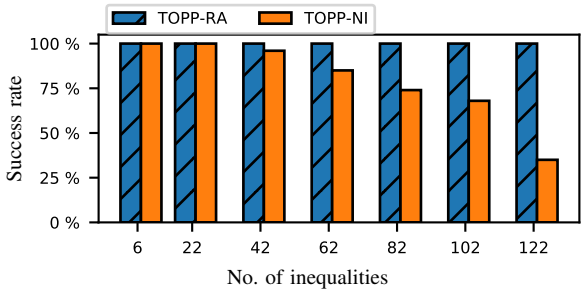

Fig. 4. Success rate for TOPP-RA and TOPP-NI. TOPP-RA enjoys consistently 100% success rate while TOPP-NI reports failure for more complex problem instances ($m \geq 40$).

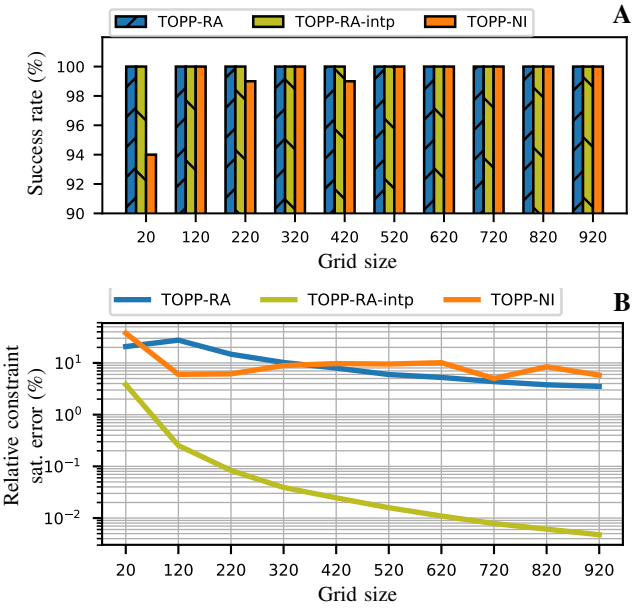

Fig. 5. **(A)**: effect of grid size on success rate. **(B)**: effect of grid size on solution quality, as measured by the relative constraint satisfaction error, defined as the ratio between the error and the respective kinematic bounds. We observe that TOPP-RA-intp returns solutions that are orders of magnitude better than TOPP-RA and TOPP-NI. While TOPP-RA and TOPP-RA-intp are very robust with respect to grid size.

by the i-th contact at point $\mathbf{p}_i$. Using the linearized friction cone, one obtains the set of feasible wrenches as a polyhedral cone

$$\{\mathbf{w}_i \mid \mathbf{F}_i \mathbf{w}_i \leq 0\},$$

for some matrix $\mathbf{F}_i$. This matrix can be found using the Cone Double Description method [21, 13]. Combining with

the equation governing rigid-body dynamics, we obtain the full dynamic feasibility constraint as follow

$$\begin{aligned}\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}+\dot{\mathbf{q}}^\top\mathbf{C}(\mathbf{q})\dot{\mathbf{q}}+\mathbf{g}(\mathbf{q}) &= \boldsymbol{\tau}+\sum_{i=1,2}\mathbf{J}_i(\mathbf{q})^\top\mathbf{w}_i,\\ \mathbf{F}_i\mathbf{w}_i &\le 0,\\ \boldsymbol{\tau}_{\min} \le \boldsymbol{\tau} &\le \tau_{\max},\end{aligned}$$

where $\mathbf{J}_i(\mathbf{q})$ is the wrench Jacobian. The convex set $\mathscr{C}(\mathbf{q})$ in Equation (1) can now be identified as a multi-dimensional polyhedron.

We considered a simple swaying motion: the robot stands with both feet lie flat on two uneven steps and shift its body back and forth, see Fig. 6. The coefficient of friction was set to $\mu = 0.5$. Start and end path velocities were set to zero. Discretization grid size was $N = 100$. The number of constraint inequalities was $m = 242$.

b) Results: Excluding computation of dynamic quantities, TOPP-RA took $267\,\mathrm{ms}$ to solve for the time-optimal path parameterization on our computer. The final parameterization is shown in Fig. 6 and computation time is presented in Table II.

