Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems

Quentin Le Lidec, Fabian Schramm, Louis Montaut, Cordelia Schmid, Ivan Laptev, Justin Carpentier

Leveraging Randomized Smoothing
for Optimal Control of Nonsmooth Dynamical Systems

Quentin Le Lidec[a,*], Fabian Schramm[a], Louis Montaut[a,b], Cordelia Schmid[a], Ivan Laptev[a], Justin Carpentier[a]

[a]*Inria and Département d'Informatique de l'Ecole Normale Supérieure, PSL Research University, Paris, France*
[b]*Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University, Prague, Czech Republic*

Abstract

Optimal control (OC) algorithms such as differential dynamic programming (DDP) take advantage of the derivatives of the dynamics to control physical systems efficiently. Yet, these algorithms are prone to failure when dealing with non-smooth dynamical systems. This can be attributed to factors such as the existence of discontinuities in the dynamics derivatives or the presence of non-informative gradients. On the contrary, reinforcement learning (RL) algorithms have shown better empirical results in scenarios exhibiting non-smooth effects (contacts, frictions, etc.). Our approach leverages recent works on randomized smoothing (RS) to tackle non-smoothness issues commonly encountered in optimal control and provides key insights on the interplay between RL and OC through the prism of RS methods. This naturally leads us to introduce the randomized Differential Dynamic Programming (RDDP) algorithm accounting for deterministic but non-smooth dynamics in a very sample-efficient way. The experiments demonstrate that our method can solve classic robotic problems with dry friction and frictional contacts, where classical OC algorithms are likely to fail, and RL algorithms require, in practice, a prohibitive number of samples to find an optimal solution.

Keywords: optimization, optimal control, reinforcement learning

---

*Corresponding author.
Email address:* `quentin.le-lidec@inria.fr` (Quentin Le Lidec)

1. Introduction

Theories and applications of optimal control (OC) and reinforcement learning (RL) are all related to the problem of minimizing a cost (resp. maximizing a reward) while fulfilling the system dynamics and constraints over a given time duration. Nonetheless, the resulting algorithms to solve OC or RL problems are based on different approaches, leading to very different performances in practice. On the one hand, in their vast majority, RL algorithms only exploit samples, leading to zero-th order approaches. On the other hand, optimal control and trajectory optimization techniques such as the iterative Linear Quadratic Regulator (iLQR) [1] and Differential Dynamic Programming (DDP) [2] rely on first-order and second-order linearization of the dynamics. Exploiting this derivative information makes them much more sample-efficient than their zero-th order counterparts from the field of RL. However, when considering complex scenarios such as robots interacting with their environments, the system dynamics may depict some non-smooth physical phenomena (dry friction, contact constraints, etc.). These properties may induce non-informative or discontinuous gradients that make gradient-based strategies fail [3]. On the contrary, RL algorithms have proven to be able to get around these non-smoothness issues in such cases, leading to impressive results when considering contact interactions [4]. By treating the dynamics as a black-box function, derivative-free algorithms such as standard RL methods circumvent the issues as mentioned above. They can transparently deal with arbitrarily complex and non-smooth dynamics. However, completely disregarding the specific structure of the dynamics comes at the cost of often requiring a large number of samples.

In a recent growing effort, differentiable physical simulators have emerged in the context of exploiting informative gradients for control [5] and estimation [6]. Gradients of rigid body dynamics were obtained by differentiating the classical rigid body algorithms [7, 8]. Simulating and differentiating physics with frictional contacts is more challenging as it requires to also solve for contact forces [5, 6] and was made possible by differentiable optimization techniques [9, 10]. Yet, these dynamics derivatives may present some discontinuities or lack of regularity, drastically impacting gradient-based optimization techniques, especially in the context of classical control algorithms [3].

In this work, we propose to leverage randomized smoothing (RS) techniques [11, 12] to cope with nonsmooth dynamical systems in optimal control problems. From a theoretical perspective, we notably demonstrate how RS
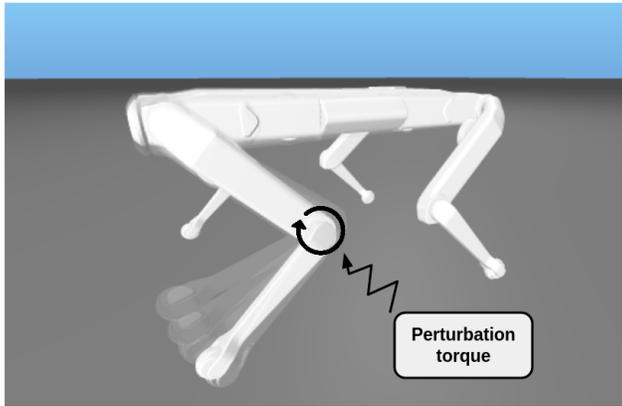
Figure 1: Illustration of randomized smoothing effects on the front left leg of the Solo robot.

methods applied to OC problems allow us to close the gap with the RL setting (Sec. 3) in general. A first connection of RL and OC for control tasks with infinite state and action spaces has been introduced by [13, 14] for the linear quadratic regulator with a focus on convergence proofs. From a practical perspective, we propose to use RS techniques within the frame of Differential Dynamic Programming to deal with nonsmooth dynamics, which are hard to handle in the vanilla setting (Sec. 4). We also introduce an adaptive strategy to automatically reduce the smoothing noise inherent to RS techniques across the optimization procedure. We experimentally show the practical benefits of our approach on various robotic tasks with increasing complexity, ranging from inverted pendulum to locomotion (Sec. 5).

Similarly to our work, [15] introduced the notion of randomized smoothing on the state and control spaces to get gradients from the dynamics through contacts. The same authors extended the approach to motion planning for complex manipulation tasks [16] by exploring the influence of zero-th and first-order gradient estimators. We take a different point of view by establishing links between RL and OC through the lens of RS, which is a first step towards a stronger interplay between these two fields. This viewpoint implies a practical difference: we interpret RS as an exploration term and only smooth the control space, leading to a sampled space of reduced dimension. Our experiments demonstrate that this is sufficient to perform well on complex robotics tasks.

2. Background

Our work builds on optimal control, reinforcement learning, and randomized smoothing techniques applied to robotics, which are briefly introduced in this section.

Optimal control. We consider the OC problem of controlling a robot by minimizing a cost l while satisfying the system dynamics f:

$$\min_{x,u} \int_0^T l(\tau, x(\tau), u(\tau))\mathrm{d}\tau \tag{1a}$$

$$\text{s.t. } \dot{x}(\tau) = f(x(\tau), u(\tau)), \tag{1b}$$

$$x(0) = \hat{x}_0, \tag{1c}$$

where $x(\tau) \in \mathcal{S}$ and $u(\tau) \in \mathcal{A}$ are the state and the control action of the system at time τ respectively, $\hat{x}_0 \in \mathcal{S}$ is a given initial state and T is the time horizon. In this paper, we call *smooth* a dynamics whose corresponding function f is differentiable everywhere. While in robotics f is often considered smooth, contact interactions give rise to points where f is only sub-differentiable [17].

Two different approaches may be used to solve (1), namely *direct* and *indirect* approaches. *Direct* approaches translate the infinite dimensional problem (1) into a finite nonlinear programming problem (NLP) and exploit off-the-shelf constrained optimization solvers [18] for solving it. In particular, this leads to a discrete numerical problem of the form:

$$\min_{\boldsymbol{x},\boldsymbol{u}} \overbrace{l_N(x_N) + \sum_{t=0}^{N-1} l_t(x_t, u_t)}^{J(\boldsymbol{x},\boldsymbol{u})} \tag{2a}$$

$$\text{s.t. } x_{t+1} = f(x_t, u_t), \ \forall t \in [0, N-1], \tag{2b}$$

$$x_0 = \hat{x}_0, \tag{2c}$$

where $\boldsymbol{x} = \{x_0, \ldots, x_N\}$ and $\boldsymbol{u} = \{u_0, \ldots, u_{N-1}\}$, with N being the number of time steps.

