Fast Dynamic Optimization of Robot Paths under Actuator Limits and Frictional Contact

Kris Hauser[1]

***Abstract*— This paper presents an algorithm for minimizing the execution time of a geometric robot path while satisfying dynamic force and torque constraints. The formulation is numerically stable, using a convex optimization that is guaranteed to converge to a unique optimum, and it is also scalable due to the use of a fast feasible set precomputation step that greatly reduces dimensionality of the optimization problem. The algorithm handles frictional contact constraints with arbitrary numbers of contact points as well as torque, acceleration, and velocity limits. Results are demonstrated in simulation on locomotion problems on the Hubo-II+ and ATLAS humanoid robots, demonstrating that the algorithm can optimize trajectories for robots with dozens of degrees of freedom and dozens of contact points in a few seconds.**

I. Introduction

Trajectory optimization is a powerful tool for generating robot motions that are fast, require low energy consumption, and obey kinematic and dynamic motion constraints. In general, trajectory optimization requires the solution to high-dimensional, nonlinear optimization problems, which is computationally expensive and sensitive to initial guesses. As a result, it has been typically been applied more successfully in offline computation rather than online use. The goal of *online optimization* is to answer dynamic queries in novel environments, and it demands that the time saved in execution is greater than the time elapsed while refining a solution. Otherwise, optimization is counterproductive. One promising approximate approach is a two-stage strategy that first generates a geometric path ignoring dynamics, and then performs dynamic time-scaling to minimize execution time while maintaining dynamic constraints. Dynamic time-scaling can be solved exactly and relatively efficiently.

To our knowledge, this is the first paper to consider the time-scaling problem under frictional contact constraints. The key contribution is a preprocessing step that computes the feasible set of time derivatives at each point. In a certain space of first and second time derivatives, these sets are convex polygons for velocity, acceleration, and torque bounds, and perhaps surprisingly, for arbitrary arrangements of frictional point contacts as well. These reduce the number of optimization variables at each point in time from $O(n + m)$, where n is the number of degrees of freedom and m is the number of contacts, to two. To perform time optimization, we extend a recent scalable and numerically-stable method for convex optimization time-scaling with box-bounds on acceleration and velocity [1]. The time domain is divided into a grid, and a simple parameter transformation yields a linearly constrained convex optimization that can be solved robustly using sequential linear programming techniques.

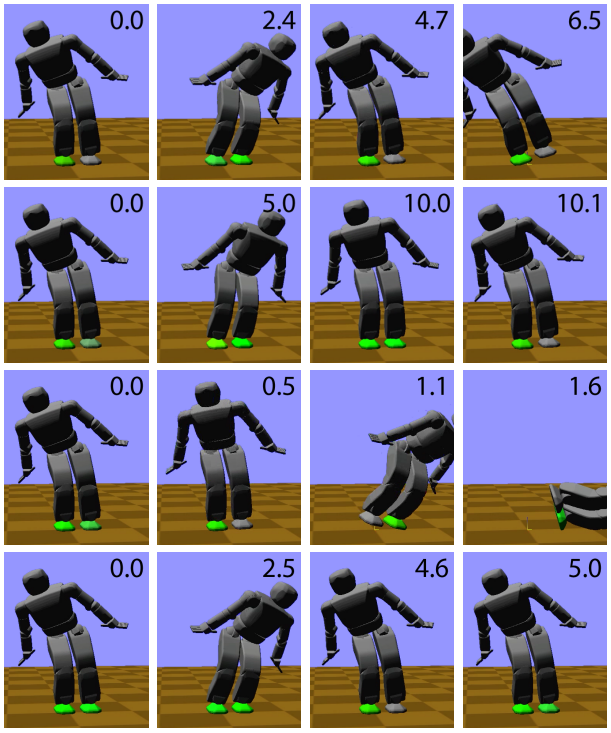

Fig. 1. A side-to-side swaying motion on a humanoid robot with an abrupt start and stop. Row 1: executed with uniform rate at 5 s duration, the robot falls over. Row 2: executed with uniform rate at 10 s duration, the robot barely stays upright, wobbling onto its right foot at the end of the motion (last frame). Row 3: with an optimized time-scaling under only acceleration constraints, the robot decelerates far too quickly and tips over halfway through the motion. Row 4: our new method. Enforcing contact/torque constraints lets the robot stay upright.

The approach supports a wide variety of articulated systems under contact, and can be applied to multi-limbed locomotion, full-body contact, and non-prehensile manipulations like lifting and pushing. Experiments demonstrate that it solves time-scaling problems in seconds for robots up to 63DOF and nearly 60 contact points; can naturally handle systems for which no quasi-static solutions exist; and it helps humanoid robots execute locomotion tasks efficiently while maintaining actuator constraints and balance (Fig. 1).

*This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) award # N65236-12-1-1005 for the DARPA Robotics Challenge. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of DARPA.

[1]School of Informatics and Computing, Indiana University at Bloomington, Bloomington, Indiana 47408, hauserk@indiana.edu

II. Background

The classical numerical approach to trajectory optimization has been used extensively for generating robot manipulator trajectories, and has also been applied to systems with contact, such as legged locomotion [2]–[6]. But global nonlinear optimization in high-dimensional parameter spaces is computationally challenging due to problem size and the prevalence of many local minima, so these techniques appear best suited for generating gait cycles [2], [5], [6] or offline primitives to be reused later [3], [4].

