Robust Hybrid Systems Differential Dynamic Programming for Worst-Case Disturbance

Pranit Mohnot*
University of California, Berkeley
Berkeley, USA
pranit_mohnot@berkeley.edu

Akshay Thirugnanam*
University of California, Berkeley
Berkeley, USA
akshay_t@berkeley.edu

Jason J. Choi*
University of California, Berkeley
Berkeley, USA
jason.choi@berkeley.edu

Koushil Sreenath
University of California, Berkeley
Berkeley, USA
koushils@berkeley.edu

Abstract

Differential Dynamic Programming (DDP) is an optimal control algorithm that takes iterative Newton steps to solve Bellman equations with quadratic approximations. While DDP has been adapted for hybrid dynamical systems such as legged robots, current methods do not address the uncertainties in reset events such as ground contacts and impacts. In this paper, we propose Robust Hybrid DDP (RH-DDP), an extension of the DDP algorithm for hybrid systems that ensures robustness against worst-case bounded disturbances during reset events. We employ log-sum-exponent functions to approximate the worst-case value function, solving the robust optimal control problem with our modified DDP algorithm. We show that for linear systems, our method results in a cost reduction up to a constant in each iteration. Through simulations, we demonstrate the effectiveness of RH-DDP in generating robust trajectories for a linear system and a simple biped robot subjected to an uncertain ground impact.

Keywords: Differential Dynamic Programming, Robust Control, Optimal Control, Legged Robots, Hybrid Systems

1 Introduction

1.1 Motivation

Legged robots are complex hybrid dynamical systems that engage in complex interactions with their environments characterized by discrete events such as ground contacts and impacts. These interactions are often difficult to model accurately, especially given the presence of uncertainties that affect the reset of the system states, such as uncharacterized geometry and compliance of terrains or varying friction coefficients. Such uncertainties can significantly impact the robot's stability and balance. As a result, planning and control algorithms for legged robots must account for these uncertainties to ensure robust and stable performance.

Differential Dynamic Programming (DDP) [13, 23] is a renowned optimal control technique employed extensively in robot motion planning and control problems. By approximating the costs and dynamics as second order functions in state and input, DDP uses iterative Newton steps to converge to a solution [7]. Its efficacy has been demonstrated across a diverse range of applications, encompassing aerial path planning [11] and robotic manipulation [34]. DDP has been further applied to contact-rich systems like legged robots [29], but its capability to deal with model errors or uncertainties associated with contact events remains limited. While explicit treatment of the impact event in the DDP algorithm has been introduced in [14, 17], the methods in these works assume accurate model of the impact dynamics. This paper introduces a novel methodology that enhances the existing DDP algorithm for hybrid systems with reset events [17]. Our approach explicitly accounts for the *worst-case* perturbations and uncertainties during reset events, thereby yielding a solution that stands robust in the face of uncertainties inherent to such events.

1.2 Related Works

Numerous methods are developed based upon various control theory paradigms, including numerical dynamic programming [4, 6], hybrid zero dynamics [27, 32], and Lyapunov stability analysis [1, 25]. Recently, deep reinforcement learning-based approaches have shown successful demonstrations of legged robots coping with various uncertainties in their contact interactions with the environments [16, 20, 26].

In this work, we focus on optimal control-based techniques, which offer constructive and interpretable methods for the synthesis of robust trajectory plans and feedback policies. Within the optimal control literature, there still exists a plethora of various kinds of solution methods, including indirect methods, multiple shooting, direct collocation, and DDP, each extensively surveyed in [31]. The main strengths of DDP compared to the other methods are twofold. First, it shows a fast quadratic convergence to a locally optimal solution due to its resemblance to Newton's method [7, 21]. Second, the DDP not only provides the optimal trajectory but

*The first three authors have contributed equally to the paper.

also provides a linear feedback policy around the solution, which can be used online to track the trajectory. Thus, DDP has been deployed successfully not only for offline trajectory optimization but also for online feedback control in a predictive control paradigm [19, 29].

DDP algorithms have been applied to whole-body motion planning and legged robot locomotion problems, as seen in works such as [5, 22, 29], to name just a few. The methods in [5, 29] employ smooth contact models to facilitate the computation of contact-constrained dynamics; however, these smooth contact models are inherently susceptible to modeling errors. The approach in [22] handles the Bellman update with respect to the contact-constrained dynamics and the impact dynamics indifferently, failing to explicitly address the potential state discontinuity induced by impact dynamics. Although the explicit treatment of impact dynamics in DDP algorithms is explored in [15, 17], none of the aforementioned works provide robustness against uncertainties in the impact model by explicitly addressing them.

The explicit treatment of robustness against disturbances in DDP algorithms has been explored in previous works. The work in [24] introduces a minimax, worst-case formulation to cope with disturbances affecting the continuous-time evolution of the system. Subsequent works such as [9, 28] have extended this approach, enhancing algorithm convergence and addressing bounded disturbances. In parallel, alternative variants of DDP algorithms have been proposed, adopting stochastic dynamics settings rather than the worst-case analysis [10, 30]. While the worst-case formulation might introduce additional conservativeness to the solution, its advantage over the stochastic setting is that it does not require knowledge of the probability distribution of the disturbance. Most importantly, none of the aforementioned works addresses disturbances in hybrid systems and reset events.

Finally, there exist various algorithms that improve the convergence or computational speed [18, 22], numerical stability [29], and constraint handling [12, 22, 33] of the DDP. Our work's primary focus is on the explicit treatment of the uncertainty in the reset events, thus, we adhere to the basic DDP algorithm and its hybrid systems extension in [17] as our baseline method, which will be explained in Section 3. Its extension to advanced DDP algorithms will be considered in the future work.

1.3 Contributions

The contributions for our paper are as follows: First, we propose Robust Hybrid DDP (RH-DDP) – a formulation of DDP that is robust to convex disturbance sets in the reset map of a hybrid system, by incorporating over-approximations of the worst-case effect of possible disturbances. Second, we provide guarantees on cost reduction after every DDP iteration for a hybrid linear quadratic regulator. Finally, we demonstrate our method on a two-link legged robot, and show that our method results in a decrease in worst-case cost, compared to non-robust DDP approaches.

2 Problem Statement

We consider a finite-horizon optimal control problem of a nonlinear system, subjected to one reset event in the time horizon. We discretize the system dynamics in time, denoting the timesteps as $k \in \{0, 1, \cdots, n\}$, and assume that a time-triggered reset event happens at $k = m$, $(m < n)$. This dynamics is described by

$$x_{k+1} = f(x_k, u_k) \text{ for all } k \neq m \tag{1a}$$

$$x_{m+1} = P(x_m; d). \tag{1b}$$

Eq. (1a) represents the evolution of trajectory along the continuous-time dynamics, discretized in time, which we refer to as the *continuous mode* of the dynamics, where $x \in \mathbb{R}^{n_x}$ is the system state, $u \in \mathbb{R}^{n_u}$ is the control input. Eq. (1b) describes the state jump happening at the *reset event*, described by the reset map P, where $d \in \mathbb{R}^{n_d}$ is a disturbance (or uncertainty) in the reset map.

We assume that the disturbance is bounded by a disturbance set given by $\mathcal{D} = \text{conv}(\{d_1, ..., d_v\})$, a convex hull of extreme disturbance vectors $d_i =\in \mathbb{R}^{n_d}, i = 1, \cdots, v$. We assume that the nominal disturbance is set to 0, and $0 \in \mathcal{D}$.

