Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems

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

Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems

Quentin Le Lidec[†], Louis Montaut[†⋆], Cordelia Schmid[†], Ivan Laptev[†] and Justin Carpentier[†]

***Abstract*— Optimal control (OC) algorithms such as Differential Dynamic Programming (DDP) take advantage of the derivatives of the dynamics to efficiently control physical systems. Yet, in the presence of nonsmooth dynamical systems, such class of algorithms are likely to fail due, for instance, to the presence of discontinuities in the dynamics derivatives or because of non-informative gradient. 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 (R-DDP) algorithm accounting for deterministic but non-smooth dynamics in a very sample-efficient way. The experiments demonstrate that our method is able to 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.**

***Index Terms*— optimization, optimal control, reinforcement learning**

I. 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 one hand, RL algorithms in their vast majority only exploits 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 scenarii such as robots in interaction 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 aforementioned issues and 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.

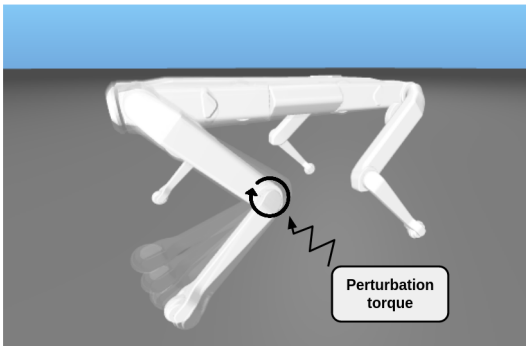

Fig. 1: Illustration of randomized smoothing effects on the front left leg of the Solo robot.

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 bodies 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, which may drastically impact 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 methods applied on OC problems allow to close the gap with the RL setting (Sec. III). 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. IV). 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

[†]The authors are with Inria and Département d'Informatique de l'Ecole Normale Supérieure, PSL Research University, Paris, France. firstname.lastname@inria.fr

[⋆] L. Montaut is with Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague.

with increasing complexity, ranging from inverted pendulum to quadrupedal locomotion(Sec. V).

Concurrently to our work, [13] introduced the notion of randomized smoothing in order to get gradients through contacts. 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.

II. Background

Our work builds on optimal control, reinforcement learning and randomized smoothing techniques, 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:

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

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

$$x(0) = \overline{x_0}, \tag{1c}$$

where $x(t) \in \mathcal{S}$ and $u(t) \in \mathcal{A}$ are the state and the control action of the system at time t respectively, 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 [14].

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 [15] for solving it. In particular, this leads to a discrete numerical problem of the form:

$$\min_{X,U} \quad \overbrace{l_T(x_T) + \sum_{t=0}^{T-1} l_t(x_t, u_t)}^{R(\boldsymbol{x},\boldsymbol{u})} \tag{2a}$$

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

$$x_0 = \overline{x_0}, \tag{2c}$$

where $\boldsymbol{x} = \{x_0, \dots, x_T\}$ and $\boldsymbol{u} = \{u_0, \dots, u_{T-1}\}$.

Alternatively, *indirect* approaches first apply the optimality conditions of optimal control problems (e.g. Hamilton-Jacobi-Bellman or Pontryagin's maximum principles) and then discritize these conditions in order to numerically solved the problems. This notably offers the advantage of highlighting the underlying sparsity of constraints induced by times. In this line of work, iLQR [1] and DDP [16], [2], [17] are the most well-known first and second order algorithms, with linear or quadratic-type convergence rates respectively. Moreover, their recent adaptations are even able to handle trajectory constraints [18], [19] and implicit dynamics [20]. More closely related to our work, sampled DDP [21] lies in-between as it uses stochastic estimates of the dynamics derivatives to deal with dynamics for which gradients are not available.