Alternatively, *indirect* approaches first apply the optimality conditions of optimal control problems (e.g., Hamilton-Jacobi-Bellman or Pontryagin's maximum principles) and then discretize these conditions in order to solve

the problems numerically. This notably offers the advantage of highlighting the underlying sparsity of constraints induced by times. In this line of work, iLQR [1] and DDP [19, 2, 20] are the most well-known first and second-order algorithms, with linear or quadratic-type convergence rates respectively. Moreover, their recent adaptations can even handle trajectory constraints [21, 22, 23] and implicit dynamics [24]. More closely related to our work, sampled DDP [25] lies in between, as it uses stochastic estimates of the dynamics derivatives to deal with dynamics for which gradients are unavailable.

Reinforcement learning considers the dynamics as a black-box function and uses parameterized stochastic policies π_θ to explore the action space better. This model-free approach can naturally handle complex or even unknown dynamics. During training, the cumulated sum of rewards R along the trajectories sampled from the distribution ρ_θ induced by the policy π_θ is maximized

the along the trajectories sampled from the distribution ρ_θ induced by the policy π_θ is minimized (which is equivalent to minimize the average cost $J = -R$), leading to the following problem:

$$\max_\theta \mathbb{E}_{(\boldsymbol{x},\boldsymbol{u})\sim\rho_\theta}\left[R(\boldsymbol{x},\boldsymbol{u})\right] \tag{3}$$

where trajectories are generated by the policy in the following way:

$$u_t \sim \pi_\theta(\cdot|x_t), \tag{4a}$$
$$x_{t+1} = f(x_t, u_t). \tag{4b}$$

Policy Gradient (PG) [26, 27] algorithms aim at maximizing (3) by using a zero-th order estimate of the gradient as a ascent direction:

$$\nabla_\theta R_{PG} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{u})\sim\rho_\theta}\left[R(\boldsymbol{x},\boldsymbol{u})\nabla_\theta \log \rho_\theta(\boldsymbol{x},\boldsymbol{u})\right] \tag{5}$$

The resulting randomness induces some variance in the gradient estimates, which slows down optimization and fosters exploratory behaviors, potentially leading to more global solutions. Finally, this makes RL an instance of random optimization, which will be discussed in more detail in Sec. 3.3.

Random optimization techniques are among the earliest optimization schemes and were developed in order to tackle problems where the objective function is discontinuous or nonsmooth. In this situation, the gradients

only provide limited information and are not suited for use in classic gradient-based optimization techniques [28]. During a random search, one classical strategy consists in sampling a random direction at each step and then moving towards the corresponding directional derivative [29], which is the basis of the Simultaneous Perturbation Stochastic Approximation algorithm [30]. Gradient Sampling (GS) is an alternative strategy that approximates the subgradient at non-differentiable points by taking the descent direction inside the convex hull of some gradients randomly sampled in a neighborhood [31]. Recent works [32] exploit the equivalence between random optimization techniques and perform stochastic gradient descent on a smoothed version of the original problem to get theoretical convergence bounds. In a parallel line of work, randomized smoothing was recently introduced in the machine learning community in order to be able to differentiate through Linear Programming (LP) *argmin* operators [33, 12]. It was also applied to make rendering [34], patch selection[35], clustering [36] and collision detection [37] operations differentiable. Unlike their classical counterparts, these perturbed optimizers are guaranteed to have non-null gradients everywhere, making it possible to use gradient-based optimization methods.

Concretely, let Z be a random variable whose probability distribution μ is a Gibbs measure. A function g can be approximated by convolving it with this probability distribution:

$$g_\epsilon(x) = \mathbb{E}_{Z\sim\mu}\left[g(x+\epsilon Z)\right] \tag{6}$$

which corresponds to the randomly smoothed counterpart of g and can be estimated with a Monte-Carlo estimator as follows:

$$g_\epsilon(x) \approx \frac{1}{M}\sum_{i=0}^{M} g(x+\epsilon Z^{(i)}) \tag{7}$$

where $\{Z^{(1)},\ldots,Z^{(M)}\}$ are i.i.d. samples and M is the number of samples. Using integration by part, we have the following expression of gradients:

$$\nabla_x g_\epsilon(x) = \mathbb{E}_{Z\sim\mu}\left[-g(x+\epsilon Z)\frac{\nabla\log\mu(Z)^\top}{\epsilon}\right] \tag{8a}$$

$$= \mathbb{E}_{Z\sim\mu}\left[\nabla g(x+\epsilon Z)\right], \tag{8b}$$

where (8a) and (8b) corresponds respectively to the zero-th and first order expressions of $\nabla_x g_\epsilon$. In practice, it is possible to approximate the smoothed

gradient by the first order Monte-Carlo (MC) estimator (9b) or, because $\mathbb{E}_{Z \sim \mu} \left[\nabla \log \mu(Z)^\top \right] = 0$, by the variance reduced zero-th order MC estimator (9a):

$$\nabla_x g_\epsilon(x) \approx \frac{1}{M} \sum_{i=1}^{M} \left(g(x) - g(x + \epsilon Z^{(i)}) \right) \frac{\nabla \log \mu(Z^{(i)})^\top}{\epsilon} \tag{9a}$$

$$\approx \frac{1}{M} \sum_{i=1}^{M} \nabla g(x + \epsilon Z^{(i)}) \tag{9b}$$

More intuitively, because of the local averaging effect of the convolution, $g_\epsilon(x)$ is always smoother than g. Indeed, g_ϵ is guaranteed to be differentiable [38], uniformly close from g and its gradient to be Lipschitz-continuous [11, 12]. Moreover, $g_\epsilon \xrightarrow{\epsilon \to 0} g$, so reducing the perturbation by decreasing ϵ leads to a reduced gap between g_ϵ and the original function g but also results in a less smooth approximation. In terms of computational complexity, evaluating ∇g_ϵ with M samples induces a complexity increased by a factor M. The computation being easily parallelizable, this, in fact, leads to a constant computational time without a critical impact on the memory footprint, as shown in the context of differentiable rendering in [34]. Adding stochasticity to gradients is also proven to help escape saddle points when optimizing non-convex functions [39], which constitutes a positive side effect of randomized smoothing. Finally, other previous works on the use of randomized smoothing demonstrate how it can improve convergence rates when optimizing non-smooth functions [11] and lead to more robust solutions [40].

Non-smooth dynamics in robotics. Generating movements for locomotion and manipulation is considered a major task in robotics. Whether it is to move itself or an object, the robot has to interact with its environment by making and breaking contacts, which induce non-smooth dynamics [41]. It is common to address these issues by making the hypothesis of bilateral contacts either between feet and the ground for locomotion [42] or between the fingers and the manipulated object [43] as detailed in Appendix 6. This hypothesis limits the range of possible movements by enabling only conservative strategies and fixing the contact modes during predetermined phases. RL approaches go beyond these constraints by not making any assumption on the underlying physical model and, hence, are said to be model-free. To go towards more dynamical movements, model-based approaches should be able to handle the switches between various contact modes better [44, 45].
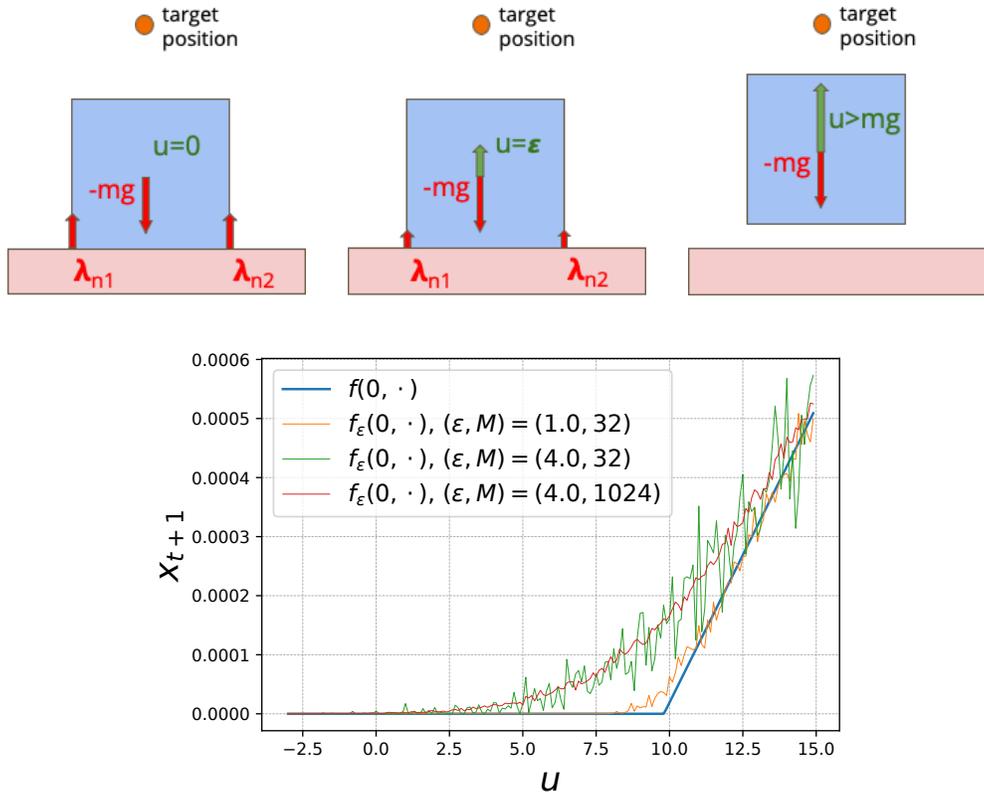
Figure 2: **Top:** A slight vertical force cannot break the unilateral contact, leaving the cube on the floor and, thus, the state unchanged. **Bottom:** The non-smoothness of physics induces null gradients $\nabla_{\boldsymbol{u}} J$ which results in the failure of classical optimization techniques (3.1).