Time-scaling is a simpler optimization problem that considers optimizing the execution speed of a given path under dynamic constraints. It is often used in two-stage trajectory generation [7], [8] in which the first stage computes a path and then the second stage optimizes its speed. This approach may fail to achieve the same quality as full trajectory optimization because there is a risk of computing a path in stage 1 that will not yield a fast time parameterization in stage 2. In practice it is still useful because computation time is improved by orders of magnitude while delivering satisfactory results. Also, the optimization of paths and time-scalings can be iterated for better results [7], [9]. The classical Bobrow method integrates the time scaling variable along dynamic limits [7] both backward and forward in time to search for a continuous time parameterization. However, as identified in [10], [11], this method was found to suffer from numerical instability issues at dynamic singularities.

Verscheure et. al. formulated time-scaling in a more numerically robust manner as a convex optimization problem, and their approach solves a second-order cone program (SOCP) [12]. Recent work presented a simpler, more scalable approach that transforms time-scaling into a convex optimization with linear constraints [1]. The approach handles velocity and acceleration bounds, and addresses scalability to very high-DOF robots using a constraint pruning method. Our optimizer builds heavily on this work, extending it to handle torque and contact force constraints as well.

A dynamic contact constraint commonly used in bipedal walking is the zero-moment point (ZMP) condition, which states that the acceleration of a robot's center of mass is directly related to its center of pressure on the ground (the ZMP). It is used by humanoid robots to optimize a center of mass trajectory so that the ZMP always lies in the support polygon [13]. Our new method significantly generalizes this approach and is applicable to uneven terrains, walking with hand support, and simultaneous navigation and manipulation.

III. Problem Statement

The time-scaling problem is that of optimizing a time-parameterization $s(t) : [0, T] \to [0, 1]$ of a given geometric path $p(s) : [0, 1] \to \mathbb{R}^n$. The trajectory duration T is unknown, and in this paper the objective is to minimize T. The time-parameterized trajectory is denoted $y(t) = p(s(t))$ and is required to satisfy various dynamic constraints of the form:

$$f(y(t), \dot{y}(t), \ddot{y}(t)) \leq 0 \tag{1}$$

For example, velocity and acceleration bounds are written:

$$v_L \leq \dot{y}(t) \leq v_U \text{ for all } t \in [0, T] \tag{2}$$

$$a_L \leq \ddot{y}(t) \leq a_U \text{ for all } t \in [0, T] \tag{3}$$

where all inequalities are taken element-wise. Constraints of the form (2–3) were handled in the previous paper [1]. In this paper we introduce two new types of constraints.

a) Torque Constraints.: For systems without contact, torques τ are calculated in the standard Lagrangian form

$$B(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau(q, \dot{q}, \ddot{q}) \tag{4}$$

where B is the mass matrix, C is the Coriolis force vector, and G is the generalized gravity vector. Hence, torque bounds are written in the form

$$\tau_L \leq \tau(y(t), \dot{y}(t), \ddot{y}(t)) \leq \tau_U. \tag{5}$$

b) Contact Constraints.: Systems with contact are modeled using arrangements of frictional point contacts. For simplicity of notation, we will assume for the moment that contact must be maintained throughout the duration of the motion, although later we will present extensions to making and breaking contact. A robot-world contact matches a point on the robot $c(q) \in \mathbb{R}^3$ with a point in space $d \in \mathbb{R}^3$, and a robot-robot contact matches two points on the robot $c(q), d(q) \in \mathbb{R}^3$. For all $s \in [0, 1]$, the input geometric path must satisfy $c_i(p(s)) = d_i(p(s))$ for all contacts $i = 1, \ldots, m$.

Rigid objects are modeled by augmenting the robot with an additional 6 generalized coordinates for translation and rotation, which are included in q. The velocity and acceleration bounds on these coordinates are infinite, while the torque bounds are zero. A free-floating base, e.g., the torso of a humanoid robot, is modeled similarly. Note that such systems are underactuated and hence there will be no solution of (5), in general. Instead, contact forces are explicitly modeled.

At each contact (c_i, d_i) a force f_i may be applied to the robot at each instant in time. By the principle of virtual work, the Lagrange equation becomes

$$B(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau + \sum_{i=1}^{m} J_i(q)^T f_i \tag{6}$$

where J_i is the Jacobian matrix of the constraint $c_i(q) - d_i(q) = 0$. In addition, the individual forces are required to respect friction cone constraints:

$$f_i \in FC_i(q) \text{ for } i = 1, \ldots, m \tag{7}$$

where each friction cone has an apex at the origin and axis equal to the contact normal (note that, particularly for manipulation tasks, the normal may change with q).

