IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Hybrid Systems Differential Dynamic Programming for Whole-Body Motion Planning of Legged Robots

He Li and Patrick M. Wensing

***Abstract*— This paper presents a Differential Dynamic Programming (DDP) framework for trajectory optimization (TO) of hybrid systems with state-based switching. The proposed Hybrid Systems DDP (HS-DDP) approach is considered for application to whole-body motion planning with legged robots. Specifically, HS-DDP incorporates three algorithmic advances: an impact-aware DDP step addressing the impact event in legged locomotion, an Augmented Lagrangian (AL) method dealing with the switching constraint, and a Switching Time Optimization (STO) algorithm that optimizes switching times by leveraging the structure of DDP. Further, a Relaxed Barrier (ReB) method is used to manage inequality constraints and is integrated into HS-DDP for locomotion planning. The performance of the developed algorithms is benchmarked on a simulation model of the MIT Mini Cheetah executing a bounding gait. We demonstrate the effectiveness of AL and ReB for handling switching constraints, friction constraints, and torque limits. By comparing to previous solutions, we show that the STO algorithm achieves 2.3 times more reduction of total switching times, demonstrating the efficiency of our method.**

I. Introduction

Many tasks in agriculture, construction, defense, and disaster response require mobile robots to traverse irregular terrains and move through narrow passages. The mobility afforded by legged robots makes them exceptionally suitable for these scenarios. Practical challenges to unlock their mobility include the highly nonlinear and hybrid nature of their multi-contact dynamics, a need for on-the-fly generation of motion plans, and the management of various constraints.

Despite these difficulties, many successful algorithms have been developed and tested in simulation and on hardware [1]–[7]. Conventional approaches often optimize the Center of Mass (CoM) trajectory and foothold locations using a reduced-order model and adopt QP-based operational space control (OSC) laws [3]–[5] to select joint torques that track the planned trajectories. Widely used reduced-order models include the Linear Inverted Pendulum (LIP) [2] and the Spring-Loaded Inverted Pendulum (SLIP) [3], [4] for determining foothold locations, with the Zero-Moment Point (ZMP) criterion used to enforce admissible CoM trajectories [1]–[3]. Centroidal dynamics models have also been used that consider the linear and angular momentum of the system as a whole [6]–[8]. Overall, these approaches have the advantage of fast computation, but the complexity of the resulting motions is limited. For example, motions such as standing up from the ground cannot be generated with a LIP model since it neglects all kinematics constraints and assumes constant height and zero angular momentum.

This work was supported by NSF Grant CMMI-1835186.

He Li and Patrick Wensing are with the Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (hli25@nd.edu, pwensing@nd.edu)

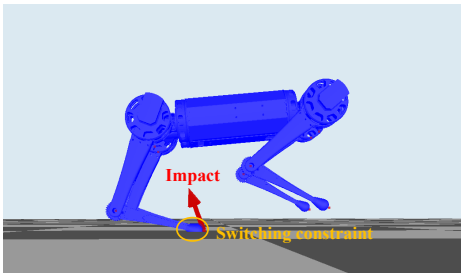

Fig. 1. Mini Cheetah bounding. This paper develops coordinated advances to the Differential Dynamic Programming (DDP) algorithm for trajectory optimization in hybrid systems. In particular, the methods focus on handling the impact event, the associated switching constraints, and the inequality constraints such as torque limits and friction constraints.

By comparison, whole-body motion planning can generate more complex behaviors. Whereas QP-based OSC only considers the instantaneous effects of joint torques, whole-body motion planning finds a sequence of torques by solving a finite-horizon trajectory optimization (TO) problem, potentially enabling recovery from larger disturbances. Despite the appeal of this approach, the curse of dimensionality caused by the high-dimensional state space of legged robots has prevented it from being popular. Recent results (e.g., [9]) using Differential Dynamic Programming (DDP) [10] have shown great promise for online use. Many complementary DDP advances have been proposed, demonstrating robustness for disturbance rejection [11] and real-time performance for whole-body motion planning [12]–[15].

Unlike conventional direct methods, which optimize over all decision variables together, DDP adopts a *divide and conquer* strategy by successively solving much smaller optimization problems [10]. This feature makes DDP exceptionally suitable for problems with long time horizons because the computational effort scales linearly with time as opposed to quadratic or cubic growth with many nonlinear programming (NLP) approaches to TO. Since DDP is a shooting method, the algorithm can also be terminated at any time while still giving a physically valid trajectory. These features and the successes of [13]–[15] together suggest the promise of DDP for online MPC over other direct methods.

Despite these benefits and promise, there are some difficulties for DDP to be used in legged locomotion planning, such as dealing with the impact discontinuity and managing various constraints. The first difficulty is addressed in [9] by approximating the impact discontinuity with a smooth transition, and in [13] by ignoring the impact in DDP but compensating for this simplification with a feedback

controller. These approaches either do not have experimental evidence or present a robustness issue. Other previous work has contributed to attacking the second difficulty by leveraging constraint-handling techniques from NLP. Box input constraints are handled by solving QPs with a Projected Newton algorithm in [16]. A penalty method is used in [13] to satisfy state constraints. This method, however, has a numerical ill-conditioning problem that results when penalty coefficients are large. Augmented Lagrangian (AL) methods (e.g., [17]) resolve this issue by adding a linear multiplier term. Lantoine et al. [18] proposed a DDP algorithm that handles terminal state constraints using AL, motivating their use to address the state-based switching for hybrid systems in this work.

In this paper, we propose a Hybrid Systems DDP (HS-DDP) approach that extends the applicability of DDP to hybrid systems. In particular, HS-DDP includes three algorithmic advances: an impact-aware DDP step that addresses impact discontinuities, an AL method for switching constraints, and a switching time optimization (STO) strategy. Further, in order to deal with the many inequality constraints in legged locomotion, a relaxed barrier (ReB) method [19], [20] is adopted and is integrated within HS-DDP. The developed algorithms are benchmarked in simulation on Mini Cheetah bounding, as shown in Fig. 1. The developed algorithms are extendable to general gaits such as trotting and galloping etc., and to other platforms such as bipeds and manipulators.

The structure of this paper is as follows. DDP background and the hybrid dynamics formulation are given in Sections II and III. Section IV discusses the main contributions of this letter, which extend DDP to hybrid systems. Section V analyzes the performance of the proposed algorithm in terms of constraint handling and efficiency of the STO as applied to quadruped bounding. Section VI provides a closing discussion.

