Time-Optimal Path Parameterization for Redundantly-Actuated Robots (Numerical Integration Approach)

Quang-Cuong Pham, Olivier Stasse

Time-Optimal Path Parameterization for Redundantly-Actuated Robots (Numerical Integration Approach)

Quang-Cuong Pham and Olivier Stasse

***Abstract*—Time-Optimal Path Parameterization (TOPP) under actuation bounds plays a fundamental role in many robotic theories and applications. This algorithm was first developed and perfected for classical serial robotic manipulators whose actuation is non-redundant. Yet, redundantly-actuated systems, such as parallel manipulators or humanoid robots in multi-contact tasks, are increasingly common in all fields of robotics. Here we extend the classical algorithm of TOPP (a.k.a. numerical integration approach) to the case of redundantly-actuated systems. As illustration, we present an application to multi-contact trajectory planning for a humanoid robot.**

I. Introduction

Given a robot and a smooth path in its configuration space, the problem of finding the Time-Optimal Parameterization of that Path (TOPP) under the robot actuation constraints (e.g. bounds on the joint torques) plays a fundamental role in many robotics theories and applications. For serial, normally-actuated manipulators (i.e. every joint is actuated), an efficient algorithm to solve this problem was first proposed in the 1980's [1], [2] and has been continuously perfected since then, see [3] for a historical review.

TOPP for different degrees of robot actuation

Normally-actuated systems constitute however only a subclass of existing robot systems : under- and redundantly-actuated systems are increasingly common in all fields of robotics. Cases of *under-actuation* arise when some of the degrees of freedom of the robot are not actuated : consider for instance a serial manipulator in which some of the joints are not actuated, or a flying robot whose non-actuated degrees of freedom correspond to the position and orientation of the base-link. In such cases, the difficulty for TOPP consists in that a given path in the robot configuration space may not be trackable at all, let alone time-parameterized. This case was essentially solved in [4], [5], where the authors give a characterization of the paths that can be time-parameterized. For a given time-parameterizable path, TOPP is no different from the normal actuation case.

Redundantly-actuated (or over-actuated) systems have more actuated joints than degrees of freedom. Actuation redundancy can be by design, such as in parallel (or closed-chain) manipulators, or can arise spontaneously when an otherwise non-redundantly-actuated robot interacts with the environment (consider e.g. a humanoid robot having two flat feet on the ground, or when it manipulates an object with both hands). In the former case, actuation redundancy is a means to eliminate singularities, to provide more homogeneous output forces, or to minimize the internal loading of the actuators [6]. As far as TOPP is concerned, actuation redundancy implies that any path in the robot configuration space can be time-parameterized (subject to actuation bounds), but contrary to the normal- and under-actuation cases, a given time-parameterization of the path may correspond to *infinitely many* possible torque profiles [7], [8], [6], [9]. One possible way to deal with that redundancy is to fix a predefined load distribution [7]. However, it is clear that such a solution is suboptimal and would give rise to slower trajectories than what can be obtained using the full actuation available to the system. Our purpose here is to extend TOPP so that it can *optimally* handle actuation redundancy.

Quang-Cuong Pham is with School of Mechanical and Aerospace Engineering, NTU, Singapore. Olivier Stasse is with LAAS-CNRS, France.

Main approaches to TOPP

There are two main approaches to TOPP : numerical integration and convex optimization. *Numerical integration* is based on the Pontryagin Maximum Principle : the optimal velocity profile in the $s - \dot{s}$ plane [$s(t)$ denotes the position on the path at a given time instant t] is known to be "bang-bang" and can thus be found by integrating successively the maximum and minimum accelerations $\ddot{s}$, see [3] and also Section II-A for details. *Convex optimization* discretizes the s-axis into N segments and subsequently converts the original problem into a convex optimization problem in $O(N)$ variables and $O(N)$ equality and inequality constraints [10]. This approach has the advantage of being versatile (it can for instance trade off time duration with other objectives such as energy or torque rate) and can rely on existing efficient convex optimization packages. However, the sheer size of the convex optimization problem makes this approach slower than its numerical integration counterpart, which exploits the bang-bang structure of the TOPP problem in order to compute directly the optimal accelerations without any search process. Experimental comparisons in the case of non-redundantly-actuated robots supported this analysis [3].

Recently, the convex optimization approach has been extended to the case of humanoid robots in multi-contact tasks [11], which is a particular case of actuation redundancy [9]. The key of this extension lies in an ingenious *polytope projection technique* [12], which converts actuation constraints into "polygon" constraints in the $\ddot{s} - \dot{s}^2$ plane.

Our contribution in the present paper is twofold. First, we show how the numerical integration approach can also handle "polygon" constraints, and implement the algorithm to do so as a new module in the TOPP library (https://github.com/quangounet/TOPP). This algorithm, combined with the polytope projection technique [12], [11], enables the numerical integration approach to address the case of redundant actuation, which in turn yields a significant improvement in computation time as compared to the convex optimization approach. Second, we show that the torque constraints for closed-chain manipulators can be reduced to the polygon form, which enables a unified, efficient, treatment of this class of robots together with humanoid robots in multi-contact tasks.

The remainder of this paper is organized as follows. In Section II, we show how to address "polygon" constraints within the numerical integration approach to TOPP and discuss how to reduce the actuation constraints of some general redundantly-actuated systems into these "polygon" constraints. In Section III, we present an application in a multi-contact task where a humanoid robot steps down a 30 cm-high podium. We show in particular that our approach enables a significant speed-up in computation time as compared to the convex optimization approach. Finally, Section IV briefly discusses the obtained results and future research directions.