Since contact forces are a priori unknown, the instantaneous force/torque constraint becomes

$$\begin{gathered} \text{There exist } (\tau, f_1, \ldots, f_m) \text{ s.t.} \\ B(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau + \sum_{i=1}^{m} J_i(q)^T f_i \\ \tau_L \leq \tau \leq \tau_U \\ f_i \in FC_i(q) \text{ for } i = 1, \ldots, m. \end{gathered} \tag{8}$$

Checking this condition at a particular state $(q, \dot{q}, \ddot{q})$ is equivalent to testing the non-emptiness of a convex subset of $\mathbb{R}^{n+3m}$. One approach is to solve an optimization problem in an arbitrary direction, e.g., a semidefinite program. A simpler optimization problem is derived by approximating the friction cones as polyhedra, because the feasible set is a convex polytope. We adopt this approximation, and handle this constraint using polytope projection techniques to precompute the set of speeds and accelerations along the path that yield feasible frictional forces. For time-scaling, this reduces the number of variables per time-domain grid point from $2 + n + 3m$ (speed, acceleration, torque, and contact forces) down to two (speed and acceleration).

IV. Precomputing Feasible Sets of First and Second Time-Scaling Derivatives

Before presenting the time scaling optimization, we will first describe a method for precomputing the feasible sets of first and second derivatives of the time-scaling function at a point along the path. We will show that the set of feasible values of $(\dot{s}^2, \ddot{s})$ that respect (2,3,5,8) are convex sets, and if friction cones are expressed as polyhedra, then these sets are also convex polygons.

Given any value s, the method outputs vectors $a(s)$, $b(s)$, and $c(s)$ such that the feasible set is expressed in the form:

$$a(s)\dot{s}^2 + b(s)\ddot{s} \leq c(s). \tag{9}$$

A. Feasible Sets for Basic Dynamic Constraints

First we will discuss how to compute $a(s)$, $b(s)$, and $c(s)$ for dynamic constraints (2,3,5). Contact constraints are more complex and will be considered in the next section.

First, as usual in time scaling, the trajectory derivatives are written in terms of the derivatives of $p(s)$ and $s(t)$ following the chain rule:

$$\dot{y}(t) = p'(s(t))\dot{s}(t) \tag{10}$$

$$\ddot{y}(t) = p''(s(t))\dot{s}(t)^2 + p'(s(t))\ddot{s}(t). \tag{11}$$

We also make use of the following relation between the time scaling derivatives and torque:

$$\tau(s, \dot{s}, \ddot{s}) = B \cdot (p''(s)\dot{s}^2 + p'(s)\ddot{s}) + C\dot{s}^2 + G \tag{12}$$

where $B(p(s))$, $C(p(s), p'(s))$, and $G(p(s))$ are independent of $\dot{s}$ and $\ddot{s}$. This equation uses the fact that the Coriolis force term satisfies $C(q, c\dot{q}) = c^2 C(q, \dot{q})$ for any scalar c.

Substituting (10–12) into (2,3,5), we obtain:

$$v_L \leq p'(s)\dot{s} \leq v_U \tag{13}$$

$$a_L \leq p''(s)\dot{s}^2 + p'(s)\ddot{s} \leq a_U \tag{14}$$

$$\tau_L \leq B \cdot (p''\dot{s}^2 + p'(s)\ddot{s}) + C\dot{s}^2 + G \leq \tau_U \tag{15}$$

where $B(p(s))$, $C(p(s), p'(s))$, and $G(p(s))$ are independent of $\dot{s}$ and $\ddot{s}$. By inspection, it is clear that constraints (14–15) are linear in the squared-rate $\dot{s}^2$ and rate derivative $\ddot{s}$. It is also a simple matter to convert (13) to squared-rate constraints:

$$0 \leq \dot{s}^2 \leq \left(\min_{i=1}^{n} \max \frac{v_{L,i}}{p'(s)_i}, \frac{v_{U,i}}{p'(s)_i} \right)^2 \tag{16}$$

where the subscript i denotes an element of each vector.

Given these constraints, the method of [1] can quickly determine the planes bounding the feasible polygon using repeated cutting. Employing this technique is extremely important for problems with high dimension n; by eliminating irrelevant constraints, the time-scaling optimization is sped up by orders of magnitude. In fact, experiments indicate that the number of relevant constraints stays nearly constant as n grows large. See [1] for more details on the cutting procedure.

B. Feasible Sets for Contact Constraints

Contact forces pose a greater challenge because force/torque constraints are no longer expressed in the form (1), and instead at each point in time require testing the non-emptiness of a high-dimensional set expressed in the form (8). We wish to compute the set of $\dot{s}^2$ and $\ddot{s}$ for which *some* set of torques and contact forces satisfy the constraints (8). To do so, we compute the projection of a $2 + n + 3m$ dimensional convex polytope onto a plane.

Performing the same transformation as (12) as on (6), we find the following instantaneous relation between time scaling derivatives, torques, and forces:

$$B \cdot (p''\dot{s}^2 + p'\ddot{s}) + C\dot{s}^2 + G = \tau + \sum_{i=1}^{m} J_i^T f_i. \tag{17}$$

Here, p', p'', B, C, G, and each J_i depend only on s and are independent of $\dot{s}$ and $\ddot{s}$, so this equation is linear in the $(\dot{s}^2, \ddot{s}, \tau, f_1, \ldots, f_m)$ space.