Direct optimization of the sequence of contacts [46, 47] avoids the tedious manual engineering by shifting the burden to mixed-integer solvers. Alternatively, the seminal work from [48] exploits an analytical smoothing of the dynamics to use classical smooth optimization techniques. This work and the concurrent [15] propose to apply randomized smoothing on the dynamics before leveraging Trajectory Optimization algorithms.

3. Bridging the gap between optimal control and reinforcement learning

In this section, we present the caveats of poorly informative gradients for classical control algorithms, which may, for instance, occur in the presence of

nonsmooth dynamical systems. To overcome these limitations, we propose to exploit the randomized smoothing approach in the trajectory optimization paradigm. We also detail a connection between randomized smoothing and RL methods, thus explaining how they effectively solve problems involving poorly informative gradients.

3.1. Locally optimal solutions of optimal control problems

As discussed in Sec. 2, several approaches may be used for solving OC problems of the form (2). In particular, one can substitute $x_1, \ldots, x_N$ thanks to the constraint on the dynamics (11) and express problem (2) only in terms of $u_0, \ldots, u_{N-1}$, leading to the following but equivalent unconstrained optimization problem:

$$\min_{\boldsymbol{u}} \ J(\boldsymbol{x}(\boldsymbol{u}), \boldsymbol{u}) \tag{10}$$

where $\boldsymbol{x}(\boldsymbol{u})$ is recursively defined by:

$$x_0 = \hat{x}_0 \text{ and } x_t(\boldsymbol{u}) = f(x_{t-1}(\boldsymbol{u}), u_t), \tag{11}$$

corresponding to an integration process (e.g., exploiting a dynamical simulator). Unrolling the successive integration steps facilitates the efficient differentiation of $\boldsymbol{x}(\boldsymbol{u})$. Solving the equivalent problem (10) can be achieved using a classical unconstrained optimization algorithm such as gradient descent, consisting of backpropagating through time [49]. In the case of a robotic system, the dynamic simulator involves making and breaking contacts and switching between sticking and sliding contacts, which induces several modes in the dynamics. More formally, the computation of f results from a Nonlinear Complementarity Problem (NCP), making it inherently non-smooth as detailed in the Appendix 6 and [50, 51]. In addition, the cost function J, which often involves the robot kinematics via a penalty on the end-effector position as done in the experiments of Sec. 5, is non-convex. Thus, solving (2) with approaches described previously can lead to local solutions because of the inherent non-convexity and non-smoothness of the problem.

To illustrate this, one can think about the problem of lifting a cube [3] illustrated in Fig. 2. When $\boldsymbol{u}$ is initialized with null control, this results in $\nabla_{\boldsymbol{u}} f = 0$ because of the complementarity constraint arising from unilateral contacts (see IV. of [3]). By supposing that $\frac{\partial J}{\partial \boldsymbol{u}} = 0$ and applying the chain rule, we have that:

$$\nabla_{\boldsymbol{u}} J(\boldsymbol{x}(\boldsymbol{u}), \boldsymbol{u}) = \frac{\partial J}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} + \frac{\partial J}{\partial \boldsymbol{u}} = \frac{\partial J}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}}. \tag{12}$$

Moreover, since $\nabla_{\boldsymbol{u}} f = 0$, we have that starting from the first time step $\frac{\partial x_1}{\partial \boldsymbol{u}} = \frac{\partial f}{\partial \boldsymbol{u}}(x_0, u_0) = 0$ and with the time recursion, this leads to:

$$\frac{\partial x_{t+1}}{\partial \boldsymbol{u}} = \frac{\partial f}{\partial \boldsymbol{u}}(x_t, u_t) + \frac{\partial f}{\partial \boldsymbol{x}}(x_t, u_t)\frac{\partial x_t}{\partial \boldsymbol{u}} = 0, \tag{13}$$

implying that $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} = 0$. Finally, because $\nabla_{\boldsymbol{u}} J = 0$, an algorithm exploiting only the local gradient information (*e.g.,* gradient or Newton descent) will stop at this point, leaving the problem unsolved and blocked at a local maximum. The classical DDP algorithm would get stuck in a similar situation for the same reasons. It is also worth mentioning that even when equipped with proximal smoothing [52], such algorithms would still rely on local information and, thus, fail to overcome the issue of non-informative gradients.

3.2. Randomized smoothing of the system dynamics

The issue highlighted above is due to the inability of deterministic control algorithms to deal with non-smooth dynamics and their non-informative gradients, which often occur for physical systems involving contact or friction.

An intuitive way to circumvent this issue consists of introducing randomization in the optimization process in order to get a more exploratory behavior by collecting samples in a larger neighborhood around a given point when compared to classic methods such as the ones relying on local gradient information. This is precisely the motivation behind Randomized Smoothing and, to some extent, behind Reinforcement Learning, as discussed in Sec. 3.3. We adapt the formulation (2) by artificially smoothing the system dynamics using randomized smoothing, leading to the following smooth but approximated problem:

$$\min_{\boldsymbol{x},\boldsymbol{u}} J(\boldsymbol{x}, \boldsymbol{u}) \tag{14a}$$

$$\text{s.t. } x_{t+1} = f_\epsilon(x_t, u_t), \ \forall t \in [0, N-1], \tag{14b}$$

$$x_0 = \hat{x}_0. \tag{14c}$$

where $f_\epsilon(x, u) = \mathbb{E}_{Z \sim \mu}\left[f(x, u + \epsilon Z)\right]$ and μ is a noise distribution. It is worth mentioning that f_ϵ corresponds to a randomized version of f, which is only perturbed with respect to the control input u. Perturbing the control but not the state ensures that only reachable states are explored. When $\epsilon \to 0$, problem (14) converges to the original problem (2).

The proposed solution of using a smoothed approximation of the system dynamics is very generic. It can be easily instantiated in most of the existing

trajectory optimization frameworks without major modifications. Yet, there is *a priori* no obvious choice of the sampling distribution μ, neither for the number of particles sufficient in the Monte-Carlo estimator of the system dynamics and gradient computations. Another difficulty lies in the proper scheduling of the noise intensity ϵ towards 0 in order to remove the effect of noise at convergence to recover the original problem.

In Sec. 4, we introduce an algorithmic variation of the so-called Differential Dynamic Programming algorithm, which relies on the smoothed dynamics f_ϵ. In particular, we propose an automatic scheduling of the noise intensity ϵ and an auto-tuning strategy of the number of particles in the Monte-Carlo estimators.

