End-to-End and Highly-Efficient Differentiable Simulation for Robotics

Quentin Le Lidec, Louis Montaut, Yann de Mont-Marin, Justin Carpentier

End-to-End and Highly-Efficient Differentiable Simulation for Robotics

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

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

Yann de Mont-Marin*
Inria, École normale supérieure
CNRS, PSL Research University
75005 Paris, France
yann.montmarin@gmail.com

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

Abstract: Over the past few years, robotics simulators have largely improved in efficiency and scalability, enabling them to generate years of simulated data in a few hours. Yet, efficiently and accurately computing the simulation derivatives remains an open challenge, with potentially high gains on the convergence speed of reinforcement learning and trajectory optimization algorithms, especially for problems involving physical contact interactions. This paper contributes to this objective by introducing a unified and efficient algorithmic solution for computing the analytical derivatives of robotic simulators. The approach considers both the collision and frictional stages, accounting for their intrinsic nonsmoothness and also exploiting the sparsity induced by the underlying multibody systems. These derivatives have been implemented in C++, and the code will be open-sourced in the Simple simulator. They depict state-of-the-art timings ranging from 5 us for a 7-dof manipulator up to 95 us for 36-dof humanoid, outperforming alternative solutions by a factor of at least 100. For videos additional details, please see our project website.

Keywords: Differentiable physics, Differentiable optimization, Simulation, Non-smooth dynamics, Model-based learning

1 Introduction

Recent progress in reinforcement learning and trajectory optimization methods in robotics extensively relies on simulation. Additional information, such as the simulator derivatives of the simulator, might be leveraged to accelerate the convergence speed of these control methods. However, simulating robotics systems interacting with their environment induces a sequence of nonsmooth operations. Typically, collision detection involved in simulators is intrinsically nonsmooth (e.g., contact points might jump from one vertex to another one when slightly changing the orientation of the geometries), and frictional contact dynamics corresponds to nonsmooth problems (e.g., when a cube switches from a sticking mode to a sliding mode).

Several approaches have been envisaged to estimate simulator derivatives. Mordatch et al. [1] leverages MuJoCo [2] and finite differences to discover new behaviors. However, computing gradients via finite differences requires as many calls to forward dynamics as the number of parameters to

*Equal contribution.

Table 1: Differentiable physics engines for robotics.

Physics engine	Contact Model	∂(Contacts)	∂(Collisions)
MuJoCo MJX [2]	CCP	Auto-diff	meshes* + primitives
Nimble [10], [8]	LCP	Implicit	meshes + primitives
Dojo[13]	NCP	Implicit	(not considered)
Ours	NCP	Implicit	meshes + primitives

*Performances are degraded for meshes over 20 vertices

differentiate, which becomes quickly prohibitive. Following the advent of the differentiable programming paradigm, DiffTaichi [3] and NeuralSim [4] propose to exploit Automatic Differentiation to differentiate through simplified contact models and geometries. In this vein, Brax [5] and MuJoCo MJX build on JAX [6] auto-diff and hardware acceleration capabilities to compute gradients through the computational graph. Because collision detection and contact forces involve iterative algorithms, the cost of computing gradients scales with the number of iterations performed during the evaluation of the forward dynamics. Alternatively, inspired by differentiable optimization [7], multiple works propose to apply implicit differentiation to a linear complementarity problem (LCP) [8, 9, 10] or mixed linear complementarity problem (MLCP) [11] relaxing the original nonlinear complementarity problem (NCP). Implicit differentiation has been extended to the NCP case in [12, 13], but it remains inefficient as it does not, for instance, exploit the structure induced by the kinematic chain. Other approaches, such as [14], rely on compliant contact models and focus on differentiating with respect to morphological parameters. For a comprehensive study of different contact models and the impact of relaxing the original NCP on gradients, we refer the reader to [15, 16].

Contributions. In this paper, we present a comprehensive framework for differentiable simulation that combines differentiable rigid-body dynamics, differentiable collision detection, and differentiable contact resolution. We notably introduce an implicit differentiation scheme to compute the gradients of the NCP associated with the frictional contact problem without any relaxation and chain it with rigid body dynamics algorithms to finely exploit the kinematic sparsity of the problem. Our approach achieves substantial computational speedups, with gradient computations up to 100 times faster than current state-of-the-art methods in robotics, while avoiding any physical relaxation or geometrical approximation on meshes. We validate the effectiveness of our method by applying it to complex inverse problems, including the estimation of initial conditions and inverse dynamics through contact. To support reproducibility and further research, we will make our code publicly available after the review process.