Including force and torque constraints, we are interested in the feasible set F of $(\dot{s}^2, \ddot{s}, \tau, f_1, \ldots, f_m)$ that satisfy the simultaneous equations:

$$\begin{gathered} B \cdot (p''\dot{s}^2 + p'\ddot{s}) + C\dot{s}^2 + G = \tau + \sum_{i=1}^{m} J_i^T f_i \\ \tau_L \leq \tau \leq \tau_U \\ f_i \in FC_i(p(s)) \text{ for } i = 1, \ldots, m. \end{gathered} \tag{18}$$

as well as velocity and acceleration constraints (13) and (14). It is slightly faster to eliminate τ as an optimization variable by moving the force terms in (17) to the left hand side and then bounding the l.h.s. by the torque limits τ_L and τ_U. This reduces the dimension of F from $2 + n + 3m$ to $2 + 3m$.

Testing whether a particular set of rates $(\dot{s}, \ddot{s})$ yields feasible forces and torques is equivalent to checking whether a $3m$-dimensional hyperplane intersects F. Another interpretation is that the value of $(\dot{s}, \ddot{s})$ must lie in the "shadow" of F when projected onto the $(\dot{s}^2, \ddot{s})$ plane. Next we show how to compute this 2-D projection, which we denote P.

C. Polytope Projection via Recursive Expansion

Since all constraints are convex, F is also a convex set, and P is a convex planar shape. With friction cones approximated as polyhedra, F is a convex polytope and P is a convex polygon (Fig. 2). We compute this polygon by recursive expansion technique presented in [14] which bears a close resemblance to the Equality Set Projection algorithm

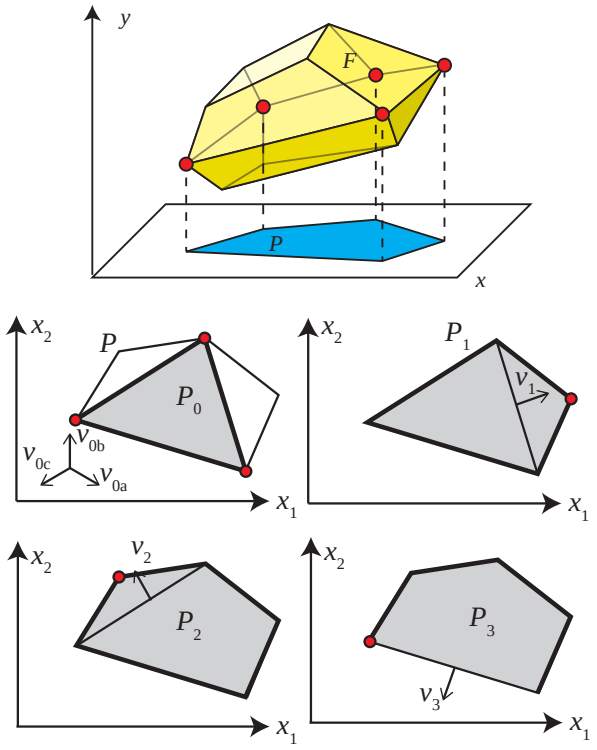

Fig. 2. Illustrating the polytope projection step. The feasible polytope F is projected along the torque and force dimensions (denoted y) onto the plane of first and second derivatives the time scaling parameter (denoted x). To do so, the x parameter is maximized in three initial directions v_{0a}, v_{0b}, v_{0c} subject to the halfplane constraints of F to obtain an initial approximation P_0. Each edge is then recursively expanded in the direction of the outward normal to obtain the next approximation. When an edge fails to expand, recursion stops.

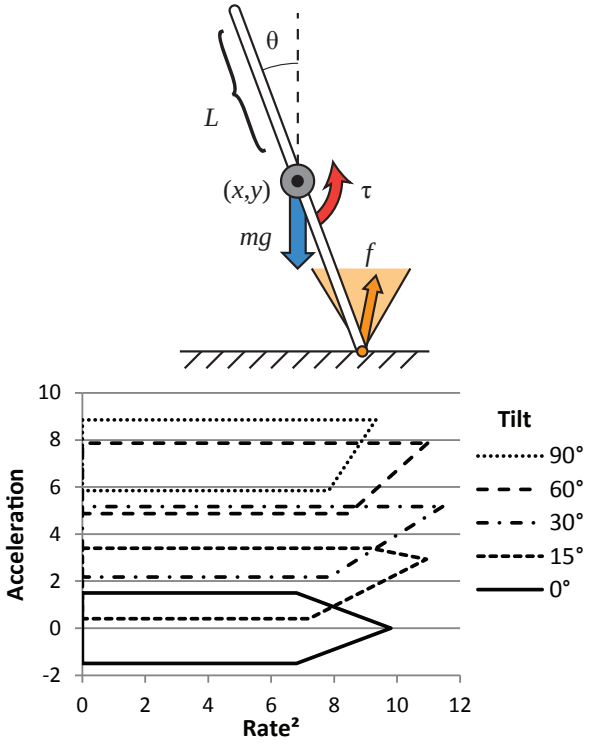

Fig. 3. Simple rod example with a point contact.

of [15]. Let x denote the $(\dot{s}^2, \ddot{s})$ component of this space and let y denote the rest. Specifically, we wish to compute $P = \{x \quad | \quad (x, y) \in F\}$.