II. Time-Optimal Path Parameterization Considering Redundant Actuation

A. Background I : TOPP for non-redundantly-actuated systems

For completeness, we briefly recall below the classical TOPP algorithm for non-redundantly-actuated systems [1], [2]. For details, the reader is referred to [3].

Consider an n-dof serial manipulator with dynamics

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \tag{1}$$

where $\mathbf{q}$ is the $n \times 1$ vector of joint values, $\mathbf{M}$ the $n \times n$ manipulator inertia matrix, $\mathbf{C}$ the $n \times n \times n$ Coriolis tensor, $\mathbf{g}$ the $n \times 1$ vector of gravity forces and $\boldsymbol{\tau}$ the $n \times 1$ vector of actuator torques.

Assume that the manipulator is subject to lower and upper bounds on the joint torques, that is, for every joint i and time instant t,

$$\tau_i^{\min} \leq \tau_i(t) \leq \tau_i^{\max}. \tag{2}$$

Consider now a path $\mathcal{P}$ – represented as the underlying path of a trajectory $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$ – in the configuration space. Assume that $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$ is C^1- and piecewise C^2-continuous. We are interested in *time-parameterizations* of $\mathcal{P}$ – or time-*re*parameterizations of $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$ – which are increasing *scalar functions* $s : [0, T'] \to [0, s_{\text{end}}]$.

To check whether a reparameterized trajectory satisfies the actuation bounds, one may differentiate $\mathbf{q}(s(t))$ with respect to t

$$\dot{\mathbf{q}} = \mathbf{q}_s \dot{s}, \quad \ddot{\mathbf{q}} = \mathbf{q}_s \ddot{s} + \mathbf{q}_{ss} \dot{s}^2, \tag{3}$$

where dots denote differentiations with respect to the time parameter t and $\mathbf{q}_s = \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}s}$ and $\mathbf{q}_{ss} = \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}s^2}$. Substituting (3) into (1) and (2) and rearranging the terms then lead to

$$\ddot{s}\mathbf{a}(s) + \dot{s}^2\mathbf{b}(s) + \mathbf{c}(s) \leq \mathbf{0}, \tag{4}$$

where $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are vectors of dimension $2n$.

Each row i of (4) is of the form

$$a_i(s)\ddot{s} + b_i(s)\dot{s}^2 + c_i(s) \leq 0. \tag{5}$$

Next,

- if $a_i(s) > 0$, then one has $\ddot{s} \leq \frac{-c_i(s)-b_i(s)\dot{s}^2}{a_i(s)}$. Define the acceleration *upper bound* $\beta_i \stackrel{\text{def}}{=} \frac{-c_i(s)-b_i(s)\dot{s}^2}{a_i(s)}$;
- if $a_i(s) < 0$, then one has $\ddot{s} \geq \frac{-c_i(s)-b_i(s)\dot{s}^2}{a_i(s)}$. Define the acceleration *lower bound* $\alpha_i \stackrel{\text{def}}{=} \frac{-c_i(s)-b_i(s)\dot{s}^2}{a_i(s)}$.

One can then define for each $(s, \dot{s})$

$$\alpha(s,\dot{s}) \stackrel{\text{def}}{=} \max_i \alpha_i(s,\dot{s}), \quad \beta(s,\dot{s}) \stackrel{\text{def}}{=} \min_i \beta_i(s,\dot{s}). \tag{6}$$

From the above transformations, one can conclude that $\mathbf{q}(s(t))_{t\in[0,T']}$ satisfies the constraints (4) [1] if and only if

$$\forall t \in [0, T'] \quad \alpha(s(t), \dot{s}(t)) \leq \ddot{s}(t) \leq \beta(s(t), \dot{s}(t)). \tag{7}$$

Next, observe that if $\alpha(s, \dot{s}) > \beta(s, \dot{s})$ then, from (7), there is no possible value for $\ddot{s}$. Thus, to be valid, every velocity profile must stay below the Maximum Velocity Curve (MVC) defined by

$$\text{MVC}(s) \stackrel{\text{def}}{=} \begin{cases} \min\{\dot{s} \geq 0 : \alpha(s,\dot{s}) = \beta(s,\dot{s})\} & \text{if } \alpha(s,0) \leq \beta(s,0), \\ 0 & \text{if } \alpha(s,0) > \beta(s,0). \end{cases} \tag{8}$$

It was shown (see e.g. [13]) that the time-minimal velocity profile is obtained by a *bang-bang*-type control, i.e., whereby the optimal profile follows alternatively the β and α fields while always staying below the MVC. More precisely, the algorithm to find the time-optimal parameterization of $\mathcal{P}$ starting and ending with the desired linear velocities v_{beg} and v_{end} is (see Fig. 1 for illustration):

1. In the $s - \dot{s}$ plane, start from $(s = 0, \dot{s} = v_{\text{beg}}/\|\mathbf{q}_s(0)\|)$ and integrate forward following β until hitting either

(i) the MVC, in this case go to step 2;
(ii) the horizontal line $\dot{s} = 0$, in this case the path is not dynamically traversable;
(iii) the vertical line $s = s_{\text{end}}$, in this case go to step 3.

2. Search forward along the MVC for the next candidate $\alpha \to \beta$ switch point (cf. [3]). From such a switch point:

(a) integrate *backward* following α, until *intersecting* a forward β-profile (from step 1 or recursively from the current step 2). The intersection point constitutes a $\beta \to \alpha$ switch point;

[1] Joint velocity constraints can also be taken into account, see [3].

