Trajectory Optimization with Implicit Hard Contacts

Jan Carius[1], René Ranftl[2], Vladlen Koltun[2], and Marco Hutter[1]

***Abstract*—We present a contact invariant trajectory optimization formulation to synthesize motions for legged robotic systems. The method is capable of finding optimal trajectories subject to whole-body dynamics with hard contacts. Contact switches are determined automatically. We make use of concepts from bilevel optimization to find gradients of the system dynamics including the constraint forces and subsequently solve the optimal control problem with the unconstrained iLQR algorithm. Our formulation achieves fast computation times and scales well with the number of contact points. The physical correctness of the produced trajectories is verified through experiments in simulation and on real hardware. We showcase our method on a single legged hopper for which jumping and forward hopping motions are synthesized with an arbitrary number of contact switches. The jumping trajectories can be tracked on the robot and allow it to safely liftoff and land.**

***Index Terms*—Dynamics, Optimization and Optimal Control, Legged Robots**

I. Introduction

MOTION planning of hybrid systems is inherently difficult due to the presence of both continuous dynamics and discrete switches. Moreover, the under-actuated nature of free-floating robots requires the dexterous use of contact forces, which are precisely the aspect that is intricate to handle mathematically.

Combining physically realistic dynamics and contact effects in an optimal control (OC) problem is an active area of research. The principal difficulty arises from complementarity conditions – contact forces may only act repulsively and only when bodies are in contact, furthermore bodies must not penetrate – which render the problem hard to solve numerically. Consequently, current approaches usually rely on pre-specified contact configurations, computationally demanding combinatorial optimization problems, or coarse approximations of the forces arising during hard contacts [1]–[4]. Thus, most established methods suffer from at least one of the following: strongly nonlinear constraints leading to slow convergence, computationally expensive combinatorial search, necessity to predefine step sequence, unphysical contact models, or coarse approximations of physics.

In this paper, we present a novel method for generating optimal trajectories for legged robots that overcomes the above drawbacks. We are able to jointly optimize over gait sequence, timings, and whole body motion at interactive rates. The central idea behind our approach is to delegate the contact constraints to the system dynamics where fast and robust methods are available to resolve them. This hides the problematic complementarity constraint from the OC problem and allows us to employ an unconstrained off-the-shelf Riccati-based iterative linear quadratic regulator (iLQR) [5] solver. As iLQR is inapt for handling the nonlinear state constraints arising from contacts, this would have been prohibitive previously.

Our formulation models hard unilateral contacts using set-valued force laws [6], which leads to an expression of forces in terms of the solution of a constrained convex optimization problem. As a consequence, our approach can be seen as an instance of bilevel optimization, where the higher-level OC problem is indirectly constrained by the solution to a lower-level problem which models the forces arising from hard contacts. We show how the resulting dynamics model can be used as a state propagation law in the iLQR framework by formulating the system dynamics in terms of an unrolled optimization algorithm [7]. Moreover, a hard contact formulation achieves configuration-independent contact dynamics, has a controllable restitution parameter, and handles contact forces and impulses in a unified manner. Finally, while many methods suffer from an exponential explosion of mode sequences with increasing number of contact points, our method scales well in this respect as only the matrix dimensions in the inner contact solving loop grow.

We demonstrate the generation of physically correct motions of a single-legged hopping robot both on real hardware and in simulation. The motion task can be intuitively specified through a cost function on the final or intermediate state, and the optimizer automatically chooses how to reach that goal. Our algorithm is capable of generating dynamic motions (e.g., a jump and forward steps) from a trivial initialization in a fraction of a second and involves very few tuning parameters.

A. Related work

A widespread approach to motion generation in legged robotics is to formulate an OC problem and subsequently obtain trajectories that represent (at least) local optima with respect to some performance criterion. The critical design choices therein are the way in which the variational OC problem is transcribed into a finite-dimensional mathematical program and how contact constraints are handled.

Among the dominant methods are formulations that divide the problem into distinct phases corresponding to different contact configurations. Each phase uses contact-consistent dynamics [8] and/or the corresponding contact constraints [1],

Manuscript received: February, 23, 2018; Revised May, 21, 2018; Accepted June, 23, 2018.

This paper was recommended for publication by Editor Dezhen Song upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Intel Labs, the Swiss National Science Foundation (SNF) through project 166232 and NCCR Robotics.

[1]First Author and Fourth Author are with the Robotic Systems Lab, ETH Zürich, Switzerland jcarius@ethz.ch

[2]Second Author and Third Author are with Intel Labs, Santa Clara, CA

Digital Object Identifier (DOI): see top of this page.

[9]–[12]. While this eliminates the hybrid nature of the problem, the sequence of contact configurations (i.e., the gait) must be known a priori. Recently, a per-leg phase switching parametrization was proposed [13], also allowing the gait sequence to be optimized in a continuous optimization framework without the use of complementarity constraints. This appears to be a promising approach, although it remains unclear how the formulation scales to a larger number of contacts points. Furthermore, the single body dynamics approximation breaks down if significant inertia resides outside the main body, e.g., for walking robots that have actuators in the knee joints.

On the other hand, methods that attempt to holistically solve the problem by directly incorporating complementarity constraints or by mixed integer programming have shown to be computationally heavy and typically require constraint smoothing for convergence [2], [3], [14]–[21]. As a consequence, these approaches are not suitable for real-time applications. Recent work by Werner et al. [21] extends [2] by using a collocation scheme with complementarity constraints to generate motions for a biped walking machine. Their approach is capable of finding dynamic movements given a suitable initialization without specifying the contact sequence. Our contribution attempts to solve a similar problem, though we explore the option of hiding the contact constraints from the optimizer by folding them into the system dynamics. This enables us to use two independent specialized routines for solving the contact and OC problem separately instead of burdening a general purpose optimizer with both problems at once. This separation yields a significant reduction in computation time.

The main inspiration for our work is drawn from a formulation by Neunert et al. [4] who introduce an explicit soft contact model in the system dynamics. Analogous to our approach, this allows the use of fast and unconstrained Riccati-style solvers for the OC problem while respecting the full body dynamics with contacts. Unfortunately, a soft contact model entails an inherent trade-off between numerical stiffness (hindering convergence and requiring small time steps) and unphysical behavior (e.g., forces acting at a distance) and therefore requires careful tuning for a given application. We address this problem by using a variant of Moreau's time stepping scheme [22] to calculate hard contact forces, which is advantageous for legged robots in terms of computational speed and accuracy [23]. We show how such a dynamics description can be used inside an OC problem, and that it leads to numerically stable results and fast evaluation.