3.3. Reinforcement learning through the prism of randomized smoothing

At this stage, one could wonder why RL demonstrated empirical success even in the case of the non-smooth dynamics evoked in 3.1. We provide the first possible explanation by drawing a parallel between the descent directions used in RL and the one from random optimization (8a) when applied to the OC problem (2).

Indeed, the ascent directions used in the classical RL algorithm REINFORCE with baseline (the same considerations remain valid for the closely related actor-critic algorithms) [53, 27, 54] can be written as:

$$\nabla_\theta R_{PG} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{u})\sim\rho_\theta}\left[\left(R(\boldsymbol{x},\boldsymbol{u}) - \hat{V}(\hat{x}_0)\right)\nabla_\theta \log \rho_\theta(\boldsymbol{x},\boldsymbol{u})\right], \qquad (15)$$

where $\hat{V}$ is an estimate of the value function, which exactly corresponds to the variance-reduced version of the randomly smoothed approximation (9a). More intuitively, in order to deal with the non-smoothness of the dynamics or reward functions, Policy Gradient adds noise in the action space by sampling actions from a stochastic policy. Concretely, this causes $\nabla_\theta R_{PG} \neq 0$ even in regions where $\nabla_{\boldsymbol{u}} R = 0$ and first or second order gradient methods fail, as discussed in Sec. 3.1. Thus, introducing some stochasticity allows RL to smooth the original problem and avoid the computation of gradients from the dynamics when they are unknown. Unfortunately, this generally comes at the cost of increased variance in the estimates of ∇R_{PG}, which induces a slower convergence rate [32]. A similar study could be done with Evolution Strategies used in [55, 56], which prefer to generate trajectories with deterministic policies but parameterized by randomly sampled parameters.

Following this analysis, in a way similar to RL, we propose using randomized smoothing to compute informative gradients even when the standard ones are non-informative. The obtained gradients can then be exploited in the context of optimal control problems with higher-order optimization algorithms in order to get improved convergence rates, as shown in the context of the widely used Differential Dynamic Programming algorithm in the following section.

In earlier works, [13, 14] derive a convergence proof for policy iteration methods in a deterministic, linear, and time-variant LQR setup for infinite state and action spaces. An algorithm was presented, that converges to the optimal policy and involves some additional exploration noise in the control due to a requirement in the underlying recursive least-square algorithm. However, the noise has a very different motivation from our usage of injected noise for randomized smoothing of non-smooth function surfaces, which is done by averaging over several noise samples.

4. Randomized Differential Dynamic Programming

This section introduces our randomized Differential Dynamic Programming algorithm. This novel formulation builds on the previous analysis to incorporate randomized smoothing in the optimal control paradigm in order to increase the exploration of classical DDP and efficiently solve problems involving non-smooth dynamical systems.

4.1. Dynamic programming with smoothed physics

The main idea behind randomized DDP consists in exploiting the formulation (14) and replacing the original, possibly non-smooth, dynamics f by its randomly smoothed approximation f_ϵ in the forward and backward passes of the vanilla DDP. Doing so allows to benefit from the efficiency of DDP even in situations where the original DDP algorithm will fail.

First, we introduce the cost-to-go function associated to our problem (14):

$$J_t(x_t, u_t, \ldots, u_{N-1}) = l_N(x_N) + \sum_{j=t}^{N-1} l_j(x_j, u_j) \tag{16}$$

Algorithm 1: Randomized DDP algorithm

Input: OC problem: J, f, initial trajectory: $\boldsymbol{x}, \boldsymbol{u}$, target noise and precision: ϵ^*, α^*, initial noise and precision: ϵ, α, adaptive scheme parameters: ρ, γ

Output: Solution $\boldsymbol{x}, \boldsymbol{u}$ of the OC problem (2)

1 repeat
2   **repeat**
3 $k, K \leftarrow$ **Backward Pass** (19);
4 $\boldsymbol{x}, \boldsymbol{u} \leftarrow$ **Forward Pass** (22);
5   **until** $\|Q_u\|_{Q_{uu}^{-1}} < \alpha$;
6 $\epsilon \leftarrow \epsilon / \rho$;
7 $\alpha \leftarrow \alpha / \gamma$;
8 until $\alpha < \alpha^*$ and $\epsilon < \epsilon^*$;

and the value function which verifies the Bellman's equation:

$$V_t(x_t) = \min_{u_t,\ldots,u_{N-1}} J_t(x_t, u_t, \ldots, u_{N-1}) \tag{17a}$$

$$= \min_{u_t} l_t(x_t, u_t) + V_{t+1}\left(f_\epsilon(x_t, u_t)\right) \tag{17b}$$

with the terminal condition $V_N(x) = l_N(x)$. Additionally, the Q-function is defined by:

$$Q_t(x, u) = l_t(x, u) + V_{t+1}\left(f_\epsilon(x, u)\right) \tag{18}$$

As done in the classical Differential Dynamic Programming algorithm, we exploit the sparsity of constraints induced by time via Bellman's equation to solve the problem (14). To do so, local second-order approximations of the value and the Q functions are built by backpropagating the Bellman's equation (17) backward in time around a reference trajectory $\overline{\boldsymbol{x}}$, leading to
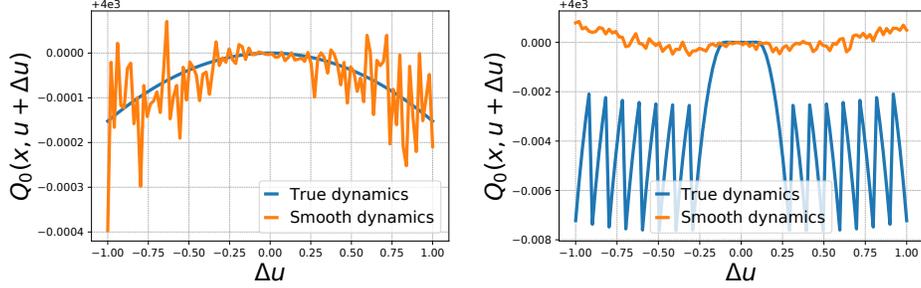
13

Figure 3: **Left:** The randomly-smoothed dynamics allow to get non-null gradients and escape the local maximum for the inverted pendulum. **Right:** $Q_0(x, \cdot)$ value function of the pendulum with dry friction exhibits plateaus where $Q_u = 0$ around $u = 0$ while randomly smoothed dynamics leads to $Q_u \neq 0$.

the backward pass equations:

$$Q_{xx} = l_{xx} + f_x^\top V'_{xx} f_x + V'^\top_x f_{xx} \tag{19a}$$

$$Q_{ux} = l_{ux} + f_u^\top V'_{xx} f_x + V'^\top_x f_{ux} \tag{19b}$$

$$Q_{uu} = l_{uu} + f_u^\top V'_{xx} f_u + V'^\top_x f_{uu} \tag{19c}$$

$$Q_x = l_x + V'^\top_x f_x \tag{19d}$$

$$Q_u = l_u + V'^\top_x f_u \tag{19e}$$

$$q = l + v', \tag{19f}$$

where $v = V_t(x)$, and and the subscript on x and u are the usual notations for the partial derivative w.r.t the state and control variables, and the superscript V' denotes the value function at the next time-step. Minimizing these local quadratic approximations of Q w.r.t u gives:

$$u = k + K(x - \bar{x}), \ k = -Q_{uu}^{-1}Q_u \text{ and } K = -Q_{uu}^{-1}Q_{ux}, \tag{20}$$

and injecting (20) in (17) gives rise to:

$$V_{xx} = Q_{xx} - K^\top Q_{uu} K \tag{21a}$$

$$V_x = Q_x - k^\top Q_{uu} K \tag{21b}$$

$$v = q - \frac{1}{2} k^\top Q_{uu} k. \tag{21c}$$

14

Finally, u is updated with a line search during the forward computation:

$$u_t^n = u_t^{n-1} + \alpha k + K(x_t^n - \overline{x_t}) \tag{22a}$$

$$x_{t+1}^n = f_\epsilon(x_t^n, u_t^n) \tag{22b}$$

with the initial condition $x_0^n = \hat{x}_0$ and where the superscript n denotes the number of optimization iterations. During line search, the noise is fixed as described in [57, 58]. Repeating the forward and backward steps by taking the new trajectory $\boldsymbol{x}^n$ as the new reference $\overline{\boldsymbol{x}}$ allows to find the optimal control $\boldsymbol{u}$ under the form of a linear policy (20) which is optimal around the trajectory.