The algorithm incrementally grows tighter approximations to P by determining an extremum of F in a direction on the $(\dot{s}^2, \ddot{s})$ plane at each iteration. Each extremizing step solves a linear program (LP) of the form

$$\max_{x,y} v^T x \text{ s.t. } (x, y) \in F \tag{19}$$

with v some direction in the plane.

The algorithm begins at an initial polygon $P_0 \subseteq P$ by extremizing P in at least three directions whose positive span contains the origin in its interior. For example, our implementation uses the vertices of an equilateral triangle: $(1, 0)$, $(\cos(120°), \sin(120°))$, and $(\cos(240°), \sin(240°))$. If any of these LPs is infeasible, then P is empty and the dynamic constraints are inconsistent.

Next, it begins a recursion to grow finer approximations $P_1, P_2, \ldots$, at each step i extremizing along a direction v_i which is determined by choosing an outward-pointing normal to an unexpanded edge of P_i. The value of x that achieves the maximum is either contained in P_i or not. In the former case, that edge is marked as expanded in P_{i+1}. Otherwise, x is added as a new vertex to obtain P_{i+1}. This continues until no unexpanded edges remain, or a maximum number of iterations is reached.

The steps of this procedure are illustrated in Fig. 2. Once the edges of P are determined, each supporting plane is converted to a halfplane equation $a\dot{s}^2 + b\ddot{s} \leq c$.

D. Comments on Complexity

F lies in a $2 + 3m$-dimensional space and is bounded by $3n + km$ halfplane constraints, where n is the robot dimensionality, m is the number of contacts, and k is the number of planes in each approximated friction cone. So, the feasible set has at most $\binom{3n+km}{2+3m}$ vertices. Experiments suggest the number of boundaries z of the projected feasible sets can be considered essentially bounded by a constant. Surprisingly, $z \approx 7$ on average in our humanoid robot examples despite having dozens of degrees of freedom and contact points. The recursive expansion algorithm is output-sensitive, only making $O(z)$ calls to solve a relatively small linear program. So, in practice it runs in polynomial time.

E. Illustration: Rod with Point Contact

Fig. 3 illustrates a planar rod of mass m, inertia H, and length $2L$ making contact with the origin at one end. Its coordinates are given by (x, y, θ), where (x, y) is the center of mass and θ is the leftward lean angle. It can produce a torque τ_θ about the center of mass, but cannot exert translational forces on the rod directly. The contact force f is subjected to Coulomb friction with coefficient μ. The path is fully determined by θ via the equations $x = -L \sin\theta$, $y = L \cos\theta$.

The mass matrix B is a diagonal matrix with entries $[m, m, H]$, the Coriolis term C is zero, and the gravity vector G is $[0, mg, 0]^T$. Assuming a path with constant velocity $\theta'(s) = 1$ and $\theta''(s) = 0$, we have the torque τ given by:

$$\begin{bmatrix} -mx \\ -my \\ 0 \end{bmatrix} \dot{s}^2 + \begin{bmatrix} -my \\ mx \\ H \end{bmatrix} \ddot{s} + G - \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -y & x \end{bmatrix} f. \quad (20)$$

Since $\tau = [0, 0, \tau_\theta]^T$, the first two rows in this matrix must be equal to zero. In other words,

$$f = -m[x, y]^T \dot{s}^2 + m[-y, x]^T \ddot{s} + [0, mg], \quad (21)$$

and the third row gives:

$$\tau_\theta = H\ddot{s} + [-y, x]^T f. \quad (22)$$

Replacing the formula for f into the above expression, we have

$$\tau_\theta = (H + mL^2)\ddot{s} + mxg \quad (23)$$

after a bit of algebra and observing that $x^2 + y^2 = L^2$. This equation signifies that with zero torque, $\ddot{s} = -mxg/(H + mL^2)$. Adding torque increases the range of attainable accelerations.

Adding in the torque constraints $|\tau_\theta| \leq \tau_{max}$ and the friction constraints $[1, \mu]f \geq 0$ and $[-1, \mu]f \geq 0$, we observe four linear constraints in the $\dot{s}^2$ and $\ddot{s}$ space. $\dot{s}^2$ must also be nonnegative.

Fig. 3 plots the feasible set for different values of θ for $m = L = 1$, $H = m/3$, $\mu = 0.5$, and the magnitude of τ_θ limited to 2. Observe that forward and backward accelerations are only attainable in a small range of inclinations about the vertical; otherwise the torque exerted by gravity overwhelms the robot's ability to balance. Also observe that at faster rotation speeds (approximately $\dot{\theta} \geq \sqrt{7}$), the feasible set is bounded by diagonals, and at even faster speeds, there are no feasible solutions. A vertex at the upper and lower bounds signifies that at faster velocities, the contact would slide because of insufficient friction. A rightward facing point between the upper and lower bounds indicates that at faster speeds, additional *downward* forces are needed to maintain contact. Without such forces, the rod would break contact and fly away.

V. Time Scaling through Convex Optimization

The time scaling algorithm minimizes T while respecting precomputed constraints of the form (9). The time-parameterization $s(t)$ is formulated as a piecewise quadratic curve, split at N grid points on the s domain. N is the main parameter of the algorithm and larger values of N will yield progressively lower execution times but higher computational costs. Using a transformation into *squared rate* parameter space, the dynamic constraints are then cast into linear constraints on adjacent grid points. The optimization problem is convex, and the optimizer converges to a unique global minimum.