Paper organization. In Sec. 2, we provide a background on collision detection, frictional dynamics, rigid multi-body dynamics, and implicit differentiation techniques. Sec. 3 corresponds to the core contribution of this paper. We specify the computational graph of a physics engine and explain how the combination of gradients of rigid-body dynamics, collision detection, and contact forces make simulation end-to-end differentiable. Additionally, we show how implicit differentiation and rigid body algorithms can be leveraged to compute the derivatives of multibody frictional contact problems efficiently, including collision geometry contributions. In Sec. 4, the efficiency of our approach is benchmarked on several advanced robotics systems, and we leverage our differentiable physics engine to tackle various estimation and control problems. Sec. 5 discusses this work's limitations and how it could set the stage for future developments in model-based approaches for robotics.

2 Background

This section reviews the two fundamental stages of modern robotic simulators: collision detection, contact modelling and multibody dynamics. We also review the notion of implicit differentiation, which is at the core of our approach.

2.1 Collision detection

The collision detection phase identifies the contact points between the colliding geometries composing a simulation scene. Given two shapes and their relative poses, a collision detection algorithm (e.g., GJK [17] combined with EPA [18]) computes a contact point and a contact normal, corresponding to the direction separating the two bodies with minimal displacement. We define the contact frame c with its origin at the contact point, and the Z axis aligned with the contact normal. Collision detection algorithms often assume the geometries to be convex but existing algorithms [19] can be employed during an offline preprocessing phase to decompose the nonconvex shapes into convex sub-shapes.

Collision detection is inherently nonsmooth for non-strictly convex geometries [20]. Concretely, this induces discontinuous contact points and normals. Thus, differentiating the contact point, the contact normal, and the contact frame w.r.t. the body poses is challenging. [21] uses a smooth approximation of the bodies to calculate the contact frame Jacobians. In contrast, [22] employs a randomized smoothing approach to compute the derivatives of contact points and normals.

2.2 Frictional contact dynamics

Given a contact frame between two bodies, let $\boldsymbol{\lambda}$ denote the contact force and $\boldsymbol{\sigma}$ the contact velocity. The Signorini condition provides a complementarity constraint $0 \leq \boldsymbol{\lambda}_N \perp \boldsymbol{\sigma}_N \geq 0$, ensuring the normal force is repulsive, bodies do not interpenetrate further, and no simultaneous separation motion and contact force exist. The maximum dissipation principle (MDP) combined with the frictional Coulomb law $\|\boldsymbol{\lambda}_T\| \leq \mu\lambda_N$ of friction μ states that $\boldsymbol{\lambda}_T \in \operatorname{argmax}_{y,\|y\|\leq\mu\boldsymbol{\lambda}_N} -y^\top\boldsymbol{\sigma}_T$ maximizes the power dissipated by the contact. These three principles are equivalent to the following nonlinear complementarity problem (NCP)

$$\begin{aligned} &\mathcal{K}_\mu \ni \boldsymbol{\lambda} \perp \boldsymbol{\sigma} + \Gamma_\mu(\boldsymbol{\sigma}) \in \mathcal{K}_\mu^*, \\ &\boldsymbol{\sigma} = G\boldsymbol{\lambda} + g, \end{aligned} \tag{1}$$

where G is the so-called Delassus matrix [23] that gives the system inverse inertia projected on the contacts. It is a linear operator mapping contact forces to contact velocities. $\boldsymbol{g}$ is the free velocity of the contact. $\mathcal{K}_\mu$ is a second-order cone with aperture angle $\operatorname{atan}(\mu)$, $\mathcal{K}_\mu^* = \mathcal{K}_{1/\mu}$ its dual cone and $\Gamma_\mu(\boldsymbol{\sigma}) = [0, 0, \mu\|\boldsymbol{\sigma}_T\|]$ is the so-called De Saxcé correction [24, 25] enforcing the Signorini condition [25, 16]. (1) can be solved by interior point methods [13], projected Gauss-Seidel [26] or ADMM-based approaches [25, 27, 28].