As detailed previously in Sec. 2, smoothing the physics makes it possible to have non-null gradients $\nabla_u f_\epsilon$ thus inducing non-null Q_u, as illustrated in Fig. 3. Alternatively, the stochasticity can also be interpreted as an exploration term that has proven to be crucial in the RL framework to escape regions where the local information from gradients does not provide exploitable insight on the problem being solved (see Sec. 3.3).

The main difference between our approach formulated at (14) and "Policy Gradient"-type algorithms lies in the scope of the randomized smoothing. Indeed, RL smooths the whole problem while we "only" smooth the dynamics f: note that the expectation is on entire trajectories in (3) while it is at the time-step level in (14) resulting in a reduced variance in the latter case [59]. Moreover, we find that preserving the original recursive structure of (2) is beneficial in the case of known dynamics f as it allows to benefit from the efficient dynamic programming backward passes (19) of DDP. While RL requires to also smooth the cost function to deal with sparse rewards, it is not necessary in our case as current trajectory optimization algorithms can handle hard constraints [22, 24, 21].

4.2. Adaptive smoothing

To enforce the convergence towards an optimal (local) solution, it remains crucial to reduce the noise injected via the randomized smoothing across the iterations. A first possible strategy [15] consists in relying on Robbins-Monro rule [60] by decreasing the variance in a way such that $\sum_k \epsilon_k^2 < \infty$ to guarantee convergence towards a local minimum. In this work, we propose to decrease ϵ_k in a way that adapts to the problem and avoids the smoothing being reduced too quickly, which would lead to performance similar to classical DDP, or too slowly, which would induce an unnecessarily large number
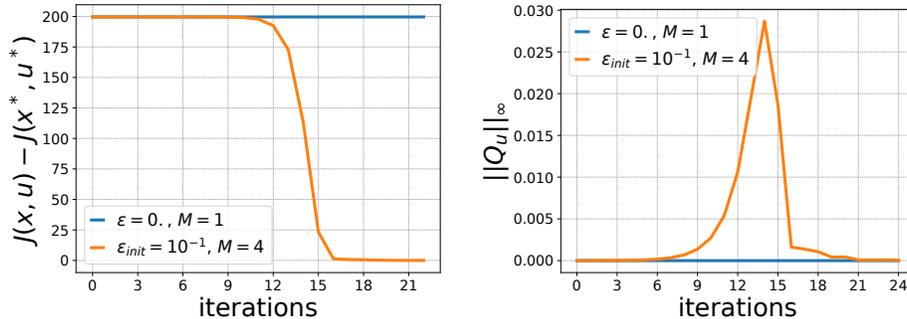
15

Figure 4: Randomized smoothing allows to escape from the local optima $(x, u) = 0$ for the inverted pendulum (see Fig 3) and the system reaches the upward position (orange). Without randomized smoothing, the gradients are null and the system remains stuck in the downward position (blue).

of iterations. We adapt the smoothing by solving a cascade of randomly smoothed DDP problems. More concretely, $\|\nabla_u Q\|_\infty$ decreases towards 0 when converging towards an optimum. Whenever $\|\nabla_u Q\|_\infty$ is under a given precision threshold α we consider the smooth sub-problem solved and thus reduce the noise intensity ϵ and the precision threshold α, by a factor ρ and γ respectively, before solving the next sub-problem. Typically, ρ and γ are taken equal and set to the value 2. This adaptive scheme is summarized in Alg. 1 and is generic, so it could be transferred to any algorithm using randomized smoothing.

5. Experiments

In this section, we demonstrate the practical benefits of our randomized DDP algorithms on a set of different systems, ranging from simple toy examples to real robotic systems. We first examine the underlying mechanisms and pendulum and a cube and extend them afterward to a quadrotor and a quadrupedal robot. Our implementation is based on the open-source frameworks Crocoddyl [20] for the DDP algorithm and on Pinocchio [61] and [8, 6] for the derivatives of the dynamics with and without contacts.

5.1. Avoiding local optima of smooth dynamics

We consider the task of raising a pendulum from the downward $(\theta = 0)$ to the upward vertical position $(\theta = \pi)$. The system's state is characterized by

the angle θ, representing the deviation of the pendulum bar from the vertical axis, and by the angular velocity $\dot{\theta}$. The tip's position also called the end-effector, can be determined through a trigonometric relation and is denoted as $p(\theta)$, where p^* is the desired position. We optimize the cost function

$$J(\boldsymbol{x}, \boldsymbol{u}) = w_p \|p(\theta_N) - p^*\|^2 + \sum_{t=0}^{N} w_u \|u_t - u^*\|^2, \qquad (23)$$

over $N = 400$ time steps and each with a duration $dt = 5 \times 10^{-3}$ s. The weights are defined as $w_p = 2$, $w_u = 2 \times 10^{-5}$ and the system is described using the generalized coordinates $x = (\theta, \dot{\theta})$ of position and velocity. As explained previously, due to the robot kinematics, the term of J involving the distance on the end-effector $\|p(\theta_N) - p^*\|$ is non-convex leading to multiple potential local optima of the control problem. We run 24 iterations of the classical DDP and the RDDP algorithm optimizing for a trajectory with 400 time steps to swing the pendulum up. For the problem (23), the solution $(u, x) = (0, 0)$ is a local extremum, and the classical DDP algorithm gets stuck at this point (Fig. 4). As discussed in Sec. 3.3, randomized smoothing using a smaller number of only $M = 4$ samples per control input to smooth and average the resulting state allows RDDP to avoid this local optimum (Fig. 3,4). Here, two distinct effects are at work: i) the Randomized Smoothing can smoothen out some local optima, and ii) the noise from the Monte-Carlo estimator helps to escape from unstable critical points as detailed in [39]. In addition to converging towards better optima in the case of smooth dynamics, we primarily designed RDDP to provide a solution when it comes to the aforementioned issues from non-smooth dynamics (see Sec. 3.1 and Appendix 6), shown in the next section.