A. Piecewise Quadratic Time Parameterization

For the output y to be twice differentiable, $\dot{s}(t)$ is required to be continuous. We also require that the trajectory start and stop with zero velocity: $\dot{s}(0) = \dot{s}(T) = 0$. The problem now becomes one of optimizing a representation of $s(t)$ subject to these conditions and (2,3,5).

Grid the path domain $[0, 1]$ into N intervals $[s_k, s_{k+1}]$, $k = 0, \ldots, N-1$. Our implementation uses uniform grids by default but applies equally well for nonuniform grids as well. $\dot{s}(t)$ is assumed to be linear on each interval, and hence $s(t)$ is piecewise quadratic. The time scaling is then parameterized by *rate parameters* $\dot{s}_0, \ldots, \dot{s}_N$ denoting the velocities $\dot{s}(t)$ at the grid points. Note that the values of t at the grid points are not yet determined, but we shall show that they are fully determined by the rate parameters, as is the total duration T.

Consider the unknown time values t_k and t_{k+1} at the endpoints of interval k. Let $\Delta s_k = s_{k+1} - s_k$ be the interval width in the path domain. We require $s(t_k) = s_k$, $s(t_{k+1}) = s_{k+1}$, $\dot{s}(t_k) = \dot{s}_k$ and $\dot{s}(t_{k+1}) = \dot{s}_{k+1}$. Simple algebra verifies that the choices $\Delta t_k = t_{k+1} - t_k = \frac{2\Delta s_k}{\dot{s}_{k+1} + \dot{s}_k}$ and

$$s(t) = \frac{\dot{s}_{k+1}^2 - \dot{s}_k^2}{4\Delta s_k}(t - t_k)^2 + \dot{s}_k(t - t_k) + s_k \quad (24)$$

satisfy all of the boundary constraints. Observe that $\ddot{s}(t) = (\dot{s}_{k+1}^2 - \dot{s}_k^2)/(2\Delta s_k)$ is a constant over this interval. Furthermore, the total duration is easily computed via

$$T = \sum_{k=1}^{N} \Delta t_k = \sum_{k=1}^{N} \frac{2\Delta s_k}{\dot{s}_{k+1} + \dot{s}_k}. \quad (25)$$

B. Transformation to Squared-Rate Parameter Space

The next step of the formulation transforms the optimization problem to *squared-rate* parameters $\theta_0 = \dot{s}_0^2, \ldots, \theta_N = \dot{s}_N^2$.

Collecting the optimization variables into $\theta = (\theta_0, \ldots, \theta_N)$ the final linearly-constrained optimization problem becomes:

$$\begin{gathered} \min_\theta T(\theta) \text{ such that} \\ 0 \leq \theta_k \leq \theta_{U,k} \text{ for } k = 0, \ldots, N \\ a(s_k)\theta_k + b(s_k)\frac{\theta_{k+1} - \theta_k}{2\Delta s_k} \leq c(s_k) \text{ for } k = 0, \ldots, N-1 \\ a(s_k)\theta_k + b(s_k)\frac{\theta_k - \theta_{k-1}}{2\Delta s_{k-1}} \leq c(s_k) \text{ for } k = 1, \ldots, N \end{gathered} \quad (26)$$

where the upper bounds $\theta_{U,k}$ on the parameters are given by the left hand side of (16). Boundary values are enforced by setting $\theta_{U,0} = \theta_{U,N} = 0$. Note that the acceleration constraints must be duplicated to account for differing second derivatives over interval k and $k-1$.

Here the objective function

$$T(\theta) = \sum_{k=1}^{N} \frac{2\Delta s_k}{\sqrt{\theta_{k+1}} + \sqrt{\theta_k}} \quad (27)$$

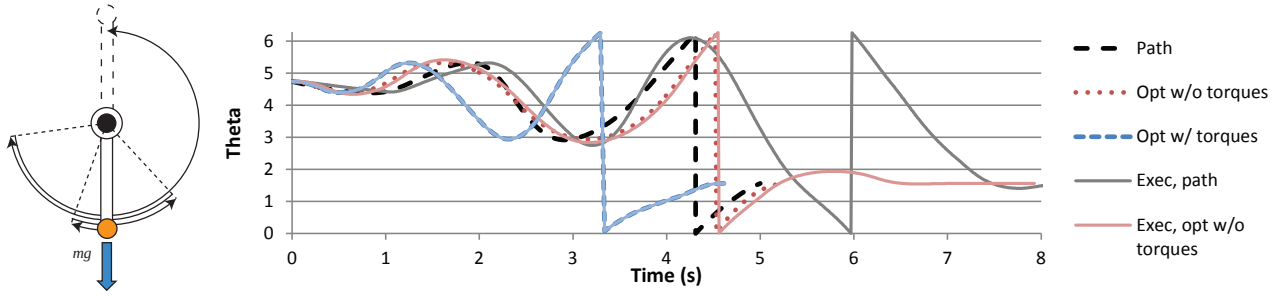

Fig. 4. A pendulum swingup problem. Executions simulated with PD control with gravity compensation. The original spline path uniformly mapped to a 5 s duration (Path) causes the robot to overshoot several times (Exec. path). At $\leq$4 s or $\geq$9 s duration the robot does not even reach the upright position (not shown). Optimizing the time-scaling without considering torque limits (Opt. w/o torque) causes the robot to overshoot the target (Exec, opt. w/o torque). Considering torque limits, the optimized time scaling (Opt. w/ torque) is almost indistinguishable from its execution (Exec, opt. w/ torque).

is convex, as proven in [1]. Hence, (26) is convex, which implies that it has a single global optimum and can be solved using sequential linear programming (SLP).