Compared to TOPP-NI and TOPP-CO, TOPP-RA had significantly better computation time, chiefly because both existing methods require an expensive polytopic projection step. Indeed, [10] reported projection time of $2.4\,\mathrm{s}$ for a similar sized problem, which is significantly more expensive than TOPP-RA computation time. Notice that in [10], computing the parameterization takes an addition $2.46\,\mathrm{s}$ which leads to a total computation time of $4.86\,\mathrm{s}$.

To make a more accurate comparison, we implement the following pipeline on our computer to solve the same problem [11]

1) project the constraint polyhedron $\mathscr{C}_i$ onto the path using Bretl's polygon recursive expansion algorithm [17];
2) parameterize the resulting problem using TOPP-NI.

This pipeline turned out to be much slower than TOPP-RA. We found that the number of LPs the projection step solved is nearly 8 times more than the number of LPs solved by TOPP-RA (which is fixed at $3N = 300$). For a more detailed comparison of computation time and parameters of the LPs, refer to Table II.

c) Obtaining joint torques and contact forces "for free": Another interesting feature of TOPP-RA is that the algorithm can optimize and obtain joint torques and contact forces "for free" without additional processing. Concretely, since joint torques and contact forces are slack variables, one can simply store the optimal slack variable at each step and obtain a trajectory of feasible forces. To optimize the forces, we can solve the following quadratic program (QP) at the i-th step of the forward pass

$$\begin{aligned}\min\quad & -u+\epsilon\|(\mathbf{w},\boldsymbol{\tau})\|_2^2\\ \text{s.t.}\quad & x = x_i\\ & (u,x)\in\Omega_i\\ & x+2\Delta_i u \in K_{i+1},\end{aligned}$$

where ϵ is a positive scalar. Figure 6's lower plot shows computed contact wrench for the left leg. We note that both existing approaches, TOPP-NI and TOPP-CO are not able to produce joint torques and contact forces readily as they "flatten" the constraint polygon in the projection step.

In fact, the above formulation suggests that time-optimality is simply a specific objective cost function (linear) of the more general family of quadratic objectives. Therefore, one can in principle depart from time-optimality in favor of more realistic objective such as minimizing torque while maintaining a certain nominal velocity x_{norm} as follow

$$\begin{aligned}\min\quad & \|x_i+2\Delta_i u - x_{\mathrm{norm}}\|_2^2+\epsilon\|(\mathbf{w},\boldsymbol{\tau})\|_2^2\\ \text{s.t.}\quad & x = x_i\\ & (u,x)\in\Omega_i\\ & x+2\Delta_i u \in K_{i+1}.\end{aligned}$$

Finally, we observed that the choice of path discretization scheme has noticeable effects on both computational cost and quality of the result. In general, TOPP-RA-intp produced smoother trajectories and better (lower) constraint satisfaction error at the cost of longer computation time. On the other hand, TOPP-RA was faster but produced trajectories with jitters [2] near dynamic singularities [4] and had worse (higher) constraint satisfaction error.

TABLE II
COMPUTATION TIME (ms) AND INTERNAL PARAMETERS COMPARISON BETWEEN TOPP-RA, TOPP-NI AND TOPP-RA-INTP (FIRST-ORDER INTERPOLATION) IN EXPERIMENT 2.

	TOPP-RA	TOPP-RA-intp	TOPP-NI
	Time (ms)		
comp. dynamic quantities	181.6	193.6	281.6
polytopic projection	0.0	0.0	3671.8
solve TOPP	267.0	1619.0	335.0
extract trajectory	3.0	3.0	210.0
total	451.6	1815.6	4497.8
	Parameters		
joint torques / contact forces avail.	yes	yes	no
No. of LP(s) solved	300	300	2110
No. of variables	64	126	64
No. of constraints	242	476	242
Constraints sat. error	$O(\Delta)$	$O(\Delta^2)$	$O(\Delta)$

VI. ADDITIONAL BENEFITS OF TOPP BY REACHABILITY ANALYSIS

We now elaborate on the additional benefits provided by the reachability analysis approach to TOPP.