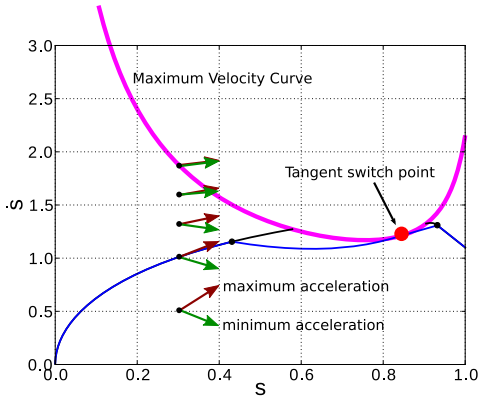

Fig. 1. Numerical integration approach to TOPP. The optimal velocity profile (blue) is obtained by integrating alternatively the maximum and minimum acceleration fields.

(b) integrate *forward* following β. Then continue as in step 1.

The resulting forward profile will be the concatenation of the intersected forward β-profile, the backward α-profile obtained in (a), and the forward β-profile obtained in (b).

3. Start from $(s = s_{\text{end}}, \dot{s} = v_{\text{end}}/\|\mathbf{q}_s(s_{\text{end}})\|)$ and integrate *backward* following α, until intersecting a forward profile obtained in steps 1 or 2. The intersection point constitutes a $\beta \to \alpha$ switch point. The final profile will be the concatenation of the intersected forward profile and the backward α-profile just computed.

B. Background II : polytope projection technique

For clarity, consider, in this section, a fixed position s_0 on the path. Inequalities (4) can be seen as constraining the vector $\ddot{s}\mathbf{a}(s_0) + \dot{s}^2\mathbf{b}(s_0) + \mathbf{c}(s_0)$ to stay in the *negative orthant* of $\mathbb{R}^{2n}$. As we shall see in Section II-D, actuation bounds in redundantly-actuated systems imply the following more general condition

$$\ddot{s}\mathbf{a}(s_0) + \dot{s}^2\mathbf{b}(s_0) + \mathbf{c}(s_0) \in \mathscr{C}, \tag{9}$$

where $\mathscr{C}$ is a (possibly unbounded) *convex polytope* of $\mathbb{R}^{2n}$.

The polytope projection technique [12], [11] enables one to compute, given the vectors $\mathbf{a}(s_0)$, $\mathbf{b}(s_0)$, $\mathbf{c}(s_0)$ and the set of linear constraints defining the convex polytope $\mathscr{C}$, the *convex polygon* $\mathscr{P}$ in the $\ddot{s} - \dot{s}^2$ plane such that

$$(\ddot{s}, \dot{s}^2) \in \mathscr{P} \quad \text{if and only if} \quad \ddot{s}\mathbf{a}(s_0) + \dot{s}^2\mathbf{b}(s_0) + \mathbf{c}(s_0) \in \mathscr{C}.$$

For completeness, we briefly recall below the polytope projection algorithm [12], [11]. Let first

$$\mathscr{C}' \stackrel{\text{def}}{=} \{(u, v, \mathbf{y}) : \ v \geq 0; \ u\mathbf{a}(s_0) + v\mathbf{b}(s_0) + \mathbf{c}(s_0) = \mathbf{y}; \ \mathbf{y} \in \mathscr{C}\}.$$

Since the constraints defining $\mathscr{C}'$ are linear, $\mathscr{C}'$ is a convex polytope. Next, given a vector $\mathbf{d}$ in the plane, one can find the extreme point of $\mathscr{P}$ (or "extremize") in the direction of $\mathbf{d}$ by solving the following Linear Program (LP)

$$\max_{(u,v)} \mathbf{d}^\top(u, v) \quad \text{such that} \quad (u, v, \mathbf{y}) \in \mathscr{C}'. \tag{10}$$

The projection algorithms starts by finding extreme points of $\mathscr{P}$ in three directions of the plane, for instance $(1, 0)$, $(\cos(2\pi/3), \sin(2\pi/3))$, $(\cos(4\pi/3), \sin(4\pi/3))$. This gives an initial polygon – which is actually a triangle. The algorithm then iteratively attempts to expand this polygon by "extremizing" in the outward normal direction of unexpanded edges, see Fig. 2A. The algorithm stops when all edges have been expanded (in this case, the

final polygon is $\mathscr{P}$), or when one has reached the maximum number of iterations (in this case, the final polygon is a subset of $\mathscr{P}$).

In [11], the author then converts the obtained polygon back into M half-plane inequalities similar to (5) and feed these inequalities into a large convex optimization problem over the entire path.

C. TOPP-Polygon : numerical integration approach to redundantly-actuated systems

The fact that torque constraints – even in the non-redundant case – define a polygon in the $\ddot{s} - \dot{s}^2$ plane has been remarked as early as in [14], [13]. However, in the non-redundant case, it is more efficient to deal directly with the analytic expressions (6) and (8) rather than to construct an explicit polygon and use this polygon in subsequent computations.

By contrast, as shown in Section II-B, one is *given* the polygon in the redundant case. One possible approach might consist in converting the polygon into M half-plane inequalities similar to (5) and then try to apply the development of Section II-A. However, besides the computational cost required by the conversion, doing so would result in an $O(M^2)$ algorithm [to compute in particular the MVC by (8)], while proceeding directly with the polygon representation yields a simpler and faster $O(M)$ algorithm, as shown below.