5.2. Controlling non-smooth dynamics

To illustrate the issues induced by the non-smoothness of unilateral contact and frictions, we consider the preliminary tasks of taking off or sliding a cube on a table (appearing in [3]). In these experiments, we use a cost function similar to the one of (23), optimizing the difference between a target configuration and the final configuration of the system at the end of the planned trajectory. As shown by Fig. 6 (left), our approach allows us to complete this task while we have shown in Sec. 3.1 that it is impossible when relying on classic gradient information.
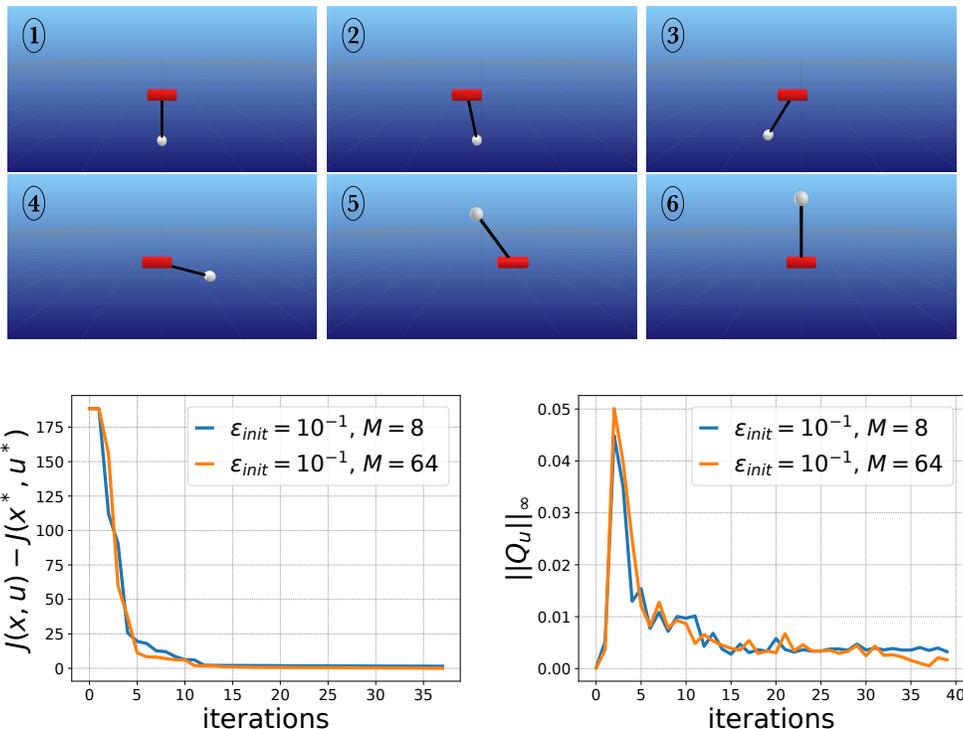
Figure 5: **Top:** Application of RDDP to precisely control a cartpole movement requiring overcoming dry friction on the joints and moving the pole straight upward. **Bottom:** Using RDDP for the cartpole task, we compare the influence of the sample size on the solution. Only a few samples (here M=8) are necessary to get reasonable results. Increasing the number of samples in the first order Monte-Carlo estimator of the gradients leads to very marginal improvement.

Similarly, it is possible to apply the RDDP algorithm to control more complex systems such as a double pendulum and a cartpole with dry friction on the joints (Fig. 5). The double pendulum and the cartpole are illustrative instances of underactuated systems whose dynamics are non-smooth due to friction. In both cases, classical DDP fails to optimize the control variable using the true dynamics. Once again, smoothing the dynamics with respect to the control u using M samples per time step allows to have non-null gradients almost everywhere and thus to apply classical DDP (Fig. 6, middle and right). Combining randomized smoothing with the DDP algorithm makes it possible to solve the problem of controlling a complex system involving contacts and friction very efficiently. Interestingly, the FDDP algorithm [20], which
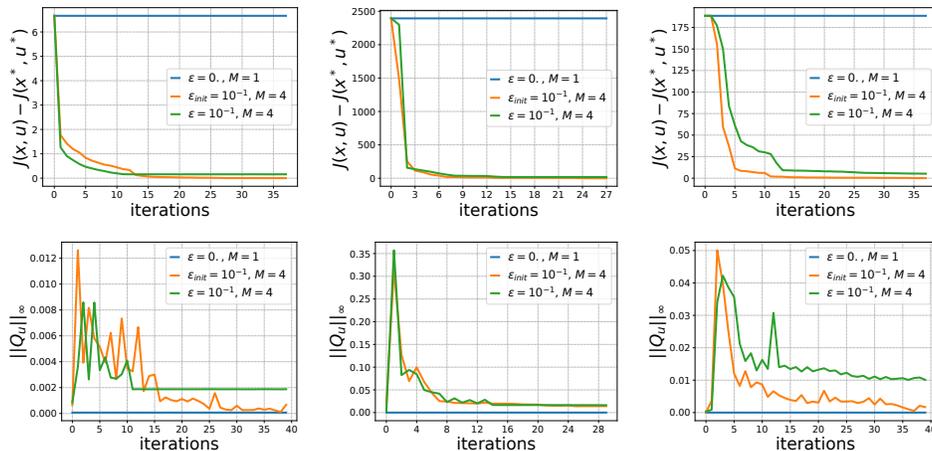
Figure 6: This figure shows the effect of our adaptive noise scheduling (orange) RDDP in comparison against fixed noise (green) RDDP and classical DDP with no noise (blue) for three different tasks. **Left:** RDDP solves tasks requiring breaking contacts such as the cube-lifting from Fig. 2 **Middle:** Similarly, the approach solves classical control problems on underactuated systems such as an inverted double pendulum or, **Right:** a cartpole with dry frictions on the joints. While both RDDP variants solve the task contrary to classical DDP, observe the improvement on the convergence criteria $\|Q_u\|_\infty$ for the adaptive noise scheduling (orange).

allows for unfeasible trajectories, also failed to overcome the non-smoothness from dry frictions. We also studied the influence of the number of samples M used for the MC estimators (Fig. 5) and the adaptive scheme (Fig. 6) on the quality of the obtained solution. We noticed that good results could be obtained with a relatively small number of samples M for the first-order MC gradient estimator, meaning that the supplementary computational costs are limited when compared to classical DDP. On the contrary, using an adaptive scheme for decreasing the smoothing perturbation leads to a more precise solution by allowing the gradients of the trajectory optimization problems to converge to zero (Fig. 6). This represents an advantage of our method when compared to classical RL approaches, as the latter requires keeping a non-null noise to estimate gradients, preventing them from converging very precisely.

5.3. Controlling contact interactions in robotics

The following experiments explore the application of our method to control complex robotics systems interacting with their environments via fric-
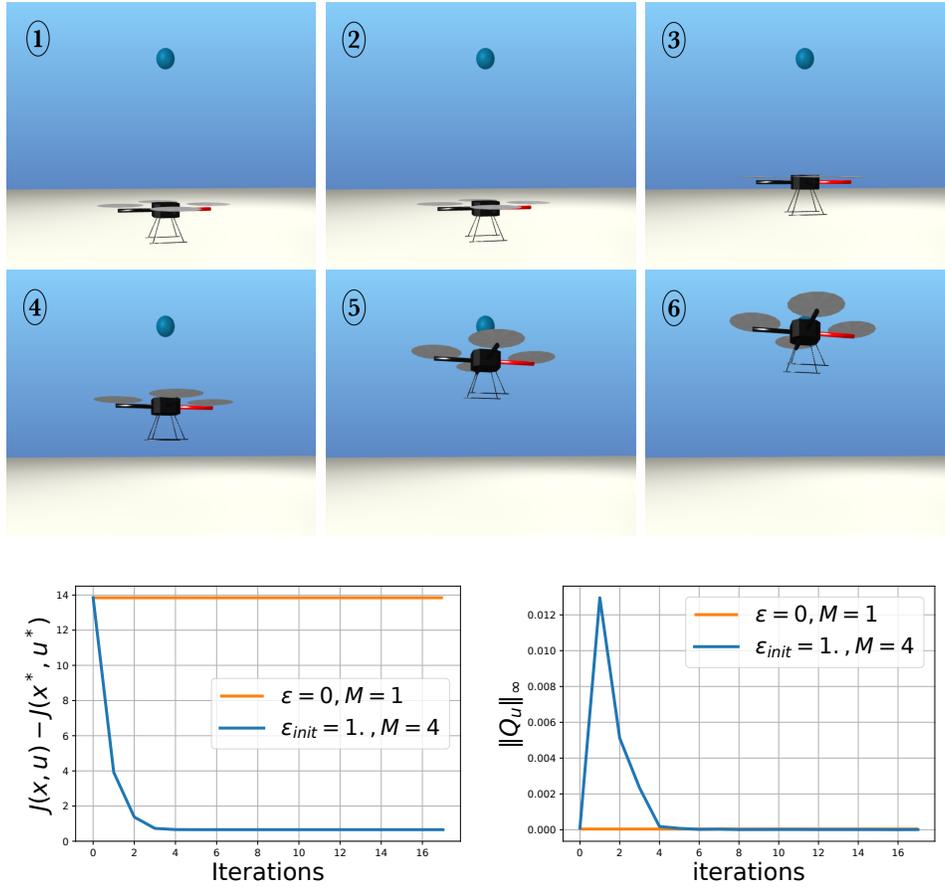
Figure 7: Comparison of RDDP (blue) against classical DDP (orange) to plan the take-off of a drone. **Top:** Sequence for take-off achieved by RDDP which allows to reach the goal position marked with a blue sphere on the pictures. **Bottom:** On the opposite, unilateral contacts induce null gradients which lead to the failure of classical DDP.

tional contacts.