II. Background: Differential Dynamic Programming

This section gives a brief introduction to DDP following [9]. Readers are referred to [10] for detailed derivation. The goal of DDP is to find an optimal control sequence $\mathbf{U}^* = \{\mathbf{u}_k^*\}_{k=0}^{N-1}$ that minimizes a cost function J of the form

$$J(\mathbf{U}) = \sum_{k=0}^{N-1} L(\mathbf{x}_k, \mathbf{u}_k) + \Phi(\mathbf{x}_N) \tag{1}$$

where $\{\mathbf{x}_k\}_{k=0}^{N}$ denotes the state trajectory, L denotes the running cost, and Φ denotes the terminal cost. The trajectory $\{\mathbf{x}_k\}_{k=0}^{N}$ is subject to the discretized dynamics

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{2}$$

where $\mathbf{x}$ and $\mathbf{u}$ respectively denote the state and control variables. DDP recursively finds $\mathbf{U}^*$ by repeatedly executing a forward sweep and a backward sweep. Given a nominal control sequence, the forward sweep computes a nominal trajectory and the associated dynamics derivatives. A backward sweep is then executed to generate a policy that is used to update the control sequence. As this process continues, the control sequence (locally) converges to $\mathbf{U}^*$. Since DDP optimizes only over the control sequence, it can be classified as a direct shooting method. Interested readers may refer to [21] for a discussion of tradeoffs with other direct methods.

Denote $V(\mathbf{x}_k)$ the value function (i.e., optimal cost-to-go) at time step k. Using Bellman's principle of optimality, $V(\mathbf{x}_k)$ is given recursively backward in time:

$$V(\mathbf{x}_k) = \min_{\mathbf{u}_k} [\underbrace{L(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1})}_{Q(\mathbf{x}_k, \mathbf{u}_k)}]. \tag{3}$$

Attempts to solve (3) directly are difficult since an analytical expression for $V(\mathbf{x}_k)$ is rarely possible due to nonlinearity of $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$. To avoid this problem, DDP considers the variation of $Q(\mathbf{x}_k, \mathbf{u}_k)$ around a nominal state-control pair $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$ under the perturbation $(\delta\mathbf{x}, \delta\mathbf{u})$. The resulting variation $\delta Q(\delta\mathbf{x}, \delta\mathbf{u})$ is approximated to the second order as:

$$\delta Q(\delta\mathbf{x}, \delta\mathbf{u}) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^T \begin{bmatrix} 0 & \mathbf{Q}_\mathbf{x}^T & \mathbf{Q}_\mathbf{u}^T \\ \mathbf{Q}_\mathbf{x} & \mathbf{Q}_{\mathbf{xx}} & \mathbf{Q}_{\mathbf{ux}}^T \\ \mathbf{Q}_\mathbf{u} & \mathbf{Q}_{\mathbf{ux}} & \mathbf{Q}_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}, \tag{4}$$

where

$$\mathbf{Q}_\mathbf{x} = \mathbf{L}_\mathbf{x} + \mathbf{f}_\mathbf{x}^T \mathbf{V}'_\mathbf{x}, \tag{5a}$$
$$\mathbf{Q}_\mathbf{u} = \mathbf{L}_\mathbf{u} + \mathbf{f}_\mathbf{u}^T \mathbf{V}'_\mathbf{x}, \tag{5b}$$
$$\mathbf{Q}_{\mathbf{xx}} = \mathbf{L}_{\mathbf{xx}} + \mathbf{f}_\mathbf{x}^T \mathbf{V}'_{\mathbf{xx}} \mathbf{f}_\mathbf{x} + \mathbf{V}'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{xx}}, \tag{5c}$$
$$\mathbf{Q}_{\mathbf{uu}} = \mathbf{L}_{\mathbf{uu}} + \mathbf{f}_\mathbf{u}^T \mathbf{V}'_{\mathbf{xx}} \mathbf{f}_\mathbf{u} + \mathbf{V}'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{uu}}, \tag{5d}$$
$$\mathbf{Q}_{\mathbf{ux}} = \mathbf{L}_{\mathbf{ux}} + \mathbf{f}_\mathbf{u}^T \mathbf{V}'_{\mathbf{xx}} \mathbf{f}_\mathbf{x} + \mathbf{V}'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{ux}}, \tag{5e}$$

in which the subscripts indicate the partial derivatives and the prime indicates the next time step. Note that $\mathbf{f}_{\mathbf{xx}}$, $\mathbf{f}_{\mathbf{uu}}$ and $\mathbf{f}_{\mathbf{ux}}$ generally are tensors. The notation '$\cdot$' denotes matrix-tensor multiplication. Omitting the third terms in the last three equations gives rise to the iLQR algorithm, which enables faster iterations but loses quadratic convergence properties. We employ iLQR in this work and use the algorithm proposed in [22], [23] to efficiently compute $\mathbf{f}_\mathbf{x}$ and $\mathbf{f}_\mathbf{u}$.

Optimizing $\delta Q(\delta\mathbf{x}, \delta\mathbf{u})$ over $\delta\mathbf{u}$ results in the optimal control increment $\delta\mathbf{u}^*$ around the nominal control $\hat{\mathbf{u}}$ as

$$\delta\mathbf{u}^* = -\mathbf{Q}_{\mathbf{uu}}^{-1}(\mathbf{Q}_\mathbf{u} + \mathbf{Q}_{\mathbf{ux}}\delta\mathbf{x}) = \boldsymbol{\kappa} + \mathbf{K}\delta\mathbf{x}, \tag{6}$$

where $\boldsymbol{\kappa}$ is the step direction and $\mathbf{K}$ is the feedback gain. Substituting (6) into the equation (4) results in update equations for the local quadratic model of V according to

$$\Delta V = \Delta V' + \frac{1}{2}\mathbf{Q}_\mathbf{u}^T \mathbf{Q}_{\mathbf{uu}}^{-1} \mathbf{Q}_\mathbf{u}, \tag{7a}$$
$$\mathbf{V}_\mathbf{x} = \mathbf{Q}_\mathbf{x} - \mathbf{Q}_{\mathbf{ux}}^T \mathbf{Q}_{\mathbf{uu}}^{-1} \mathbf{Q}_\mathbf{u}, \tag{7b}$$
$$\mathbf{V}_{\mathbf{xx}} = \mathbf{Q}_{\mathbf{xx}} - \mathbf{Q}_{\mathbf{ux}}^T \mathbf{Q}_{\mathbf{uu}}^{-1} \mathbf{Q}_{\mathbf{ux}}. \tag{7c}$$

where ΔV denotes the expected cost reduction.