C. Convex Optimization via Sequential Linear Programming

We briefly summarize the SLP solver here. For more details, see [1]. SLP is an iterative technique for optimization problems with nonlinear objective functions but linear constraints. It operates by starting at an initial guess θ, then linearizing the objective function about θ to form a linear program (LP). The LP is solved to obtain a descent direction, θ is moved in that direction with a given step size, and the process repeats. Our implementation uses a trust region technique to choose a step size that ensures convergence toward a global optimum.

SLP is numerically stable because it makes use of widely available and stable LP packages (e.g., CPLEX, GLPK). It is also fast, because 1) only a few iterations are typically needed to terminate, and 2) each LP solve after the first can be initialized with a warm-start basis from the prior iteration, leading to faster convergence.

D. Enforcing Exact Dynamic Feasibility

The collocation point method only enforces dynamic feasibility at collocation points, which has the potential to miss constraint violations in-between. The severity of these violations shrinks as N grows, but it may be desirable to obtain dynamic feasibility regardless of the value of N.

The simplest approach to this problem is to add safety margins onto each constraint. This also helps the method tolerate execution and modeling errors. We slightly shrink the torque limits and friction coefficient estimates by a user-defined parameter. Also, we add a small offset to the each friction cone in the normal direction which enforces that each contact must apply a minimum force. Currently these parameters are chosen by manual tuning but in future work we would be interested in guaranteed feasibility. The experiments in this paper shrink friction by 50% and require a minimum of 1 kg force on each limb in contact.

A more sophisticated method would use interval analysis to bound the coefficients of the optimization parameters along each grid interval. We have implemented this technique for velocity and acceleration constraints, but extension to torque and force constraints is a matter for future work.

VI. Experiments

Numerical tests and simulations were conducted to evaluate the speed of the method and adherence to physical constraints on a variety of problems. All simulations are performed on the Klamp't simulator [16], which extends the Open Dynamics Engine rigid body simulator by improving contact handling for triangle-mesh collisions, and incorporates emulation tools for robot actuators and sensors. All timing results are conducted on a 2.67 GHz PC.

A. Pendulum Swingup Example

Fig. 4 shows optimized paths and simulated execution results for a simple pendulum swing-up task. The pendulum has a mass of 1 kg, length of 1 m and moves under the influence of gravity. The input path is a cubic Bezier curve and is not quasistatically stable due to the motor's torque limit of 5.6 Nm. Without torque limits and approximate acceleration bounds of 5.6 rad / s^2, the optimized path had duration 5.28 s and does not take advantage of the dynamic effects of gravity. As a result, during execution the robot overshoots the target. With torque bounds, the optimized path has duration 4.6 s and is executed accurately. The optimized path is solved in 89 ms with N=100.

B. Humanoid Robot Examples

Fig. 1 demonstrates the importance of dynamic contact force constraints to keep humanoids balanced. A side-to-side configuration-space path on the Hubo-II+ is executed in simulation using four time parameterizations:

- *Uniform, 5 s duration*: the robot slips at the beginning of the path as it jerks to a start, and falls over at the end of the path.
- *Uniform, 10 s duration*: ultimately, the robot does not fall over, but it still wobbles on its feet.
- *Time-scaling, acceleration constraints only*: the robot slows down too quickly and tips over.
- *Our new method*: the robot stays upright with an optimized duration of 4.99 s.

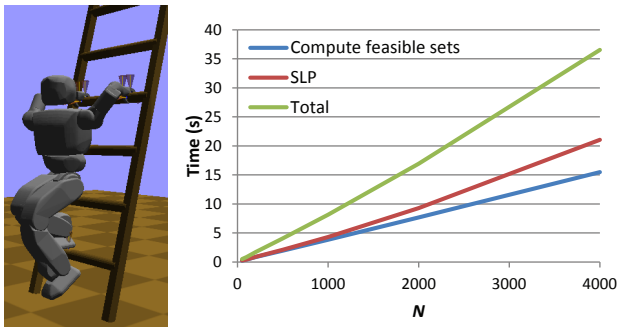

Fig. 5. A ladder-climbing motion for the Hubo-II+ humanoid robot. Variation in running time vs. the number of grid points N suggests a roughly linear relationship.