We consider a general finite-horizon cost function consisting of a running cost and a terminal cost,

$$J(x_0, \boldsymbol{u}_{0:n-1}; d) = \sum_{\substack{k=0 \\ k\neq m}}^{n-1} l(x_k, u_k) + l_f(x_n), \tag{2}$$

where $\boldsymbol{u}_{k_1:k_2} = \{u_{k_1}, ..., u_{k_2}\}$ denotes the control sequence from timestep k_1 to k_2, l is a stage cost, and l_f is a terminal cost. We assume that l is jointly convex in (x, u) and strongly convex in u for each fixed x, and l_f is convex.

We consider an optimal control problem under the worst-case disturbance, in which we assume that the value of the disturbance can be taken such that it maximizes the cost function. We assume that the disturbance value is taken only in a non-anticipative way, meaning the disturbance cannot use any information about the future control inputs. Thus, formally, the problem is described as finding the optimal control sequence $\boldsymbol{u}^*_{0:n-1}$ that solves

$$\min_{\boldsymbol{u}_{0:m}} \max_{d\in\mathcal{D}} \min_{\boldsymbol{u}_{m+1:n-1}} J(x_0, \boldsymbol{u}_{0:n-1}; d). \tag{3}$$

Remark 1. *Problem in* (3) *describes a special case of a dynamic feedback Stackelberg game between the control and the disturbance [8], where the disturbance can take an action only at the reset event. Introduction of the disturbance to the continuous mode of the dynamics* (1a) *is discussed in detail in [8], and the DDP solution to such problem for systems without reset events are discussed in [9, 24, 28]. Although combining these methods to our framework is possible since the DDP update for the continuous mode remains unchanged in our method,*

to distinguish from the existing work, we only consider the disturbance in the reset event.

Remark 2. *(Time-triggered vs. event-triggered reset events) Many physical systems involving reset events do not have a fixed reset event schedule or timing (m), but rather, the reset event happens when the state meets a particular condition. These conditions are known as guards or switching surfaces, and the time when the state meets these conditions may alter the value of m. In this paper, we are not concerned with updating m, but its update can be done in the outer loop of the DDP algorithm using a switching time optimization [17], or by using the notion of a saltation matrix as in [14].*

3 Background

3.1 Classical DDP

Classical DDP seeks to iteratively minimize a cost function via a series of backward passes (where a new control sequence is generated) and forward passes (where the new control sequence generates a new trajectory).

The dynamics and the cost function are given by (1a) and (2), where the reset event is not considered in the classic literature. We define the value function as the optimal cost-to-go function, given as

$$V(x_k) := J(x_k, \boldsymbol{u}^*_{k:n-1}), \text{ where } \boldsymbol{u}^*_{k:n-1} = \arg\min J(x_k, \boldsymbol{u}_{k:n-1}).$$

The forward pass takes a nominal control sequence $\boldsymbol{u}_{0:n-1}$ from the initial state x_0 and computes the trajectory. By Bellman's principle of optimality, the value function at step k satisfies

$$V(x_k) = \min_{u_k}[l(x_k, u_k) + V(f(x_k, u_k))]. \tag{4}$$

DDP takes a Newton method approach to solve (4) quickly, where local quadratic approximations of the value function are used. Note that the timestep subscripts are dropped for simplicity below, and the primes ($'$) denote the next time step. We define the argument of the min of (4) under the state and control input perturbation δx, δu as

$$Q(\delta x, \delta u) = l(x + \delta x, u + \delta u) + V(f(x + \delta x, u + \delta u)), \tag{5}$$

and consider a second-order variation

$$\delta Q(\delta x, \delta u) = Q(\delta x, \delta u) - Q(0,0) \approx \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^\top \begin{bmatrix} 0 & Q_x^\top & Q_u^\top \\ Q_x & Q_{xx} & Q_{ux}^\top \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix},$$

where the partial derivatives (denoted by subscripts) are

$$Q_x = L_x + f_x^\top V_x' \tag{6a}$$
$$Q_u = L_u + f_u^\top V_x' \tag{6b}$$
$$Q_{xx} = L_{xx} + f_x^\top V_{xx}' f_x \tag{6c}$$
$$Q_{uu} = L_{uu} + f_u^\top V_{xx}' f_u \tag{6d}$$
$$Q_{ux} = L_{ux} + f_u^\top V_{xx}' f_x. \tag{6e}$$

Eqs. (6c), (6d), and (6e) have a tensor-matrix product, which can be omitted for numerical stability. In the backward pass, we optimize Q w.r.t δu, to obtain

$$\begin{aligned} \delta u^* &= \arg\min_{u^*}[Q(\delta x, \delta u)] \\ &= -Q_{uu}^{-1}(Q_u + Q_{ux}\delta x) \\ &=: \kappa + K\delta x. \end{aligned} \tag{7}$$

Substituting (7) into (6), we obtain

$$\Delta V = \Delta V' + \frac{1}{2} Q_u^\top Q_{uu}^{-1} Q_u \tag{8a}$$
$$V_x = Q_x - Q_{ux}^\top Q_{uu}^{-1} Q_u \tag{8b}$$
$$V_{xx} = Q_{xx} - Q_{ux}^\top Q_{uu}^{-1} Q_{ux} \tag{8c}$$

where ΔV denotes the expected reduction in value function this iteration. During the next forward pass, $\boldsymbol{u}_i$ is updated through an Armijo backtracking line search by

$$\bar{x_0} = x_0 \tag{9a}$$
$$\bar{u_i} = u_i + \epsilon\kappa_i + K_i(\bar{x_i} - x_i) \tag{9b}$$
$$\bar{x}_{i+1} = f(\bar{x_i}, \bar{u_i}), \tag{9c}$$

where ϵ is the backtracking search parameter, and is initialized as 1. It is geometrically reduced by a factor c (we use $c = 0.5$, for $\epsilon \leftarrow 0.5\epsilon$), until the Armijo condition, given below is satisfied:

$$J(x_0, \bar{u}_{0:n-1}) \leq J(x_0, \boldsymbol{u}_{0:n-1}) - cb\Delta V_0, \tag{10}$$

where ΔV_0 is given by (8a) for $k = 0$, and b is a hyperparameter of the optimization, which we choose to use 0.5.

The forward pass is complete once the above line search is done. It passes the new trajectory to the backward pass. This process continues until convergence to a locally optimal control sequence $\boldsymbol{u}^*_{0:n-1}$.

3.2 Hybrid Systems DDP

The method developed in [17] explicitly accounts for the reset events in the dynamics (1). Note that this method does not account for the disturbance term. To simplify notations, we denote the entities before the reset event, happening at $k = m$, with a superscript $(-)$, and entities after the reset event at $k = m + 1$, with a superscript $(+)$.

The paper shows that the value function update across the reset map during the backward pass is given by

$$\Delta V^- = \Delta V^+ \tag{11a}$$
$$V_{xx}^- \approx P_x^\top V_{xx}^+ P_x \tag{11b}$$
$$V_x^- = P_x^\top V_x^+ \tag{11c}$$

Where P_x is the Jacobian of the reset map with respect to the state before the reset, x^-.

3.3 Log-Sum-Exp

The log-sum-exponent (LSE) operator is an effective smooth approximation of a max operator, and is used in our method to approximately solve $\max_{d\in\mathcal{D}}$ in (3).