A. Admissible Velocity Propagation

Admissible Velocity Propagation (AVP) is a recent concept for kinodynamic motion planning [3]. Specifically, given a path and an initial interval of velocities, AVP returns exactly

[2]Our experiments show that singularities do not cause parameterization failures for TOPP-RA and the jitters can usually be removed easily.

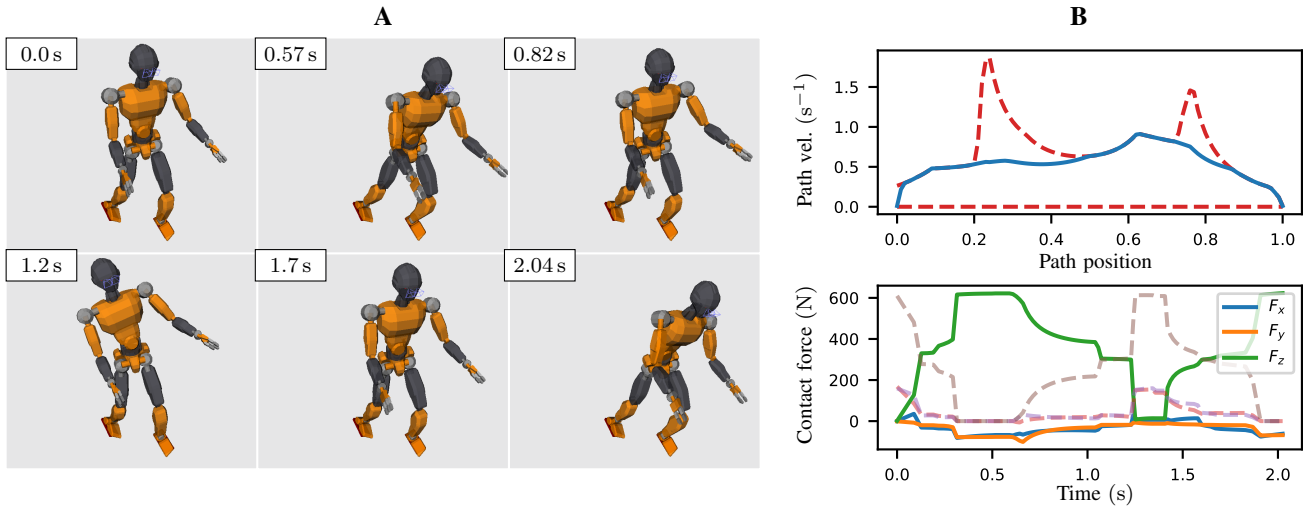

Fig. 6. Time-parameterization of a legged robot trajectory under joint torque bounds and multi-contact friction constraints. **(A)**: Snapshots of the retimed motion. **(B)**: Optimal velocity profile computed by TOPP-RA (blue) and upper and lower limits of the controllable sets (dashed red). **(C)**: The optimal joint torques and contact forces are obtained "for free" as slack variables of the optimization programs solved in the forward pass. Net contact forces for the left foot are shown in colors and those for the right foot are shown in transparent lines.

the interval of all the velocities the system can reach after traversing the path while respecting the system kinodynamic constraints. Combined with existing *geometric* path planners, such as RRT [22], this can be advantageously used for *kinodynamic* motion planning: at each tree extension in the configuration space, AVP can be used to guarantee the eventual existence of admissible path parameterizations.

Suppose that the initial velocity interval is $\mathbb{I}_0$. It can be immediately seen that, what is computed by AVP is exactly the reachable set $\mathcal{R}_N(\mathbb{I}_0)$ (cf. Section III-B). Furthermore, what is computed by AVP-Backward [23] given a desired final velocity interval $\mathbb{I}_N$ is exactly the controllable set $\mathcal{K}_0(\mathbb{I}_N)$ (cf. Section III-C). In terms of complexity, $\mathcal{R}_N(\mathbb{I}_0)$ and $\mathcal{K}_0(\mathbb{I}_N)$ can be found by solving respectively $4N$ and $2N$ LPs. We have thus re-derived the concepts of AVP at no cost.