To follow the numerical integration approach, one needs to compute the acceleration bounds α and β as well as the MVC. For a given $\dot{s}_0$, one can compute $\alpha(s_0, \dot{s}_0)$ and $\beta(s_0, \dot{s}_0)$ by simply finding the intersection of the horizontal line $\dot{s}^2 = \dot{s}_0^2$ with the edges of the polygon, see Fig. 2B. This can be done in time $O(M)$ where M is the number of edges of the polygon. To compute $\mathrm{MVC}(s_0)$, it suffices to determine the highest vertex of the polygon, which can also be done in time $O(M)$.

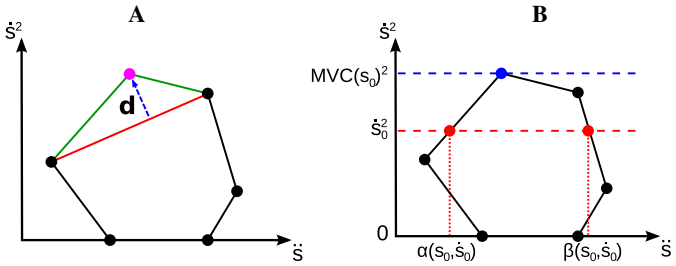

Fig. 2. **A** : Expanding an edge. At a given step of the algorithm, one chooses an edge that has not yet been expanded (red segment). One then computes the outward normal direction $\mathbf{d}$ to this edge and solves the LP defined in (10). The solution of this LP gives a new vertex (magenta disk), and one can finally replace the old edge (red segment) by two new edges (green segments). **B** : The constraint polygon at a given s_0 in the $\ddot{s} - \dot{s}^2$ plane.

The numerical integration approach requires furthermore determining the switch points on the MVC (cf. [3]). Determining the "discontinuous" and "tangent" switch points can be done as in [3] by stepping along the MVC. Regarding the "singular" switch points, it is not possible here to consider "zero-inertia" points [14], [13], [3] – which are the points where $a_i = 0$ in (5) – since the a_i are not defined[2]. Thus, one has to step along the MVC and determine numerically the points where the MVC is continuous but not differentiable. Finally, to compute the optimal accelerations at the singular points, one cannot use the analytical value suggested in [3], also because the a_i, b_i and c_i are not defined. Instead, we suggest here to search numerically for an *approximate* optimal acceleration as follows :

- Discretize the space of possible accelerations $(-\infty, +\infty)$ into a large number of values (100 in our implementation);
- For each discretized acceleration value β_0, integrate forward a small number of steps using β_0, evaluate the acceleration β_1 at the end of the integration, and record the absolute difference $|\beta_0 - \beta_1|$;
- Choose the acceleration value β_0 that minimizes the absolute difference $|\beta_0 - \beta_1|$.

Doing so ensures that the obtained profile is reasonably smooth around the singular switch point (see Fig. 3B).

We implemented the methods described above as a new TOPP-Polygon module in the TOPP library, see https://github.com/quangounet/TOPP.

As a sanity check, we compared TOPP-Polygon with the original algorithm in [3], on a model of a serial, normally-actuated 7-dof manipulator (since the original TOPP cannot handle redundantly-actuated systems). Fig. 3 shows that the two versions produced nearly identical results. The profile computed by TOPP-Polygon was slightly less smooth near the dynamic singularity (around $s = 0.6$) and more false-positive singularities were detected (green dots). However, TOPP-Polygon was 50% faster in terms of computation time. This shows in particular that it would be suboptimal to convert the polygon constraints back into half-plane inequalities and apply the original TOPP on these inequalities.

A detailed comparison of TOPP-Polygon with the convex optimization approach, in a redundant actuation setting, is reported in Section III.

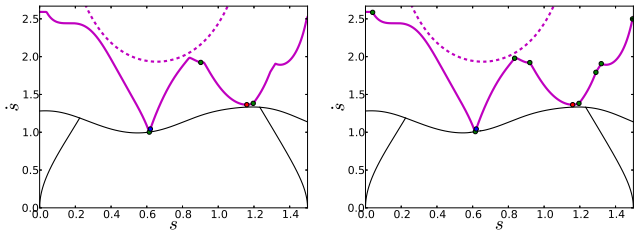

Fig. 3. Comparison of the original TOPP (**Left**) and TOPP-Polygon (**Right**) on a model of a serial 7-dof manipulator. The two methods produced nearly identical results (MVC in magenta and velocity profiles in black), but TOPP-Polygon was 50% faster. Note that the singular switch point (green disk near $s = 0.6$ could be appropriately addressed using the method described in the main text.

D. Reductions of some redundantly-actuated systems to TOPP-Polygon form

We now discuss how to reduce some general classes of redundantly-actuated robots to the form of equation (9).

1) Closed-chain manipulators: Consider a closed-chain (or parallel) manipulator as in Fig. 4. Let $\mathbf{q}$ denote the joint values of the whole chain and $\mathbf{q}_A$ the joint values of the n_A actuated joints. Assume that the torques of the actuated joints are subject to lower and upper bounds, that is, for all $i \in [1, n_A]$,

$$\tau_{Ai}^{\min} \leq \tau_{Ai} \leq \tau_{Ai}^{\max}. \tag{11}$$

Following [8], let $\mathbf{q}_O$ denote the joint values of the *tree structure* obtained by cutting the chain at some specific joints. Consider a motion $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ of the chain. Assume that the tree structure

[2]Even if one converts the polygon into half-plane inequalities, there would be, in general, no way to index these inequalities so that the a_i, b_i and c_i are continuous, since polygons computed at different points s_0 on the path may have different numbers of edges. Yet, the continuity of a_i is necessary to determine "zero-inertia" points, and the differentiability of a_i, b_i and c_i are required to compute the optimal acceleration near a dynamic singularity, see [3].

