Progress and Prospects in Simulation for Robotics

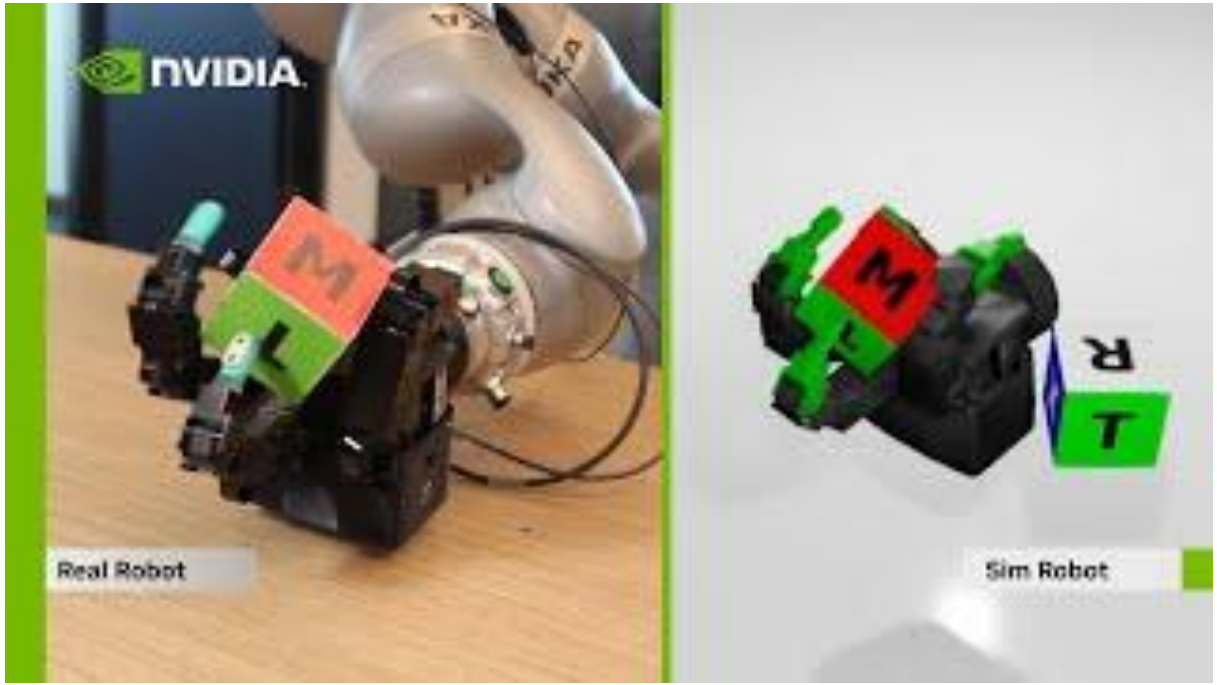
Quentin Le Lidec, Louis Montaut, Ivan Laptev, Cordelia Schmid and Justin Carpentier

Outline

1. Simulation in robotics
2. Modeling the physics of contacts
3. Optimizing through contacts

Simulation in robotics

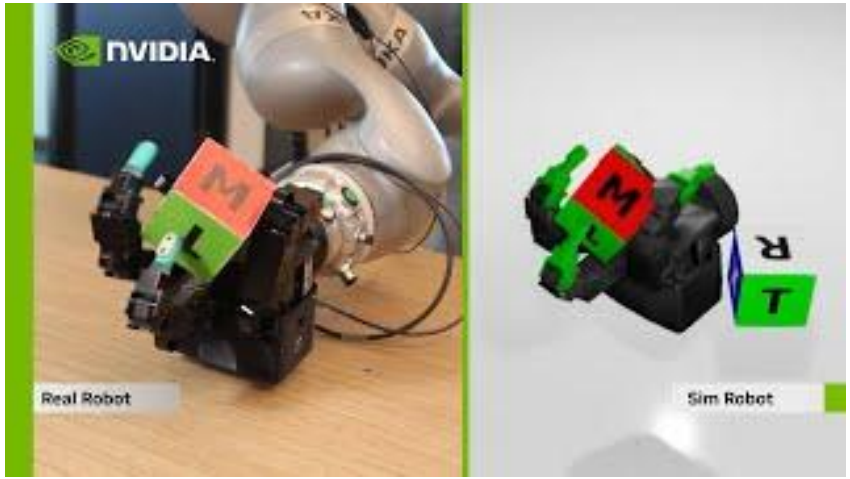
Models in the era of “model-free” RL



Models in the era of “model-free” RL

Parameter	Type	Distribution	Initial Range	ADR-Discovered Range
Hand				
Mass	Scaling	uniform	[0.4, 1.5]	[0.4, 1.5]
Scale	Scaling	uniform	[0.95, 1.05]	[0.95, 1.05]
Friction	Scaling	uniform	[0.8, 1.2]	[0.54, 1.58]
Armature	Scaling	uniform	[0.8, 1.02]	[0.31, 1.24]
Effort	Scaling	uniform	[0.9, 1.1]	[0.9, 2.49]
Joint Stiffness	Scaling	loguniform	[0.3, 3.0]	[0.3, 3.52]
Joint Damping	Scaling	loguniform	[0.75, 1.5]	[0.43, 1.6]
Restitution	Additive	uniform	[0.0, 0.4]	[0.0, 0.4]
Object				
Mass	Scaling	uniform	[0.4, 1.6]	[0.4, 1.6]
Friction	Scaling	uniform	[0.3, 0.9]	[0.01, 1.60]
Scale	Scaling	uniform	[0.95, 1.05]	[0.95, 1.05]
External Forces	Additive	Refer to [1]	–	–
Restitution	Additive	uniform	[0.0, 0.4]	[0.0, 0.4]
Observation				
Obj. Pose Delay Prob.	Set Value	uniform	[0.0, 0.05]	[0.0, 0.47]
Obj. Pose Freq.	Set Value	uniform	[1.0, 1.0]	[1.0, 6.0]
Obs Corr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.12]
Obs Uncorr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.14]
Random Pose Injection	Set Value	uniform	[0.3, 0.3]	[0.3, 0.3]
Action				
Action Delay Prob.	Set Value	uniform	[0.0, 0.05]	[0.0, 0.31]
Action Latency	Set Value	uniform	[0.0, 0.0]	[0.0, 1.5]
Action Corr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.32]
Action Uncorr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.48]
RNA α	Set Value	uniform	[0.0, 0.0]	[0.0, 0.16]
Environment				
Gravity (each coord.)	Additive	normal	[0, 0.5]	[0, 0.5]