Let $y_i \in \mathbb{R}, i \in \{1, ..., v\}, y = [y_1, ..., y_v]^T$, and $\hat{y} = \max_i\{y_i\}$. Then, the LSE function for some parameter α is a *smooth* approximation of $\hat{y}$ defined as

$$\mathrm{LSE}_\alpha(y) = \frac{1}{\alpha}\cdot\log\left(\sum_{i=1}^{v}\exp(\alpha\cdot y_i)\right). \tag{12}$$

The parameter α controls the tightness of the approximation and provides the upper and lower bound of $\hat{y}$ as,

$$\mathrm{LSE}_\alpha(y) - \frac{\log(v)}{\alpha} \le \hat{y} < \mathrm{LSE}_\alpha(y). \tag{13}$$

For real-valued functions $h_i : \mathbb{R}^n \to \mathbb{R}$, let $H = (h_1, ..., h_v)$ and $H(x) = [h_1(x), ..., h_v(x)]^\top$. Then we define the operator $\mathrm{LSE}_\alpha(H)$ as

$$\mathrm{LSE}_\alpha(H)(x) = \mathrm{LSE}_\alpha(H(x)).$$

For this work, we will take $h_i(x)$ of the form $\frac{1}{2}x^\top Ax + B_i x + C_i$, $A \succ 0$. All the h_i are quadratic functions, with a common quadratic term that is assumed to be positive definite, but possibly different linear and constant terms. These terms will be defined contextually in Section 4.

$\mathrm{LSE}_\alpha(H)$ then provides a smooth over-approximation of the point-wise maximum operator, $\max_i h_i$. For sufficiently small α, the second order expansion of $\mathrm{LSE}_\alpha(H)$ is also an over-approximation of $\max_i h_i$, and can be used in the DDP's Newton step optimization approach.

To find this expansion, define $DH(x) \in \mathbb{R}^{v\times n}$ as the Jacobian of H, which is

$$DH(x) = \begin{bmatrix} h_{1_x}^\top(x) \\ \vdots \\ h_{v_x}^\top(x) \end{bmatrix} = \begin{bmatrix} (Ax + B_1^\top)^\top \\ \vdots \\ (Ax + B_v^\top)^\top \end{bmatrix},$$

and define $\mathrm{SM}_\alpha(H)(x) := \mathrm{softmax}(\alpha H(x)) \in \mathbb{R}^v$. The i-th entry of $\mathrm{SM}_\alpha(H)(x)$ is given by

$$[\mathrm{SM}_\alpha(H(x)]_i = \frac{\exp(\alpha h_i(x))}{\sum_{j=1}^{v}\exp(\alpha h_j(x))}.$$

Then, the gradient and Hessians of $\mathrm{LSE}_\alpha(H(x))$ can be calculated as

$$\nabla_x \mathrm{LSE}_\alpha(H)(x) = DH(x)^\top \mathrm{SM}_\alpha(H)(x), \tag{14a}$$

$$\nabla_{xx}\mathrm{LSE}_\alpha(H)(x) = \alpha DH(x)^\top[\mathrm{diag}\{\mathrm{SM}_\alpha(H)(x)\} - \mathrm{SM}_\alpha(H)(x)\ \mathrm{SM}_\alpha(H)(x)^\top]DH(x) + A. \tag{14b}$$

The matrix $(\mathrm{diag}\{\mathrm{SM}_\alpha(H)(x)\} - \mathrm{SM}_\alpha(H)(x)\ \mathrm{SM}_\alpha(H)(x)^\top)$ is positive semi-definite [3, Pg. 74], so $\nabla_{xx}\mathrm{LSE}_\alpha(H)(x) \succeq A$. Additionally, we note that (14a) and (14b) can be computed using a numerically stable implementation of softmax [2].

4 Method

At each time step in the backward pass, DDP requires a positive definite quadratic local approximation of the Q and value functions. Similar to LQR, the Bellman update at time step k in the continuous mode preserves the quadratic approximation of the value function at time step $k-1$. However, before the reset (at time step m) the quadratic approximation no longer holds, because the worst-case disturbance tries to maximize the value function at time step $m+1$, (3). This means that the backward pass iteration cannot be directly performed at time step m.

The key idea of our method and the flow of this section is as follows: first, we show that the value function at time step m (just before the reset) is convex and we compute a smooth over-approximation of the value function. However, this approximate smooth function is not quadratic, so we compute a second-order approximation to obtain a quadratic function. We show that this quadratic function is an over-approximation of the actual value function at time step m. The DDP backward pass can now proceed using this quadratic function. For the special case of linear-quadratic regulators with a linear reset map, we can show that our method results in a guaranteed decrease in the cost (3) up to a constant term.

4.1 Backward pass

Our method runs the backward pass as normal until it encounters a reset event. Upon the reset event, we would like to pass back the worst-case value function with respect to the disturbance. We examine a second order expansion of V^+, the value function just after reset, with respect to x^+, the state just after reset:

$$\delta Q^-(\delta x^-, \delta d) := \delta V^+(\delta x^+) \approx V_x^{+\top}\delta x^+ + \frac{1}{2}\delta x^{+\top}V_{xx}^+\delta x^+. \tag{15}$$

We also examine a first-order expansion of x^+ with respect to the state before the reset, x^-, and the nominal disturbance 0 in the reset event.

$$\delta x^+ \approx P_x(x^-, 0)\delta x^- + P_d(x^-, 0)\delta d. \tag{16}$$

Here, δd is an infinitesimal disturbance about the nominal disturbance 0 and is assumed to lie in the set $\delta\mathcal{D} = \mathrm{conv}\{\delta d_1, ..., \delta d_v\}$. We comment on how big the disturbance δd can be Remark 3.

Substituting (16) in (15), we find an expansion of Q^- as a function of $(\delta x^-, \delta d)$ about $(x^-, 0)$:

$$\delta Q^-(\delta x^-, \delta d) = \frac{1}{2}\delta x^{-\top}A\delta x^- + B(\delta d)\delta x^- + C(\delta d), \tag{17}$$

where

$$A = P_x^\top V_{xx}^+ P_x, \tag{18a}$$
$$B(\delta d) = \delta d^\top P_d^\top V_{xx}^+ P_x + V_x^{+\top} P_x, \tag{18b}$$
$$C(\delta d) = \frac{1}{2}\delta d^\top P_d V_{xx}^+ P_d \delta d + V^{+\top} P_d \delta d. \tag{18c}$$

To account for the worst-case disturbance, we set the value function after the reset to be the maximum over all the disturbances:

$$\delta V^-(\delta x^-) = \max_{\delta d \in \delta \mathcal{D}} \delta Q^-(\delta x^-, \delta d).$$

Note that δV^- is the point-wise maximum of a family of convex functions, and thus is convex [3, Sec. 3.2.3]. Moreover, because $\delta\mathcal{D}$ is a compact polyhedral set and the maximum of a convex function on a compact polyhedral set is achieved on at least one of its vertices,

$$\delta V^-(\delta x^-) = \max_{1 \le i \le v} \delta Q^-(\delta x^-, \delta d_i). \tag{19}$$

At this point, we have shown that δV^- is a convex function, but it is not quadratic and thus cannot be directly used for the backward pass. So, we find a quadratic over-approximation of δV^-. We begin by computing the smooth LSE over-approximation of δV^-.