B. Robustness to parametric uncertainty

In most works dedicated to TOPP, including the development of the present paper up to this point, the parameters appearing in the dynamics equations and in the constraints are supposed to be exactly known. In reality, those parameters, which include inertia matrices or payloads in robot manipulators, or feet positions or friction coefficients in legged robots, are only known up to some precision. An admissible parameterization for the nominal values of the parameters might not be admissible for the actual values, and the probability of constraints violation is even higher in the *optimal* parameterization, which saturates at least one constraint at any moment in time.

TOPP-RA provides a natural way to handle parameter uncertainty. Assume that the constraints appear in the following form

$$\begin{aligned} &\mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathscr{C}_i, \\ &\forall (\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathscr{C}_i) \in \mathcal{E}_i, \end{aligned} \tag{14}$$

where $\mathcal{E}_i$ contains all the possible values that the parameters might take at path position s_i.

Implementation remark 6. Consider for instance the manipulator with torque bounds of equation (2). Suppose that, at path position i, the inertia matrix is uncertain, i.e., that it might take any values $\mathbf{M}_i \in B(\mathbf{M}_i^{\text{nominal}}, \epsilon)$, where $B(\mathbf{M}_i^{\text{nominal}}, \epsilon)$ denotes the ball of radius ϵ centered around $\mathbf{M}_i^{\text{nominal}}$ for the max norm. Then, the first component of $\mathcal{E}_i$ is given by $\{\mathbf{M}_i \mathbf{q}'(s_i) \mid \mathbf{M}_i \in B(\mathbf{M}_i^{\text{nominal}}, \epsilon)\}$, which is a convex set.

In legged robots, uncertainties on feet positions or on friction coefficients can be encoded into a "set of sets", in which $\mathscr{C}_i$ can take values ■

We can now define the robust variants of reachable and controllable sets. For simplicity, we only detail the reachable variants, the controllable ones can be constructed by duality.

Definition 5 (i-stage *robust* reachable set)**.** Consider a set of starting states $\mathbb{I}_0$. The *i-stage robust reachable set* $\widehat{\mathcal{L}}_i(\mathbb{I}_0)$ is the set of states $x \in \mathcal{X}_i$ such that there exist a state $x_0 \in \mathbb{I}_0$ and a sequence of controls $u_0, \dots, u_{i-1}$ that steers the system from x_0 to x, where $u_0, \dots, u_{i-1}$ are admissible *for all possible values* of the parameters in $\mathcal{E}_0 \times \cdots \times \mathcal{E}_{i-1}$ ■

Denote the i-stage set of *robust admissible* control-state pairs by

$$\widehat{\Omega}_i := \{(u, x) \mid \text{Equation (14) holds}\}.$$

The sets of robust admissible states $\widehat{\mathcal{X}}_i$ and robust admissible controls $\widehat{\mathcal{U}}_i(x)$ given a state x can be defined similarly. Next, one can define the robust reach set.

Definition 6 (*Robust* reach set)**.** Consider a set of states $\mathbb{I}$. The *robust reach set* $\widehat{\mathcal{R}}_i(\mathbb{I})$ is the set of states x such that there

exist a state $\tilde{x} \in \mathbb{I}$ and a robust admissible control $u \in \widehat{\mathcal{U}}_i(\tilde{x})$ that steers the system from $\tilde{x}$ to x ■

Conceptually, the problem of *robust path parameterization* problem is solved if one can compute the robust reach sets and robust one-step sets.

Implementation remark 7. Computing the robust one-step set and the robust reach set involves solving LPs with uncertain constraints of the form (14). In general, these constraints may contain hundreds of inequalities, making them difficult to handle by generic methods. In the mathematical optimization literature, they are known as "Robust Linear Programs", and specific methods have been developed to handle them efficiently, when the robust constraints are [24]

1) polyhedra;
2) ellipsoids;
3) Conic Quadratic re-presentable (CQr) sets.

The first case can be treated as normal LPs with appropriate slack variables, while the last two cases are explicit Conic Quadratic Program (CQP). For more information on this conversion, refer to the first and second chapters of [24] ■

VII. Conclusion

We have presented a new approach to solve the Time-Optimal Path Parameterization (TOPP) problem based on Reachability Analysis. The key insight is to compute, in a first pass, the sets of controllable states, for which admissible controls allowing to reach to goal are guaranteed to exist. Time-optimality can then be obtained, in a second pass, by a simple greedy strategy. We have shown, through theoretical complexity analysis and extensive experiments, that the proposed algorithm is extremely robust (100% success rate) and competitive in terms of computation time as compared to the fastest known TOPP implementation [4]. Finally, the new approach yields additional benefits: no need for polytopic projection in the redundantly-actuated case, Admissible Velocity Projection, and robustness to parameter uncertainty.