The equations (5) and (7) are computed recursively starting at the final state, constituting the backward pass of DDP. The nominal control is then updated using the resulting control policy (6) as follows,

$$\mathbf{u}_k = \hat{\mathbf{u}}_k + \epsilon\boldsymbol{\kappa} + \mathbf{K}(\mathbf{x}_k - \hat{\mathbf{x}}_k), \tag{8}$$

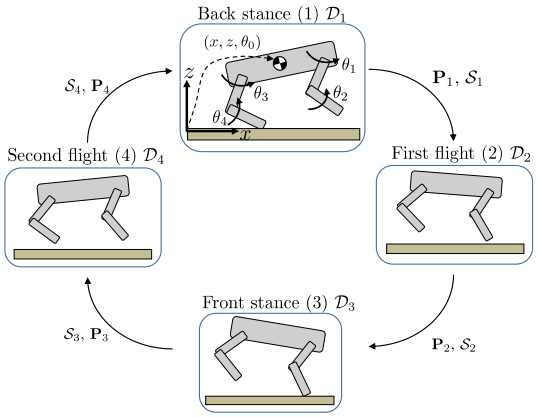

Fig. 2. Mode sequence of a quadruped bounding gait. The gait cycle is assumed to start from the back stance for simplicity of presentation. The generalized coordinates for this 2D quadruped are $\mathbf{q} = [x, z, \theta_0, \theta_1, \theta_2, \theta_3, \theta_4]^T$.

where $0 < \epsilon \leq 1$ is a line search parameter, $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$ and $(\mathbf{x}, \mathbf{u})$ respectively are the nominal and new state-control pair. A backtracking line search method is used to select ϵ [9] and a regularization strategy as in [9] is employed, ensuring a decrease of the cost in each iteration. The forward-backward process above is repeated until the algorithm converges or a certain number of iterations is reached.

III. Background: Dynamics Modeling

This section presents a hybrid system model for bounding quadrupeds. Figure 2 shows one gait cycle of quadruped bounding with four continuous modes and a reset map between every two consecutive modes. Denote $\mathcal{P}(n) = \{1, \cdots, n\}$ the mode sequence where n represents the total number of modes. Then, $\mathcal{P}(4)$ denotes one gait cycle. The continuous dynamics in mode i, denoted by $\bar{\mathbf{f}}_i$, takes place on domain $\mathcal{D}_i$. The reset map $\mathbf{P}_i$ takes place on the switching surface $\mathcal{S}_i$ at the boundary of $\mathcal{D}_i$. Mathematical definitions of $\mathcal{D}_i$ and $\mathcal{S}_i$ are introduced later. Denote $\mathbf{q}$ the generalized coordinates of the quadruped and $\mathbf{x} = [\mathbf{q}^T, \mathbf{v}^T]^T$ the state vector where $\mathbf{v} = \dot{\mathbf{q}}$. The hybrid model is given as

$$\begin{cases} \dot{\mathbf{x}} & = \bar{\mathbf{f}}_i(\mathbf{x}, \mathbf{u}), \ \mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}^+ & = \mathbf{P}_i(\mathbf{x}^-), \ \mathbf{x}^- \in \mathcal{S}_i \end{cases}, \tag{9}$$

where '-' and '+' indicate pre- and post-transition states.

Denote c_i the contact foot in mode i and $\bar{c}_i$ the other foot. During a flight mode, c_i represents the foot scheduled to touch down at the end of flight. The sets $\mathcal{D}_i$ and $\mathcal{S}_i$ are defined for one gait cycle as in Fig. 2,

$$\mathcal{D}_i = \{\mathbf{x} \in \mathcal{TQ} \,|\, g_{c_i}(\mathbf{x}) = 0, \dot{g}_{c_i}(\mathbf{x}) = 0\}, i = 1, 3, \tag{10a}$$
$$\mathcal{D}_i = \{\mathbf{x} \in \mathcal{TQ} \,|\, g_{c_i}(\mathbf{x}) > 0, g_{\bar{c}_i}(\mathbf{x}) > 0\}, i = 2, 4, \tag{10b}$$
$$\mathcal{S}_i = \{\mathbf{x} \in \mathcal{TQ} \,|\, g_{c_i}(\mathbf{x}) = 0, |\dot{g}_{c_i}(\mathbf{x})| \neq 0\}, \forall i, \tag{10c}$$

where $g(\cdot)$ is a function measuring the vertical distance of the corresponding foot to the ground, $\mathcal{TQ}$ denotes the tangent bundle of the configuration space $\mathcal{Q}$. Although the hybrid model (9) and (10) considers one gait cycle for simplicity, it can be extended to multiple gait cycles.

A. Continuous Dynamics

The continuous dynamics in (9) varies depending on which legs are in stance. However, these dynamics can be formulated with a unified structure as follows:

$$\begin{bmatrix} \mathbf{H} & -\mathbf{J}_{c_i}^T \\ -\mathbf{J}_{c_i} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda}_{c_i} \end{bmatrix} = \begin{bmatrix} \mathbf{S}^T\boldsymbol{\tau} - \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}_g \\ \dot{\mathbf{J}}_{c_i}\dot{\mathbf{q}} \end{bmatrix}, \tag{11}$$

where $\mathbf{H}$, $\mathbf{C}\dot{\mathbf{q}}$, $\boldsymbol{\tau}_g$, $\mathbf{S}$, and $\boldsymbol{\tau}$ denote the inertia matrix, Coriolis force, gravity force, selection matrix, and actuation torque, respectively. $\mathbf{J}_{c_i}$ and $\boldsymbol{\lambda}_{c_i}$ represent the contact Jacobian and contact force associated with the contact foot c_i. The matrix on the left side of (11) is known as the KKT matrix, since the equation (11) can be obtained via KKT conditions [12]. When the robot is in flight, $\mathbf{J}_{c_i}$ and $\boldsymbol{\lambda}_{c_i}$ are not meaningful anymore, and the KKT matrix degenerates to the inertia matrix $\mathbf{H}$. The state-space representation of (11) is obtained by pre-multiplying both sides of (11) by the inverse of the KKT matrix and separating out the solution for $\ddot{\mathbf{q}}$.

B. Reset Maps

While the generalized coordinates remain unchanged across impact events, velocities change instantaneously at each *touch down*. The impact dynamics are modeled as

$$\begin{bmatrix} \mathbf{H} & -\mathbf{J}_{c_i}^T \\ -\mathbf{J}_{c_i} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}^+ \\ \hat{\boldsymbol{\lambda}}_{c_i} \end{bmatrix} = \begin{bmatrix} \mathbf{H}\mathbf{v}^- \\ e\mathbf{J}_{c_i}\mathbf{v}^- \end{bmatrix}, \tag{12}$$

where $e \in [0, 1]$ denotes the coefficient of restitution. Perfect inelastic collision with $e = 0$ is assumed in this work, meaning that the contact foot sticks to the ground after impact. The vector $\hat{\boldsymbol{\lambda}}_{c_i}$ denotes the impulse acting on the contact foot that is scheduled to touch down at the end of flight. Note that there is no control present in the model (12) since the actuators cannot generate impulsive outputs. By separating $\mathbf{v}^+$ out, the state-space representation of the reset map at impact is $\mathbf{x}^+ = \mathbf{P}_i(\mathbf{x}^-)$ where

$$\mathbf{P}_i(\mathbf{x}^-) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} - \mathbf{H}^{-1}\mathbf{J}_{c_i}^T(\mathbf{J}_{c_i}\mathbf{H}^{-1}\mathbf{J}_{c_i}^T)^{-1}\mathbf{J}_{c_i} \end{bmatrix} \mathbf{x}^-. \tag{13}$$

Note that the transition from stance to flight is continuous, and, thus, $\mathbf{P}_i(\mathbf{x}^-) = \mathbf{x}^-$ when i denotes a stance phase.