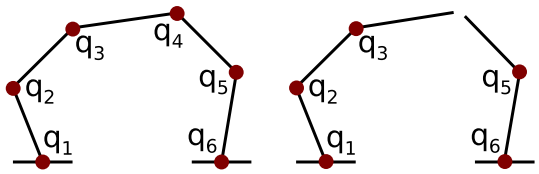

Fig. 4. **Left**: A closed chain manipulator. If all joints are actuated, then $\mathbf{q}_A = (q_1, q_2, q_3, q_4, q_5, q_6)$. **Right**: Tree structure obtained by cutting the chain at joint 4. Here the joints of the tree structures are $\mathbf{q}_O = (q_1, q_2, q_3, q_5, q_6)$.

makes the same motion as the closed chain. One can calculate the joint torques $\boldsymbol{\tau}_O$ of the tree structure by usual inverse dynamics algorithms [15]

$$\boldsymbol{\tau}_O = \mathbf{M}(\mathbf{q}_O)\ddot{\mathbf{q}}_O + \dot{\mathbf{q}}_O^\top \mathbf{C}(\mathbf{q}_O)\dot{\mathbf{q}}_O + \mathbf{g}(\mathbf{q}_O), \tag{12}$$

where $(\mathbf{q}_O, \dot{\mathbf{q}}_O, \ddot{\mathbf{q}}_O)$ are simply extracted from $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$.

Next, it was shown that the torques of the actuated joints $\boldsymbol{\tau}_A$ satisfy

$$\mathbf{S}^\top \boldsymbol{\tau}_A = \mathbf{W}^\top \boldsymbol{\tau}_O, \tag{13}$$

where $\mathbf{S}$ and $\mathbf{W}$ are *sensitivity* matrices of appropriate dimensions computed from the kinematics of the chain and the choice of the actuated joints, see [8] for details on the computations of these matrices.

As in Section II-A, introducing the path parameter s, differentiating (12) with respect to t, and rearranging the terms lead to

$$\boldsymbol{\tau}_O = \ddot{s}\mathbf{a}(s) + \dot{s}^2\mathbf{b}(s) + \mathbf{c}(s), \tag{14}$$

where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ are vectors of appropriate dimensions. Letting $\mathbf{a}^* \stackrel{\text{def}}{=} \mathbf{W}^\top\mathbf{a}$, $\mathbf{b}^* \stackrel{\text{def}}{=} \mathbf{W}^\top\mathbf{b}$, $\mathbf{c}^* \stackrel{\text{def}}{=} \mathbf{W}^\top\mathbf{c}$ and left-multiplying equation (14) by $\mathbf{W}^\top$ yield

$$\ddot{s}\mathbf{a}^*(s) + \dot{s}^2\mathbf{b}^*(s) + \mathbf{c}^*(s) = \mathbf{W}^\top\boldsymbol{\tau}_O = \mathbf{S}^\top\boldsymbol{\tau}_A. \tag{15}$$

If there is no actuation redundancy, then $\mathbf{S}^\top$ is invertible, and one can compute $\boldsymbol{\tau}_A$ by

$$\boldsymbol{\tau}_A = \mathbf{S}^{-\top}\left(\ddot{s}\mathbf{a}^*(s) + \dot{s}^2\mathbf{b}^*(s) + \mathbf{c}^*(s)\right).$$

Thus, the torque bounds on the actuated joints can be rewritten as: for all $i \in [1, n_A]$,

$$\tau_{Ai}^{\min} \leq \ddot{s}a_i^\circ(s) + \dot{s}^2 b_i^\circ(s) + c_i^\circ(s) \leq \tau_{Ai}^{\max},$$

where $a_i^\circ(s)$, $b_i^\circ(s)$, $c_i^\circ(s)$ are the i-th component of the vectors $\mathbf{S}^{-\top}\mathbf{a}^*(s)$, $\mathbf{S}^{-\top}\mathbf{b}^*(s)$, $\mathbf{S}^{-\top}\mathbf{c}^*(s)$. Thus, in this case, one has reverted to the classical form of (4).

In case of actuation redundancy, $\mathbf{S}^\top$ is non-square, hence non-invertible. Remark that the torque bounds of (11) can be alternatively written as $\boldsymbol{\tau}_A \in \mathscr{C}$, where $\mathscr{C} = [\tau_{A1}^{\min}, \tau_{A1}^{\max}] \times \cdots \times [\tau_{An_A}^{\min}, \tau_{An_A}^{\max}]$. Let next $\mathscr{C}^* = \mathbf{S}^\top\mathscr{C}$. Since $\mathbf{S}^\top$ is linear and $\mathscr{C}$ is a parallelotope, $\mathscr{C}^*$ is clearly a convex polytope. We have thus shown that the motion $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ satisfies the torque constraints if and only if

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

which has the form of (9) in Section II-C.