Our formulation of hard contacts leads to a so-called bilevel optimization problem, i.e., an optimization problem that has another optimization problem embedded in its constraints. While these types of problems are in general intrinsically hard due to their non-convex nature, several approaches exist to find approximate solutions [24]. Our approach is based on unrolling an iterative optimization algorithm that solves the lower-level problem together with backpropagation to derive the necessary gradients for the higher-level problem. Similar formulations have been successfully used in the context of image processing and machine learning [7], [25], [26].

B. Contribution

The central contribution of this work is a contact-invariant trajectory optimization (TO) formulation that reasons about whole body motion, gait sequence and timing, and foot placement while respecting hard contact constraints (incl. friction and impacts) and being computationally competitive. To the best of our knowledge, no other method is currently capable of achieving this while preserving the temporal structure of the OC problem which leads to linear complexity in the time horizon, independent of the number of contact switches. Our principal finding is that hard contact constraints can be efficiently solved in the dynamics description of an OC problem. This advancement paves the way for online motion planning for legged systems with contacts.

The physical correctness is verified through hardware experiments on a single-legged hopping robot attached to a vertical rail. Our focus lies on the generation of jumping motions because discovering a multi-phase motion with contact switches is the key competence of a locomotion planner. Furthermore, to emphasize that our method is suitable for locomotion tasks, we additionally unlock the forward degree of freedom in simulation and generate walking motions without pre-specifying the duration and number of steps.

II. Method

Our algorithm solves a discrete-time optimal control problem of the general form

$$\underset{\boldsymbol{\tau}[\cdot]}{\text{minimize}} \quad J = \Phi(\boldsymbol{x}[n_f]) + \sum_{n=0}^{n_f-1} L(\boldsymbol{x}[n], \boldsymbol{\tau}[n], n) \;, \tag{1}$$

$$\text{s.t.} \quad \begin{cases} \boldsymbol{x}[n+1] = & \boldsymbol{f}(\boldsymbol{x}[n], \boldsymbol{\tau}[n]) \;, \\ \boldsymbol{x}[0] = & \boldsymbol{x}_0 \;. \end{cases} \tag{2}$$

We assume the initial state $\boldsymbol{x}_0$ and the cost functions Φ and L to be given. The system dynamics $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau})$ are computed by a Moreau Time-Stepping integration scheme [22], which allows modeling contacts as hard unilateral constraints. The unknown inputs $\boldsymbol{\tau}[\cdot]$ at each step are found by the unconstrained iLQR algorithm, which scales linearly in the number of steps n_f.

A. Rigid body dynamics under unilateral hard contacts

The equations of motion (EoM) of a free-floating rigid body system with m_j joints can be written as

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} - \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{\tau}) = \sum_{i=0}^{m_{\text{ext}}} \boldsymbol{J}_i^\top(\boldsymbol{q})\, \boldsymbol{f}_i^{\text{ext}} \;, \tag{3}$$

with mass matrix $\boldsymbol{M} \in \mathbb{R}^{(6+m_j)\times(6+m_j)}$, centrifugal, Coriolis, gravity, and actuation ($\boldsymbol{\tau}$) terms $\boldsymbol{h} \in \mathbb{R}^{6+m_j}$, and external forces $\boldsymbol{f}_i^{\text{ext}} \in \mathbb{R}^3$ with corresponding Jacobians $\boldsymbol{J}_i \in \mathbb{R}^{3\times(6+m_j)}$. For clarity, we drop the dependency on generalized coordinates $\boldsymbol{q} \in SE(3) \times \mathbb{R}^{m_j}$ and velocities $\boldsymbol{u} \in \mathbb{R}^{6+m_j}$ below.

By choosing appropriate coordinates $\boldsymbol{q}$, we can assume without loss of generality that the external forces only arise from active unilateral constraints[1] (i.e., contact forces) and we

[1] other external forces that can be expressed as a function of $\{n, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{\tau}\}$ can be included in $\boldsymbol{h}$

denote the index set of active contacts $\mathcal{A}$. The complementarity condition between separation (gap between end-effector and terrain) $\boldsymbol{g}_i \in \mathbb{R}^3$ and force of a hard unilateral contact is conveniently expressed as a normal cone ($\mathcal{N}$) inclusion on velocity level [22]

$$-\dot{\boldsymbol{g}}_i \in \mathcal{N}_{\mathcal{F}_i}(\boldsymbol{f}_i^{\text{ext}}) \quad \forall \text{ contacts } i \in \mathcal{A} \,, \tag{4}$$

as this directly allows to incorporate friction laws which are naturally expressed in terms of velocities. One may split the contact forces into normal and tangential components $\boldsymbol{f}_i^{\text{ext}} = \begin{bmatrix} f_{i,n} & \boldsymbol{f}_{i,t}^\top \end{bmatrix}^\top \in \mathcal{F}_{i,n} \times \mathcal{F}_{i,t}$. The force reservoirs $\mathcal{F}_i$ allow nonnegative normal forces ($\mathcal{F}_{i,n} = \mathbb{R}_{0+}$) and tangential forces obeying Coulomb friction ($\mathcal{F}_{i,t} = \{\boldsymbol{f}_t \in \mathbb{R}^2 \mid ||\boldsymbol{f}_t|| < \mu |f_{i,n}|\}$) with friction coefficient μ. The intuitive explanation of this formulation is that as long as the force stays in the interior of its permissible reservoir, the contact velocity must be zero. Conversely, the velocity can only be non-zero if the forces are at the border of their permissible set, i.e., zero normal force or maximum friction force opposing the direction of motion.

To complete the contact dynamics, it is helpful to express the equation of motion (3) in local contact coordinates (task space). This is achieved by realizing that

$$\dot{\boldsymbol{g}} = \boldsymbol{J_c} \boldsymbol{u} \,, \tag{5}$$

where $\boldsymbol{g}$ and $\boldsymbol{J_c}$ are the stacked contact separations and Jacobians, respectively. By taking the time derivative of (5) and substituting (3) we arrive at

$$\ddot{\boldsymbol{g}} = \underbrace{\boldsymbol{J_c} \boldsymbol{M}^{-1} \boldsymbol{J_c}^\top}_{\boldsymbol{G}} \boldsymbol{f}^{\text{ext}} + \underbrace{\dot{\boldsymbol{J}}_{\boldsymbol{c}} \boldsymbol{u} + \boldsymbol{J_c} \boldsymbol{M}^{-1} \boldsymbol{h}}_{\boldsymbol{c}} \,, \tag{6}$$

where $\boldsymbol{G}$ (Delassus matrix) represents the apparent inverse inertia at the contact points and $\boldsymbol{c}$ collects all terms that are independent of the stacked external forces $\boldsymbol{f}^{\text{ext}}$. The principle of least action now states that the contact forces are given by the solution of a constrained optimization problem:

$$\underset{\{\boldsymbol{f}_i \in \mathcal{F}_i\}}{\text{minimize}} \quad \frac{1}{2} \boldsymbol{f}^{\text{ext}\top} \boldsymbol{G} \boldsymbol{f}^{\text{ext}} + \boldsymbol{f}^{\text{ext}\top} \boldsymbol{c} \,. \tag{7}$$