Defining $B_i = B(\delta d_i)$ and $C_i = C(\delta d_i)$, let

$$\begin{aligned} h_i(\delta x^-) &:= \delta Q^-(\delta x^-, \delta d_i), \\ &= \frac{1}{2}\delta x^{-\top} A \delta x^- + B_i \delta x^- + C_i, \end{aligned} \tag{20a}$$
$$H(\delta x^-) := [h_1(\delta x^-), ..., h_v(\delta x^-)]^\top. \tag{20b}$$

Then, we can approximate δV^- using (19) as

$$\delta V^-(\delta x^-) = \mathrm{LSE}_\alpha(H)(\delta x^-), \tag{21}$$

based on Section 3.3. Since $\mathrm{LSE}_\alpha(H)$ is not quadratic, we compute its second-order approximation about x^-, as

$$\delta V^-(\delta x^-) \approx \mathrm{LSE}_\alpha(H)(0) + V_x^{-\top}\delta x^- + \frac{1}{2}\delta x^{-\top} V_{xx}^- \delta x^-, \tag{22}$$

where

$$V_x^- = \nabla_x \mathrm{LSE}_\alpha(H)(0), \tag{23a}$$
$$V_{xx}^- = \nabla_{xx} \mathrm{LSE}_\alpha(H)(0) + \beta I_{n_x \times n_x}. \tag{23b}$$

The derivatives (23a, 23b) can be calculated using (14a, 14b). The regularization constant $\beta \ge 0$ is used to guarantee that $\delta V^-(\delta x^-) \ge h_i(\delta x^-)\ \forall \delta x^-, i$, i.e. to ensure that δV^- is an over-approximation of the actual value function.

β can be calculated as follows: First, note that since $\mathrm{LSE}_\alpha(H)$ over-approximates h_i, $\delta V^-(0) > h_i(0)\ \forall i \Rightarrow \mathrm{LSE}_\alpha(H)(0) > C_i\ \forall i$. Noting from Sec. 3.3 that $V_{xx}^- \succeq A + \beta I$, a sufficient condition to ensure $\delta V^-(\delta x^-) \ge h_i(\delta x^-)\ \forall \delta x^-$ is,

$$\mathrm{LSE}_\alpha(H)(0) + V_x^{-\top}\delta x^- + \frac{1}{2}\beta\|\delta x^-\|_2^2 \ge B_i \delta x^- + C_i.$$

It is easily shown that this inequality holds for all $\delta x^-, i$ when

$$\beta \ge \max_i \frac{\|V_x^- - B_i^\top\|_2^2}{2(\mathrm{LSE}_\alpha(H)(0) - C_i)}. \tag{24}$$

With (22), (23), and (24), we have computed a quadratic approximation δV^- of the actual value function, which can now be used in the backward pass. Our backward pass method is summarized in Algorithm 1.

4.2 Forward pass

The forward pass is mostly the same as that of the classical DDP, described in Section 3.1. The only major difference is that in each step of the Armijo backtracking line search, to ensure proper convergence, we must conduct a rollout under each d_i, and use the worst-case cost to check the Armijo Condition, which becomes

$$\max_{1 \le i \le v} J(x_0, \bar{u}_{0:n-1}; d_i) \le J_{\max}^{\text{prev}} - cb\Delta V_0, \tag{25}$$

where $J_{\max}^{\text{prev}}$ is the maximum cost of the previous forward pass. Once the Armijo condition is satisfied, we pass the trajectory under the disturbance that incurred the worst-case cost to the backward pass and reiterate the process. The forward pass is described in Algorithm 2 and the entire algorithm is summarized in Algorithm 3.

Algorithm 1: Backward Pass

Input: $\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1}$,
Output: $\boldsymbol{\kappa}_{0:n-1}, \boldsymbol{K}_{0:n-1}, \Delta\mathbf{V}_{0:n-1}$
$H \leftarrow$ Horizon length, $R \leftarrow$ timestep of reset map
Initialize $\Delta\mathbf{V}_{0:n-1}, \boldsymbol{V}_{\boldsymbol{x},0:n-1}, \boldsymbol{V}_{\boldsymbol{xx},0:n-1}, \boldsymbol{\kappa}_{0:n-1}, \boldsymbol{K}_{0:n-1}$
for $i = n-1;\ i \ge m+1;\ i \leftarrow i-1$ **do**
  $\Delta\mathbf{V}_i, \boldsymbol{V}_{\boldsymbol{x},i}, \boldsymbol{V}_{\boldsymbol{xx},i}, \boldsymbol{\kappa}_i, \boldsymbol{K}_i$ updated per (7, 8)
end
$\Delta\mathbf{V}_m, \boldsymbol{V}_{\boldsymbol{x},m}, \boldsymbol{V}_{\boldsymbol{xx},m}$ updated per (22), (23), and (24)
for $i = m-1;\ i \ge 0;\ i \leftarrow i-1$ **do**
  $\Delta\mathbf{V}_i, \boldsymbol{V}_{\boldsymbol{x},i}, \boldsymbol{V}_{\boldsymbol{xx},i}, \boldsymbol{\kappa}_i, \boldsymbol{K}_i$ updated per (7, 8)
end
return $\boldsymbol{\kappa}_{0:n-1}, \boldsymbol{K}_{0:n-1}, \Delta V_0$

Algorithm 2: Forward Pass

Input: $\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1}, \boldsymbol{\kappa}_{0:n-1}, \boldsymbol{K}_{0:n-1}, \Delta V_0, J_{\max}$
Output: $\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1}, J_{\max}$
$J_{\max}^{\text{prev}} \leftarrow J_{\max}$
while *Line Search Not Satisfied* **do**
  $J_{\max} \leftarrow$ Rollout trajectory and update control for each δd_i; choose maximum cost
  Compare $J_{\max}$ against $\Delta V_0, J_{\max}^{\text{prev}}$
end
$\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1} \leftarrow$ Trajectory that incurred the maximum cost
return $\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1}, J_{\max}$

Algorithm 3: Robust Hybrid DDP (RH-DDP)

Input: $\boldsymbol{u}^{\text{init}}_{0:n-1}, x_0, \mathcal{D}$
Output: $\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1}, \boldsymbol{K}_{0:n-1}, J_{\max}$
$X \leftarrow$ roll out trajectory with control U_{WS}, initial state x_0, and disturbance d_{nom}.
while *not converged* **do**
  $\boldsymbol{\kappa}_{0:n-1}, \boldsymbol{K}_{0:n-1}, \Delta V_0 \leftarrow$ Backward Pass$(\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1})$
  $\boldsymbol{\kappa}_{0:n-1}, \boldsymbol{K}_{0:n-1}, \Delta V_0 \leftarrow$ Forward Pass$(\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1}, \boldsymbol{\kappa}_{0:n-1}, \boldsymbol{K}_{0:n-1}, \Delta V_0, J_{\max})$
end
return $\boldsymbol{x}_{0:n}, \boldsymbol{u}_{0:n-1}, \boldsymbol{K}_{0:n-1}, J_{\max}$

4.3 Linear System Analysis

In this section, we consider a linear quadratic regulator with a linear reset map. We show that for any compact polyhedral disturbance set $\mathcal{D}$, one iteration of our method (i.e., backward pass + forward pass) results in a decrease in cost up to a constant. Towards this end, consider a discrete-time linear hybrid system:

$$x_{k+1} = Fx_k + Gu_k \quad \text{for all } k \neq m, \tag{26a}$$
$$x_{m+1} = Px_m + d, \tag{26b}$$

Consider the optimal control problem (3), with the cost function at time step t as

$$J_t(x_t, \boldsymbol{u}_{t:n-1}; d) = x_n^\top S_n x_n + \sum_{\substack{k=t \\ k\neq m}}^{n-1} x_k^\top \mathrm{N}\, x_k + u_k^\top \mathrm{M}\, u_k, \tag{27}$$

where the trajectory $\boldsymbol{x}_{t:n}$ is computed using (26), and $\mathrm{N}, S_n \succeq 0$ and $\mathrm{M} \succ 0$ are the cost matrices. Let

$$V_t(x_t) = \min_{\boldsymbol{u}_{t:m}} \max_{d\in\mathcal{D}} \min_{\boldsymbol{u}_{m+1:n-1}} J_t(x_t, \boldsymbol{u}_{t:n-1}; d) \tag{28}$$

be the optimal cost-to-go at time step t from state x_t. When $t = 0$, this is the same as the cost function (2). For $t \geq m+1$, (28) is an LQR problem, so for $t \geq m+1$, $V_t(x_t) = x_t^\top S_t x_t$, where $S_t \succeq 0$ is obtained from the Riccati Difference Equation (RDE).