Table 3: Domain randomisation parameter ranges for policy learning

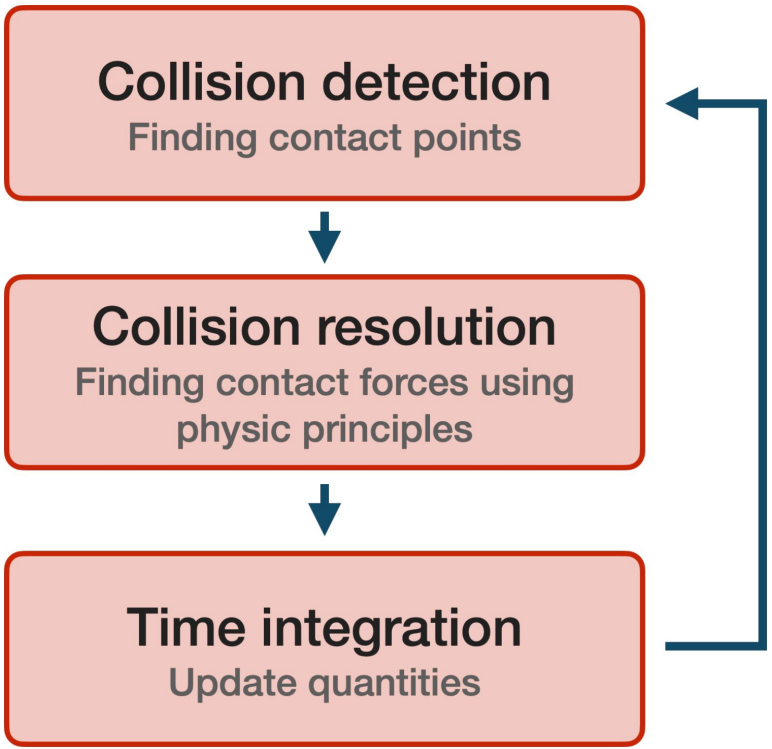


1 training: 60 hours with 16384 environments

Approx. 100e9 calls to the simulator ~ 55 years of simulation

1000$, 55 kg CO2e ~ 300km by car

What is a simulator ?



Simulators in robotics

- Bullet (Google): computer graphics
- MuJoCo (Deepmind): invertible, RL
- RaiSim (ETHZ): quadrupedal locomotion
- Isaac Gym (NVIDIA): GPU acceleration
- Simple (Inria Willow): soon… efficient and realistic CPU simulation

These simulators have been designed for different purposes: they are not interchangeable in general !

Difference may come from both physical modeling and numerical implementations

Modeling the physics of contacts

Free motion with generalized coordinates

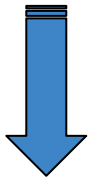
Lagrangian equations of motion:

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

Free motion with generalized coordinates

Lagrangian equations of motion:

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



Discrete version (time-stepping methods):

$$Mv^{t+1} = Mv^{t} + (\tau - Cv - g)\Delta t$$

Contact modelling hypothesis

Contacts in modern simulators are modelled via 4 elementary principles:

- Rigid bodies
- Unilateral contacts
- Coulomb’s law of friction
- Maximum Dissipation Principle

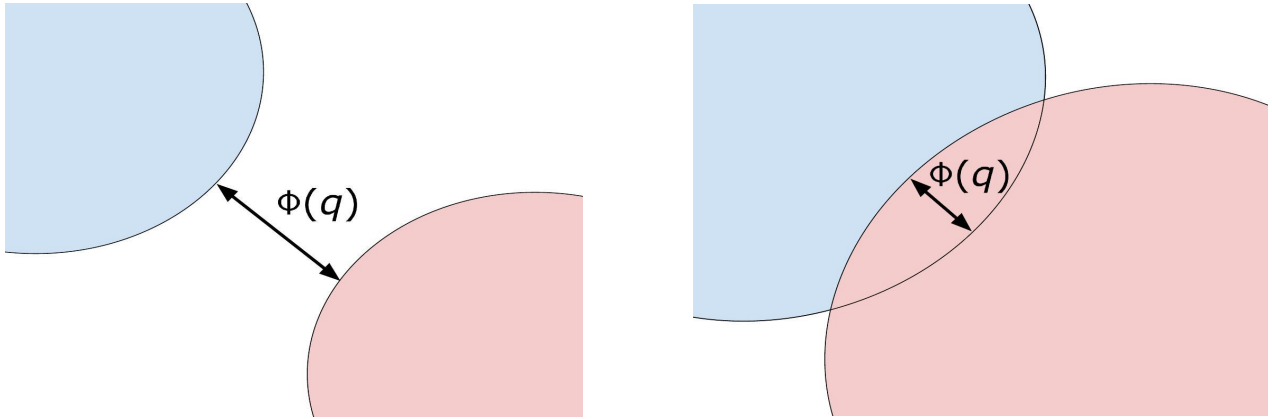
Contact modelling hypothesis

Contacts in modern simulators are modelled via 4 elementary principles:

- Rigid bodies
- Unilateral contacts
- Coulomb’s law of friction
- Maximum Dissipation Principle