C. Time-Switched Hybrid System

We associate the pre-determined mode sequence $\mathcal{P}(n)$ with a switching time vector $\mathbf{t} = [t_1, \cdots, t_n]$ where t_i represents the terminating time of the i^{th} mode. Along any trajectory of the state-switched hybrid system, (9) and (10) are equivalent to the time-switched hybrid system:

$$\begin{cases} \dot{\mathbf{x}}(t) & = \bar{\mathbf{f}}_i(\mathbf{x}(t), \mathbf{u}(t)), \ t \in [t_{i-1}^+, t_i^-] \\ \mathbf{x}^+(t) & = \mathbf{P}_i(\mathbf{x}^-(t)), \ t \in [t_i^-, t_i^+] \end{cases}, \tag{14}$$

with the enforced switching constraint:

$$g_{c_i}(\mathbf{x}(t_i^-)) = 0. \tag{15}$$

In this work, this time-switched reformulation is considered, where variables t_i are optimized under switching constraints.

IV. Hybrid Systems Differential Dynamic Programming

This section discusses three algorithmic advances for HS-DDP and presents the ReB method for inequality constraints. An overview of HS-DDP and ReB is shown in Fig. 3. The HS-DDP algorithm takes a two-level optimization strategy. In the bottom level, the switching times are fixed and the AL algorithm is executed. This algorithm continuously calls the impact-aware DDP. Once DDP converges, the constraint violations are remeasured and added to the cost function, and another DDP call is executed. The AL algorithm terminates when all switching constraints are satisfied. The output from this loop is then utilized by the STO algorithm to update the switching times. This process repeats until the switching times are optimal. The ReB algorithm is executed whenever the AL algorithm is executed. The entire framework is presented to plan trajectories for the quadruped bounding model introduced in the previous section.

A. Whole-body Motion Planning Problem

To find an optimal trajectory, we formulate a TO problem

$$\min_{\mathbf{u}(\cdot),\mathbf{t}} \sum_{i=1}^{n} \left[\int_{t_{i-1}^{+}}^{t_i^{-}} l_i(\mathbf{x}(t),\mathbf{u}(t))\mathrm{d}t + \Phi_i(\mathbf{x}(t_i^{-})) \right] \tag{16}$$

where l_i and Φ_i respectively denote the continuous-time running cost and the terminal cost for the i^{th} mode. In whole-body TO, the minimization of (16) is subject to the full-order dynamics (14) and other various constraints. A common way to solve (16) is to formulate a discrete-time optimal control problem (OCP) with integration time step h as follows

$$\begin{aligned}
\min_{\mathbf{U},\mathbf{t}} \quad & J(\mathbf{U},\mathbf{t}) && \text{(17a)}\\
\text{subject to} \quad & \mathbf{x}_{k+1} = \mathbf{f}_i(\mathbf{x}_k,\mathbf{u}_k), && \text{(17b)}\\
& \mathbf{x}_{N_i}^{+} = \mathbf{P}_i(\mathbf{x}_{N_i}^{-}), && \text{(17c)}\\
& g_{c_i}(\mathbf{x}_{N_i}^{-}) = 0, && \text{(17d)}\\
& |u_{k,j}| \le u_{\max}, && \text{(17e)}\\
& \lambda_{c,z} \ge 0, && \text{(17f)}\\
& |\lambda_{c,x}| \le \mu\lambda_{c,z}, && \text{(17g)}
\end{aligned}$$

where

$$J(\mathbf{U},\mathbf{t}) = \sum_{i=1}^{n} \left(\Phi_i(\mathbf{x}_{N_i}^{-}) + \sum_{k=N_{i-1}}^{N_i-1} L_i(\mathbf{x}_k,\mathbf{u}_k) \right), \tag{18}$$

and $L_i = hl_i$ approximates the integral of the running cost over integration time step h, and $N_i = \frac{t_i}{h}$ denotes the number of time steps in the time horizon up to the i^{th} mode. Equations (17b) and (17c) represent the dynamics and reset map constraints, respectively, where $\mathbf{f}_i$ is obtained via forward integration of $\bar{\mathbf{f}}_i$. This work uses a forward Euler method, but all algorithmic advances hold with other integration schemes. Equation (17d) represents the switching constraint, and inequalities (17e)-(17g) represent the torque limit, non-negative normal GRF, and friction cone constraint, respectively.

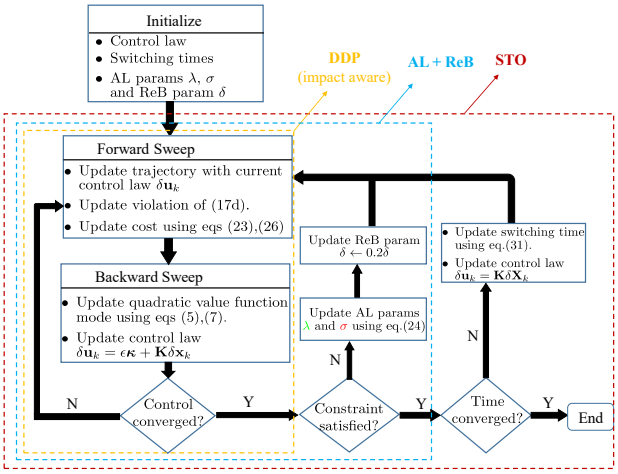

Fig. 3. Overview of the HS-DDP algorithmic framework.