According to the Bellman principle, and by (19),

$$V_m(x_m) = \max_{d\in\mathcal{D}} \ (Px_m + d)^\top S_{m+1}(Px_m + d), \tag{29a}$$
$$= \max_{i\in\{1,\dots,v\}} (Px_m + d_i)^\top S_{m+1}(Px_m + d_i). \tag{29b}$$

Now, (28) at $t = 0$ can be written as,

$$V_0(x_0) = \min_{\substack{\boldsymbol{x}_{0:m} \\ \boldsymbol{u}_{0:m-1}}} \sum_{k=0}^{m-1} (x_k^\top \mathrm{N}\, x_k + u_k^\top \mathrm{M}\, u_k) + V_m(x_m), \tag{30a}$$
$$\text{s.t} \quad x_{k+1} = Fx_k + Gu_k,\ k \in \{0, \dots, m-1\}, \tag{30b}$$

which is a (non-smooth) convex optimization problem.

Our method, described in Sec. 4.1 and Sec. 4.2, solves this problem iteratively using LSE to over-approximate the value function V_m using the quadratic function $\tilde{V}_m$, (22). Let $(\boldsymbol{x}^j_{0:m}, \boldsymbol{u}^j_{0:m-1})$ be the current iterate for the DDP algorithm, and $(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1})$ be the next iterate obtained after one step of our algorithm. Then, Robust Hybrid DDP solves for $(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1})$ by solving the following LQR problem,

$$\tilde{V}_0(x_0) = \min_{\substack{\boldsymbol{x}_{0:m} \\ \boldsymbol{u}_{0:m-1}}} \sum_{k=0}^{m-1} (x_k^\top \mathrm{N}\, x_k + u_k^\top \mathrm{M}\, u_k) + \tilde{V}_m(x_m; x_m^j), \tag{31a}$$
$$\text{s.t} \quad x_{k+1} = Fx_k + Gu_k,\ k \in \{0, \dots, m-1\}, \tag{31b}$$

where $\tilde{V}_m(\cdot\,; x_m^j)$ is the quadratic over-approximation of V_m with the second-order approximation of the LSE performed at x_m^j, (22). Note that (31) is an LQR problem.

Before proving our result, we define the cost functions for a trajectory $(\boldsymbol{x}_{0:m}, \boldsymbol{u}_{0:m-1})$ corresponding to (30) and (31), respectively, as

$$J_0^m(\boldsymbol{x}_{0:m}, \boldsymbol{u}_{0:m-1}) = \sum_{k=0}^{m-1} (x_k^\top \mathrm{N}\, x_k + u_k^\top \mathrm{M}\, u_k) + V_m(x_m),$$
$$\tilde{J}_0^m(\boldsymbol{x}_{0:m}, \boldsymbol{u}_{0:m-1}) = \sum_{k=0}^{m-1} (x_k^\top \mathrm{N}\, x_k + u_k^\top \mathrm{M}\, u_k) + \tilde{V}_m(x_m; x_m^j).$$

Now, we can prove the following result, which guarantees that the cost corresponding to the trajectory $(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1})$ cannot increase by more than an amount inversely proportional to α.

Theorem 1. *Let $\mathcal{D}$ be any compact polyhedral disturbance set, and consider the optimal control problem (30) subject to (26). For any dynamically feasible trajectory iterate $(\boldsymbol{x}^j_{0:m}, \boldsymbol{u}^j_{0:m-1})$, let the next iterate $(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1})$ be obtained by solving the LQR problem (31), where $\tilde{V}_m(\cdot\,; x_m^j)$ is obtained as in (22). Then,*

$$J_0^m(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1}) \leq J_0^m(\boldsymbol{x}^j_{0:m}, \boldsymbol{u}^j_{0:m-1}) + \frac{\log(v)}{\alpha}. \tag{32}$$

Proof. The quadratic over-approximation function $\tilde{V}_m(\cdot\,; x_m^j)$ satisfies the following properties, as shown in Sec. 4.1:

(a) $\tilde{V}_m(x_m^j; x_m^j) \leq V_m(x_m^j) + \frac{\log(v)}{\alpha}$ (by LSE property (13)).
(b) $V_m(\cdot) \leq \tilde{V}_m(\cdot\,; x_m^j)$ (by construction in Sec. 4.1).

By properties (a) and (b) respectively, we have that for the trajectories $(\boldsymbol{x}^j_{0:m}, \boldsymbol{u}^j_{0:m-1})$ and $(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1})$,

$$\tilde{J}_0^m(\boldsymbol{x}^j_{0:m}, \boldsymbol{u}^j_{0:m-1}) \leq J_0^m(\boldsymbol{x}^j_{0:m}, \boldsymbol{u}^j_{0:m-1}) + \frac{\log(v)}{\alpha}, \tag{33a}$$
$$J_0^m(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1}) \leq \tilde{J}_0^m(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1}). \tag{33b}$$

Finally, we note that the LQR process produces the optimal solution in one step, and thus leads to a reduction in cost. This means that,

$$\tilde{J}_0^m(\boldsymbol{x}^{j+1}_{0:m}, \boldsymbol{u}^{j+1}_{0:m-1}) \leq \tilde{J}_0^m(\boldsymbol{x}^j_{0:m}, \boldsymbol{u}^j_{0:m-1}). \tag{34}$$

Combining inequalities (33a), (33b), and (34), we get (32). □

Theorem 1 guarantees that the cost of the next iterate does not increase by more $\frac{\log(v)}{\alpha}$. If we increase α, the tightness parameter, we can reduce this error. However, using a large value of α initially can blow up the β parameter (24). Therefore, we propose annealing of the α parameter, where it is increased by a factor every few DDP iterations. Furthermore, by (34), if the LQR problem (31) results in a decrease of more than $\frac{\log(v)}{\alpha}$, then the cost of the next iterate is guaranteed to be lower. This tells us that whenever the cost reduction from a fixed value α is less than $\frac{\log(v)}{\alpha}$, we can perform an annealing step.