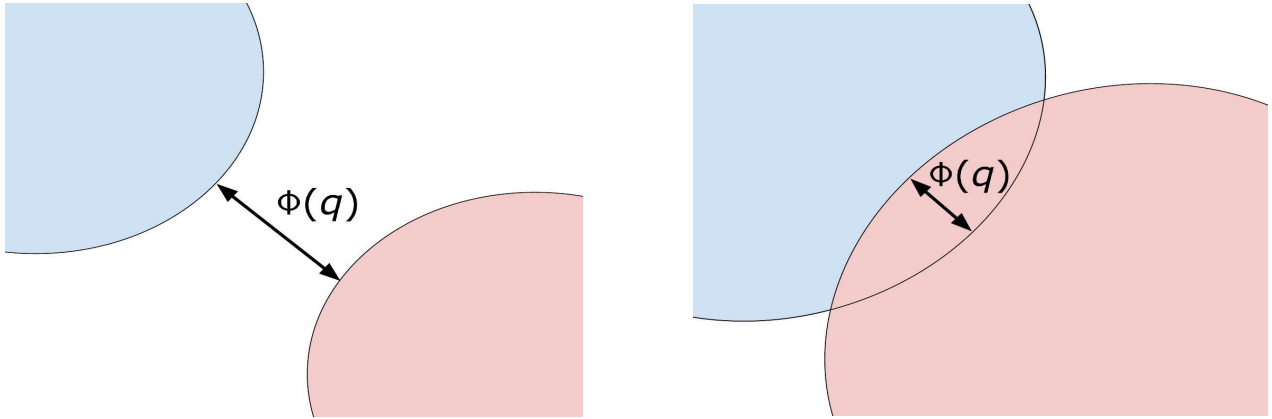
} Signorini condition

Modelling unilateral contacts



No interpenetration: $\Phi(q)_N \geq 0$

Modelling unilateral contacts

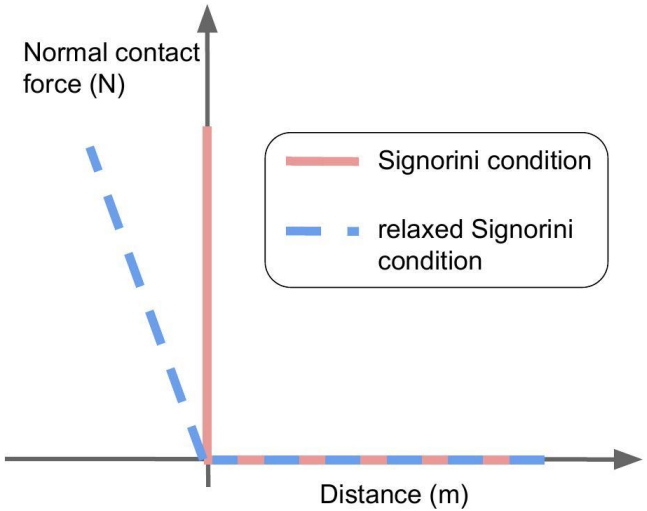


No interpenetration: $\Phi(q)_N \geq 0 \Rightarrow c_N - c_N^* \geq 0$

where: $c = Jv^{t+1}$ and $J = \partial\Phi/\partial q$

Modelling unilateral contacts

Signorini condition: $0 \leq \lambda_N \perp c_N - c_N^* \geq 0$

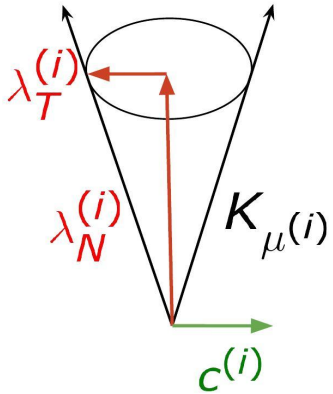


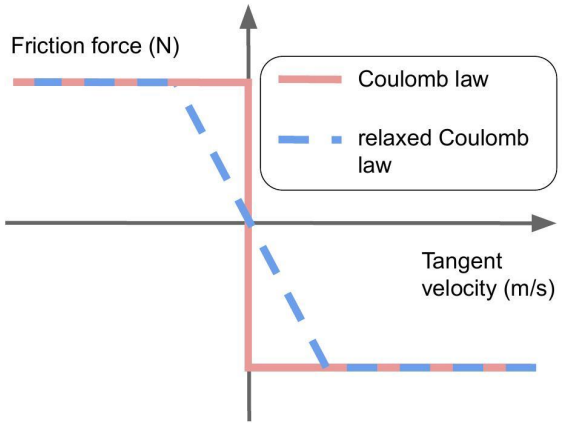
Modelling frictions

Coulomb's law of friction: $\lambda \in K_\mu = \prod_{i=1}^{n_c} K_{\mu^{(i)}}$

$$K_{\mu^{(i)}} = \left\{ \lambda | \lambda_N \geq 0, \|\lambda_T\|_2 \leq \mu^{(i)} \lambda_N \right\}$$

Maximum dissipation principle: $\forall i,\ \lambda_T^{(i)} = -\mu^{(i)}\, \lambda_N^{(i)} \frac{c_T^{(i)}}{\|c_T\|}, \text{if } \|c_T^{(i)}\| > 0$





The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} \end{cases}$$

where: $c = G\lambda + g$, $G = JM^{-1}J^{\top}$ and $g = Jv^f$

The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} \end{cases}$$

➡ Sticking

where: $c = G\lambda + g$, $G = JM^{-1}J^\top$ and $g = Jv^f$

The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 \Rightarrow \text{Sticking} \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 \Rightarrow \text{Breaking} \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} \end{cases}$$

where: $c = G\lambda + g$, $G = JM^{-1}J^\top$ and $g = Jv^f$

The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 & \Rightarrow \text{Sticking} \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 & \Rightarrow \text{Breaking} \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} & \Rightarrow \text{Slipping} \end{cases}$$