Extracting the time-optimal parameterization is only one of the possible applications of the Reachability Analysis of the path-projected dynamics. Another potentially very useful application could be to compute the "parameterizability index", which would quantify "how much" the parameterization has failed when it fails. This could be obtained as the distance between the forward-propagated reachable sets and the backward-propagated controllable sets – when the path is parameterizable, there exists at least a path position where the two sets intersect, yielding a zero distance. Such a parameterizability index would be helpful to guide the selection of the underlying path.

A recognized disadvantage of the classical TOPP formulation is that the time-optimal trajectory contains hard acceleration switches, corresponding to infinite jerks. Solving TOPP subject to jerk bounds, however, is not possible using CO-based approach as the problem becomes non-convex [9]. Some prior works proposed to either extend the NI-based approach [25, 26] or to represent the parameterization as a spline and optimize directly over the parameter space [27, 28]. Exploring how Reachability Analysis can be extended to handle jerk bounds is another direction of our future research.

Acknowledgment

This work was partially supported by grant ATMRI:2014-R6-PHAM awarded by NTU and the Civil Aviation Authority of Singapore to the last author.

Appendix

A. Relation between TOPP-RA and TOPP-NI

TOPP-RA and TOPP-NI are subtly related: they in fact compute the same velocity profiles, but in different orders. To facilitate this discussion, let us first recall some terminologies from the classical TOPP literature (refer to [4] for more details)

- Maximum Velocity Curve (MVC): a mapping from a path position to the highest dynamically feasible velocity;
- Integrate forward (or backward) following α (or β): for each tuple $s, \dot{s}$, $\alpha(s, \dot{s})$ is the smallest control and $\beta(s, \dot{s})$ is the greatest one; we integrate forward and backward by following the respective vector field (α or β);
- $\alpha \to \beta$ switch point: there are three kinds of switch points: tangent, singular, discontinuous;
- $\dot{s}_{\text{beg}}, \dot{s}_{\text{end}}$: starting and ending velocity at path positions 0 and s_{end} respectively.

TOPP-NI proceeds as follows

1) determine the $\alpha \to \beta$ switch points;
2) from each $\alpha \to \beta$ switch point, integrate forward following β and backward following α to obtain the Limiting Curves (LCs);
3) take the lowest value of the LCs at each position to form the Concatenated Limiting Curve (CLC);
4) from $(0, \dot{s}_{\text{beg}})$ integrate forward following β; from $(s_{\text{end}}, \dot{s}_{\text{end}})$ integrate backward following α until their intersections with the CLC; then return the combined $\beta - \text{CLC} - \alpha$ profile.

We now rearrange the above steps so as to compare with the two passes of TOPP-RA, see Fig. 7

Backward pass

1) determine the $\alpha \to \beta$ switch points;
2a) from each $\alpha \to \beta$ switch point, integrate backward following α to obtain the Backward Limiting Curves (BLCs);
2b) from the point $(s_{\text{end}}, \dot{s}_{\text{end}})$ integrate backward following α to obtain the last BLC;
3) take the lowest value of the BLC's *and the* MVC at each position to form the upper boundary of the controllable sets.

Forward pass

4a) set the current point to the point $(s_{\text{beg}}, \dot{s}_{\text{beg}})$;
4b) repeat until the current point is the point $(s_{\text{end}}, \dot{s}_{\text{end}})$, from the current point integrate forward following β until hitting a BLC, set the corresponding switch point as the new current point.