Quadrotor. In our first experiment, we control the takeoff of a quadrotor. The drone starts in an initial state, landing on the ground, and the goal is to reach a target position one meter higher in the air. This case of a drone with four actuated thrusts represents a realistic and challenging robotics task as one could need to control its trajectory without having to plan for takeoff and landing phases, which would require breaking or making unilateral contacts. Concretely, the cost of the control problem J is similar to (23) but with p
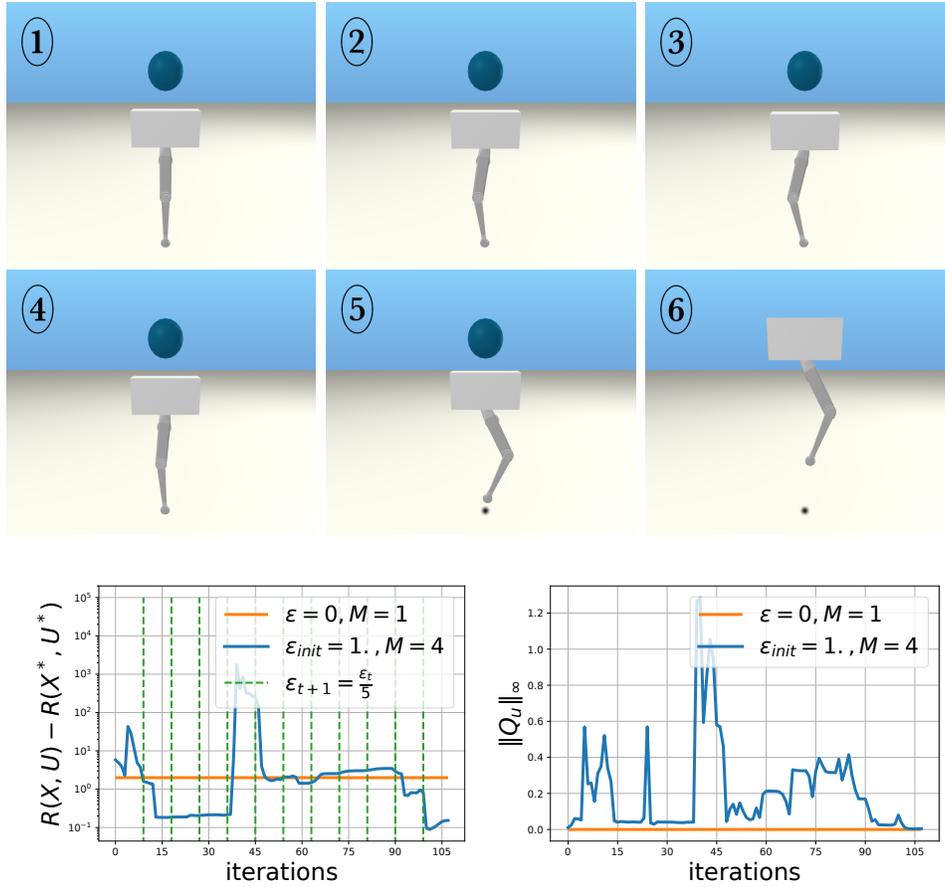
Figure 8: RDDP allows executing a single-legged jump (on a vertical plane) towards a target position marked with a blue sphere. **Top:** The leg initially bends downwards and then extends to perform the jumping motion. **Bottom:** The cost J and the norm $\|Q_u\|$ are constant when using classical DDP (orange) while RSDDP (blue) achieves to lower the cost. The vertical green lines denote updates in the noise level used in randomized smoothing.

now denoting the position of the base of the quadrotor with $w_p = 4$ and $w_u = 10^{-3}$. As shown by Fig. 7, the classical DDP algorithm fails due to the issue of non-informative gradients whenever the drone rests on the ground. In contrast, our approach is quickly (less than 10 optimization steps) able to generate a trajectory that accurately reaches the target position.

Solo robot. Finally, we apply our algorithm to solve a task on a legged

robotic system with frictional contacts. We use one leg of the Solo robot [62] similar to [63], which results in a hopper with 3 degrees of freedom, namely the hip and knee joint, and a vertical translation of the base. Therefore, the state is fully described by the position $q \in \mathbb{R}^3$ and velocity $\dot{q} \in \mathbb{R}^3$. The system is underactuated as only the knee and the hip of the robot are actuated. The task is to make the system jump up from an initial stretched configuration (Fig. 8, **Left**) to a position 30cm above the ground along the vertical axis using a similar formulation for the control cost R as in (23). Without randomized smoothing, the DDP algorithm has no gradient information in this configuration; the algorithm converges to a local optimum, and the system is not moving. On the contrary, using RDDP achieves solving the task by first bending the leg downward (Fig. 8, 3rd image), to be then able to apply a control torque that leads to contact forces with the ground moving the leg up to the target position. It is worth noting that this movement was generated without pre-specifying contact phases, as the optimizer does the contact planning entirely. This represents an encouraging first step towards automatically generating locomotion movements via trajectory optimization.

6. Conclusion

Analyzing reinforcement learning via the theory of randomized smoothing allows us to understand how crucial the exploratory characteristic of these algorithms is to solve control problems with non-smooth dynamics. By transferring these ideas to the field of trajectory optimization, we have leveraged randomized smoothing in this paper to propose an approximate and smooth formulation of the original optimal control problem. Exploiting this new formulation with the well-established DDP algorithm results in an approach that can cope with the presence of contact interactions and friction in a sample-efficient manner. We've also demonstrated the capacity of our method to properly cope with standard robotics systems (pendulum, cart pole, drone, Solo robot), including non-smooth dynamical effects (frictions, contacts) where classic optimization-based are likely to fail. In future work, we plan to investigate possible uses of the information contained in the variance of the several particles of the Monte-Carlo estimator. In particular, we believe this could help to build a more robust adaptive scheme by detecting when increasing the noise ϵ, the precision threshold α, or even the number of particles M is necessary.

References

[1] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[2] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[3] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics engine for articulated rigid bodies with contact constraints," in *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, D. A. Shell, M. Toussaint, and M. A. Hsieh, Eds., 2021. [Online]. Available: https://doi.org/10.15607/RSS.2021.XVII.034

[4] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[5] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/842424a1d0595b76ec4fa03c46e8d755-Paper.pdf

[6] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, 2021.

[7] R. Featherstone, *Rigid body dynamics algorithms.* Springer, 2014.

[8] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems (RSS 2018)*, 2018.

[9] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning.* PMLR, 2017, pp. 136–145.

[10] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9562–9574, 2019.

[11] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 674–701, 2012.

[12] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach, "Learning with differentiable pertubed optimizers," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 9508–9519. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/6bb56208f672af0dd65451f869fedfd9-Paper.pdf

[13] S. Bradtke, "Reinforcement learning applied to linear quadratic regulation," *Advances in neural information processing systems*, vol. 5, 1992.

[14] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, "Adaptive linear quadratic control using policy iteration," in *Proceedings of 1994 American Control Conference-ACC'94*, vol. 3. IEEE, 1994, pp. 3475–3479.

[15] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4000–4007, 2022.

[16] T. Pang, H. J. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on Robotics*, pp. 1–21, 2023.

[17] B. Brogliato, *Nonsmooth mechanics.* Springer, 1999.

[18] 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.

[19] 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.

[20] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2020, pp. 2536–2542.

[21] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2019, pp. 7674–7679. [Online]. Available: https://ieeexplore.ieee.org/document/8967788/

[22] S. Kazdadi, J. Carpentier, and J. Ponce, "Equality constrained differential dynamic programming," in *2021-IEEE International Conference on Robotics and Automation*, 2021.