Reinforcement learning considers the dynamics as a black-box function and uses parameterized stochastic policies π_θ to better explore the action space. This model-free approach can natually handle complex, or even unknown, dynamics. During training, the average cost R along the trajectories sampled from the distribution ρ_θ induced by the policy π_θ is minimized (which is equivalent to maximizing $-R$, the cumulated sum of rewards, as in the RL literature), leading to the following problem:

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

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) [22], [23] algorithms aim at minimizing (3) by using a zero-th order estimate of the gradient as a descent 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 gradients estimates, which in turns slows down optimization but also fosters exploratory behaviours, potentially leading to more global solutions. Finally, this makes RL an instance of random optimization, which will be discussed more in details in Sec. III-C.

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 a use in classic gradient-based optimization techniques [24]. During a random search, one classical strategy consists in sampling a random direction at each step and then moving towards the corresponding directional derivative [25]. An alternative strategy, called Gradient Sampling (GS), approximates the sub-gradient at non-differentiable points by taking the descent direction inside the convex hull of some gradients randomly sampled in a neighbourhood [26]. Recent works [27] exploits the equivalence between random optimization techniques and performing a stochastic gradient descent on a smoothed version of the original problem in order 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 [28], [12], [29], [30]. 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 $\{Z^{(1)},\dots,Z^{(M)}\}$ are i.i.d. samples and M is the number of samples. Using an 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 [31], uniformly close from f 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 f 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 parallelisable, 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 [29]. Adding stochasticity to gradients is also proven to help escape saddle points when optimizing non-convex functions [32] which constitutes a positive side-effect of randomized smoothing. Finally, other previous works on the use of randomized smoothing demonstrates how it can improve convergence rates when optimizing non-smooth functions [11] and lead to more robust solutions [33].

III. 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.

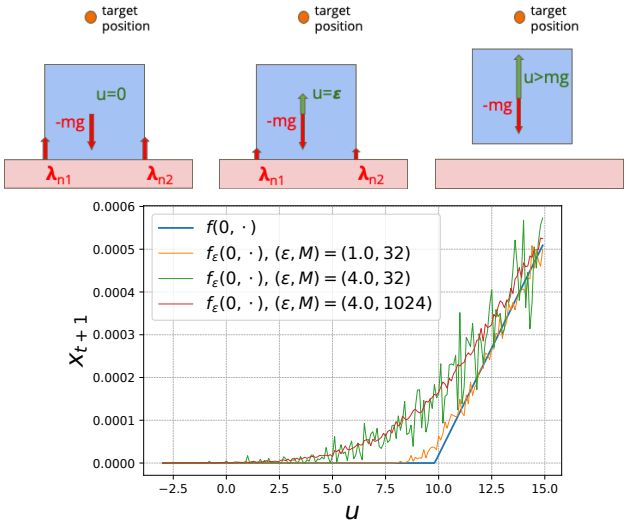

Fig. 2: **Top:** A slight vertical force is not able to break the unilateral contact, leaving cube on the floor and, thus, the state unchanged. **Bottom:** The non-smoothness of physics induces null gradients $\nabla_{\boldsymbol{u}} R$ which results in the failure of classical optimization techniques (III-A).

A. Locally optimal solutions of optimal control problems

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

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

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

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

corresponding to an integration process (e.g. exploiting a dynamical simulator). This also means that, by unrolling the successive integration steps, $\boldsymbol{x}(\boldsymbol{u})$ can be efficiently differentiated. This equivalent problem (10) can then be solved using a classical unconstrained optimization algorithm such as gradient descent, consisting in backpropagating through time [34].

In the case of a robotic system, 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 u is initialized with null control, this results in $\nabla_u f = 0$ because of the complementarity constraint arising from unilateral contacts (see IV. of [3]). By supposing that $\frac{\partial R}{\partial \boldsymbol{u}} = 0$ and

applying the chain rule, we have that:

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

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

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

implying that $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} = 0$. Finally, because $\nabla_u R = 0$, an algorithm exploiting only the local gradient information (*e.g.* gradient or Newton descent) will stop at this point, leaving the problem unsolved, blocked at a local maxima. For exactly the same reasons, the classical DDP algorithm would get stuck in a similar situation.