B. Impact-Aware Value Function Update in DDP

Impact-aware DDP extends DDP to address the impact effect, but does not consider constraints (17d) - (17g). While the impact-aware DDP executes the same forward sweep as DDP, it modifies the update equations (7) for the quadratic value function model at the switching surface. Suppose that ΔV, $\mathbf{V_x}$, and $\mathbf{V_{xx}}$ are known at t_i^{+}, which can be computed from DDP. The dependency of all variables on i is ignored here for simplicity. Since there is no control applied at t_i^{-}, according to Bellman's Principle of Optimality

$$V(\mathbf{x}^{-}) = \Phi(\mathbf{x}^{-}) + V(\mathbf{x}^{+}). \tag{19}$$

Since $\mathbf{x}^{-}$ and $\mathbf{x}^{+}$ can be computed from the forward sweep, the variation of (19) around $\mathbf{x}^{-}$ and $\mathbf{x}^{+}$ is considered, i.e.,

$$V(\mathbf{x}^{-}+\delta\mathbf{x}^{-}) = \Phi(\mathbf{x}^{-}+\delta\mathbf{x}^{-}) + V(\mathbf{x}^{+}+\delta\mathbf{x}^{+}), \tag{20}$$

where

$$\delta\mathbf{x}^{+} = \mathbf{P}(\mathbf{x}^{-}+\delta\mathbf{x}^{-}) - \mathbf{P}(\mathbf{x}^{-}) \approx \mathbf{P_x}\delta\mathbf{x}^{-} \tag{21}$$

in which $\mathbf{P_x}$ is the Jacobian of $\mathbf{P}$ with respect to $\mathbf{x}^{-}$. Approximating both sides of (20) to the second order, we obtain

$$\begin{aligned}
\Delta V^{-} &= \Delta V^{+}, && \text{(22a)}\\
\mathbf{V}_{\mathbf{xx}}^{-} &\approx \Phi_{\mathbf{xx}^{-}} + \mathbf{P}_{\mathbf{x}}^{T}\mathbf{V}_{\mathbf{xx}}^{+}\mathbf{P}_{\mathbf{x}}, && \text{(22b)}\\
\mathbf{V}_{\mathbf{x}}^{-} &= \Phi_{\mathbf{x}^{-}} + \mathbf{P}_{\mathbf{x}}^{T}\mathbf{V}_{\mathbf{x}}^{+}. && \text{(22c)}
\end{aligned}$$

The equations (22) establish the model update equations at the switching surface, which, together with (7), constitute the model update equations of V for hybrid systems.

C. Augmented Lagrangian for Switching Constraints

The impact-aware DDP solves unconstrained optimization problems. Nevertheless, it can be combined with various constraint-handling techniques from NLP for constrained optimization. In this section, we are particularly interested in the switching equality constraint (17d). Penalty methods [13] to manage this constraint add a squared term of the constraint violation to the cost function. However, a numerical ill-conditioning issue could happen as the penalty increases. An

Algorithm 1 Pseudocode combining AL and ReB

Given
Mode sequence $\mathcal{P}(n)$ and switching time $\mathbf{t}$.
Cost function, dynamics, and switching constraints in (17).
Initial control sequence $\mathbf{U}$ (e.g., zeros).
Initialization
AL parameters σ, λ_i, β, and ReB parameters ϵ_B, δ
Run DDP to convergence and compute g_{c_i}.
while $\sum g_{c_i}^2(\mathbf{x}_{N_i}^-) > \epsilon_{AL}$ **do**
    Update $\sigma \leftarrow \beta\sigma$, $\lambda \leftarrow \lambda + \sigma g_{c_i}$, $\delta \leftarrow 0.2\delta$.
    Update (23) and (26).
    Update initial guess for DDP.
    Run DDP to convergence.
    Compute $g_{c_i}(\mathbf{x}_{N_i}^-)$.
end while

AL method is employed in this work, which, in addition to the quadratic term, adds a linear Lagrange multiplier term to the cost function, avoiding the numerical ill-conditioning.

With the AL technique, the cost function now becomes

$$J(\mathbf{U},\mathbf{t}) + \left(\frac{\sigma_\eta}{2}\right)^2 \sum_{i\in\mathcal{I}_c} g_{c_i}^2(\mathbf{x}_{N_i}^-) + \sum_{i\in\mathcal{I}_c} \lambda_{\eta,i} g_{c_i}(\mathbf{x}_{N_i}^-) \quad (23)$$

where $\mathcal{I}_c$ denotes the set of all touch down indices, σ and λ denote the penalty and the Lagrange multipliers, respectively. The subscript 'η' is the AL iteration. The AL approach begins with certain initial values for σ and λ, and solves the resulting TO problem using impact-aware DDP. The parameters σ and λ are then updated and the new TO problem is re-solved using the previous optimal control as a warm start. The update equations are

$$\sigma_{\eta+1} = \beta\sigma_\eta \quad \text{and} \quad \lambda_{\eta+1,i} = \lambda_{\eta,i} + \sigma_\eta g_{c_i}(\mathbf{x}_{N_i}^-), \quad (24)$$

where $\beta > 1$ is the penalty update parameter. This process is repeated until g_{c_i} is within the threshold ϵ_{AL}. To make a distinction, one execution of the forward sweep and backward sweep of DDP is called one DDP iteration. Pseudocode for the AL algorithm is shown in Algorithm 1.

D. Relaxed Barrier Function for Inequality Constraints

We employ a relaxed barrier (ReB) method [19], [20] to manage the inequality constraints in (17). Given any inequality constrained optimization problem as below

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{subject to} \quad & c_j(\mathbf{x}) \geq 0, j = 1, \cdots, m, \end{aligned} \quad (25)$$

ReB attacks (25) by successively solving the unconstrained optimization

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + \epsilon_B \sum_{j=1}^{m} B_\delta(c_j(\mathbf{x})), \quad (26)$$

where $\epsilon_B > 0$ is a weighting parameter and

$$B_\delta(z) = \begin{cases} -\log(z) & z > \delta \\ \frac{k-1}{k}\left[\left(\frac{z-k\delta}{(k-1)\delta}\right)^k - 1\right] - \log\delta & z \leq \delta \end{cases}, \quad (27)$$

is called a ReB function where $\delta > 0$ is the relaxation parameter. The function $B_\delta(z)$ smoothly extends the logarithmic barrier function $-\log(z)$ over the entire real domain with a polynomial of order k. In many cases, $k = 2$ works well [19]. Consequently, when applied to a TO problem, the ReB method allows the objective function to be evaluated for an infeasible trajectory, which cannot be done with a standard barrier method. Note that δ is updated toward zero in an outer loop. This drives the resulting trajectory toward feasibility.

With this technique, the inequality constraints (17e) - (17g) are turned into ReB functions and added to the objective function $J(\mathbf{U},\mathbf{t})$. Combing this technique with AL, the constrained TO problem (17) is converted into an unconstrained optimization problem, which is solved using the impact-aware DDP. The AL parameters λ, σ and the ReB parameter δ are updated in an outer loop as shown in Algorithm 1.