After obtaining the optimal parameterization $s(t)_{t\in[0,T']}$ from TOPP, one can compute at each s, the *generalized* torques

$$\boldsymbol{\tau}_G \stackrel{\text{def}}{=} \ddot{s}\mathbf{a}^*(s) + \dot{s}^2\mathbf{b}^*(s) + \mathbf{c}^*(s).$$

To obtain the torques at each actuated joint, one may solve for instance the quadratic program

$$\min_{\boldsymbol{\tau}_A \in \mathscr{C}} \|\boldsymbol{\tau}_A\|^2 \quad \text{such that} \quad \mathbf{S}^\top\boldsymbol{\tau}_A = \boldsymbol{\tau}_G.$$

In the above optimization, other objectives rather than the instantaneous square torque are possible. For instance, one may seek to exploit redundancy to minimize torque changes between two consecutive time steps. In any case, the key point here is that such optimizations are *guaranteed* to find at least one solution since we have taken care to ensure that $\boldsymbol{\tau}_G \in \mathbf{S}^\top\mathscr{C}$.

2) Humanoid robots in multi-contact tasks: Consider the equation of motion of a humanoid robot with n joints

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top\mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \sum_{k=1}^{K}\mathbf{J}_k^\top\mathbf{f}_k, \tag{16}$$

where $\mathbf{q}$ is a vector of dimension $(n+6)$ representing the joint angles and the 6 degrees of freedom of the free-flying base-link, $\boldsymbol{\tau}$ is the vector of torques – whose 6 coordinates corresponding to the free-flying base-link are set to 0, K is the number of contact points, $\mathbf{J}_k$ and $\mathbf{f}_k$ are the contact Jacobian and the contact force at contact point k.

For the robot to respect the torque constraints and be dynamically balanced, one must ensure that, for all $i \in [1, n]$

$$\tau_i^{\min} \leq \tau_i \leq \tau_i^{\max},$$

and for all $k \in [1, K]$,

$$\mathbf{f}_k \in \mathcal{F}_k,$$

where $\mathcal{F}_k$ is the friction pyramid at the contact point. For computational efficiency, we consider friction pyramids instead of friction cones.

As previously, introducing the path parameter s, differentiating (16) with respect to t, and rearranging the terms lead to

$$\ddot{s}\mathbf{a}(s) + \dot{s}^2\mathbf{b}(s) + \mathbf{c}(s) = \boldsymbol{\tau} + \sum_{k=1}^{K}\mathbf{J}_k^\top\mathbf{f}_k, \tag{17}$$

where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ are vectors of appropriate dimensions.

Note that, if the robot is free-flying or has one non-flat contact with the environment, then the system is under-actuated. If it has exactly one flat contact with the environment, then the system is normally-actuated and (17) can be inverted and put back into the classical form of (4). If the robot has more than one flat contact with the environment – e.g. when both feet lie flat on the ground or one foot lies flat on the ground and one hand lies flat on a table top, see Fig. 5 – then the system is redundantly-actuated [9]. In this case, denoting by $\mathscr{C}$ the *convex polytope* [11]

$$\underbrace{[\tau_1^{\min}, \tau_1^{\max}] \times \cdots \times [\tau_n^{\min}, \tau_n^{\max}]}_{n \text{ actuated coordinates}} \times \underbrace{\{0\} \times \cdots \times \{0\}}_{6 \text{ unactuated coordinates}} + \sum_{k=1}^{K}\mathbf{J}_k^\top\mathcal{F}_k,$$

enables one to transform equation (17) into the form of (9).

After obtaining the optimal parameterization $s(t)_{t\in[0,T']}$ from TOPP, one may recover the actuated torques and contact forces by solving a quadratic program as in Section II-D1. It is also possible to perform more sophisticated optimizations as in [9]; the key point here is that the existence of at least one solution is guaranteed by the TOPP procedure.

III. Experimental results

We now present an example of optimal time parameterization for a 30-dof humanoid robot (HRP2-14, Kawada Industries) in a multi-contact task consisting in stepping down a 30 cm-high podium[3], see Fig. 5. During the whole task, the left foot of the robot should lie flat

[3] A minor part of this example application – including, in particular, neither the physical simulation nor the actual robot experiment nor the detailed explanation of the algorithm – was briefly mentioned in [3].

on the podium, while its right hand should lie flat on the handrail. As discussed in Section II-D2, this double flat contact creates a situation of actuation redundancy.

We first generated the initial configuration and the final configuration (first and last snapshots in the third row of Fig. 5). The flat contact constraints for the left foot and the right hand were enforced by inverse kinematics. We then generated, using a simple algorithm presented in the Appendix, a smooth trajectory connecting the two configurations while keeping fixed the positions of the left foot and the right hand.

Next, based on the developments of Section II, we computed the time-optimal reparameterization of that trajectory respecting (i) joint velocity bounds, (ii) joint torque bounds, (iii) friction pyramid constraints for the contact forces at the left foot and right hand. For safety, we constrained the normal components of the contact forces to be $\geq 1\,\mathrm{N}$ instead of > 0.

The optimal time-parameterization of the trajectory had duration 0.65 s, see Fig. 5. We now discuss the time taken by our implementation to compute that optimal parameterization. To calculate the MVC and to perform the numerical integration, we considered the grid size $N = 100$. The average number of edges per constraint polygon was 11.8. The time-parameterization step (excluding dynamics computations and polytope projection) took 0.005 s on our computer (Intel Core i5 3.2 GHz, 3.8 GB memory, GNU/Linux). On a comparable computer installation, also using C++, and for a humanoid multi-contact problem where $N = 100$ and the average number of edges per constraint polygon was 9.1, the convex optimization approach took 2.46 s (also excluding dynamics computation and polytope projection, cf. page 1247 in [11]), which is about 500 times slower than our implementation.