B. 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 occurs for physical systems involving contact or frictions.

An intuitive way to circumvent this issue consists in introducing randomization in the optimization process in order to get a more exploratory behaviour, by collecting samples in a larger neighbourhood around a given point, when compared to classic methods 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. III-C. We adapt the formulation (2) by artificially smoothing the system dynamics using randomized smoothing, leading to the following smooth but approximated problem:

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

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

$$x_0 = \overline{x_0}. \tag{14c}$$

where $f_\epsilon(x, u) = \mathbb{E}_{Z \sim \mu}[f(x, u + \epsilon Z)]$ 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 state 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 and 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. IV, 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.

C. 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 III-A. We provide a 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 descent directions used in the classical RL algorithm REINFORCE with baseline (the same considerations remain valid for the closely related actor-critic algorithms) [35], [23], [36] can be written as:

$$\nabla_\theta R_{PG} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{u}) \sim \rho_\theta} \Bigg[\Big(R(\boldsymbol{x}, \boldsymbol{u}) - \hat{V}(\overline{x_0}) \Big) \nabla_\theta \log \rho_\theta(\boldsymbol{x}, \boldsymbol{u}) \Bigg], \tag{15}$$

where $\hat{V}$ is an estimate of the value function and this 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. III-A . 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, in general, this comes at the cost of an increased variance in the estimates of ∇R_{PG}, which induces a slower convergence rate [27]. A similar study could be done with Evolution Strategies used in [37], [38] 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 to use randomized smoothing to compute informative gradients even in situations 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.

IV. 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.

Algorithm 1: Randomized DDP algorithm

Input: OC problem: R, f, Initial trajectory: $u, \overline{x}$, Target noise and precision: ϵ^*, α^*, Initial noise and precision: ϵ, α, Adaptive scheme parameters: ρ, γ

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

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

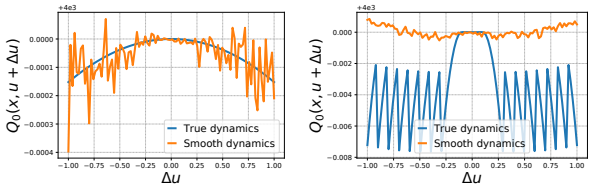

Fig. 3: **Left:** The randomly-smoothed dynamics allows to get non null gradients and escape the local maximum for the inverted pendulum. **Right:** $Q_0(x, \cdot)$ value function of the pendulum with dry friction exhibits plateaus where $Q_u = 0$ around $u = 0$ while randomly smoothed dynamics leads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

A. 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 benefits 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):

$$J_t(x_t, u_t, \dots, u_{T-1}) = l_T(x_T) + \sum_{\tau=t}^{T-1} l_\tau(x_\tau, u_\tau) \tag{16}$$

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

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

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

with the terminal condition $V_T(x) = l_T(x)$. 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 the 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 the backward pass equations:

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

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

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

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

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

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

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:

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

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

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

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

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

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

$$u_i^n = u_i + \alpha k + K(x_i^n - \overline{x_i}) \tag{22a}$$

$$x_0^n = \overline{x_0} \tag{22b}$$

$$x_{i+1}^n = f_\epsilon(x_i^n, u_i^n), \tag{22c}$$

where the superscript n relates to the updated quantities after applying a Newton step. During line search, the noise is fixed as described in [39], [40]. 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. II, 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 which 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. III-C).

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 [41]. Moreover, we find that preserving the

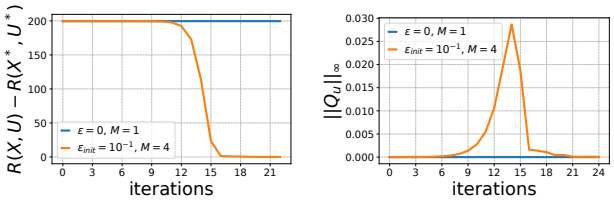

Fig. 4: Randomized smoothing allows to escape from the local optima $(x, u) = 0$ for the inverted pendulum (see Fig 3). In blue, the gradient is null and the systems remains stuck into the downward position. In orange, the system is able to reach the upward position thanks to randomized smoothing.