2.3 Multibody frictional contact dynamics

We briefly introduce the simulation of rigid bodies in contact, a core component of physics engines. We refer to [16] for a more detailed background. Let $\boldsymbol{q} \in \mathcal{Q} \cong \mathbb{R}^{n_q}$ denotes the joint position vector with $\mathcal{Q}$ the configuration space, i.e., the space of minimal coordinates. The equations of a constrained motion writes

$$M(\boldsymbol{q})\dot{\boldsymbol{v}} + b(\boldsymbol{q}, \boldsymbol{v}) - \boldsymbol{\tau} = J_c^\top(\boldsymbol{q}, c(\boldsymbol{q}))\boldsymbol{\lambda}, \tag{2}$$

where we denote by $\boldsymbol{v} \in \mathcal{T}_q\mathcal{Q} \cong \mathbb{R}^{n_v}$ and $\boldsymbol{\tau} \in \mathcal{T}_q^*\mathcal{Q} \cong \mathbb{R}^{n_v}$ the joint velocity vector and the joint torque vector. $\dot{\boldsymbol{v}}$ is the time derivative of $\boldsymbol{v}$. $M(\boldsymbol{q})$ is the joint-space inertia matrix, and $b(\boldsymbol{q}, \boldsymbol{v})$ includes terms related to the gravity, Coriolis, and centrifugal effects. $J_c(\boldsymbol{q}, c(\boldsymbol{q}))$ is the contact Jacobian associated with the contact frame $c(\boldsymbol{q})$ given by the collision detection on the system bodies using the configuration $\boldsymbol{q}$. In the following, we drop the dependency on the parameters when it is explicit.

To deal with rigid-body dynamics and impacts, we use an impulse-based formulation [29] obtained with the Euler symplectic scheme

$$\boldsymbol{v}^+ = \boldsymbol{v} + \Delta t \left(\dot{\boldsymbol{v}}_f + M^{-1}J_c^\top\boldsymbol{\lambda}\right), \tag{3}$$

where $\dot{\boldsymbol{v}}_f = M^{-1}(\boldsymbol{\tau} - b)$ is the free acceleration term and Δt is the time step. The acceleration term $\dot{\boldsymbol{v}}_f + M^{-1}J_c^\top \boldsymbol{\lambda}$ of (3) can be efficiently computed with the Articulated Body Algorithm (ABA) [30]. Next, we use the shorthand $\text{ABA}(\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}, \boldsymbol{\lambda}) = \dot{\boldsymbol{v}}_f + M^{-1}J_c^\top \boldsymbol{\lambda}$.

Multiplying (3) by J_c, we recover from the definition of the Jacobian $J_c \boldsymbol{v}^+ = \boldsymbol{\sigma} = G\boldsymbol{\lambda} + g$ associated to the NCP (1) in the case of multibody dynamics. This yields the expression of the Delassus matrix $G = J_c M^{-1} J_c^\top$ and the free contact velocity vector $g = J_c(\boldsymbol{v} + \Delta t \dot{\boldsymbol{v}}_f)$. For poly-articulated rigid-body systems, G depends on $\boldsymbol{q}$, and g depends on $\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}$. In this respect, the associated NCP is conditioned by $\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}$. Note also that the contact Jacobian J_c depends on $\boldsymbol{q}$, first through the kinematic structure of the system and second through the contact frame $c(\boldsymbol{q})$.

2.4 Implicit differentiation

As previously mentioned, the physically accurate contact forces denoted by $\boldsymbol{\lambda}^*$ are implicitly defined as the solution of the NCP (1), we write $0 = \text{NCP}(\boldsymbol{\lambda}^*; \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau})$. By deriving the optimality conditions, the implicit function theorem allows the computation of their gradients and corresponds to the concept of implicit differentiation [7]. The theorem provides locally the derivatives of the solution $\mathrm{d}\boldsymbol{\lambda}^*$ as a linear function of the other variable derivatives.

This approach has been successfully applied to differentiate Quadratic Programming (QP) problems in [31] and generalized to convex cone programs [32] and LCPs [8]. More generally, it allows incorporating optimization layers in the differentiable programming paradigm. In Sec. 3.2, we extend this approach to the NCP case and propose a method to compute the gradients of the contact forces efficiently.