E. Switching time optimization based on DDP

While Algorithm 1 finds the optimal control $\mathbf{U}^*$ for the OCP (17) (equivalently (16)) for fixed $\mathbf{t}$, the switching time optimization (STO) algorithm developed in this section updates $\mathbf{t}$ toward an optimal value under a fixed control policy (6). This approach is different from [24], where the control sequence $\mathbf{U}$ is fixed without feedback from the state.

The STO algorithm reformulates the OCP (16) on fixed time intervals of length one, and augments the state vector with an extra state representing the time span of each mode. Denote $\mathbf{z}$ the time state, $\bar{\mathbf{x}}$ the scaled system state, and $\mathbf{X} = [\bar{\mathbf{x}}^T, \mathbf{z}^T]^T$ the augmented state. Then, Algorithm 1 can be used to find $\mathbf{V}_{\bar{\mathbf{x}}}$, $\mathbf{V}_{\bar{\mathbf{x}}\bar{\mathbf{x}}}$, $\mathbf{V}_{\mathbf{z}}$, $\mathbf{V}_{\mathbf{zz}}$, and $\mathbf{V}_{\bar{\mathbf{x}}\mathbf{z}}$ in the backward sweep. Following the convergence of Algorithm 1, the values of $\mathbf{V}_{\mathbf{z}}$ and $\mathbf{V}_{\mathbf{zz}}$ are then used to update the switching times using Newton's method.

We first discuss the reformulation of the OCP (16) on fixed time intervals and then the derivation of the STO under the new formulation. Let $T_i = t_i - t_{i-1}$ and $\mathbf{z} = [T_1, \cdots, T_n]^T$. With the change of variable $\tau = \frac{t-t_{i-1}}{T_i} + i - 1$, time-scaled dynamics are obtained as

$$\begin{cases} \dot{\bar{\mathbf{x}}}(\tau) & = T_i \bar{\mathbf{f}}_i(\bar{\mathbf{x}}(\tau), \mathbf{u}(\tau)),\ \tau \in [(i-1)^+, i^-] \\ \bar{\mathbf{x}}^+(\tau) & = \mathbf{P}_i(\bar{\mathbf{x}}^-(\tau)),\ \tau \in [i^-, i^+] \\ \dot{\mathbf{z}}(\tau) & = 0,\ \tau \in [0, n], \end{cases} . \quad (28)$$

with the switching constraint

$$g_{c_i}(\bar{\mathbf{x}}(i^-)) = 0\,. \quad (29)$$

The timing state $\mathbf{z}$ has the initial condition $\mathbf{z}(0) = \mathbf{z}_0$. The cost function in the OCP (16) now becomes

$$\sum_{i=1}^{n} \left[\int_{(i-1)^+}^{i^-} T_i l_i(\bar{\mathbf{x}}(\tau), \mathbf{u}(\tau)) \mathrm{d}\tau + \Phi_i(\bar{\mathbf{x}}(i^-)) \right]. \quad (30)$$

We can now apply Algorithm 1 to minimize (30) under the fixed initial condition $\mathbf{z}(0)$. Once Algorithm 1 converges, it implies that 1) the control sequence and trajectory are (locally) optimal and 2) the quadratic model of V is a valid approximation of V. The gradient $\mathbf{V}_{\mathbf{z}}$ and Hessian $\mathbf{V}_{\mathbf{zz}}$ are

Algorithm 2 STO algorithm

Given
Mode sequence $\mathcal{P}(n)$.
Cost function (30), scaled dynamics (28) and switching constraints (29).
Initialization
Initialize control sequence $\mathbf{U}$ and time state $\mathbf{z}_0$.
Execution
Execute Algorithm 1 to obtain optimal control $\mathbf{U}^*$, feedback gain $\mathbf{K}$ in (6), $\mathbf{V_z}$ and $\mathbf{V_{zz}}$.
Line search using $\mathbf{z}(0) = \mathbf{z}(0) - \epsilon_z \mathbf{V}_{\mathbf{zz}}^{-1}(0)\mathbf{V}_{\mathbf{z}(0)}$.

then obtained from the quadratic model. Since $\mathbf{z}$ only affects the dynamics via its initial condition, $\mathbf{z}(0)$ is updated using

$$\mathbf{z}_{\text{new}}(0) = \mathbf{z}_{\text{old}}(0) - \epsilon_z \mathbf{V}_{\mathbf{zz}}^{-1}(0)\mathbf{V}_{\mathbf{z}}(0). \tag{31}$$

where $0 < \epsilon_z \leq 1$ denotes the switching time line search parameter. Similar to DDP, we perform a backtracking line search to select ϵ_z and ensure cost reduction with (31).

Algorithms 1 and 2 are combined to solve the OCP (17) simultaneously for optimal $\mathbf{U}^*$ and $\mathbf{t}^*$, as shown in Fig. 3, constituting HS-DDP. Note that the STO algorithm is executed after the AL algorithm converges, which implies that the feedforward term in equation (8) becomes zero, and, thus, the control policy $\mathbf{u}_k = \hat{\mathbf{u}}_k + \mathbf{K}\delta\mathbf{X}_k$ is used in the line search for the timing variables and in the next forward sweep. The major difference between our method and the approach in [24] is the inclusion of this feedback term in the control law. The control policy used in this work allows (31) to make more aggressive updates, and consequently achieves faster convergence. The reason behind this is that any change in $\mathbf{z}(0)$ will create perturbations to the locally optimal trajectory. The effect of the change in $\mathbf{z}(0)$ on optimality is reduced by including the feedback term in control to account for perturbations. More details on this aspect are discussed in Sec. V-D.

V. Results: Bounding with a 2D Quadruped

A. Model and Simulation

The developed HS-DDP algorithm is tested on a 2D model of the simulated MIT Mini Cheetah [25], as in Fig. 2. We consider two trajectory optimization tasks. The first task fixes the switching times and applies Algorithm 1 on quadruped bounding for five gait cycles. We compare the results with those when AL + ReB is disabled and demonstrate satisfaction of constraints (17d) - (17g) within four AL iterations. The second task applies the HS-DDP to quadruped bounding for one gait cycle and demonstrates the efficiency of the STO.

B. Five Gait Cycle Bounding with AL and ReB

In this task, Algorithm 1 is applied to 2D quadruped bounding for five gait cycles. The robot starts in the back-stance mode and is desired to run at an average forward speed of 1.0 m/s. A constant reference configuration is assigned to each mode, which mimics the robot's posture at the end of the mode and is selected heuristically. All desired joint velocities and the desired body vertical velocity are set to zero.