where: $c = G\lambda + g$, $G = JM^{-1}J^\top$ and $g = Jv^f$

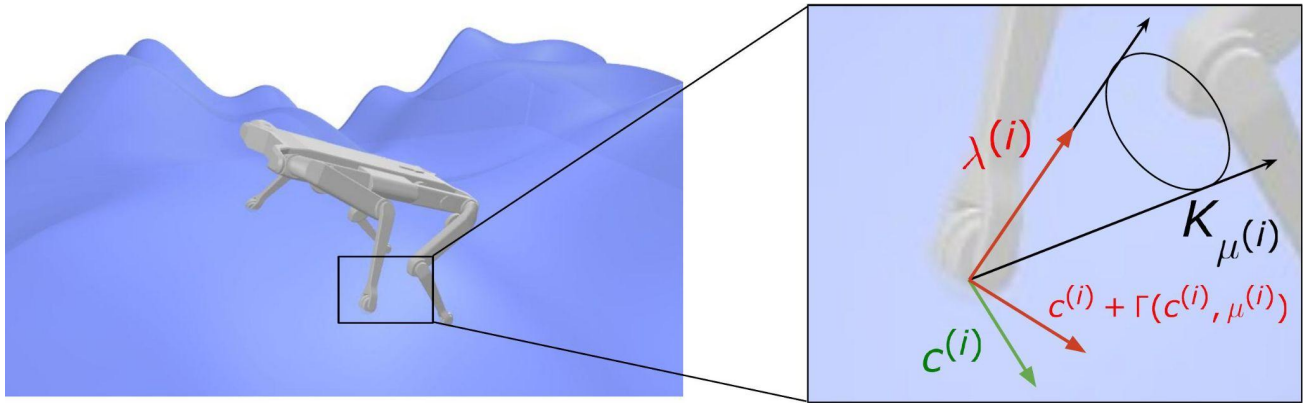
The frictional contact problem

Simulating contacts and frictions requires to solve:

$$\forall i, K_{\mu^{(i)}} \ni \lambda^{(i)} \perp c^{(i)} + \Gamma\left(c^{(i)}, \mu^{(i)}\right) \in K^*_{\mu^{(i)}}$$

$$c = G\lambda + g$$

where the de Saxcé correction is: $\Gamma : (c, \mu) \in \mathbb{R}^3 \times \mathbb{R} \mapsto [0, 0, \mu \|c_T\|_2]$



Simulation for robotics

Contact Models in Robotics: a Comparative Analysis

Quentin Le Lidec[1,†], Wilson Jallet[1,2], Louis Montaut[1,3], Ivan Laptev[1], Cordelia Schmid[1], and Justin Carpentier[1]

arXiv:2304.06372v1 [cs.RO] 13 Apr 2023

***Abstract*—Physics simulation is ubiquitous in robotics. Whether in model-based approaches (*e.g.*, trajectory optimization), or model-free algorithms (*e.g.*, reinforcement learning), physics simulators are a central component of modern control pipelines in robotics. Over the past decades, several robotic simulators have been developed, each with dedicated contact modeling assumptions and algorithmic solutions. In this article, we survey the main contact models and the associated numerical methods commonly used in robotics for simulating advanced robot motions involving contact interactions. In particular, we recall the physical laws underlying contacts and friction (*i.e.*, Signorini condition, Coulomb's law, and the maximum dissipation principle), and how they are transcribed in current simulators. For each physics engine, we expose their inherent physical relaxations along with their limitations due to the numerical techniques employed. Based on our study, we propose theoretically grounded quantitative criteria on which we build benchmarks assessing both the physical and computational aspects of simulation. We support our work with an open-source and efficient C++ implementation of the existing algorithmic variations. Our results demonstrate that some approximations or algorithms commonly used in robotics can severely widen the reality gap and impact target applications. We hope this work will help motivate the development of new contact models, contact solvers, and robotic simulators in general, at the root of recent progress in motion generation in robotics.**

***Index Terms*—Physical simulation, Numerical optimization.**

I. Introduction

SIMULATION is a fundamental tool in robotics. Control algorithms, like trajectory optimization (TO) or model predictive control (MPC) algorithms, rely on physics simulators to evaluate the dynamics of the controlled system. Reinforcement Learning (RL) algorithms operate by trial and error and require a simulator to avoid time-consuming and costly failures on real hardware. Robot co-design aims at finding optimal hardware design and morphology and thus extensively rely on simulation to prevent tedious physical validation. In practice, roboticists also usually perform simulated safety checks before running a new controller on their robots. These applications are evidence for a wide range of research areas in robotics where simulation is critical.

To be effective and valuable in practice, robot simulators must meet some fidelity or efficiency levels, depending on the use case. For instance, trajectory optimization algorithms, *e.g.* iLQR[1] or DDP [2], [3], use physics simulation to evaluate the system dynamics and leverage finite differences or the recent advent of differentiable simulators [4], [5], [6], [7], [8] to compute derivatives. If the solution lacks precision, the real and planned trajectories may quickly diverge, impacting *de facto* the capacity of such control solutions to be deployed on real hardware. To absorb such errors, the Model Predictive Control (MPC) [9], [10] control paradigm exploits state feedback by repeatedly running Optimal Control (OC) algorithms at high-frequency rates (e.g., 1kHz) [11], [12]. The frequency rate is one factor determining the robustness of this closed-loop algorithm to modeling errors and perturbations; thus, the efficiency of the simulation becomes critical. Although RL [13] is considered as a model-free approach, physical models are still at work to generate the samples that are indispensable for learning control policies. In fact, the vast number of required samples is the main bottleneck during training, as days or years of simulation, which corresponds to billions of calls to a simulator, are necessary [14], [15], [16]. Therefore, the efficiency of the simulator directly determines

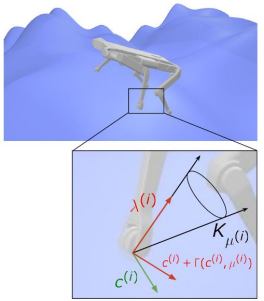


Fig. 1. **Illustration of the dynamics of frictional contacts** between rigid bodies which are governed by the Signorini condition, the Coulomb's law, and the maximum dissipation principle. The combination of these three principles leads to the Non-linear Complementarity Problem (14).