In the same framework, impulse laws can be used to calculate instantaneous velocity changes and impulsive forces. By imposing a time discretization and equality of continuous and impulsive measures [6], one can unify impulses and force integrals to a quantity called percussions $\boldsymbol{p}$. This is the core of a time-stepping method [6], [22] which integrates the system dynamics (3, 4, 7) over an interval Δt by internally resolving the contact forces.

The essential steps are summarized in Alg. 1. A single call to this procedure propagates the given starting positions $\boldsymbol{q}_0$ and velocities $\boldsymbol{u}_0$ over the discretization period Δt. An initial forward-Euler half step in generalized positions determines the configuration in the middle of the interval

$$\boldsymbol{q}_m = \boldsymbol{q}_0 + (\Delta t/2) \boldsymbol{F}(\boldsymbol{q}_0) \boldsymbol{u}_0 \,, \tag{8}$$

where $\boldsymbol{F}(\boldsymbol{q}_0)$ maps the generalized velocities to the derivative of the generalized coordinates. The quantities in the middle of the interval are then used inside the proximal point iteration whose results are the percussions $\boldsymbol{p}$ that shall act over this interval. The

Algorithm 1 Moreau's time-stepping algorithm

Input: State $\{\boldsymbol{q}_0, \boldsymbol{u}_0\}$ at beginning and torques $\boldsymbol{\tau}$ acting during this interval
Forward-Euler half step:
- $\boldsymbol{q}_m = \boldsymbol{q}_0 + (\Delta t/2) \boldsymbol{F}(\boldsymbol{q}_0) \boldsymbol{u}_0$
- $\boldsymbol{M}_m = \boldsymbol{M}(\boldsymbol{q}_m)$
- $\boldsymbol{h}_m = \boldsymbol{h}(\boldsymbol{q}_m, \boldsymbol{u}_0, \boldsymbol{\tau})$

For active contacts $i \in \mathcal{A}$**, assemble Jacobians and local dynamic quantities**:
- $\boldsymbol{J_c}^\top = \begin{bmatrix} \cdots & \boldsymbol{J}_i(\boldsymbol{q}_m)^\top & \cdots \end{bmatrix}$
- $\boldsymbol{G} = \boldsymbol{J_c} \boldsymbol{M}_m^{-1} \boldsymbol{J_c}^\top$
- $\boldsymbol{c} = (\boldsymbol{I} + \boldsymbol{\varepsilon}) \boldsymbol{J_c} \boldsymbol{u}_0 + \boldsymbol{J_c} \boldsymbol{M}_m^{-1} \boldsymbol{h}_m \Delta t$

Iteratively solve for percussions of active contacts:
Init: $\boldsymbol{p}_i \leftarrow$ steady state solution in normal direction, zero tangential percussion
for iter = 1:maxIter **do**
  for contact i in $\mathcal{A}$ **do**
    $\boldsymbol{p}_i \leftarrow \text{prox}_{\mathcal{F}_i} \left[\boldsymbol{p}_i - r_i \left(\sum_{j \in \mathcal{A}} \boldsymbol{G}_{ij} \boldsymbol{p}_j + \boldsymbol{c}_i \right) \right]$ (*)
  end for
end for
Velocity and position update step:
- $\boldsymbol{u}_e = \boldsymbol{u}_0 + \boldsymbol{M}_m^{-1} (\boldsymbol{h}_m \Delta t + \boldsymbol{J_c}^\top \boldsymbol{p})$
- $\boldsymbol{q}_e = \boldsymbol{q}_0 + \Delta t \, \boldsymbol{F}(\boldsymbol{q}_0) \, (\boldsymbol{u}_0 + \boldsymbol{u}_e)/2$

Output: State $\boldsymbol{q}_e, \boldsymbol{u}_e$ at end of interval

central iteration scheme (*) combines a fixed-point iteration with a proximal point projection. The procedure corresponds to an overrelaxed Gauss-Seidel iteration scheme which was shown suitable for legged robots [23]. When converged, the percussions are contained in their respective reservoirs while fulfilling the complementarity condition (4). The algorithm is completed by a Euler step in the velocities using the EoM (3) and applying the midpoint discretization on position level.

Alg. 1 gives us direct control over the restitution coefficients $\boldsymbol{\varepsilon} = \text{diag}\{\varepsilon_i\}$ of each contact. Furthermore, the time-stepping scheme unifies the treatment of contact states (open, stick, slip). This means a subsequent optimization algorithm does not require additional constraints such as no-slip conditions. Instead, it has control over the contact state and may exploit this to find optimal motions that may include slipping. The only tuning parameter is the number of proximal point iterations, which is easy to choose for a given system by inspecting convergence plots for $\boldsymbol{p}$. The factor r_i only affects convergence speed and the choice $r_i = (\sum_j |\boldsymbol{G}_{ij}|)^{-1}$ gives reliable performance in practice.

By defining our state as $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}^\top & \boldsymbol{u}^\top \end{bmatrix}^\top$, this algorithm directly represents the state-propagation law $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau})$ in the OC problem (2). We use a fixed number of iterations for solving the percussions which allows taking the derivatives $\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau})}{\partial \boldsymbol{x}}$, $\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau})}{\partial \boldsymbol{\tau}}$ by backpropagation. The derivative is well-defined almost everywhere except at the kink of the contained absolute value functions, at which we assume the derivative to be zero ($0 \in \partial \, \text{abs}(x)|_{x=0}$). Moreover, our implementation avoids any branching in the computational graph to make it suitable for algorithmic differentiation.