Quadratic running cost and terminal cost are used in (16),

$$l_i(\mathbf{x}, \mathbf{u}) = (\mathbf{x} - \mathbf{x}_{\text{ref},i})^T \mathbf{Q}_i (\mathbf{x} - \mathbf{x}_{\text{ref},i}) + \mathbf{u}^T \mathbf{R}_i \mathbf{u}, \tag{32}$$

$$\Phi_i(\mathbf{x}(t_i)) = (\mathbf{x}(t_i) - \mathbf{x}_{\text{ref},i})^T \mathbf{Q}_{fi} (\mathbf{x}(t_i) - \mathbf{x}_{\text{ref},i}), \tag{33}$$

where $\mathbf{Q}_i$ and $\mathbf{R}_i$ are weighting matrices for state deviation and energy consumption in running cost, respectively, and $\mathbf{Q}_{fi}$ is the weighting matrix for the terminal cost (of the i^{th} mode). In this simulation, we have zero penalty on forward position, and relatively larger penalty on forward speed, body height, and joint velocities than the other states. The integration time step $h = 1$ ms is used, and the switching times are selected such that the flight mode (and the front-stance mode) runs for 72 ms and the back stance mode runs for 80 ms. The initial guess for Algorithm 1 is given by a heuristic controller, which implements the PD control in flight mode such that a predefined joint configuration is maintained. In stance mode, the heuristic controller constructs stance leg forces following a SLIP model and converts the Ground Reaction Force (GRF) thus obtained to joint torques. The AL and ReB parameters are initialized as $\sigma = 5$, $\lambda_i = 0$, $\beta = 8$, and $\sigma = 0.5$, and the convergence threshold is set to $\epsilon_{AL} = 10^{-4}$.

C. AL and ReB Simulation Results

When AL is active and ReB is disabled, it takes three AL iterations for the constraint violation to decrease within ϵ_{AL}. The convergence of the total cost (excluding the penalty term and the Lagrangian term) and switching constraint violation are shown in Fig. 4. The blue square markers and the red circle markers indicate the beginning of the corresponding AL iteration. Figure 4 demonstrates that at least one of the total cost and the constraint violation is reduced at every DDP iteration. Further, the algorithm spends more effort in minimizing the total cost at the beginning and switches to the constraint violation after the total cost is converged.

Figure 5 compares the bounding gaits that are generated by three methods: 1) A heuristic controller that is used to warm start the optimization, 2) DDP (with impact-aware value function update) that ignores switching constraints, and 3) AL that enforces switching constraints. It demonstrates

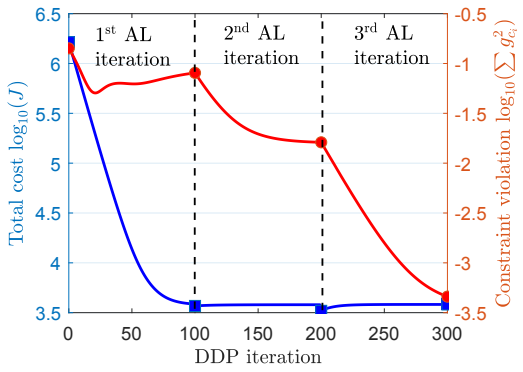

Fig. 4. Convergence of the total cost and constraint violation.

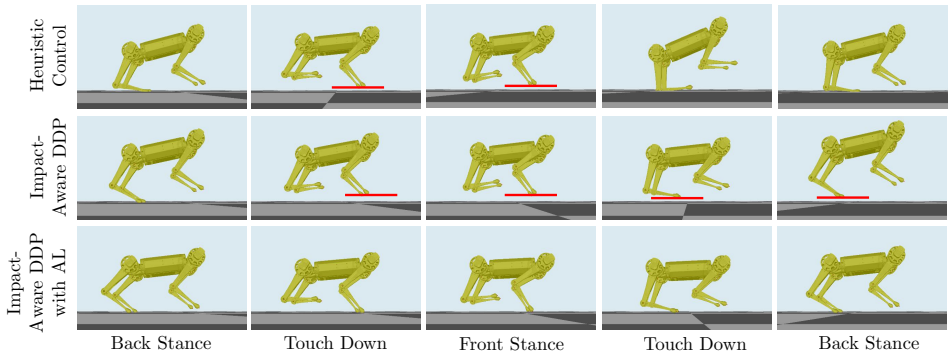

Fig. 5. Sequential snapshots of the generated bounding motion for Mini Cheetah. Top: Motion generated by the heuristic controller that is used to warm start AL. Middle: Motion generated by DDP (without AL) ignoring switching constraints. Bottom: Motion generated with AL enforcing switching constraints. The first two methods incorrectly regard the red lines as the ground, and thus, dynamics are reset on this 'virtual ground'.

that the developed AL algorithm achieves the desired performance. Though the motion generated by DDP is more smooth and realistic compared to the heuristic controller, the robot still violates the switching constraints, and the error accumulates over time. This behavior is because the first two methods do not enforce switching constraints, and thus, the robot does not correctly recognize the ground, but the simulator still resets the dynamics.

Figure 6 depicts the normal and tangential GRF for the front leg (top), and the torques for every joint (bottom) when the ReB is activated. The algorithm terminates at four AL iterations. It shows that the normal GRF is always non-negative and that the friction and joint torques are confined within their boundaries, demonstrating effectiveness of the ReB method. Similar results are observed for the back leg.

D. One-Gait Bounding with Time Optimized

In this task, HS-DDP is applied to the generation of one bounding gait for the Mini Cheetah. Different from the previous task, where only the control is optimized, switching times are also optimized in this task. Only one gait cycle is studied here in the sense that, in many situations, the switching times found for one gait cycle can be extended to the succeeding gait cycles. The cost function, initial control sequence, initial switching times, AL parameters, and terminating conditions all remain the same in this task as in the previous one.

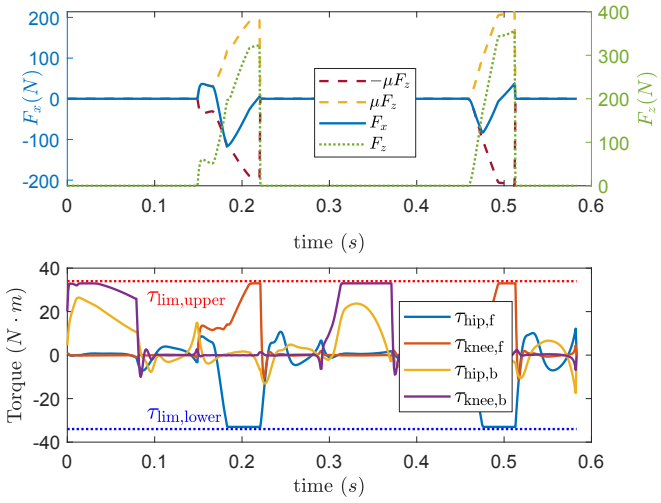

Fig. 6. GRF and joint toques for 2D Mini Cheetah bounding. Top: Normal and tangential GRF for the front leg. Bottom: Joint torques. With AL and ReB, the non-negativity of normal GRF, friction, and torque limit constraints are satisfied in four AL iterations.