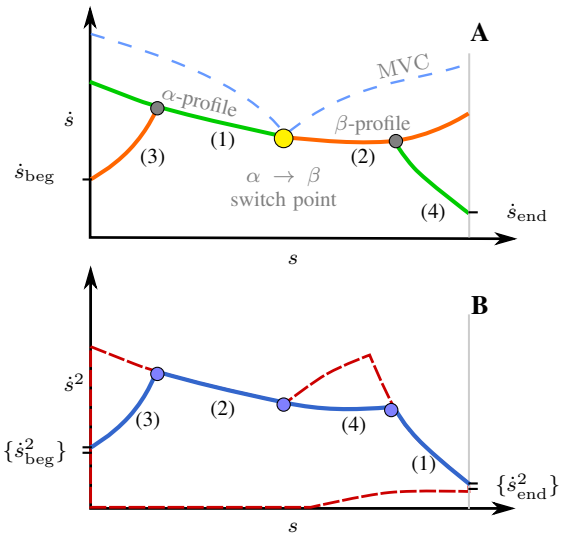

Fig. 7. TOPP-NI **(A)** and TOPP-RA **(B)** compute the time-optimal path parameterization by creating similar profiles in different ordering. (1,2,3,4) are the orders in which profiles are computed.

The key idea in this rearrangement is to not compute β profiles immediately for each switch point, but delay until needed. The resulting algorithm is almost identical to TOPP-RA except for the following points

- since TOPP-RA does not require to explicitly compute the switch points (they are implicitly identified by computing the controllable sets) the algorithm avoids one of the major implementation difficulties of TOPP-NI;
- TOPP-RA requires additional post-processing to remove the jitters.

For a more detailed remark on the last point, see the last paragraph of Appendix B .

B. Proof of the monotonicity of the maximal transition functions

The monotonicity of the maximal transition functions is the subtle point where dynamic singularities – another main issue associated with TOPP-NI – appear in our analysis. Our proof of the monotonicity relies on the procedure proposed by [4] to handle dynamic singularities.

Proof. (Monotonicity property)

Without dynamic singularities: We first consider the case where there is no singularities. The maximal function $T_\beta(i,x)$ returns the optimal objective of the following LP

$$\begin{aligned} \max \quad & (2\Delta, 1)(u,x)^\top \\ \text{s.t.} \quad & (x,u) \in \Omega_i. \end{aligned}$$

Geometrically, this LP looks for the furthest vertex along the direction $(2\Delta, 1)$. By inspection of Figure 8, we see that monotonicity of the optimal value is ensured if

$$\frac{\mathbf{a}_i[k]}{\mathbf{b}_i[k]} \geq \frac{2\Delta_i}{1}, \tag{15}$$

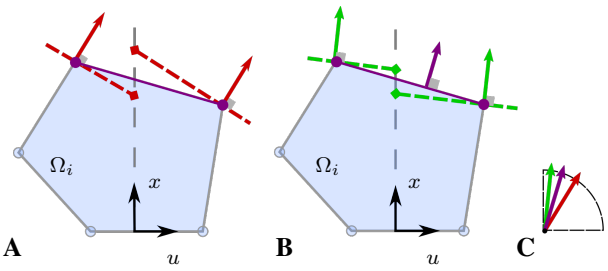

Fig. 8. The monotonicity property: at a given stage i, the maximal transition functions $T_\alpha(i,x)$ and $T_\beta(i,x)$ can be made monotone by increasing N or equivalently decreasing Δ_i sufficiently. **A**: smaller N/greater Δ_i case; the transition function $T_\beta(i,x)$ maximizes at a velocity different from the highest admissible velocity (red). **B**: greater N/smaller Δ_i case; $T_\beta(i,x)$ maximizes at the highest admissible velocity (green). **C**: Comparing the maximizing directions (red and green) with the normal of the active constraint (purple).

where k is the constraint that is active at the greatest admissible velocity, and that $\mathbf{a}_i[k], \mathbf{b}_i[k] \geq 0$.

As discussed in [4], when there is no dynamic singularity, there exists a lower bound on $\mathbf{a}(s)[k], s \in [0, s_{\text{end}}]$ (note that k is redefined for each different position s). Thus we simply choose N so that condition (15) is respected.

With dynamic singularities: First, notice that the above argument does not work in this case, as there exists singular positions such that $\mathbf{a}(s)[k]$ becomes zero.