3 Efficient differentiable simulation

This section details the core contribution of this paper, namely a comprehensive framework for differentiable simulation that combines differentiable rigid-body dynamics, differentiable collision detection, and differentiable contact resolution We show the link between the derivatives of the multibody dynamics, the frictional contact dynamics, and the derivatives of collision detection. We introduce an efficient algorithmic solution to compute the derivatives associated with the contact NCP by solving a reduced system of equations of minimal dimension resulting from its implicit differentiation.

3.1 Chaining rigid-body dynamics derivatives and NCP derivatives

From Sec. 2, the simulation equations (3) can be restated using (ABA) and the solution to the (NCP) problem (1) as

$$0 = \text{NCP}(\boldsymbol{\lambda}^*; \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}) \tag{4}$$

$$\boldsymbol{v}^+ = \boldsymbol{v} + \Delta t\, \text{ABA}(\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}, \boldsymbol{\lambda}^*). \tag{5}$$

Next, we consider forward-mode differentiation setup [33], i.e., we aim to compute the Jacobian $\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}$ where $\theta \in \mathbb{R}^{n_p}$ represents any subset of the inputs $\{\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}\}$ or physical parameters. Our approach can be efficiently adapted to the reverse mode by applying the computational trick introduced in [31].

Differentiating (5) leads to

$$\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta} = \underbrace{\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}\theta} + \Delta t \left(\frac{\partial \text{ABA}}{\partial \boldsymbol{q}} \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\theta} + \frac{\partial \text{ABA}}{\partial \boldsymbol{v}} \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}\theta} + \frac{\partial \text{ABA}}{\partial \boldsymbol{\tau}} \frac{\mathrm{d}\boldsymbol{\tau}}{\mathrm{d}\theta} \right)}_{\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\Big|_{\lambda=\lambda^*}} + \Delta t \frac{\partial \text{ABA}}{\partial \boldsymbol{\lambda}} \frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta}, \tag{6}$$

where we identify the term $\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\Big|_{\lambda=\lambda^*}$ of derivatives, considering that $\boldsymbol{\lambda}^*$ does not vary. The ABA derivatives $\frac{\partial \text{ABA}}{\partial \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}}$ and $\frac{\partial \text{ABA}}{\partial \boldsymbol{\lambda}} = M^{-1}(\boldsymbol{q}) J_c^\top(\boldsymbol{q})$ can be efficiently computed via rigid-body algorithms [34] and are, for instance, available in Pinocchio [35]. At this stage, it is worth noting that

$\frac{\partial \text{ABA}}{\partial \boldsymbol{q}}$ also depends on the geometry of the contact through the contact Jacobian $J_c(\boldsymbol{q}, c(\boldsymbol{q}))$. The classical ABA rigid body algorithm gives the derivatives related to the first variable, and we need to add a term related to the variations of J_c induced by the variation of the contact point $c(\boldsymbol{q})$ (see Section 3.4).

Computing the sensitivity of the contact forces $\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta}$ is also challenging as $\boldsymbol{\lambda}^*$ is obtained implicitly by solving a NCP (1) which depends on $\boldsymbol{q}$, $\boldsymbol{v}$ and $\boldsymbol{\tau}$ through G and g. Notably, the solutions of the NCP are intrinsically nonsmooth, corresponding to the solution of a differential inclusion problem [25]. To understand the nonsmoothness of the NCP solutions, consider the case of a contact force either (i) saturating the Coulomb cone or (ii) lying strictly inside the cone. In case (i), the contact force variations must lie on the tangent plane to the cone at this force value, while in case (ii), no restriction on the contact force variations applies. The derivatives do not lie on the constraint manifolds in these two cases. Next, we detail how implicit differentiation of (4) can be leveraged to compute them precisely at a limited computational cost.

3.2 Implicit differentiation of the NCP

The dynamics induced by the NCP (1) is inherently nonsmooth as it can switch on three modes. These modes correspond to the active set of (1) and result in different gradients for the contact dynamics. Our approach considers scenarios with multiple contact points and requires to identify the mode for each contact. For clarity purposes, we present the equations for a single contact point in the case of each of the three modes.