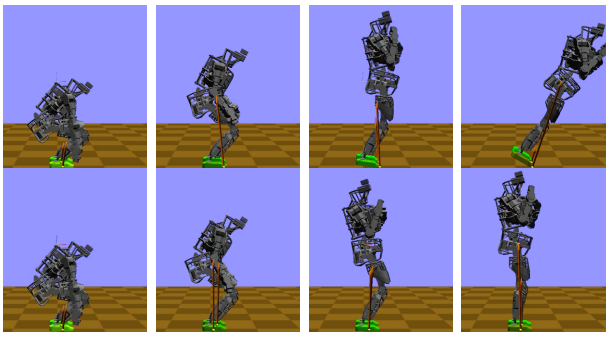

Fig. 6. Standing-up motions on a simulated ATLAS humanoid robot. Top row: the robot tips over while executing an optimized time-scaling that does not consider contact and torque constraints. Bottom row: the new method considers dynamic contact force constraints, keeping the robot balanced.

In our method we consider 59 contact points and $N = 100$. It takes 2.40 s to precompute feasible sets and 2.46 s to solve the SLP. The shade of green on the feet indicates the force magnitude; note that in our method the robot decelerates as quickly as possible by shifting all of its weight onto its right foot (second to last frame, last row).

Fig. 5 shows a 63 DOF humanoid robot ladder climbing motion. It consists of 16 changes of contact and climbs up two rungs of the ladder, and has an average of 32 contact points. The optimized motion is approximately 27 s in duration. Experiments (not shown here) indicate running times are nearly linear in N. Despite the high dimensionality of the problem and large number of contact points, the average number of feasible set boundaries per grid point is approximately 5. Hence, the SLP step is primarily only dependent on N. Feasible set computation time increases in N, the number of DOFs, and the number of contact points.

In Fig. 6 the Boston Dynamics ATLAS robot stands up from a squat. The input motion has 45 DOF and the stance contains 30 contacts. The robot stays upright during execution when motion is optimized considering force and torque constraints, and falls over otherwise. The optimized trajectory was generated in 1.44s and has duration 6.63 s.

VII. Conclusion

This paper presented a fast method for optimizing time-scalings of robot paths under dynamic contact and torque constraints. It is based on a polytope projection method to compute feasible sets of the timing parameter, which are then employed in a convex optimization time-scaling algorithm. The algorithm computes optimized trajectories in seconds for high-dimensional humanoid robots and complex contact formations. In future work we plan to study its resolution / robustness tradeoffs, and to apply it to other problems such as dynamic locomotion on uneven terrain and nonprehensile manipulation.

Source code is available in the open source Mintos library at `http://www.iu.edu/~motion/mintos/`.

References

[1] K. Hauser, "Fast interpolation and time-optimization on implicit contact submanifolds," in *Robotics: Science and Systems*, Berlin, Germany, June 2013.

[2] P. H. Channon, S. H. Hopkins, and D. T. Pham, "A variational approach to the optimization of gait for a bipedal robot," *Proc. Inst. Mech. Eng., Part C: J. Mech. Eng. Sci.*, vol. 210, no. 2, pp. 177–186, 1996. [Online]. Available: http://pic.sagepub.com/content/210/2/177.abstract

[3] A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning support contact-points for acyclic motions and experiments on hrp-2," in *Int. Symp. Experimental Robotics*, 2008, pp. 293–302. [Online]. Available: https://sites.google.com/site/adrienescandehomepage/publications/2008_ISER_Escande.pdf

[4] K. Harada, K. Hauser, T. Bretl, and J.-C. Latombe, "Natural motion generation for humanoid robots," in *IEEE/RSJ Int. Conf. Intel. Rob. Sys.*, Oct. 2006, pp. 833 –839.

[5] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Workshop Algo. Found. Robotics*, Cambridge, MA, 2012.

[6] L. Roussel, C. Canudas-de Wit, and A. Goswami, "Generation of energy optimal complete gait cycles for biped robots," in *IEEE Int. Conf. Rob. Aut.*, vol. 3, May 1998, pp. 2036 –2041.

[7] J. E. Bobrow, "Optimal robot path planning using the minimum-time criterion," *IEEE J. of Robotics and Automation*, vol. 4, pp. 443–450, 1988.

[8] E. A. Croft, B. Benhabib, and R. G. Fenton, "Near-time optimal robot motion planning for on-line applications," *J. Robotic Systems*, vol. 12, no. 8, pp. 553–567, 1995. [Online]. Available: http://dx.doi.org/10.1002/rob.4620120805

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

[10] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," in *Robotics: Science and Systems*, July 2012.

[11] Z. Shiller and H. Lu, "Computation of path constrained time-optimal motions with dynamic singularities," *ASME J. Dyn. Syst. Measure. Control*, vol. 114, pp. 34–40, 1992.

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

[13] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE Int. Conf. Rob. Aut.*, vol. 2. IEEE, 2003, pp. 1620–1626.

[14] T. Bretl and S. Lall, "A fast and adaptive test of static equilibrium for legged robots," in *IEEE Int. Conf. Rob. Aut.*, Orlando, FL, 2006.

[15] C. Jones, E. Kerrigan, and J. Maciejowski, "Equality set projection: A new algorithm for the projection of polytopes in halfspace representation," ETH Zurich, Tech. Rep. CUED/F-INFENG/TR.463, 2004. [Online]. Available: http://infoscience.epfl.ch/record/169768/files/resp_mar_04_15.pdf

[16] K. Hauser, "Robust contact generation for robot simulation with unstructured meshes," in *International Symposium on Robotics Research*, Singapore, Dec. 2013.