This significant improvement in computation time may come from two main factors. First, the numerical integration approach is inherently more efficient than the convex optimization approach since it exploits the "bang-bang" structure of the time optimization problem [3]. Indeed, at each point $(s, \dot{s})$ of the phase plane, the optimal acceleration can be *analytically computed* [by equation (6)], instead of being *iteratively searched* as part of a large optimization problem. Second, addressing directly the polygon form of the constraints as we did in Section II-C enables a further computational improvement with respect to handling half-plane inequalities as in the original TOPP algorithm.

To validate the results of our algorithm, we executed the reparameterized trajectory in OpenHRP, a highly realistic physical simulation environment. Snapshots of the movement (third row of Fig. 5) show that the robot maintained flat contacts at the left foot and the right hand throughout the whole movement.

The contact forces under the left foot and the right hand are also shown in Fig. 5. One can observe that the contact forces under the left foot saturated the positivity constraint between $s = 0$ and $s = 0.05$. Thus, the theory predicts that a higher execution speed will violate this constraint. That was indeed what we observed : when scaling up the velocity profile given by our algorithm by respectively 1.2 times and 2 times, we observed in the OpenHRP simulation that the left foot lifted up from the podium near the beginning of the trajectory, slightly at $1.2\times$ and significantly at $2\times$ (second row of Fig. 5, see also the video at http://youtu.be/4DVE1cUCh4M). Note that the motion is quasi-statically executable : scaling *down* the velocity profile by arbitrary coefficients will not violate the constraints. Planning motions that are dynamically executable but not quasi-statically executable using TOPP was suggested in [19]. Extending this approach to humanoid robots in multi-contact based on the results of the present paper is the topic of ongoing research.

Finally, we executed the optimally-reparameterized trajectory on the actual robot. For safety reasons, we restricted the execution speed to 90% of the maximum speed calculated by our algorithm. The actual robot movement indeed respected the constraints (i), (ii) and (iii) mentioned earlier, see fourth row of Fig. 5 and also the video [4].

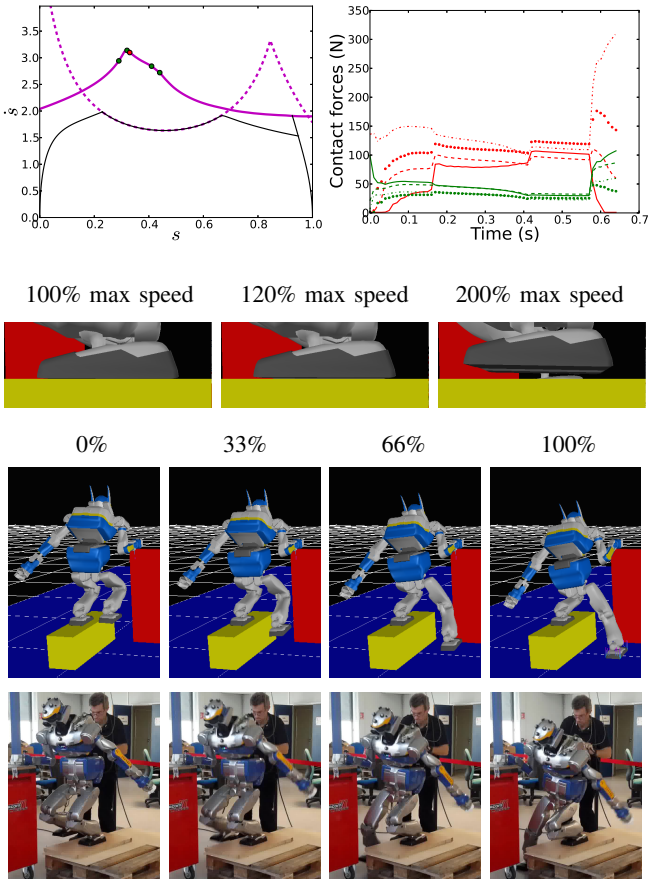

Fig. 5. Time-optimal path parameterization for a humanoid robot stepping down a 30 cm-high podium. **Top row**, *Left* : Maximum Velocity Curves in magenta (plain line : MVC caused by the torque bounds and friction constraints; dashed line : MVC caused by the velocity bounds) and optimal velocity profile (black) calculated by our algorithm. Circles on the MVC indicate possible switch points (red : tangent switch point, green : singular switch points). *Right* : Normal components of the contact forces under the left foot (red) and the right hand (green) as calculated by our algorithm. The four points where the contact forces were computed were the front-left corner (plain lines), front-right corner (dashed lines), back-left corner (dotted lines) and back-right corner (dashed-dotted lines). **Second row :** Maximum elevation of the left foot near the beginning of the trajectory at different execution speeds (simulation in OpenHRP). Note that, at 120% of the maximum speed, the foot slightly lifted up from the podium, and at 200% of the maximum speed, the foot largely lifted up from the podium. **Third row :** Snapshots of the motion executed in OpenHRP at 100% of maximum speed. **Fourth row :** Snapshots of the motion executed on the actual robot at 90% of maximum speed.

IV. Conclusion

We have presented an extension of the classical Time-Optimal Path Parameterization algorithm to the practically-important class of redundantly-actuated robots, which comprises in particular closed-chain (or parallel) manipulators and humanoid robots in multi-contact tasks. We implemented this extension within the framework

[4] In the video, the right foot of the robot makes a rather strong contact with the ground at the end of the movement. This was caused by the intrinsic flexibility of the robot, which makes movements at very high speed difficult to control, and has nothing to do with the contact conditions – on the left foot and the right hand – that are considered in the present work.

of the TOPP library and tested it in a multi-contact task where a 30-dof humanoid robot steps down from a 30 cm-high podium. We showed that our implementation enabled a significant speed-up in computation time as compared to existing implementations. This improvement is particularly crucial for the *global* trajectory optimization problem (i.e. where the path is not fixed), which requires calling the TOPP routine thousands of times [16].