[1]Inria - Département d'Informatique de l'École normale supérieure, PSL Research University. Email: firstname.lastname@inria.fr
[2]LAAS-CNRS, 7 av. du Colonel Roche, 31400 Toulouse
[3]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University, Prague, Czech Republic
[†]Corresponding author

Simulation for robotics

arXiv:2405.17020v1 [cs.RO] 27 May 2024

From Compliant to Rigid Contact Simulation: a Unified and Efficient Approach

Justin Carpentier*
Inria, École normale supérieure
CNRS, PSL Research University
75005 Paris, France
Email: justin.carpentier@inria.fr

Quentin Le Lidec*
Inria, École normale supérieure
CNRS, PSL Research University
75005 Paris, France
Email: quentin.le-lidec@inria.fr

Louis Montaut*
Inria, École normale supérieure
CNRS, PSL Research University
75005 Paris, France
Email: louis.montaut@inria.fr

***Abstract*—Whether rigid or compliant, contact interactions are inherent to robot motions, enabling them to move or manipulate things. Contact interactions result from complex physical phenomena, that can be mathematically cast as Nonlinear Complementarity Problems (NCPs) in the context of rigid or compliant point contact interactions. Such a class of complementarity problems is, in general, difficult to solve both from an optimization and numerical perspective. Over the past decades, dedicated and specialized contact solvers, implemented in modern robotics simulators (e.g., Bullet, Drake, MuJoCo, DART, Raisim) have emerged. Yet, most of these solvers tend either to solve a relaxed formulation of the original contact problems (at the price of physical inconsistencies) or to scale poorly with the problem dimension or its numerical conditioning (e.g., a robotic hand manipulating a paper sheet). In this paper, we introduce a unified and efficient approach to solving NCPs in the context of contact simulation. It relies on a sound combination of the Alternating Direction Method of Multipliers (ADMM) and proximal algorithms to account for both compliant and rigid contact interfaces in a unified way. To handle ill-conditioned problems and accelerate the convergence rate, we also propose an efficient update strategy to adapt the ADMM hyperparameters automatically. By leveraging proximal methods, we also propose new algorithmic solutions to efficiently evaluate the inverse dynamics involving rigid and compliant contact interactions, extending the approach developed in MuJoCo. We validate the efficiency and robustness of our contact solver against several alternative contact methods of the literature and benchmark them on various robotics and granular mechanics scenarios. Overall, the proposed approach is shown to be competitive against classic methods for simple contact problems and outperforms existing solutions on more complex scenarios, involving tens of contacts and poor conditioning. Our code is made open-source at https://github.com/Simple-Robotics/Simple.**

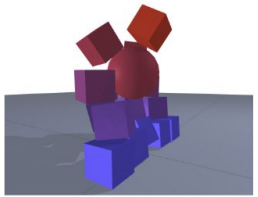


Fig. 1. **Simulation of an ill-conditioned stack of cubes.** Our approach robustly solves high-dimensional and ill-conditioned problems such as the ones involved when simulating a stack of 15 cubes involving more than 60 contact points and objects' masses varying from 1 kg (bottom row, in light blue) to 10.000 kg (highest row, in bright red). Simulation videos at https://youtu.be/l_qp9cTs0NY?si=NGtx1hiYrIGtHXSK.

I. Introduction

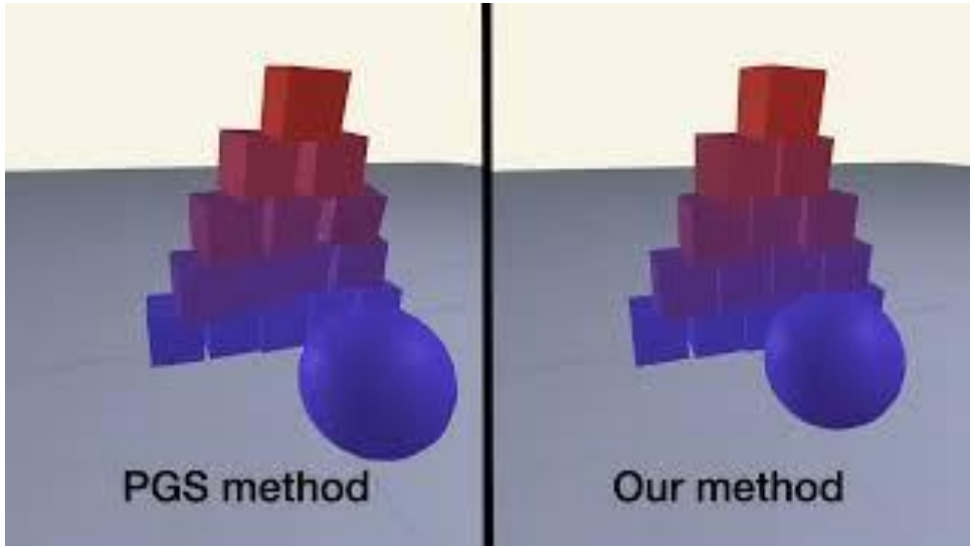
Contact interactions are the substrate of movement. When performing locomotion or manipulation tasks, one naturally makes and breaks contact with the environment. Providing robots with such abilities would require them to precisely apprehend the physics of contact. In this respect, simulators have been key components for various robotics applications. Reinforcement Learning (RL) policies for manipulation [1], [23] or locomotion [34] often rely on years of simulated trajectories during training. Alternatively, Model Predictive Control (MPC) techniques call the simulator and its derivatives at high frequency at runtime to achieve a reactive behavior [31], [29].

Simulating rigid contact interactions with friction requires solving a Nonlinear Complementarity Problem (NCP) which has been recognized as a hard problem both from an optimization and numerical perspective [2]. For this reason, simulators proceed to tradeoffs between physical realism, robustness, and efficiency. In this respect, each physics engine embeds its own contact model and contact solver, which come with their underpinning physical hypotheses and numerical capabilities.

Earlier simulators like ODE [48] and Bullet [13] have been developed for graphical purposes. Graphical applications mainly require the physics engine to provide visually consistent trajectories. Nowadays, the requirements in robotics applications are different: the physics of contacts should be as close as possible to reality, and the contact solver should be numerically efficient. Indeed, this would reduce training time and enhance the transferability of RL while making MPC more reactive.