Mode 1 - Breaking contact (brk). This mode corresponds to the case where the contact is separating ($\boldsymbol{\sigma}_N > 0$), which is induced by the Signorini condition that $\boldsymbol{\lambda}^* = 0$. This mode can be treated separately from the other two modes, as the contact force is zero and the contact point velocity is not constrained, yielding

$$\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta} = 0. \tag{7}$$

Mode 2 - Sticking contact (stk). In this mode, the contact point is not moving ($\boldsymbol{\sigma} = 0$), which yields the same equations as a bilateral constraint of an attached point $G\boldsymbol{\lambda}^* + g = 0$. Differentiating this constraint gives the following linear equations on $\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta}$

$$G\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta} = -\left(\frac{\mathrm{d}G}{\mathrm{d}\theta}\boldsymbol{\lambda}^* + \frac{\mathrm{d}g}{\mathrm{d}\theta}\right). \tag{8}$$

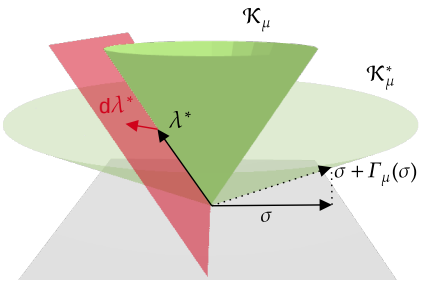

Figure 1: Illustration of the sliding mode. $\boldsymbol{\lambda}^*$ lives in the boundary of the cone $\mathcal{K}_\mu$ in the direction opposite to $\boldsymbol{\sigma} = \boldsymbol{\sigma}_T$ and the variation $\mathrm{d}\boldsymbol{\lambda}^*$ lies inside the tangent plane.

Mode 3 - Sliding contact (sld). When evolving in this regime, the contact point moves on the contact surface, which implies a null normal velocity ($\boldsymbol{\sigma}_N = 0$) and a non-null tangential velocity ($\|\boldsymbol{\sigma}_T\| > 0$). Moreover, from the MDP, tangential contact forces should lie on the boundary of the cone and in the opposite direction of the tangential velocity ($\boldsymbol{\lambda}^*_T = -\mu\boldsymbol{\lambda}^*_N \frac{\boldsymbol{\sigma}_T}{\|\boldsymbol{\sigma}_T\|}$). This additionally implies that $\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta}$ should be in the plane tangent to the friction cone as illustrated in Fig. 3.2 and allows reducing the search space to a 2D plane via a simple change of variable $\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta} = R\frac{\mathrm{d}\tilde{\boldsymbol{\lambda}}}{\mathrm{d}\theta}$ where $R = \begin{pmatrix} \frac{\boldsymbol{\lambda}}{\|\boldsymbol{\lambda}\|} & \boldsymbol{e}_z \times \frac{\boldsymbol{\sigma}_T}{\|\boldsymbol{\sigma}_T\|} \end{pmatrix} \in \mathbb{R}^{3\times 2}$. Therefore, differentiating these equations yields the following conditions on the gradients

$$\tilde{G}\frac{\mathrm{d}\tilde{\boldsymbol{\lambda}}}{\mathrm{d}\theta} = -R^\top P\left(\frac{\mathrm{d}G}{\mathrm{d}\theta}\boldsymbol{\lambda}^* + \frac{\mathrm{d}g}{\mathrm{d}\theta}\right), \tag{9}$$

where $P = \begin{pmatrix} H(\boldsymbol{\sigma}_T) & 0_{2\times1} \\ 0_{1\times2} & 1 \end{pmatrix} \in \mathbb{R}^{3\times3}$, $H(x) = \frac{1}{\alpha}\left(\mathrm{Id} - \frac{x}{\|x\|}\frac{x}{\|x\|}^\top\right) \in \mathbb{R}^{2\times2}$, $\tilde{G} = R^\top P G R + Q$ with $Q = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{2\times2}$ and $\alpha = \frac{\|\boldsymbol{\sigma}_T\|}{\mu \boldsymbol{\lambda}_N}$. We refer to the appendix for the detailed derivation.