Remark 3. *(Nonlinear system case) While we present results of our approach on nonlinear systems in 5.2, however the above analysis on linear systems is not directly applicable to nonlinear systems due to several key factors. Firstly, the quadratic approximations of the value function (6) are only accurate within a specific neighborhood of the state-input trajectory, the extent of which hinges on the cost functions and the nonlinear dynamics. This limitation is a practical consideration for DDP algorithms in all nonlinear systems, necessitating a line search in the forward pass (10) to ensure proper cost reduction in each iteration. Secondly, the accuracy of the reset map linearization is similarly confined to a neighborhood around the linearization point when the reset map is nonlinear. Consequently, the worst-case disturbance may not occur at the vertices of the disturbance set $\mathcal{D}$ and at the time step m. Therefore, our algorithm is most effective when the disturbance set is not excessively large.*

5 Results

In this section, we present the application of our algorithm (Algorithm 3) to two examples. To the extent of our knowledge, there exist no other method in the literature that explicitly addresses the uncertainties in the reset event which we can compare against. Thus, we mainly compare our method, denoted as *RH-DDP*, against the method developed in [17], denoted as *HS-DDP*, described in Section 3.2. The HS-DDP algorithm can be considered as a special case of Algorithm 3 where the disturbance set $\mathcal{D}$ is set to $\{0\}$. Note that the original work in [17] has additional outer loops of trajectory optimization that impose constraints to the trajectory and adjust the time of the reset events, which are not included in this comparison, since we do not consider the constraints or event-triggered reset events.

In both examples, we show the efficacy of our method in producing a more robust trajectory than the baseline method, by confirming the cost reduction in the solution under the worst-case disturbance scenario. In the first example, we apply our algorithm to a linear system, to show our method's compliance with the analysis conducted in Section 4.3. In the second example, we demonstrate our algorithm on a simple two-link legged robot system, a nonlinear system subjected to a reset event as its swinging leg hits the ground.

5.1 Linear System Example: Double Integrator

We analyze a point mass constrained to one dimension as a simple double integrator, whose continuous mode dynamics is given as

$$x_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} u_k,$$

where the state x consists of the position and the velocity, the control input u represents the acceleration, and $\Delta t = 0.01$ is the time step for the Euler discretization. The time horizon is set as $n = 1000$, and the reset is scheduled at $m = 800$. Upon the reset, a fixed value of 0.5 and the disturbance value d is added to the velocity, given as

$$P(x^-; d) = x^- + \begin{bmatrix} 0 \\ 0.5 + d \end{bmatrix}.$$

We consider the disturbance bound $\mathcal{D} = [-0.5, 0.5]$. Finally, the cost terms are as follows:

$$l(x,u) = x^\top \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} x + 2.5u^2; \quad l_f(x) = x^\top \begin{bmatrix} 500 & 0 \\ 0 & 1000 \end{bmatrix} x,$$

which penalize the trajectories deviation from the origin, which is considered as the goal state, and the control effort.

After solving for the optimal control sequence $\boldsymbol{u}_{0:n-1}$ and the local linear feedback controller provided by $\boldsymbol{K}_{0:n-1}$ with RH-DDP and HS-DDP, we rollout the obtained controllers under each d values in $\mathcal{D}$ and compute the cost J for each controller. The results are shown in Figure 1. The maximum cost of HS-DDP over this disturbance set is 360.71, incurred at $d = 0.5$. Meanwhile, the maximum cost of our method is 321.01 incurred symemtrically at $d = -0.5$ and $d = 0.5$, corresponding to a cost reduction of 11.0% at the worst-case instance. Under the nominal case where there is no disturbance ($d = 0$), our method incurs a slight increase in the cost (9.5%) compared to the baseline method, which is an expected symptom of robust control, since it would improve the worst-case performance at the expense of sacrificing the overall performance.

Figure 2 shows a position-velocity phase plot, under the worst-case disturbance that incurs maximum cost of the HS-DDP trajectory (top), and under the nominal case without any disturbance (bottom). Our method tends to produce a trajectory whose velocity is close to zero right before the reset, so that when a positive disturbance is applied during the reset, it reduces a cost associated with the velocity magnitude. In the nominal case, this affects the terminal state to slightly reach less close to the origin, thereby resulting in a slight increase in the total cost.

5.2 Nonlinear System Example: Two-link Walker

To demonstrate the application of our algorithm to robust control of legged robots, we consider a simple two-link walker system, illustrated in Figure 3. The continuous mode dynamics of the robot (1a) is derived from the Euler-Lagrange

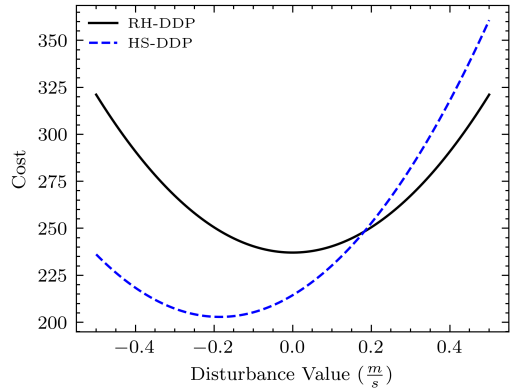

Figure 1. Double integrator example: incurred costs of DDP trajectories with respect to various disturbance values at the reset event.

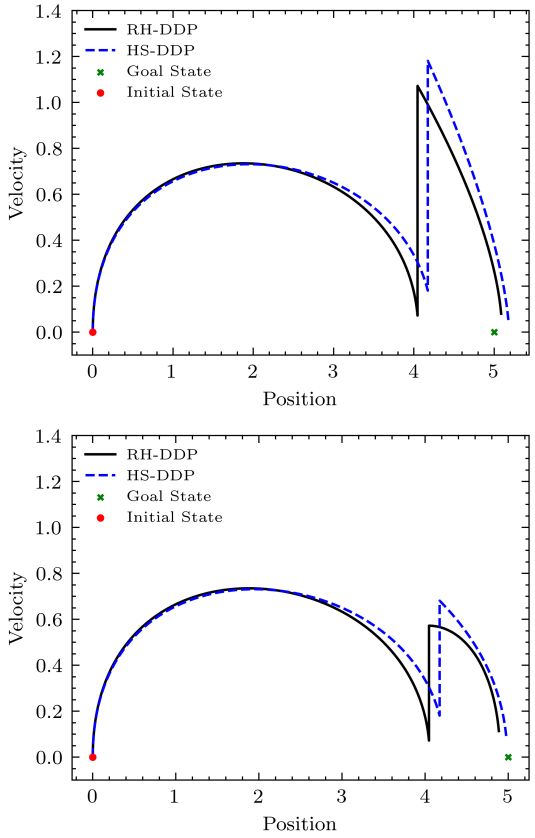

Figure 2. Double integrator example: position-velocity phase plot of the trajectories under worst-case disturbance ($d = 0.5$, top) and without disturbance ($d = 0$, bottom). The straight lines represent the discontinuity of the states incurred at the reset event.

equation, constrained by the contact force between the stance leg and the ground. The general structure of this dynamics

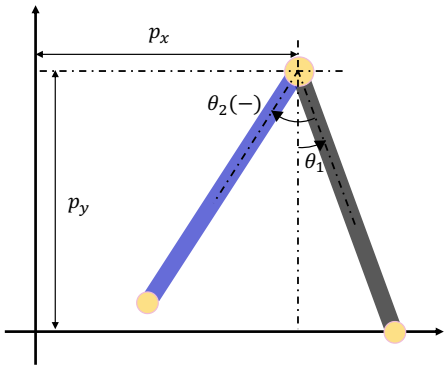

Figure 3. Configuration of the two-link walker.