B. iLQR optimization

The iLQR algorithm is a single-shooting method and is explained in detail in [5]. Each iteration consists of three parts: first, a rollout step that computes the nominal state trajectory $\boldsymbol{x}^{\text{ref}}[\cdot]$ given the current input sequence $\boldsymbol{\tau}[\cdot]$ with the propagation law (2). An initial non-diverging policy is therefore required. Second, a linear quadratic (LQ) approximation of the OC problem is computed around the nominal state and input trajectory, which requires quadratic approximation of the cost function and linearization of the system dynamics, represented

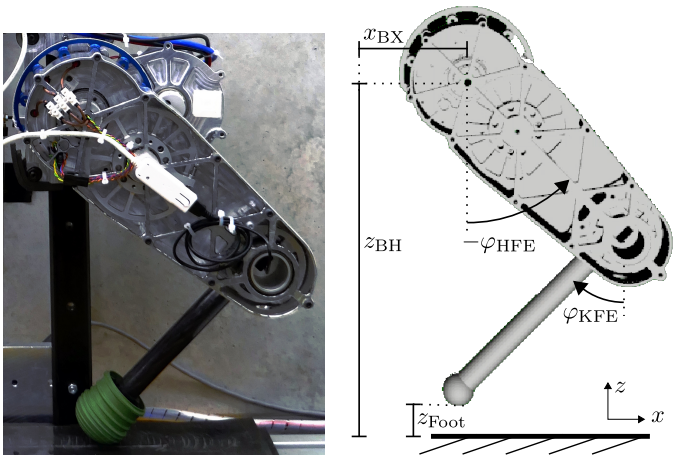

Fig. 1. Image of the Capler hardware and schematic drawing with the relevant coordinates and modeling quantities.

by Alg. 1. Third, the resulting quadratic value function leads to a Bellman equation that can be solved easily in the backwards pass, resulting in the $\boldsymbol{\tau}^{\text{ref}}[\cdot]$ and $\boldsymbol{K}[\cdot]$ terms of a linear quadratic regulator (LQR)-like feedback control law

$$\boldsymbol{\tau}[n] = \boldsymbol{\tau}^{\text{ref}}[n] + \boldsymbol{K}[n](\boldsymbol{x}^{\text{ref}}[n] - \boldsymbol{x}[n]) \; . \tag{9}$$

Finally, we use line search over the update of $\boldsymbol{\tau}^{\text{ref}}[\cdot]$ by evaluating the trajectory cost (1) at different update step sizes for the new $\boldsymbol{\tau}^{\text{ref}}$. For a more detailed description on the iLQR implementation, the reader is referred to [4], [27].

An important observation is that by formulating the dynamics as in Sec. II-A, we can generate feasible trajectories without introducing any additional constraints and can, therefore, employ an off-the-shelf iLQR implementation. Thus, the special structure of the OC problem is retained, resulting in linear complexity in the time horizon. For different applications, it may be desirable to incorporate additional state or input constraints. In this setting, linearizations of the constraints around the trajectory need to be computed in the forward pass to construct a constrained linear quadratic optimal control (LQOC) problem. Efficient solvers exist for the backward pass that still maintain linear time complexity [28], [29]. The feedback law (9) can also be recovered but does not guarantee admissible inputs under arbitrary state deviations.

III. Application on a Single-Legged Hopper

A. System

We apply our method to a single-legged hopper called *Capler* [30], which has three degrees of freedom (DOFs) ($\boldsymbol{q} = [z_{\text{BH}},\, \varphi_{\text{HFE}},\, \varphi_{\text{KFE}}]^\top$) and is actuated by two direct-drive motors at the hip and knee joints (see Fig. 1). Though seemingly simple, this system exhibits the main difficulties of legged robots (nonlinear dynamics, contact switching). Encoders in the joints and a draw wire sensor for the base height allow direct measurements of the system state. For the purpose of demonstrating forward hopping motions in simulation, we additionally allow sideway displacement in the x direction and augment the generalized coordinates to $\boldsymbol{q} = [z_{\text{BH}},\, x_{\text{BX}},\, \varphi_{\text{HFE}},\, \varphi_{\text{KFE}}]^\top$.

As is typical for robotic systems, we set the restitution coefficients to zero and assume a Coulomb friction coefficient of $\mu = 0.7$. In Fig. 2 we show how the proximal point iteration converges rapidly and consequently choose to limit the number of iterations in Alg. 1 to 30. Given our time discretization of $\Delta t = 0.01\,\text{s}$, the update magnitude to $\boldsymbol{p}$ after 30 iterations amounts to less than 0.05 % of the final magnitude, even in the case when the hopper is landing from a 15 cm jump and the percussion has to encounter a 1.77 m/s incident velocity. We initialize the percussion with $\boldsymbol{p} = [0,\, 0.5]^\top$, which is near the steady state resting solution.

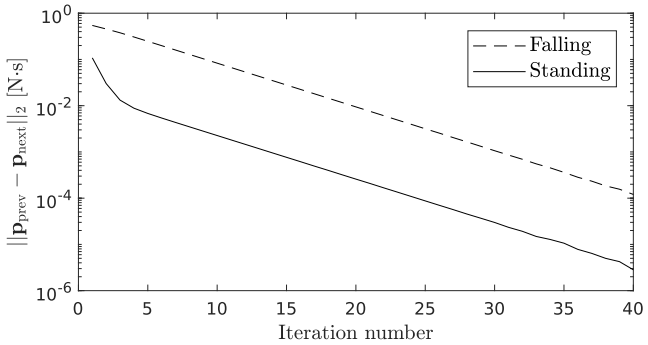

Fig. 2. Norm of the update to the percussion vector $\boldsymbol{p}$ at each prox iteration. Convergence is faster in steady-state standing than with an incident velocity.

B. Software implementation

Key to solving (1) efficiently are fast evaluations of the system dynamics and its derivatives. We use the code generator RobCoGen [31] to compute the EoM terms in (3), including the analytic mass matrix inverse required in Alg. 1. Subsequently, we use the algorithmic differentiation tool CppAD [32] to calculate the partial derivatives of the system dynamics including contact effects. This tool additionally allows the generation of code for the evaluation of $\boldsymbol{f}$ and its derivatives. This procedure eliminates unnecessary computations and function calls and condenses mathematical operations, which provides a significant speedup [33].

The complete iLQR routine with parallelized line search and the algorithmic differentiation tools are implemented in the open-source Control Toolbox (CT) [27]. In this work, we generalize the CT to handle continuous and discrete (e.g., as a result of Alg. 1) system dynamics invariantly while maintaining the same solver implementation.