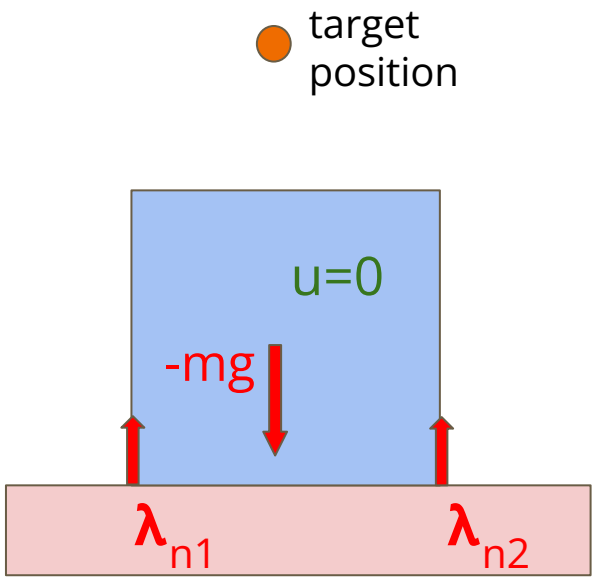
More recently, robotics-oriented engines were developed

*Equal contribution. Authors are listed in alphabetical order.

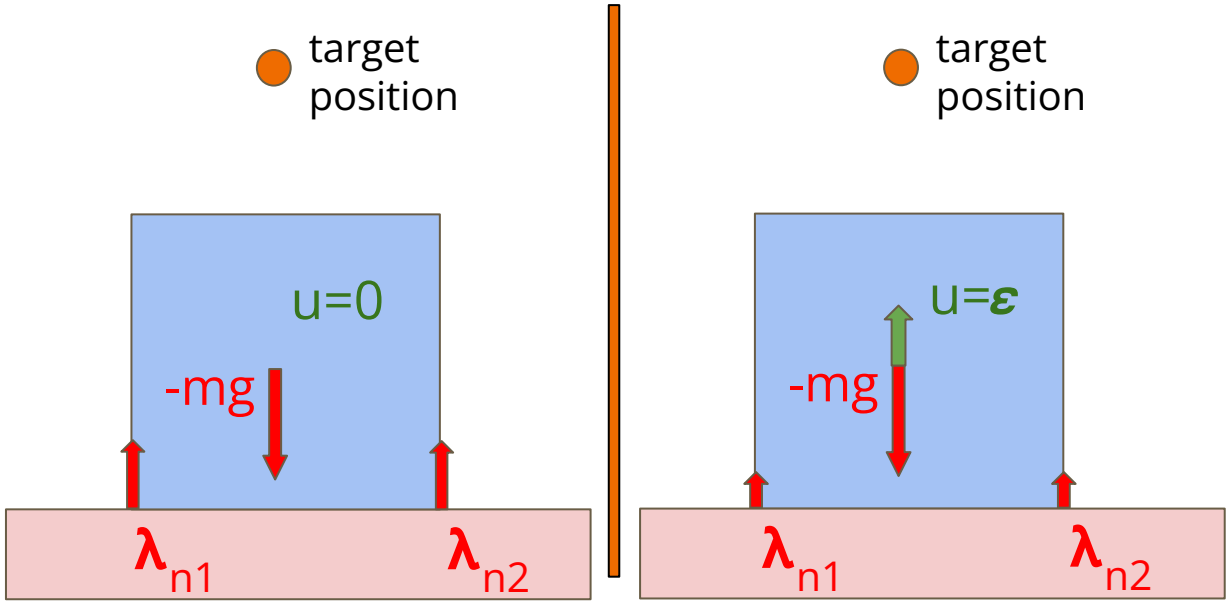


Differentiable simulation and optimization through contacts

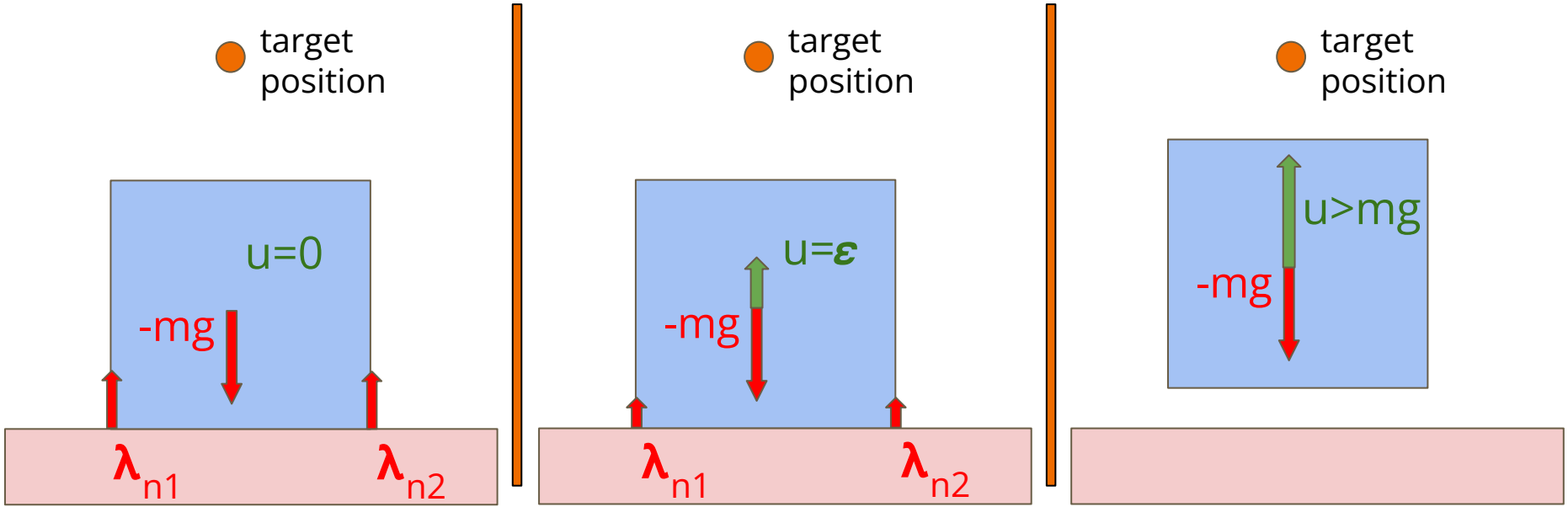
Example of non smooth dynamics: unilateral contacts



Example of non smooth dynamics: unilateral contacts



Example of non smooth dynamics: unilateral contacts



➔ Dynamics are non smooth and gradients are null wrt to control u...