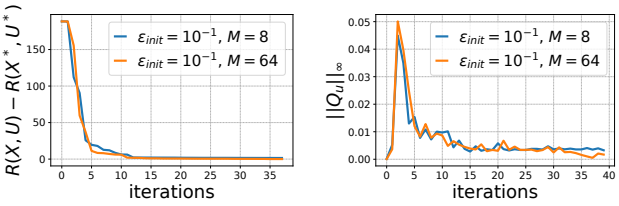

Fig. 5: Only a few samples (here M=8) are necessary to get reasonable results on the cartpole task. Increasing this number leads to very marginal improvement.

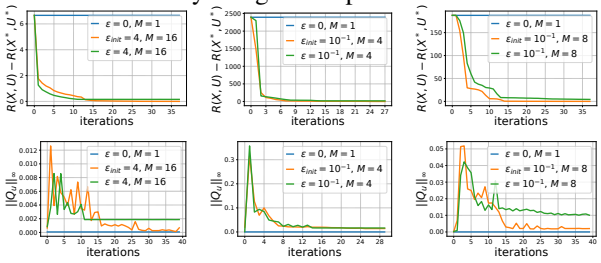

Fig. 6: **Left:** Randomized smoothing allows to solve tasks requiring to break contacts such as the one from Fig. 2 (Top) **Middle:** Complex systems such as an inverted double pendulum **Right:** or cartpole with dry frictions on the joints.

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 are able to handle hard constraints [19], [20], [18].

B. 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 [13] consists in relying on Robbins-Monro rule [42] by decreasing the variance in a way such that $\sum_k \epsilon_k^2 < \infty$ to guarantee convergence towards a local minima. 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 unnecessary large number of iterations. We adapt the smoothing by solving a cascade of randomly-smoohted DDP problem. 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.

V. Experiments

In this section, we demonstrate the practical benefits of our randomized DDP algorithms on a set of robotics systems. Our implementation is based on the open-source frameworks: Crocoddyl [17] for the DDP algorithm and on Pinocchio [43] and [8], [6] for the derivatives of the dynamics with and without contacts.

A. Avoiding local optima of smooth dynamics

We consider the task of raising a pendulum from the downwards to the upwards vertical position p^*. For this problem, we optimize the following cost function:

$$R(\boldsymbol{x}, \boldsymbol{u}) = w_p \|p(\theta_T) - p^*\|^2 + \sum_{t=0}^{T} w_u \|u_t - u^*\|^2, \quad (23)$$

where $T = 400$, $dt = 5 \times 10^{-3}$ s, $w_p = 2$, $w_u = 2 \times 10^{-5}$, $x = (\theta, \dot{\theta})$ and $p(\theta)$ corresponds to the position of the end-effector. For the problem (23), the solution $(u, x) = (0, 0)$ is a local extrema and the classical DDP algorithm gets stuck at this point (Fig. 4) as discussed in Sec. III-C. randomized smoothing allows R-DDP to avoid this local optima (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 [32]. Note that, for this problem, only $M = 4$ samples are required to obtain the presented results.

B. Controlling systems with non-smooth dynamics: contacts and friction in robotics

In addition to converging towards better optima in the case of smooth dynamics, we primarily designed R-DDP to be an elegant solution when it comes to the aforementioned issues from non-smooth dynamics (see Sec. III-A). In these experiments, we use a cost function similar to the one of (23). 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]). As shown by Fig. 6, our approach allows to complete this task while we have shown in Sec. III-A that it is not possible when relying on classic gradient information. In a similar way, it is possible to apply the R-DDP algorithm to control more complex systems such as a double pendulum

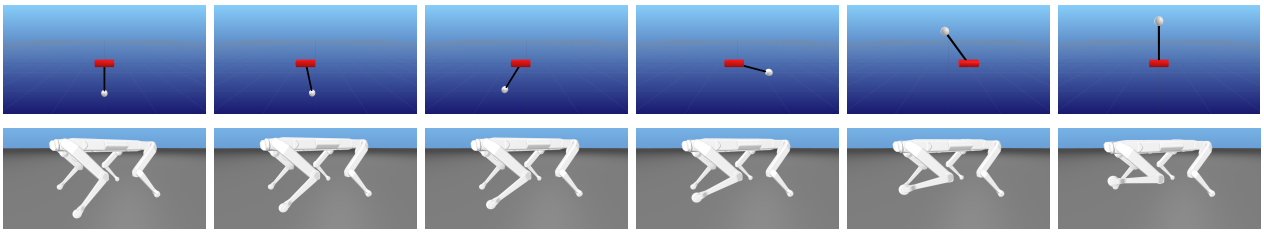

Fig. 7: Randomized-DDP leverages the smooth dynamics to precisely schedules movements requiring to break contacts (**Bottom**) and dry friction (**Top**).