is given as

$$\begin{bmatrix} \mathrm{D} & -\mathrm{J}_{st}^T \\ -\mathrm{J}_{st} & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda_{st} \end{bmatrix} = \begin{bmatrix} \mathrm{B}u - \mathrm{C}\dot{q} - \mathrm{G} \\ \dot{\mathrm{J}}_{st}\dot{q} \end{bmatrix},$$

where q denotes the generalized coordinates, u is the actuation torque that is considered as the input to the system, and D, C$\dot{q}$, G, B are inertia matrix, Coriolis force, gravity force, and input mapping matrix, respectively. J_{st} and λ_{st} represents the contact Jacobian and contact force associated with the stance foot. The robot state consist of $x = [q; \dot{q}]$, and the dynamics f in (1a) can be derived by taking the inverse of the matrix in the left hand side and separating out the solution for $\ddot{q}$, and then applying the Euler discretization. The timestep we use is $\Delta t = 0.003$s. For the two-link walker, $q = [p_x, p_y, \theta_1, \theta_2]$, where (p_x, p_y) is the torso position, θ_1 is the stance leg angle, and θ_2 is the relative angle between the legs (Figure 3), and $\dot{q} = [v_x, v_y, \dot{\theta}_1, \dot{\theta}_2]$, where (v_x, v_y) is the torso velocity.

The impact dynamics are modeled based on the rigid impact model, given as

$$\begin{bmatrix} \mathrm{D} & -\mathrm{J}_{sw}^T \\ -\mathrm{J}_{sw} & 0 \end{bmatrix} \begin{bmatrix} \dot{\hat{q}}^+ \\ \hat{\lambda}_{sw} \end{bmatrix} = \begin{bmatrix} \mathrm{D}\dot{q}^- \\ 0 \end{bmatrix}, \tag{35}$$

where $\dot{q}^-$ and $\dot{\hat{q}}^+$ are the generalized velocity before and after the impact, respectively, and J_{sw} and $\hat{\lambda}_{sw}$ represent the contact Jacobian and impulse at the swing foot. Note that under the rigid impact, the generalized coordinate q^- is unchanged. Based on the symmetry of the robot, the post-reset state x^+ is obtained after relabeling the swing leg to the stance leg, and vice versa, given as $x^+ = [q^+; \dot{q}^+] = [\mathrm{R}q^-;\ \mathrm{R}\dot{\hat{q}}^+]^\top$, where R is the relabeling matrix. Finally, we assume that the post-impact velocity $\dot{q}^+$ can be perturbed by the disturbance. We factor v_x by $(0.8 + d_1)$ and $\dot{\theta}_2$ by $(0.8 + d_2)$, where $d = (d_1, d_2) \in \mathcal{D} = [-0.2, 0.2] \times [-0.2, 0.2]$, which captures a change in torso and swing leg velocity due to unmodeled factors. Note that v_y and $\dot{\theta}_1$ is also changed accordingly to meet the constraint that the stance foot is fixed on the ground. Such impact happens when the swing leg hits the ground, which is an event-triggered reset event.

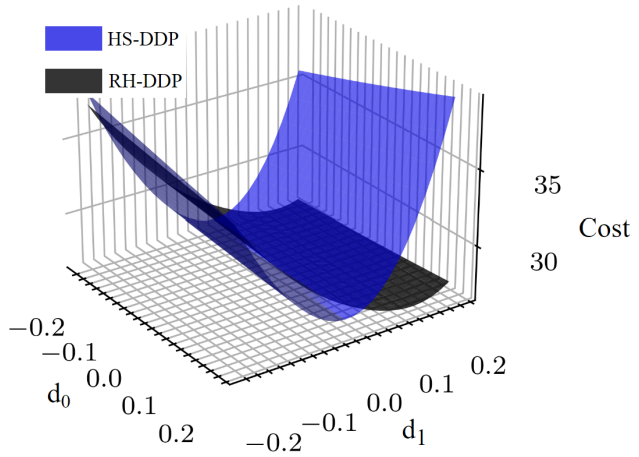

Figure 4. Two-link walker: incurred costs of DDP trajectories with respect to various disturbance values at the reset event.

To correctly time this reset, we set the initial state of the robot to be exactly one timestep before the reset, so that the reset always happen at $m = 1$.

The DDP solves an optimal control problem that stabilizes the robot from the initial state to a complete stop, at which the torso transverse velocity becomes $v_x = 0$. To achieve this, the running cost l penalizes the control effort and the terminal cost l_f penalizes $|2\theta_1 - \theta_2|$ (the configuration in which both feet touch the ground), and the velocity $|v_x|$.

After obtaining the HS-DDP and RH-DDP solutions, we evaluate the controllers under various disturbance values in the disturbance set, and the resulting incurred costs are plotted in Figure 4. At a disturbance of $d = (0.2, 0.2)$, the HS-DDP controller incurs its maximum cost of 39.55. In contrast, the our method incurs only 27.50, a significant reduction in the cost of 30.3%. Under some other disturbance values, mostly when d_1 is negative, RH-DDP induce a slight increase in the cost compared to HS-DDP, but its maximal increase rate is 5.8%, under $d = (-0.03, -0.2)$. Thus, while our method significantly improves the cost of the trajectory under the worst-case disturbance, the penalty incurred at the other disturbance values are negligible. In Figure 5, the phase portrait and the v_x history of the trajectories under the worst-case disturbance are displayed. It is worth noting that under the worst-case disturbance, RH-DDP is more effective in yielding the torso to a velocity close to zero.

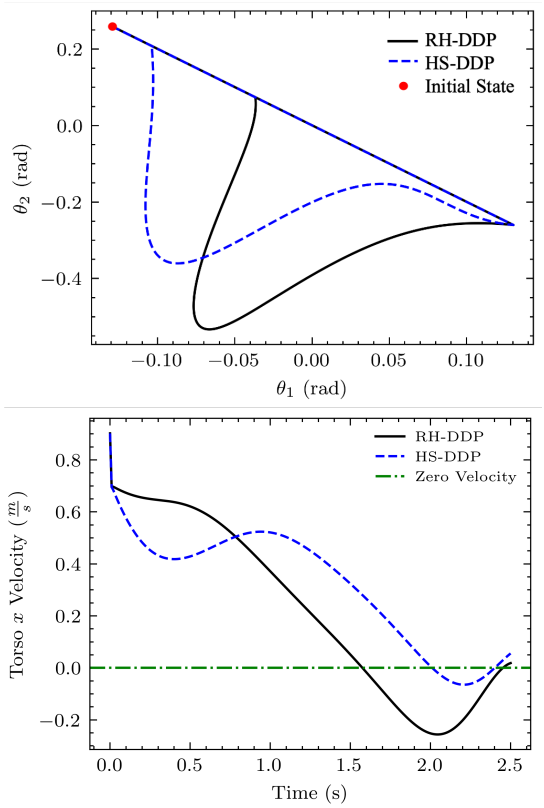

Figure 5. Two-link walker: $\theta_1 - \theta_2$ phase portraits of the trajectories (top) and the transverse velocity of the torso (bottom) under the worst-case disturbance ($d = (0.2, 0.2)$). The straight lines represent the discontinuity of the states incurred by the impact incurred immediately after the initial state.

6 Conclusion

This paper presented an extension of the DDP algorithm for hybrid systems, as introduced in [17], to incorporate robustness against worst-case disturbances in the reset event. This formulation is particularly valuable when the system is subject to uncertain reset events; for example, in legged robots, where the impact with the ground may vary unpredictably. We proved that for linear systems, our algorithm guarantees cost reduction up to a constant in each iteration. This was then validated through simulation of two distinct systems, including a stabilization task for a simple nonlinear legged robot.