Optimal Control Problem (OCP)

We want to minimize a cost (or equivalently maximize a reward) under the dynamics :

$$\begin{aligned} &\min_{x,u} \; R(x,u) \\ &s.t. \; x_{t+1} = f(x_t, u_t), \; \forall t \in [1, T-1], \\ &\quad x_0 = x(0) \end{aligned}$$

Solving the OCP via single shooting

State x can be substituted by using the dynamics constraint:

$$\min_u \; R_2(u) = R(x(u), u)$$

where:

$$\begin{aligned} x_1(u) &= f(x_0, u_0), \\ x_2(u) &= f(x_1(u), u_1) = f(f(x_0, u_0), u_1), \\ &\vdots \\ x_T(u) &= f(x_{T-1}(u), u_{T-1}) = f(...f(x_0, u_0)u_1), ...u_{T-1}) \end{aligned}$$

This unconstrained optimization problem can be solved by using GD

Failure mode: non-smooth dynamics

We need informative gradients to solve the unconstrained optimization problem:

$$\min_u \; R_2(u) = R(x(u), u)$$

In the case of a system with contact and frictions, the dynamics is non-smooth (gradients of f may be null) which leads to null gradients during optimization

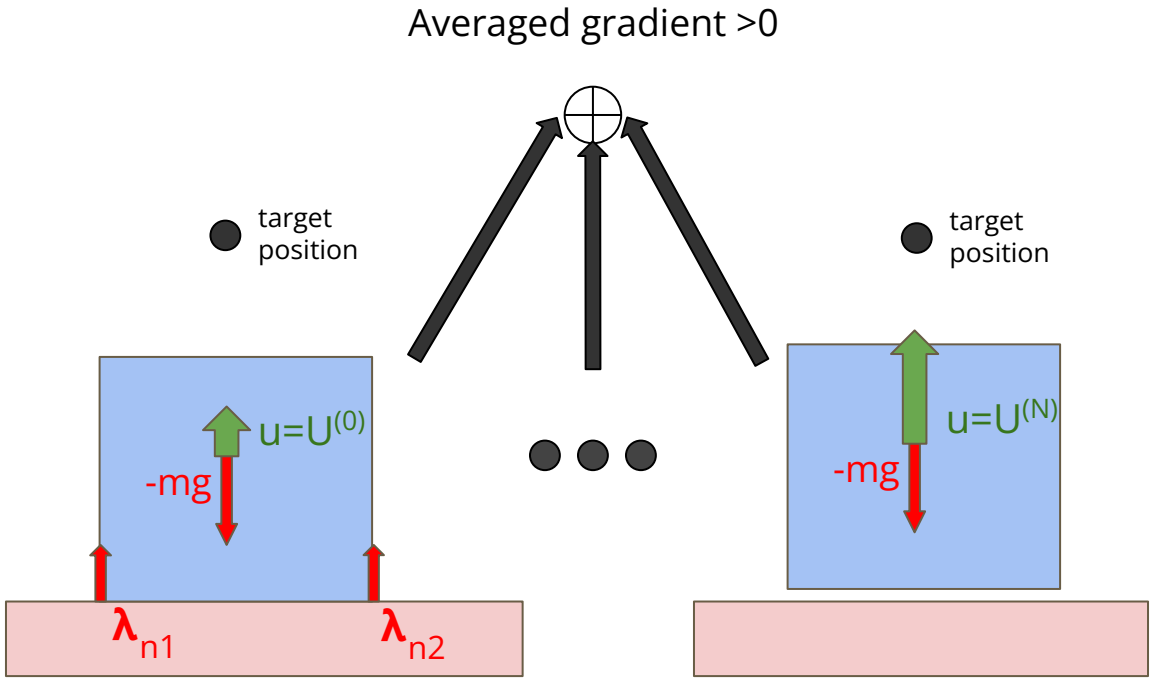
$$\begin{aligned}\nabla_u R_2(u) &= \frac{\partial x}{\partial u}\frac{\partial R}{\partial x} + \frac{\partial R}{\partial u} \\ &= \frac{\partial x}{\partial u}\frac{\partial R}{\partial x}\end{aligned}$$

and we have $\frac{\partial x}{\partial u} = 0 \qquad \nabla_u R_2 = 0$

Gradient descent and DDP fail to solve this problem...