C. Control

The iLQR procedure returns a sequence of nominal control inputs $\boldsymbol{\tau}^{\text{ref}}[\cdot]$ together with a time-varying state-feedback matrix $\boldsymbol{K}[\cdot]$. When the nominal inputs are naively applied to the real system, model mismatches result in a deviation from the planned reference trajectory $\boldsymbol{x}^{\text{ref}}$, which is implicitly given by $\boldsymbol{\tau}^{\text{ref}}[\cdot]$ and (2). To compensate these errors, we consider two possible control laws, LQR and proportional derivative (PD), which both use the state tracking error $\delta\boldsymbol{x} := \boldsymbol{x}^{\text{ref}} - \boldsymbol{x}$ for feedback and the torque trajectory as feed-forward commands:

$$\boldsymbol{\tau}_{\text{LQR}} = \boldsymbol{K}\,\delta\boldsymbol{x} + \boldsymbol{\tau}^{\text{ref}} \; , \tag{10}$$

$$\boldsymbol{\tau}_{\text{PD}} = \begin{bmatrix} k_p\,\delta\varphi_{\text{HFE}} + k_d\,\delta\dot{\varphi}_{\text{HFE}} \\ k_p\,\delta\varphi_{\text{KFE}} + k_d\,\delta\dot{\varphi}_{\text{KFE}} \end{bmatrix} + \boldsymbol{\tau}^{\text{ref}} \; . \tag{11}$$

The time-varying linear feedback gains $\boldsymbol{K}$ yield an adaptive controller: they adjust the stiffness depending on flight or stance phases and prioritize the tracking of those quantities that were important in the optimization. For example if base height was paramount in the OC problem, it would automatically sacrifice hip and knee joint tracking to keep the height deviation at a minimum. However, there are no theoretical guarantees that (10) stabilizes the system under model error and disturbances and we experience that the gains are too aggressive for a jumping motion on the real system. For a fair comparison, we, therefore, use a hand-tuned PD controller (11) in all of our experiments.

To minimize delays, the host computer manages all time-critical communication through shared memory and transmits the motor readings and commands via EtherCAT. The PD controller updates at 1000 Hz and linearly interpolates the reference state and input trajectories. A low-level current tracking loop on each actuator is closed at 20 kHz to produce the commanded torques. When tested in simulation, the joint references are tracked very closely, suggesting that the computed trajectory is feasible, but a discrepancy in base height is visible. This stresses the fact that, even in simulation, the contact point location is not precisely matching our internal model (which assumes a point foot) and therefore the robot stays a few centimeters short of the expected jump height.

D. Trajectory generation

To generate motion plans with our proposed algorithm, it is sufficient to use only quadratic cost function terms (cf. (1))

$$\Phi(\boldsymbol{x}[n_f]) = (\boldsymbol{x}[n_f] - \bar{\boldsymbol{x}})^\top \boldsymbol{Q}_f (\boldsymbol{x}[n_f] - \bar{\boldsymbol{x}}) \,, \tag{12}$$

$$\begin{aligned} L(\boldsymbol{x}[n], \boldsymbol{\tau}[n], n) &= (\boldsymbol{x}[n] - \bar{\boldsymbol{x}})^\top \boldsymbol{Q} (\boldsymbol{x}[n] - \bar{\boldsymbol{x}}) \\ &+ (\boldsymbol{x}[n] - \bar{\boldsymbol{x}}_a)^\top a[n] \boldsymbol{Q}_a (\boldsymbol{x}[n] - \bar{\boldsymbol{x}}_a) \\ &+ \boldsymbol{\tau}[n]^\top \boldsymbol{R} \boldsymbol{\tau}[n] \,, \end{aligned} \tag{13}$$

where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}_{\text{jump}}$ are constant reference states. The running cost matrices $\boldsymbol{Q} \succeq 0, \boldsymbol{R} \succ 0$ serve only as regularization terms, while $\boldsymbol{Q}_f \succeq 0$ penalizes deviation from a desired final state. To encode behaviors such as jumping, we use a time activation term $a[n]$ which is zero almost everywhere and takes the value one for a short period of time together with a cost matrix $\boldsymbol{Q}_a$ which has a strong penalty on a specific part of the state, e.g., the base height error.

To enforce torque limits in the optimization, we clamp the control inputs $\boldsymbol{\tau}$ to the permissible interval $[-60, 60]\ Nm$ inside the time-stepping dynamics. The optimizer therefore automatically respects the torque limits even without hard constraints because increasing the inputs beyond the maximum only incurs a greater cost at no additional benefit. In practice, this yields the same result as explicitly enforcing input constraints in the backward pass but avoids a performance penalty in the solver. Moreover, we observe a natural convergence behavior to no-slip contact even though slipping would be possible.

IV. Results

We conduct experiments in a simulation environment and on real hardware to evaluate how our method compares to competing approaches, its ability to discover physically meaningful trajectories at competitive speeds, and to verify that the motions generalize to the physical system.

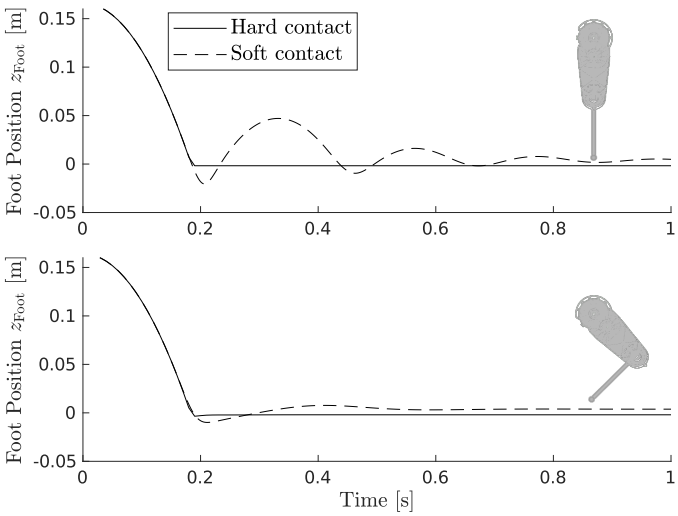

Fig. 3. Comparison between a hard and soft contact model in forward simulation. In each case, the robot is dropped with its foot 16 cm above the ground with PD controls on a fixed hip and knee position. We show both a fall in a stretched out (top) and a crouched (bottom) configuration.

A. Soft contact comparison

We evaluate our dynamics model against the soft contact model proposed by Neunert et al. [4], which was also used for TO and is openly accessible in [27]. An important initial observation is that naively using soft contacts with default settings and a discretization of 10 ms yield a highly unstable behavior for our monopod. The stiff spring dynamics force us to use a sampling interval of 1 ms. Additionally, appropriate tuning of the relevant parameters (sigmoid smoothing, damping, stiffness) proves tedious.