[23] W. Jallet, A. Bambade, E. Arlaud, S. El-Kazdadi, N. Mansard, and J. Carpentier, "PROXDDP: Proximal Constrained Trajectory Optimization," Dec. 2023, working paper or preprint. [Online]. Available: https://inria.hal.science/hal-04332348

[24] W. Jallet, N. Mansard, and J. Carpentier, "Implicit Differential Dynamic Programming," in *2022 International Conference on Robotics and Automation (ICRA).* Philadelphia, United States: IEEE Robotics and Automation Society, May 2022.

[25] J. Rajamäki, K. Naderi, V. Kyrki, and P. Hämäläinen, "Sampled differential dynamic programming," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1402–1409.

[26] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[28] J. Matyas *et al.*, "Random optimization," *Automation and Remote control*, vol. 26, no. 2, pp. 246–253, 1965.

[29] B. Polyak, *Introduction to Optimization*, 07 2020.

[30] J. C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, 2005.

[31] J. V. Burke, A. S. Lewis, and M. L. Overton, "A robust gradient sampling algorithm for nonsmooth, nonconvex optimization," *SIAM Journal on Optimization*, vol. 15, no. 3, pp. 751–779, 2005.

[32] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[33] J. Abernethy, C. Lee, and A. Tewari, "Perturbation techniques in online learning and optimization," *Perturbations, Optimization, and Statistics*, p. 233, 2016.

[34] Q. Le Lidec, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable Rendering with Perturbed Optimizers," in *Neural Information Processing Systems*, Sydney, Australia, Dec. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03378451

[35] J.-B. Cordonnier, A. Mahendran, A. Dosovitskiy, D. Weissenborn, J. Uszkoreit, and T. Unterthiner, "Differentiable patch selection for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2351–2360.

[36] L. Stewart, F. S. Bach, F. L. López, and Q. Berthet, "Differentiable clustering with perturbed spanning forests," *arXiv preprint arXiv:2305.16358*, 2023.

[37] L. Montaut, Q. Le Lidec, A. Bambade, V. Petrik, J. Sivic, and J. Carpentier, "Differentiable collision detection: a randomized smoothing approach," in *2023 - IEEE International Conference on Robotics and Automation (ICRA)*, London, May 2023. [Online]. Available: https://arxiv.org/abs/2209.09012

[38] D. P. Bertsekas, "Stochastic optimization problems with nondifferentiable cost functionals," *Journal of Optimization Theory and Applications*, vol. 12, no. 2, pp. 218–231, 1973.

[39] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points—online stochastic gradient for tensor decomposition," in *Conference on learning theory*. PMLR, 2015, pp. 797–842.

[40] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

[41] B. Brogliato, "Modeling, analysis and control of robot–object nonsmooth underactuated lagrangian systems: A tutorial overview and perspectives," *Annual Reviews in Control*, vol. 55, pp. 297–337, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578822001390

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

[43] M. T. Mason and J. K. Salisbury Jr, "Robot hands and the mechanics of manipulation," 1985.

[44] J. Carpentier and P.-B. Wieber, "Recent progress in legged robots locomotion control," *Current Robotics Reports*, vol. 2, no. 3, pp. 231–238, 2021.

[45] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, 2023.

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

[47] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.

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

[49] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[50] V. Acary, M. Brémond, and O. Huber, "On solving contact problems with Coulomb friction: formulations and numerical comparisons," INRIA, Research Report RR-9118, Nov. 2017. [Online]. Available: https://hal.inria.fr/hal-01630836

[51] Q. L. Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, "Contact models in robotics: a comparative analysis," 2023.

[52] N. Parikh, S. Boyd, *et al.*, "Proximal algorithms," *Foundations and trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[53] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning." *Journal of Machine Learning Research*, vol. 5, no. 9, 2004.

[54] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[55] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[56] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 1805–1814.

[57] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien, "Painless stochastic gradient: Interpolation, linesearch, and convergence rates," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips. cc/paper/2019/file/2557911c1bf75c2b643afb4ecbfc8ec2-Paper.pdf

[58] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[59] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[60] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400 – 407, 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729586

[61] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *International Symposium on System Integration (SII)*, 2019.

[62] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[63] J. Viereck, J. Kozolinsky, A. Herzog, and L. Righetti, "Learning a structured neural network policy for a hopping task," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4092–4099, 2018.

Appendix

This section presents the physical principles commonly used for rigid body simulation with point contact to show where non-smoothness and non-convexity arise during the computation of f in (11). In general, we describe the state of a system with its generalized coordinates $q \in \mathcal{Q} \cong \mathbb{R}^{n_q}$ and denote by $v \in \mathcal{T}_q \mathcal{Q} = \mathbb{R}^{n_v}$ the joint velocity. The principle of least action states that the path followed by a dynamical system should minimize the action functional which induces the Lagrangian equations of motion:

$$M(q)\dot{v} + C(q,v)v + g(q) = \tau \tag{.1}$$

where $M \in \mathbb{R}^{n_v \times n_v}$ represents the joint space inertia matrix of the system, $C(q,v)v$ accounts for the centrifugal and Coriolis effects, and g is the generalized gravity. In the following, we express the problem in terms of velocities rather than acceleration, thus discretizing (.1) to

$$Mv_{t+1} = Mv_t + (\tau - Cv - g)\Delta t \tag{.2}$$

where C and g are evaluated explicitly at (q_t, v_t) We note v_f the free velocity which is defined as the solution of (.2).
Constrained dynamics for robotic systems with kinematic loops or that establish contacts with the environment can define these constraints implicitly via

$$\Phi(q) = 0 \tag{.3}$$

where $\Phi : \mathbb{R}^{n_q} \to \mathbb{R}^m$ is the implicit constraint function of dimension m depending on the nature of the constraint. (.3) is then derived w.r.t time to express the constraint as a constraint on joint velocities in the sequel:

$$c - c^* = 0 \tag{.4}$$

where $c = Jv_{t+1}$ is the constraint velocity, $J = \frac{\partial \Phi}{\partial q}$ is the constraint Jacobian and c^* is the reference velocity of the constraint which is set to either model physical effects or stabilize the numerics. Such a constraint is enforced by the

action of the environment on the system via the contact vector impulse $\lambda \in \mathbb{R}^m$ and should be incorporated into the Lagrangian equations of motion (.1):

$$Mv_{t+1} = Mv_f + J^T \lambda \tag{.5}$$

Unilateral contacts are usually used when a system is in contact with its environment and enforce the normal component of the separation vector, i.e. the signed distance function, to be non-negative:

$$\Phi(q)_N \geq 0 \tag{.6}$$

where $\Phi(q) \in \mathbb{R}^{3 \times n_c}$, n_c is the number of contacts, and the subscripts N and T respectively account for the normal and tangential components. Similarly to (.4), this can be expressed as:

$$c_N - c_N^* \geq 0 \tag{.7}$$

where $c = Jv_{t+1} \in \mathbb{R}^{3 \times n_c}$ is the velocity of contact points and c_N^* is set to model physical effects or improve the numerical accuracy of the solutions (commonly referred to as Baumgarte stabilization). Unilateral contacts constrain the possible efforts λ depending on the state of the system and the friction coefficient. In general, the contact forces λ can only be repulsive as they should not act in a glue-like fashion (the environment can only push and not pull on the feet of a legged robot) and, thus, are forced to be non-negative. An impulse cannot occur when an object takes off, implying that the normal velocity and impulse cannot be non-null simultaneously.

In a frictionless situation, the tangential forces are null, which implies that $\lambda_T = 0$. Combining these conditions, we obtain the so-called Signorini condition:

$$0 \leq \lambda_N \perp c_N - c_N^* \geq 0 \tag{.8}$$

However, such a condition does not define a mapping between λ_N and c_N, i.e., the contact forces are not a function of the penetration error. Indeed, its representation is an infinitely steep graph and this problem is referred to as a Linear Complementarity Problem (LCP). Therefore, the computation of f in (11) which corresponds to the computation of v_{t+1} in (.5) and q_{t+1} via integration is facing the issues of non-smoothness and non-convexity.

If friction is included, additional conditions on λ_T, i.e. lying in a friction cone and satisfying the maximum dissipation principle, lead to a reformulation of the problem as a nonlinear complementarity problem (NCP), see [51] for further details.