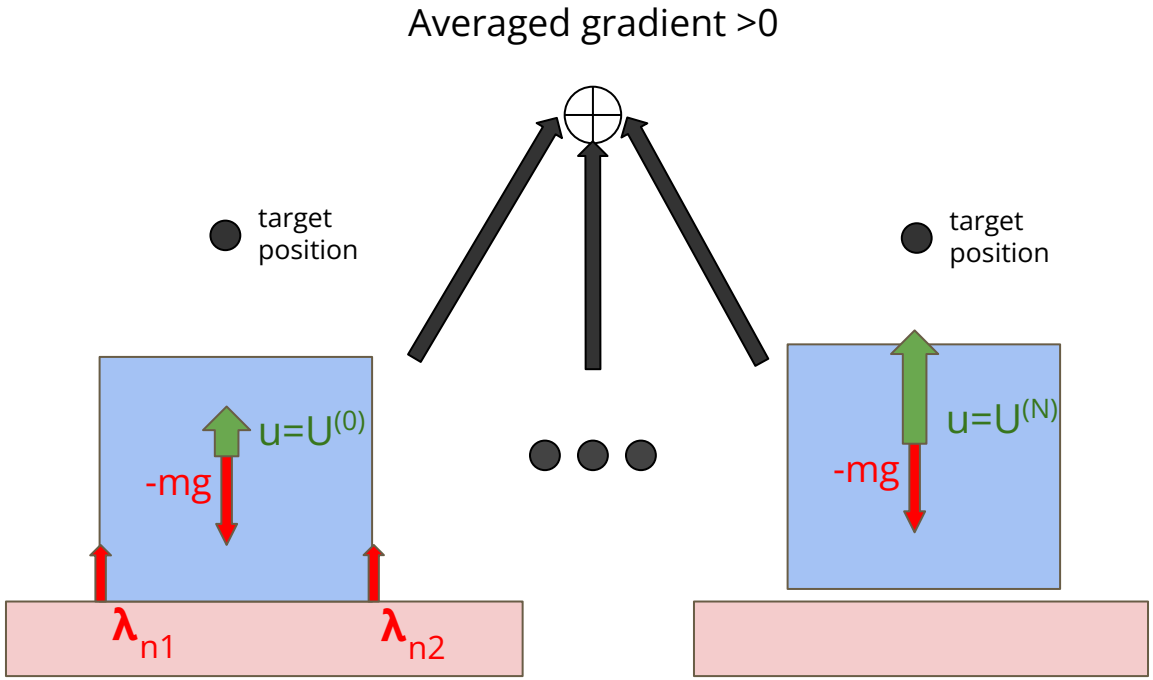
Smoothed dynamics

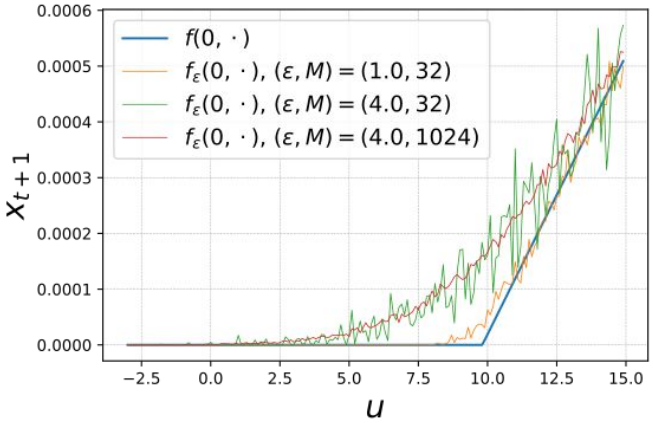
Perturbations smoothen contact dynamics (this could also be interpreted as an exploration term):



Smoothed dynamics

Perturbations smoothen contact dynamics (this could also be interpreted as an exploration term):





0th order gradient estimate:

$$\nabla_x^{(0)} f_\epsilon(x) = \frac{1}{M}\sum_{i=0}^{M} -f(x+\epsilon z^{(i)})\frac{\nabla \log \mu(z^{(i)})^\top}{\epsilon}$$

1st order gradient estimate:

$$\nabla_x^{(1)} f_\epsilon(x) = \frac{1}{M}\sum_{i=0}^{M} \nabla f(x+\epsilon z^{(i)})$$

How RL solves non-smooth problems

RL smoothes the problem by introducing stochasticity in the policy:

$$\min_{\theta} R_3(\theta) = \mathbb{E}_{\zeta}\left[R(x(\theta,\zeta), u(\theta,\zeta))\right]$$

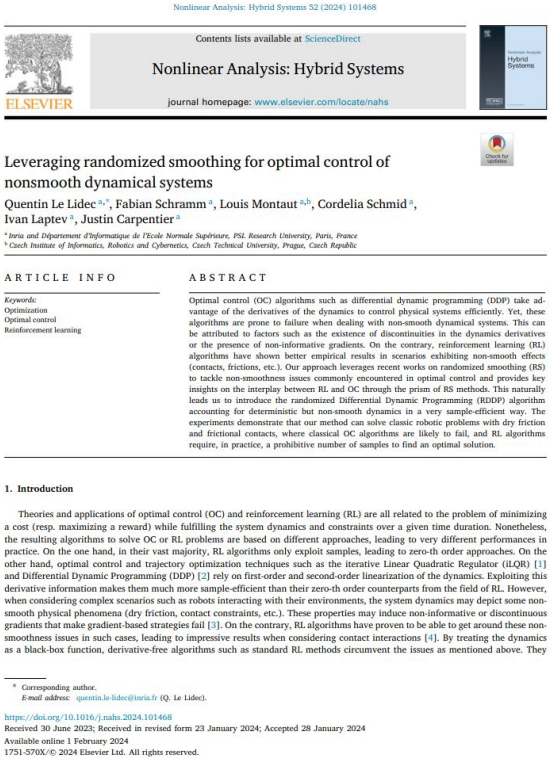
where:

$$\begin{aligned}
&u_t(\theta,\zeta) = \pi_\theta(x_t,\zeta_t)\ \forall t \in [1,T],\\
&x_1(\theta,\zeta) = f(x_0,\pi_\theta(x_0,\zeta_0)),\\
&x_2(\theta,\zeta) = f(x_1(\theta),\pi_\theta(x_1,\zeta_1)) = f(f(x_0,\pi_\theta(x_0\zeta_0)),\pi_\theta(x_1,\zeta_1)),\\
&\qquad\vdots\\
&x_T(\theta,\zeta) = f(...f(x_0,\pi_\theta(x_0,\zeta_0))\pi_\theta(x_1,\zeta_1)),...\pi_\theta(x_{T-1},\zeta_{T-1}))
\end{aligned}$$

Policy gradients:

$$\nabla_\theta R_{PG} = \mathbb{E}_{\zeta}\left[R(x(\theta,\zeta),u(\theta,\zeta))\nabla_\theta \log p_\theta(u(\theta,\zeta))\right]$$

Randomized smoothing for trajectory optimization



Nonlinear Analysis: Hybrid Systems 52 (2024) 101468

Contents lists available at ScienceDirect

Nonlinear Analysis: Hybrid Systems

journal homepage: www.elsevier.com/locate/nahs

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

ARTICLE INFO

Keywords:
Optimization
Optimal control
Reinforcement learning

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.

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

* Corresponding author.
E-mail address: quentin.le-lidec@inria.fr (Q. Le Lidec).

https://doi.org/10.1016/j.nahs.2024.101468
Received 30 June 2023; Received in revised form 23 January 2024; Accepted 28 January 2024
Available online 1 February 2024
1751-570X/© 2024 Elsevier Ltd. All rights reserved.

Conclusion

- Simulating rigid bodies with contacts and frictions requires to solve a NCP (non convex) which leads to non-smooth dynamics
- Optimizing through these dynamics is challenging
- Future works: computing smooth and informative gradients