Our model provides a direct handle on the restitution behavior whereas for soft contacts this cannot be controlled for all possible interactions. This can be seen in Fig. 3, where we plot the foot height time series for a crouched and stretched-out landing position from a 16 cm fall. It is immediately evident that the soft contact model has substantially different behavior for different configurations (i.e., apparent inertia and stiffness of the system). For the shown plot we choose parameters to almost eliminate the distant force effect – hence no deviation in the initial flight phase – with the drawback of ground penetration and rebound. We were unable to find a set of parameters that leads to a sensible tradeoff between distant force, penetration, and restitution in both configurations. Our method, on the other hand, is virtually invariant to the configuration and only shows a small difference in the foot resting position on the ground. This offset is because ground reaction forces only act once the foot has penetrated ($z_{\text{Foot}} < 0$) and we happen to 'catch' the system at a slightly different instance due to time discretization.

B. Scalability with respect to the number of contact points

To demonstrate how our method behaves under an increasing number of contact points, we construct a multi-leg version of Capler as shown at the top of Fig. 4. Multiple instances of the

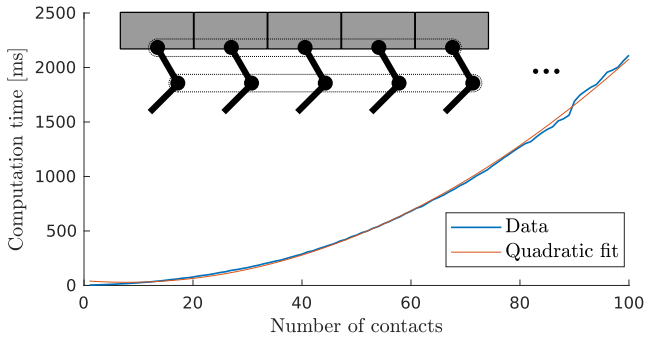

Fig. 4. Computation times for 1000 time steps forward simulation for a varying number of contact points (see schematic at top). The fitted second order polynomial confirms a quadratic computational complexity.

same system are chained next to each other and all base, hip and knee DOFs are linked to each other (like a 1:1 transmission). Furthermore, a commanded hip or knee torque acts on all hips or knees, respectively. This construction emulates a system of constant size regarding state and input but with an arbitrary number of contact points. The corresponding percussions at each foot are then solved for by the time stepping algorithm. Such a scenario is similar to the case of a more complicated walking robot for which contact points are added at the knees, feet, hands, or elbows.

We forward simulate the dynamics for 1000 steps and measure the elapsed computation time. We expect computation time to grow quadratically with the number of contact points since the Delassus matrix ($\boldsymbol{G}$) size depends quadratically on the contact points. This conjecture is confirmed by fitting a quadratic model to the recorded times and reproduces the result of state-of-the-art simulators [34]. Our method is hence well suited for tasks that require many contact interactions.

C. Motion generation with jumps

Our method is able to find jumping and walking motions by only imposing costs on the base position without any prior hints that the foot can or should be lifted. We initialize all motions with a trivial standing trajectory by setting the initial policy to a PD controller whose reference is a crouched position. The time horizon in the following examples is chosen to be of 3 s length (300 time steps). It should be noted, however, that also longer motions with multiple jumps can be synthesized as shown in the accompanying video[2].

The runtime for a single iteration (with line search) for the 3 s horizon is 2.18 ms on average (std. dev. 0.25 ms) on a standard laptop computer (Intel Core i7, 4 x 2.50 GHz), which makes our method a promising candidate for use in a model predictive control (MPC) setting. The speed is on par with a direct collocation scheme [8] applied to our 3-DOF system with fixed gait sequence. A method [17] using strongly simplified humanoid dynamics also achieve similar speed while optimizing over the gait sequence. Most competing approaches for synthesizing dynamic contact invariant motions [2], [13], [14], [21] report times that are orders of magnitude larger, albeit using models with more DOFs in some cases which makes comparisons difficult.

[2]https://youtu.be/mSpiRdPU0VE

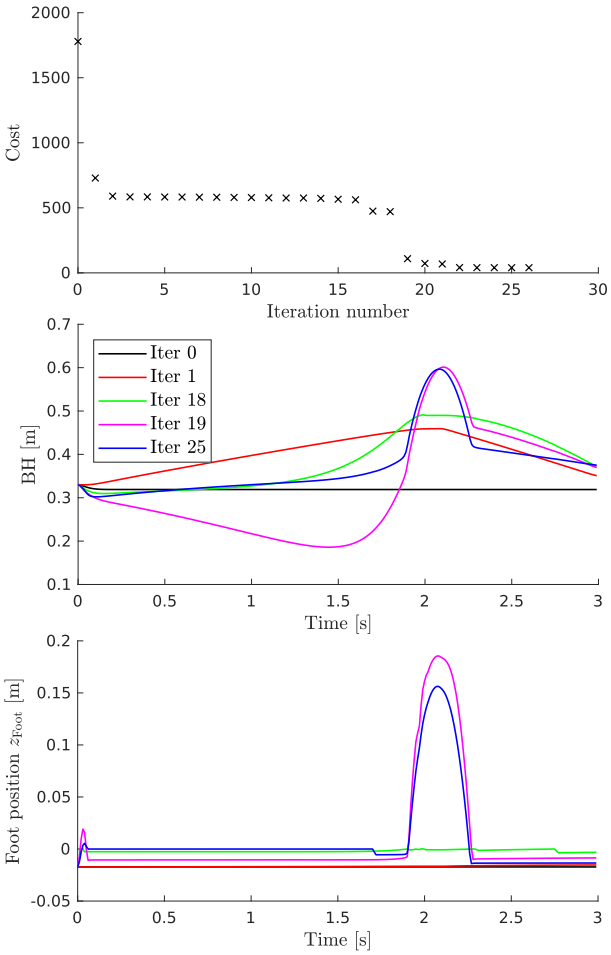

Fig. 5. Progression of optimization at different iteration numbers. The algorithm is initialized (iteration 0) with a standing behavior and converges to a jumping motion after 25 iterations.

1) 1D vertical hopping: For illustration purposes and due to our hardware setup, we first demonstrate a vertical jumping motion with the 3 DOF system. For this task, the cost parameters are (cf. III-D)

$$\begin{aligned}
\boldsymbol{Q} &= \text{diag}[0, 0.01, 0.01, 0.1, 3, 3],\ \boldsymbol{R} = 0.02 \cdot \text{diag}[1, 1]\,, \\
\boldsymbol{Q}_f &= \text{diag}[1000, 10, 10, 10, 10, 10]\,, \\
\bar{\boldsymbol{x}} &= [0.25, -\pi/4, \pi/2, 0, 0, 0]\,, \\
\boldsymbol{Q}_a &= \text{diag}[8 \cdot 10^5, 0, 0, 0, 0, 0],\ a[n] = \begin{cases} 1 & \text{if } n \in [200, 210], \\ 0 & \text{otherwise,} \end{cases} \\
\bar{\boldsymbol{x}}_a &= [0.6, -\pi/4, \pi/2, 0, 0, 0]\,. \qquad (14)
\end{aligned}$$

A time-activated term acts for 0.1 s and penalizes the deviation from a base height beyond reach, which provokes a jump.