E. HS-DDP Simulation Results

The optimal switching times obtained via the STO algorithm in HS-DDP are shown in Fig. 7. The algorithm reduces the time of the first flight mode and the front-stance mode. Figure 7 also compares the switching times obtained via the STO algorithm with the algorithm proposed in [24] where the feedback control is not used. Both algorithms are terminated at the 30^{th} iteration. With HS-DDP, the overall time spent on the entire motion is 0.2335 s, a 21.1% reduction of the initial overall time, whereas only a 6.3% reduction is observed with the algorithm in [24], showing that the HS-DDP is more efficient in the sense of taking larger steps.

Figure 8 explains why the two-level optimization strategy is adopted in HS-DDP. With the scaled optimization structure (28), (29), and (30), it is reasonable to update the control using (8) and the switching times using (31) simultaneously since the gradient and Hessian information are all available in the backward sweep of DDP. If the actual cost reduction is less than zero and is close to the predicted cost reduction, then the quadratic model of the value function is considered valid. The quadratic approximation, however, is more sensitive to the switching time line search parameter ϵ_z than the control line search parameter ϵ, as shown in Fig. 8. This figure indicates that ϵ has to be as small as ϵ_z if (8) and (31) are executed simultaneously, at the price of much less cost

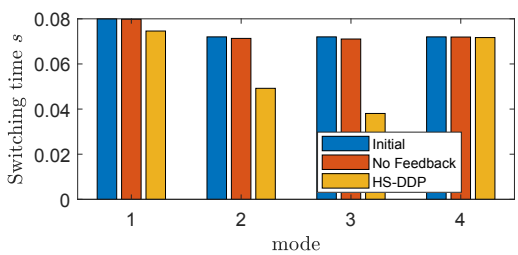

Fig. 7. Time spent in each mode for the one-gait-cycle bounding task.

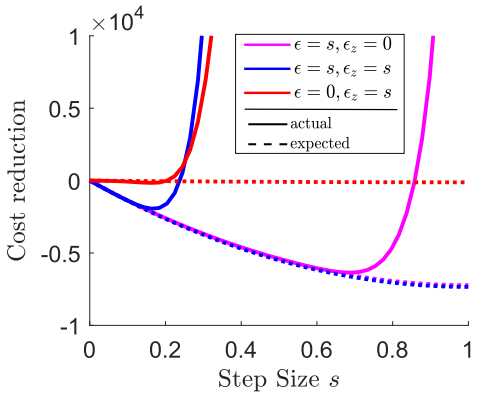

Fig. 8. Change in cost with respect to step size. Solid lines: actual cost reduction. Dashed lines: predicted cost reduction. Red: $\epsilon = 0$, $\epsilon_z = s$. Pink: $\epsilon = s$, $\epsilon_z = 0$. Blue: $\epsilon = \epsilon_z = s$.

reduction per iteration, thus decreasing the convergence rate. Although this behavior is not observed for all iterations, it can significantly slow down the optimization if a small step size is continuously used for multiple iterations.

VI. Conclusions and Future Works

The proposed HS-DDP framework combines three algorithmic advances to DDP for legged locomotion. It addresses the discontinuity at impacts by incorporating an impact-aware value function update in the backward sweep. By combing AL and DDP, HS-DDP reduces either the total cost or the constraint violation in every iteration, enforcing the switching constraint as the algorithm proceeds. Further, with the developed STO algorithm, HS-DDP can efficiently find the optimal switching times alongside the optimal control. A ReB method is combined with HS-DDP to manage the inequality constraints. The five-gait-cycle bounding example shows the promise of HS-DDP in rapidly satisfying the switching constraint in just a few iterations, and demonstrates the effectiveness of ReB for enforcing inequality constraints. The one-gait-cycle bounding example compares the developed STO algorithm to the previous solutions, demonstrating that our method is more efficient due to the inclusion of the feedback control in the forward sweep.

Though forward Euler integration is used in this work for dynamics simulation, the developed HS-DDP is independent of the integration scheme. Implicit or higher-order methods can be used if the computation time is not the primary concern. The current implementation of HS-DDP is MATLAB based, and so future work will benchmark its computational performance with C++ and realize the developed algorithm in experiments for real-time control with the Mini Cheetah. We are also interested in comparing ReB and AL in terms of their abilities for inequality constraint management.

References

[1] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.

[2] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006, pp. 137–142.

[3] T. Apgar, P. Clary, K. Green, A. Fern, and J. W. Hurst, "Fast online trajectory optimization for the bipedal robot Cassie." in *Robotics: Science and Systems*, 2018.

[4] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3D-SLIP model," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 5134–5140.

[5] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1047–1062, 2014.

[6] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 1–9.

[7] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. robots*, vol. 40, no. 3, pp. 429–455, 2016.

[8] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Auton. robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[9] 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*, 2012, pp. 4906–4913.

[10] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[11] J. Morimoto, G. Zeglin, and C. G. Atkeson, "Minimax differential dynamic programming: Application to a biped walking robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 2, 2003, pp. 1927–1932.

[12] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2018, pp. 1–9.

[13] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 93–100.

[14] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 3346–3351.

[15] M. Neunert, M. Stäuble, 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*, vol. 3, no. 3, pp. 1458–1465, 2018.

[16] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Int. Conf. on Robotics and Automation*, 2014, pp. 1168–1175.

[17] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: a fast solver for constrained trajectory optimization," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 7674–7679.

[18] G. Lantoine and R. P. Russell, "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: theory," *Journal of Optimization Theory and Applications*, vol. 154, no. 2, pp. 382–417, 2012.

[19] J. Hauser and A. Saccon, "A barrier function method for the optimization of trajectory functionals with constraints," in *IEEE Conf. on Decision and Control*, 2006, pp. 864–869.

[20] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 4730–4737.

[21] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[22] A. Jain and G. Rodriguez, "Linearization of manipulator dynamics using spatial operators," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 1, pp. 239–248, 1993.

[23] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and Systems*, 2018.

[24] X. Xu and P. J. Antsaklis, "A dynamic programming approach for optimal control of switched systems," in *IEEE Conf. on Decision and Control*, vol. 2, 2000, pp. 1822–1827.

[25] B. Katz, J. Di Carlo, and S. Kim, "Mini Cheetah: A platform for pushing the limits of dynamic quadruped control," in *IEEE Int. Conf. on Robotics and Automation*, 2019, pp. 6295–6301.