To overcome this issue, we augment the list of constraints near a singularity with a set of maximal velocity bounds computed from the singularity itself. In particular, we do the following:

- compute all singularities at s_i^*;
- integrate backward following α and forward following β from each singularity (following the procedure described in [4]), denote the profiles by $\rho_i^*(s) : s \in [s_i^* - \delta, s_i^* + \delta]$;
- augment the following constraint

$$x = \dot{s}^2 \leq \rho_i^*(s), \text{ if } s \in [s_i^* - \delta, s_i^* + \delta], \forall i.$$

To finish the proof we need to show that (i) the augmented set of constraints is equivalent to the original set and (ii) the argument we used in the first case is now applicable.

Indeed, (i) follows from the discussion in [4] and (ii) follows trivially by noting that the modified admissible set in the vicinity of singularities have "flat top": its top edge is horizontal. □

Implementation remark 8. In practice, finding singular switch points is undesirable as it increases the algorithm's complexity level. We find that not accounting for singularities does not affect TOPP-RA significantly if first-order interpolation discretization scheme is used. For collocation discretization scheme, while one would observe some jitters on the final profile near singularities, the algorithm will not fail as guaranteed by the existence of the controllable sets. One can then remove the jitters by simple smoothing.

C. Error analysis for different discretization schemes

1) First-order interpolation scheme: We have presented in the main text the collocation scheme to discretize the constraints. Before proceeding to analyzing the errors, we first introduce another the another scheme, namely the first-order interpolation scheme.

For each interval $[s_i, s_{i+1}]$, one interpolates the system dynamics using a first-order approximation, and requires the constraints to be verified at the end points s_i and s_{i+1}. For our specific problem, this scheme imposes that (u, x) and $(u, x + 2\Delta_i u)$ satisfy the constraints at $s = s_i$ and $s = s_{i+1}$ respectively, i.e., for $i = 0 \dots N-1$

$$\begin{bmatrix} \mathbf{a}(s_i) \\ \mathbf{a}(s_{i+1}) + 2\Delta_i \mathbf{b}(s_{i+1}) \end{bmatrix} u + \begin{bmatrix} \mathbf{b}(s_i) \\ \mathbf{b}(s_{i+1}) \end{bmatrix} x + \begin{bmatrix} \mathbf{c}(s_i) \\ \mathbf{c}(s_{i+1}) \end{bmatrix} \in \begin{bmatrix} \mathscr{C}(s_i) \\ \mathscr{C}(s_{i+1}) \end{bmatrix} \tag{16}$$

At $i = N$, one uses only the top half of the above equations. After appropriately lumping together the different vectors and sets, the above equations can finally be rewritten as

$$\mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathscr{C}_i. \tag{17}$$

2) Error analysis: We now analyze the constraints satisfaction errors for the collocation and the first-order interpolation discretization schemes. On the interval $[s_0, s_1]$, the parameterization is given exactly by

$$x(s; u_0, x_0) = x_0 + 2su_0,$$

where x_0 is the state at s_0 and u_0 is the constant control along the interval.

Let us define the constraint satisfaction error function as follow

$$\mathbf{e}(s; u_0, x_0) = \mathbf{a}(s)u_0 + \mathbf{b}(s)x(s) + \mathbf{c}(s). \tag{18}$$

Different discretization schemes enforce different conditions on $\mathbf{e}(\cdot)$. In particular, we have

- *collocation scheme*:

$$\mathbf{e}(s_0; u_0, x_0) \leq 0,$$

- *first-order interpolation scheme*:

$$\mathbf{e}(s_0; u_0, x_0) \leq 0, \quad \mathbf{e}(s_1; u_0, x_0) \leq 0.$$

where $s_0 = 0, s_1 = \Delta_0$. From here on we omit u_0 and x_0 in the error functions.

Using the classic result on Error of Polynomial Interpolation [29, Theorem 2.1.4.1], we obtain the following estimation of the error function for the *collocation scheme*:

$$\mathbf{e}(s) = \mathbf{e}(s_0) + (s - s_0)\mathbf{e}'(\epsilon) = s\mathbf{e}'(\epsilon),$$

for some $\epsilon \in [s_0, s]$. Suppose the derivatives of $\mathbf{a}(\cdot), \mathbf{b}(\cdot), \mathbf{c}(\cdot)$ are bounded, we have

$$\max_{s \in [0, \Delta_0]} \mathbf{e}(s) = O(\Delta_0).$$

Using the same theorem, we obtain the following estimation for constraint error of first-order interpolation scheme