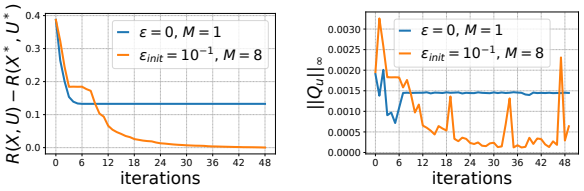

Fig. 8: Randomized DDP makes it possible to achieve complex tasks requiring breaking contacts with the floor on Solo robot such as lifting a leg (see Fig. 7)

or a cartpole with dry friction on the joints. Using the true dynamics, DDP is unable to optimize the control variable because of the non-smoothness induced by the dry friction. Once again, smoothing the dynamics with respect to the control u allows to have non-null gradients almost everywhere and thus to apply classical DDP (Fig. 6). Doing so, it is possible to solve very efficiently with the DDP algorithm the problem of controlling a complex system involving contacts and frictions. Interestingly, the FDDP algorithm [17] which allows for unfeasible trajectories also failed to overcome the non-smoothness from dry frictions. We also studied the influence of the number of samples 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 small number of samples, meaning that the supplementary computational costs is 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).

Finally, we apply our algorithm to solve a task on the Solo robot [44], a 18-DoF robotics system, with frictional contacts. Here, the goal is to reach a final target pose $\overline{x_T}$ (lifting of the tip of one leg while keeping the three others on the ground), which requires breaking some initial existing contacts, as illustrated in Fig. 7. R-DDP with adaptive smoothing allows to solve this task while classical DDP algorithm fails (Fig. 8). The observed performance gap can be explained by the impossibility of classical DDP to precisely apprehend contacts because of non-informative gradients of the dynamics. Typically, the control obtained from DDP will only approach this pose while maintaining the feet on the ground which results in Solo bending its leg instead of lifting it.

VI. Conclusion

Analyzing reinforcement learning via the theory of randomized smoothing allows to understand how crucial the exploratory characteristic of these algorithms is to solve control problem with non-smooth dynamics. By transferring these ideas to the field of trajectory optimization, we have in this paper leveraged randomized smoothing 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 able to cope with the presence of frictional contacts or frictions, in a sample efficient way. We've also demonstrated the capacity of our method to correctly solved standard robotics systems (pendulum, cartpole, Solo robot) including non-smooth dynamical effects (frictions, contacts) and where classic optimization-based will likely fail. In a future work, it would be interesting to investigate possible uses of the information contained in the variance of the several particle 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.

Acknowledgements

This work was supported in part by L'Agence d'Innovation Défense, the HPC resources from GENCI-IDRIS(Grant AD011012215), the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute), the European Regional Development Fund under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15 003/0000468), the Grant Agency of the Czech Technical University in Prague, grant No. SGS21/178/OHK3/3T/17, and Louis Vuitton ENS Chair on Artificial Intelligence.

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 for articulated rigid bodies with contact," *arXiv preprint arXiv:2103.16021*, 2021.

[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] H. J. T. Suh, T. Pang, and R. Tedrake, "Bundled gradients through contact via randomized smoothing," 2021.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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/

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

[20] W. Jallet, N. Mansard, and J. Carpentier, "Implicit Differential Dynamic Programming," Sept. 2021, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03351641

[21] 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.

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

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

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

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

[26] 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.

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

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

[29] 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

[30] 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.

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

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien, "Painless stochastic gradient: Interpolation, line-search, 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

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

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

[42] 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

[43] 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.

[44] 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.