Time optimality implies that, at any time instant, at least one of the constraints is saturated. Thus, in presence of uncertainties in the models of the robot or of the environment, it is advisable to consider constraints that are tighter than in reality : for example, we constrained the normal components of the contact forces to be $\geq 1\,\mathrm{N}$ instead of > 0 (cf. Section III).

We are currently investigating how to leverage the performance gain to plan *through* contact changes [17], [18]. In particular, integrating the results obtained here with the technique of Admissible Velocity Propagation [19] may enable one to efficiently plan truly dynamic multi-contact motions for humanoid robots. Another research direction consists in studying how higher-order terms (such as jerk) can be taken into account [20], in order to gain control authority on the flexibility of the robots.

Acknowledgments

This paper has benefited from discussions with S. Caron, N. Mansard and L. Righetti. The authors would like to thank M. Naveau for help with the humanoid experiment. The first author is grateful to J.-P. Laumond for his kind invitation to visit LAAS-CNRS in the summer of 2014, which made the present work possible.

APPENDIX

A. Interpolating smooth trajectories respecting closed-chain constraints

Consider a link L contained in a closed chain and a partition of the joints angles $\mathbf{q}$ of the chain into $\mathbf{q}_X$ and $\mathbf{q}_Y$. The linear and angular velocities of link L is given by $\mathbf{J}_X\dot{\mathbf{q}}_X$ where $\mathbf{J}_X$ is the Jacobian of the position and orientation of link L with respect to $\mathbf{q}_X$. Similarly, these velocities are also given by $\mathbf{J}_Y\dot{\mathbf{q}}_Y$. Thus the closed chain constraint is given by $\mathbf{J}_X\dot{\mathbf{q}}_X = \mathbf{J}_Y\dot{\mathbf{q}}_Y$.

Assume that link L and the partition were chosen such that $\mathbf{J}_X$ is square and invertible. For instance, for the closed chain of Fig. 4, this can be achieved by choosing L to be the link comprised between q_3 and q_4, $\mathbf{q}_X = (q_1, q_2, q_3)$, $\mathbf{q}_Y = (q_4, q_5, q_6)$. In this case, $\mathbf{q}_X$ is called the *dependent*, and $\mathbf{q}_Y$ the *independent*, joint angles. Next, a given small displacement $\delta\mathbf{q}_Y$ of the independent angles can be "compensated" by a displacement $\delta\mathbf{q}_X$ of the dependent angles computed by

$$\delta\mathbf{q}_X = (\mathbf{J}_X^{-1}\mathbf{J}_Y)\delta\mathbf{q}_Y. \tag{18}$$

Based on the above development, we propose the following scheme to interpolate smoothly between two configurations $\mathbf{q}^0$ and $\mathbf{q}^1$ of the closed chain :

- interpolate smoothly and freely the independent angles between $\mathbf{q}_Y^0$ and $\mathbf{q}_Y^1$ (using e.g. third-degree polynomials);
- compute $\mathbf{q}_X$ by integrating $\delta\mathbf{q}_X$ given by equation (18).

This scheme yields a valid interpolation if (see page 105 of [7])

- $\mathbf{q}_X^0$ and $\mathbf{q}_X^1$ are one the same branch of the inverse kinematics;
- during the integration process, $\mathbf{J}_X$ never traverses a singularity.

Note that, if the independent trajectory $\mathbf{q}_Y$ is C^k-continuous, then the compensating trajectory $\mathbf{q}_X$ is also C^k-continuous, and bounds on the derivatives of $\mathbf{q}_X$ may be obtained from equation (18). This contrasts with the interpolation method of [11], which cannot offer such guarantees.

REFERENCES

[1] J. Bobrow, S. Dubowsky, and J. 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] K. Shin and N. McKay, "Selection of near-minimum time geometric paths for robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 501–511, 1986.

[3] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 6, pp. 1533–1540, 2014. [Online]. Available: http://arxiv.org/abs/1312.6533

[4] K. M. Lynch, N. Shiroma, H. Arai, and K. Tanie, "Collision-free trajectory planning for a 3-dof robot with a passive joint," *The International Journal of Robotics Research*, vol. 19, no. 12, pp. 1171–1184, 2000.

[5] F. Bullo and K. M. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 402–412, 2001.

[6] H. Cheng, Y.-K. Yiu, and Z. Li, "Dynamics and control of redundantly actuated parallel manipulators," *Mechatronics, IEEE/ASME Transactions on*, vol. 8, no. 4, pp. 483–491, 2003.

[7] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1990.

[8] Y. Nakamura and K. Yamane, "Dynamics computation of structure-varying kinematic chains and its application to human figures," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 2, pp. 124–134, 2000.

[9] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[10] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

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

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

[13] Z. Shiller and H. Lu, "Computation of path constrained time optimal motions with dynamic singularities," *Journal of dynamic systems, measurement, and control*, vol. 114, p. 34, 1992.

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

[15] M. Walker and D. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, p. 205, 1982.

[16] J. Bobrow, "Optimal robot path planning using the minimum-time criterion," *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 443–450, 1988.

[17] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[18] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[19] Q.-C. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via velocity interval propagation," in *Robotics: Science and System*, 2013.

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