With these settings and initialization, the algorithm requires 25 iterations to converge to a jumping motion. We plot the cost evolution and trajectories of selected intermediate iterations in Fig. 5. The initialization is evidently sub-optimal for the jump objective, yet the algorithm understands within two iterations that it can use the hip and knee torques to generate ground reaction forces that push the base upwards. The flat cost region between iteration 2 and 16 corresponds to stretching out the leg as far as possible but without pushing off. At iteration 19, the algorithm discovers that it is possible to jump by lifting off the foot and refines this motion in the last few iterations.

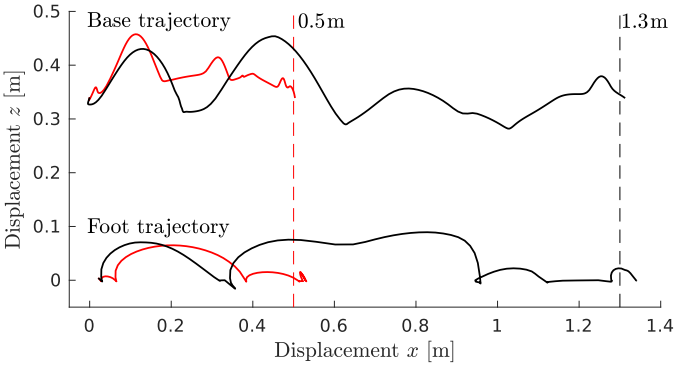

Fig. 6. Forward jumping trajectories of base and foot for goal distances of 0.5 m (red) and 1.3 m (black).

Clearly, initializing with a motion that already includes a flight phase would have reduced the number of iterations drastically, but it is interesting to see that our method can escape the local minimum (corresponding to only extending the leg without liftoff) and autonomously discover that the foot can be lifted. Interestingly, after discovering that the lowest costs are possible when including a flight phase at around $t = 2\,\mathrm{s}$, the optimized trajectory additionally repositions the foot with a short sidestep at the beginning to be in a more cost-effective configuration for the subsequent large jump. This repositioning behavior can be observed for various starting positions. Note that foot positions below zero (seen in Fig. 5) are possible as our contact solver acts only on velocity level.

2) 2D forward hopping: We can generate a forward hopping motion with a variable number of steps. For that purpose, we use the extended model with unlocked sideways DOF and impose a cost on the base displacement in x direction. No time-activated terms are necessary:

$$\begin{aligned} &\boldsymbol{Q} = 0.01 \cdot \mathrm{diag}[1, 10, 2, 2, 1, 1, 4, 4],\ \boldsymbol{R} = 0.001 \cdot \mathrm{diag}[1, 1]\,, \\ &\boldsymbol{Q}_f = 400 \cdot \mathrm{diag}[5, 2, 1, 1, 5, 5, 5, 5]\,, \\ &\bar{\boldsymbol{x}} = [0.35, x_{\mathrm{Goal}}, -\pi/4, \pi/2, 0, 0, 0, 0]\,. \end{aligned} \tag{15}$$

We plot the resulting motion plans for $x_{\mathrm{Goal}} = 0.5\,\mathrm{m}$ and $1.3\,\mathrm{m}$ in Fig. 6. The optimizer requires a similar number of iterations for these motions and the base reaches the goal with centimeter accuracy. We can also generate motions over longer distances as can be seen in the video.

D. Transfer to the physical system

To verify that our trajectories are physically consistent, we compute a vertical jump motion (see Sec. IV-C1) and execute it on the physical Capler system. The tracking controller PD gains can be chosen relatively weak because they are only needed for small corrections while the feed-forward torques fulfill gravity compensation and give rise to the required accelerations under the nominal model. We use controller parameters $k_p = 35.0\,\mathrm{N\,m\,rad^{-1}}$, $k_d = 1.0\,\mathrm{N\,m\,s\,rad^{-1}}$.

The reference tracking performance can be observed in Fig. 8 for the three generalized positions. The primary objective of this motion is to reach the desired base height and one may see from the figure that the nominal jump height of $z_{\mathrm{BH}} = 63\,\mathrm{cm}$ was achieved within a few centimeters.

It is no surprise that the hip and knee position tracking is significantly more accurate than the base' because the corrective action of the PD controller acts on individual joint level and does not compensate miscalibrated ground position and incorrect values for the link lengths. Such behavior can, in theory, be expected from using the iLQR state feedback controller (10). Still, even though small angular offsets in hip and knee propagate to an error in base height and there may be calibration errors in the draw wire encoder, the overall motion is followed very closely. Fig. 7 shows a sequence of images of the real system during the jump.

We may conclude from this section that the motions are physically feasible and generalize to a physical system without further modifications like smoothing. Our method produces jumping motions that allow the system to lift off and land safely even without adjusting the trajectory online. Still, we expect a further performance improvement when recomputing the trajectory during execution with a measured initial state.

V. Conclusion and Future Work

We have presented a method for generating motion plans for robotic systems that are subject to hard contacts. The method reasons about whole body motions while being invariant to contact switches. Our key insight is that the hybrid OC problem can be solved efficiently when contact constraints are resolved in the system dynamics instead of exposing them to the OC solver directly. The algorithm was shown to produce trajectories that can be followed accurately in simulation and also generalize to a real robot with an appropriate tracking controller.

We plan to leverage this method to find trajectories for legged robots for which it would be difficult to pre-specify a gait schedule, for example on uneven terrain and for climbing motions that require many contact points. We believe the ability to find a multi-phase motion – as demonstrated in this paper – is the key component for generating more sophisticated walking behaviors. Conveniently, the generalization to rough terrain only requires a different gap function $\boldsymbol{g}$ that accounts for location-dependent ground height. No further constraints are necessary. We expect to achieve superior tracking and disturbance rejection behavior by extending to a full MPC-style control implementation.

One may draw a parallel between our method and reinforcement learning approaches to locomotion. In both cases, an optimizer queries a simulation ('dynamics integrator') to find the optimal control policy. However, the proposed method is much more sample efficient because we can directly evaluate the gradient of the simulation with respect to the state-input pair and therefore infer the direction of improvement instead of estimating the gradient from several samples.