There are several promising directions for future work. First, our method can be extended to trajectory optimization involving multiple reset events, however, a more detailed investigation into the impact of over-approximation gaps of the log-sum-exponent operation on the cost reduction is required. Also, the incorporation of equality and inequality constraints is essential to enhance the applicability of our methods to more practical systems and problems. Finally, an exciting direction is to integrate our method with multiple shooting variants of the DDP, as seen in [18, 22]. The current single shooting implementation is inherently more sensitive to initial conditions and guesses, especially when the system is subjected to reset events.

References

[1] Aaron D Ames, Kevin Galloway, Koushil Sreenath, and Jessy W Grizzle. 2014. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Trans. Automat. Control* 59, 4 (2014), 876–891.

[2] Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. 2020. Accurately computing the log-sum-exp and softmax functions. *IMA J. Numer. Anal.* 41, 4 (08 2020), 2311–2330. https://doi.org/10.1093/imanum/draa038

[3] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[4] Katie Byl and Russ Tedrake. 2008. Approximate optimal control of the compass gait on rough terrain. In *2008 IEEE International Conference on Robotics and Automation*. 1258–1263. https://doi.org/10.1109/ROBOT.2008.4543376

[5] Iordanis Chatzinikolaidis and Zhibin Li. 2021. Trajectory Optimization of Contact-Rich Motions Using Implicit Differential Dynamic Programming. *IEEE Robotics and Automation Letters* 6, 2 (2021), 2626–2633. https://doi.org/10.1109/LRA.2021.3061341

[6] Jason J. Choi, Ayush Agrawal, Koushil Sreenath, Claire J. Tomlin, and Somil Bansal. 2022. Computation of Regions of Attraction for Hybrid Limit Cycles Using Reachability: An Application to Walking Robots. *IEEE Robotics and Automation Letters* 7, 2 (2022), 4504–4511. https://doi.org/10.1109/LRA.2022.3151143

[7] JFA DE O. PANTOJA. 1988. Differential dynamic programming and Newton's method. *Internat. J. Control* 47, 5 (1988), 1539–1553.

[8] Lawrence C Evans and Panagiotis E Souganidis. 1984. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana University mathematics journal* (1984).

[9] Dennis Gramlich, Carsten W. Scherer, and Christian Ebenbauer. 2022. Robust Differential Dynamic Programming. In *2022 IEEE 61st Conference on Decision and Control (CDC)*. 1714–1721. https://doi.org/10.1109/CDC51059.2022.9992569

[10] Astghik Hakobyan and Insoon Yang. 2023. Distributionally Robust Differential Dynamic Programming With Wasserstein Distance. *IEEE Control Systems Letters* 7 (2023), 2329–2334. https://doi.org/10.1109/LCSYS.2023.3286431

[11] Matthew D. Houghton, Alexander B. Oshin, Michael J. Acheson, Evangelos A. Theodorou, and Irene M. Gregory. [n. d.]. *Path Planning: Differential Dynamic Programming and Model Predictive Path Integral Control on VTOL Aircraft*. https://doi.org/10.2514/6.2022-0624 arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2022-0624

[12] Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. 2019. ALTRO: A Fast Solver for Constrained Trajectory Optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 7674–7679. https://doi.org/10.1109/IROS40897.2019.8967788

[13] David H. Jacobson and David Q. Mayne. 1970. *Differential Dynamic Programming*. American Elsevier Publishing Company.

[14] Nathan J. Kong, George Council, and Aaron M. Johnson. 2021. iLQR for Piecewise-Smooth Hybrid Dynamical Systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*. 5374–5381. https://doi.org/10.1109/CDC45484.2021.9683506

[15] Nathan J. Kong, J. Joe Payne, James Zhu, and Aaron M. Johnson. 2023. Saltation Matrices: The Essential Tool for Linearizing Hybrid Dynamical Systems. arXiv:2306.06862 [cs.RO]

[16] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. 2020. Learning quadrupedal locomotion over challenging terrain. *Science robotics* 5, 47 (2020), eabc5986.

[17] He Li and Patrick M. Wensing. 2020. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters* 5, 4 (2020), 5448–5455.

[18] He Li, Wenhao Yu, Tingnan Zhang, and Patrick M. Wensing. 2023. A Unified Perspective on Multiple Shooting In Differential Dynamic Programming. arXiv:2309.07872 [cs.RO]

[19] He Li, Tingnan Zhang, Wenhao Yu, and Patrick M. Wensing. 2023. Versatile Real-Time Motion Synthesis via Kino-Dynamic MPC With Hybrid-Systems DDP. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 9988–9994. https://doi.org/10.1109/ICRA48891.2023.10160221

[20] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. 2021. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2811–2817.

[21] L-Z Liao and Christine A Shoemaker. 1991. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Trans. Automat. Control* 36, 6 (1991), 692–706.

[22] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. 2020. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2536–2542. https://doi.org/10.1109/ICRA40945.2020.9196673

[23] David Q. Mayne. 1966. A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems. *Internat. J. Control* 3 (1966), 85–95.

[24] Jun Morimoto, Garth Zeglin, and Christopher G. Atkeson. 2003. Minimax differential dynamic programming: application to a biped walking robot. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, Vol. 2. 1927–1932 vol.2. https://doi.org/10.1109/IROS.2003.1248926

[25] Michael Posa, Mark Tobenkin, and Russ Tedrake. 2016. Stability Analysis and Control of Rigid-Body Systems With Impacts and Friction. *IEEE Trans. Automat. Control* 61, 6 (2016), 1423–1437. https://doi.org/10.1109/TAC.2015.2459151

[26] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. 2021. Blind bipedal stair traversal via sim-to-real reinforcement learning. In *Robotics: Science and Systems (RSS)*.

[27] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. 2011. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on MABEL. *The International Journal of Robotics Research* 30, 9 (2011), 1170–1193.

[28] Wei Sun, Yunpeng Pan, Jaein Lim, Evangelos A Theodorou, and Panagiotis Tsiotras. 2018. Min-max differential dynamic programming: Continuous and discrete time formulations. *Journal of Guidance, Control, and Dynamics* 41, 12 (2018), 2568–2580.

[29] Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4906–4913. https://doi.org/10.1109/IROS.2012.6386025

[30] Evangelos Theodorou, Yuval Tassa, and Emo Todorov. 2010. Stochastic Differential Dynamic Programming. In *Proceedings of the 2010 American Control Conference*. 1125–1132. https://doi.org/10.1109/ACC.2010.5530971

[31] Patrick M. Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. 2023. Optimization-Based Control for Dynamic Legged Robots. *IEEE Transactions on Robotics* (2023), 1–20. https://doi.org/10.1109/TRO.2023.3324580

[32] Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. 2003. Hybrid zero dynamics of planar biped walkers. *IEEE transactions on automatic control* 48, 1 (2003), 42–56.

[33] Zhaoming Xie, C. Karen Liu, and Kris Hauser. 2017. Differential dynamic programming with nonlinear constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 695–702. https://doi.org/10.1109/ICRA.2017.7989086

[34] Cheng Zhou, Yanbo Long, Lei Shi, Longfei Zhao, and Yu Zheng. 2023. Differential Dynamic Programming based Hybrid Manipulation

Strategy for Dynamic Grasping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 8040–8046. https://doi.org/10.1109/ICRA48891.2023.10160817