$$\begin{aligned} \mathbf{e}(s) =& \mathbf{e}(s_0) + (s - s_0)\frac{\mathbf{e}(s_1) - \mathbf{e}(s_0)}{s_1 - s_0} \\ &+ \frac{(s - s_0)(s - s_1)\mathbf{e}''(\epsilon)}{2!} \\ =& \frac{s(s - \Delta_0)\mathbf{e}''(\epsilon)}{2!}, \end{aligned}$$

for some $\epsilon \in [s_0, s_1]$. Again, since the derivatives of coefficients are assumed to be bounded, we see that the maximum error satisfies

$$\max_{s \in [0, \Delta_0]} \mathbf{e}(s) = O(\Delta_0^2).$$

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] Z. Shiller and S. Dubowsky, "On computing the global time-optimal motions of robotic manipulators in the presence of obstacles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 785–797, 1991.

[3] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 44–67, 2017. [Online]. Available: http://dx.doi.org/10.1177/0278364916675419

[4] 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. [Online]. Available: http://ieeexplore.ieee.org/document/6895310/

[5] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 2, pp. 115–123, 1987.

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

[7] Z. Shiller and H.-H. Lu, "Computation of path constrained time optimal motions with dynamic singularities," *Journal of dynamic systems, measurement, and control*, vol. 114, no. 1, pp. 34–40, 1992.

[8] L. Zlajpah, "On time optimal path control of manipulators with bounded joint velocities and torques," in *IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1996, pp. 1572–1577.

[9] 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.

[10] K. Hauser, "Fast interpolation and time-optimization with contact," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1231–1250, aug 2014.

[Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364914527855

[11] 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. [Online]. Available: http://ieeexplore.ieee.org/document/7083769/

[12] S. Caron, Q.-C. Pham, and Y. Nakamura, "Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics," *Robotics: Science and System*, 2015.

[13] ——, "ZMP Support Areas for Multicontact Mobility Under Frictional Constraints," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 67–80, feb 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7782743/

[14] E. C. Kerrigan, "Robust constraint satisfaction: Invariant sets and predictive control," Ph.D. dissertation, University of Cambridge, 2001.

[15] 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. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0005109871900665

[16] S. V. Rakovic, E. C. Kerrigan, D. Q. Mayne, and J. Lygeros, "Reachability analysis of discrete-time systems with disturbances," *IEEE Transactions on Automatic Control*, vol. 51, no. 4, pp. 546–561, 2006.

[17] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *Robotics, IEEE Transactions on*, vol. 24, no. 4, pp. 794–807, 2008.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[19] J.-J. E. Slotine and H. S. Yang, "Improving the efficiency of time-optimal path-following algorithms," *Robotics and Automation, IEEE Transactions on*, vol. 5, no. 1, pp. 118–124, 1989.

[20] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, dec 2014. [Online]. Available: http://link.springer.com/10.1007/s12532-014-0071-1

[21] K. Fukuda and A. Prodon, "Double description method revisited," in *Combinatorics and computer science*. Springer, 1996, pp. 91–111.

[22] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[23] P. Lertkultanon and Q.-C. Pham, "Dynamic nonprehensile object transportation," in *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*. IEEE, 2014, pp. 1392–1397.

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

[25] M. Tarkiainen and Z. Shiller, "Time optimal motions of manipulators with actuator dynamics," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 725–730.

[26] H. Pham and Q.-C. Pham, "On the Structure of the Time-Optimal Path Parameterization Problem with Third-Order Constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, A. M. Okamura, Ed. Singapore: IEEE, 2017. [Online]. Available: http://arxiv.org/abs/1609.05307

[27] D. Costantinescu 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.

[28] M. Oberherber, H. Gattringer, and A. Müller, "Successive dynamic programming and subsequent spline optimization for smooth time optimal robot path tracking," *Mechanical Sciences*, vol. 6, no. 2, pp. 245–254, 2015.

[29] J. Stoer and R. Bulirsch, "Introduction to Numerical Analysis," p. 96, 1982. [Online]. Available: http://link.aip.org/link/SIREAD/v24/i1/p96/s1{&}Agg=doi