References

[1] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[2] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The Int. Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

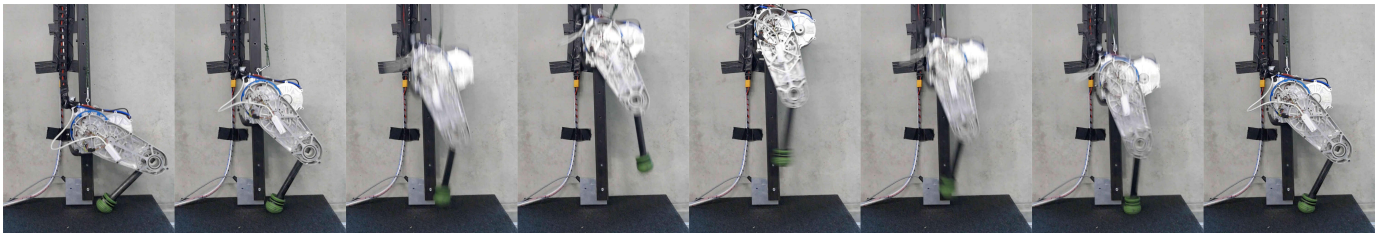

Fig. 7. Execution of a jumping motion on the physical Capler hopper.

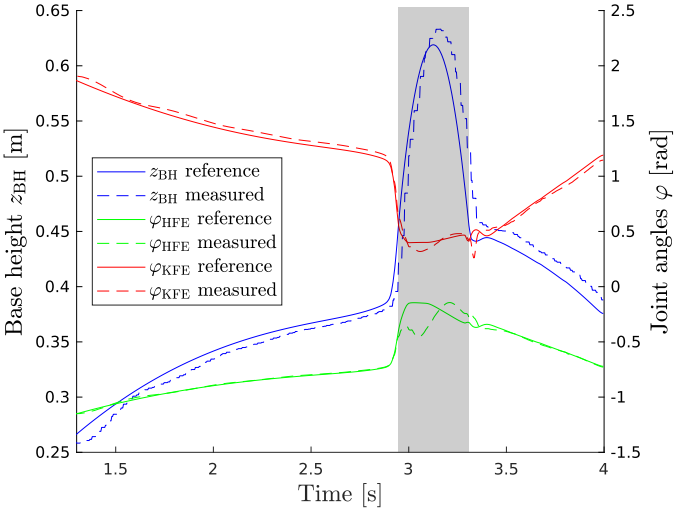

Fig. 8. Real tracking performance on the physical hopper. The measured values correspond to the dashed lines while the references are solid. Shaded areas indicate a predicted flight phase.

[3] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *Int. Symposium on Robotics Research (ISRR)*, 2017.

[4] M. Neunert, M. Stauble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1458–1465, 2018.

[5] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Int. Conf. on Informatics in Control, Automation and Robotics, (ICINCO)*, 2004.

[6] C. Studer, *Numerics of unilateral contacts and friction: modeling and numerical time integration in non-smooth dynamics*. Springer Science & Business Media, 2009, vol. 47.

[7] J. Domke, "Generic methods for optimization-based modeling," in *Int. Conf. on Artificial Intelligence and Statistics, AISTATS*, 2012.

[8] D. Pardo, M. Neunert, A. Winkler, R. Grandia, and J. Buchli, "Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion," in *Robotics: Science and Systems (RSS)*, 2017.

[9] H.-W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Robotics: Science and Systems (RSS)*, 2015.

[10] M. Kudruss, M. Naveau, O. Stasse, N. Mansard, C. Kirches, P. Soueres, and K. Mombaur, "Optimal control for multi-contact, whole-body motion generation using center-of-mass dynamics for multi-contact situations." in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2015.

[11] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[12] F. Farshidian, M. Kamgarpour, D. Pardo, and J. Buchli, "Sequential linear quadratic optimal control for nonlinear switched systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1463 – 1469, 2017.

[13] A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1560–1567, 2018.

[14] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 43:1–43:8, 2012.

[15] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[16] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The Int. Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013.

[17] A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from mixed-integer model predictive control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[18] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014.

[19] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014.

[20] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hierarchical planning of dynamic movements without scheduled contact sequences," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[21] A. Werner, W. Turlej, and C. Ott, "Generation of locomotion trajectories for series elastic and viscoelastic bipedal robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[22] J. J. Moreau, *Unilateral Contact and Dry Friction in Finite Freedom Dynamics*. Vienna: Springer, 1988, pp. 1–82.

[23] C. Gehring, R. Diethelm, R. Siegwart, G. Nützi, and R. I. Leine, "An evaluation of moreaus time-stepping scheme for the simulation of a legged robot," in *ASME Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf.* American Society of Mechanical Engineers, 2014.

[24] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals OR*, vol. 153, no. 1, pp. 235–256, 2007.

[25] P. Ochs, R. Ranftl, T. Brox, and T. Pock, "Techniques for gradient-based bilevel optimization with non-smooth lower level problems," *Journal of Mathematical Imaging and Vision*, vol. 56, no. 2, pp. 175–194, 2016.

[26] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," *CoRR*, vol. abs/1502.03240, 2015.

[27] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli, "The Control Toolbox - an open-source C++ library for robotics, optimal and model predictive control," https://adrlab.bitbucket.io/ct, 2018, arXiv:1801.04290.

[28] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Conf. on Decision and Control (CDC)*, 2012.

[29] G. Frison, "Algorithms and methods for high-performance model predictive control," Ph.D. dissertation, Technical University of Denmark (DTU), 2015.

[30] J. Hwangbo, V. Tsounis, H. Kolvenbach, and M. Hutter, "Cable-driven actuation for highly dynamic robotic systems," 2018, arXiv:1806.10632 [cs.RO].

[31] M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "RobCoGen: a code generator for efficient kinematics and dynamics of articulated robots, based on Domain Specific Languages," *Journal of Software Engineering for Robotics (JOSER)*, vol. 7, no. 1, pp. 36–54, 2016.

[32] B. M. Bell, "CppAD: A package for C++ algorithmic differentiation," *Computational Infrastructure for Operations Research*, vol. 57, 2012.

[33] M. Giftthaler, M. Neunert, M. Stäuble, M. Frigerio, C. Semini, and J. Buchli, "Automatic differentiation of rigid body dynamics for optimal control and estimation," *Advanced Robotics*, vol. 31, no. 22, pp. 1225–1237, 2017.

[34] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 895–902, 2018.