3.3 Efficient computation: exploiting kinematic-induced sparsity

One should first identify the active contact modes to obtain the equations of all contact points. We denote $\mathcal{A}_{\text{brk}}$, $\mathcal{A}_{\text{stk}}$, and $\mathcal{A}_{\text{sld}}$ as the sets of contact indices corresponding to the breaking, sticking, and sliding contacts respectively. The dynamics of the different contacts are coupled through the Delassus matrix G. Thus, we construct a matrix $A \in \mathbb{R}^{(3n_{stk}+2n_{sld})\times(3n_{stk}+2n_{sld})}$ where from G we remove the blocks related to $\mathcal{A}_{\text{brk}}$ and modify the lines and columns of G related to $\mathcal{A}_{\text{sld}}$ by following the pattern of $\tilde{G}$ presented in (20). Once this is done, the reduced linear system on X, the stacking of $\mathrm{d}\boldsymbol{\lambda}^*$ and $\mathrm{d}\tilde{\boldsymbol{\lambda}}$, is obtained by concatenating the corresponding right-hand side of (8) and (20). We obtain the linear system corresponding to implicit differentiation

$$AX = -B\left(\frac{\mathrm{d}G}{\mathrm{d}\theta}\boldsymbol{\lambda}^* + \frac{\mathrm{d}g}{\mathrm{d}\theta}\right), \tag{10}$$

where B is block diagonal with identity for blocks of $\mathcal{A}_{\text{stk}}$ and the basis change $R^\top P$ for blocks of $\mathcal{A}_{\text{sld}}$; the complete construction is given in the Appendix.

Computing the right-hand side of (10) requires evaluating $\frac{\mathrm{d}G}{\mathrm{d}\theta}\boldsymbol{\lambda}^* + \frac{\mathrm{d}g}{\mathrm{d}\theta}$. By recalling that $G\boldsymbol{\lambda}^* + g = J_c\boldsymbol{v}^+$ is the contact point velocity, this term exactly corresponds to the derivatives wrt θ of the contact point velocity with $\boldsymbol{\lambda}^*$ taken constant. We have

$$\frac{\mathrm{d}G}{\mathrm{d}\theta}\boldsymbol{\lambda}^* + \frac{\mathrm{d}g}{\mathrm{d}\theta} = J_c\left.\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{\lambda=\lambda^*} + \left.\frac{\mathrm{d}J_c\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{v=v^+}. \tag{11}$$

The first term is already computed thanks to the ABA derivatives (6) [34], and the second term $\left.\frac{\mathrm{d}J_c\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{v=v^+}$, which is the derivatives of the contact velocity with $\boldsymbol{v}^+$ assumed to be constant, can be computed at a reduced cost via the partial derivatives of the forward kinematics [34] evaluated in $\boldsymbol{q}, \boldsymbol{v}^+$. This allows us to avoid the expensive computation of $\frac{\mathrm{d}G}{\mathrm{d}\theta}$ as it is a tensor in general, while only its product with $\boldsymbol{\lambda}$ is required. It is worth noting at this stage that the term $\left.\frac{\mathrm{d}J_c\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{v=v^+}$ also depends on the geometry of the contact. Therefore, computing gradients w.r.t. to the configuration $\boldsymbol{q}$ requires evaluating an additional term for the variations of J_c induced by the variations of contact points on the local geometries [22] and presented next.

Finally, to obtain $\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta}$, we solve the linear system (10) using a QR decomposition of A before projecting back the reduced variables $\mathrm{d}\tilde{\boldsymbol{\lambda}}$ in $\mathbb{R}^3$. At this stage, it is worth noting that these gradients are computed given the current active set, and thus they do not capture the information on the contact modes boundary. Still, this is possible by combining our approach with smoothing techniques explored in [36, 37, 38].

3.4 Collision detection contribution

The collision detection phase depends on the body poses induced by the configuration $\boldsymbol{q}$. Thus, when θ depends on $\boldsymbol{q}$, one must consider the variation of the contact given variations of $\boldsymbol{q}$. Recent work on differentiable collision detection [22, 21] allows computing the derivative of the normal and contact point w.r.t. the poses of the bodies.

By choosing a function that constructs a contact frame c from a contact point and its normal, we compute the derivative of this frame w.r.t. the body poses. Chaining this derivative with the usual kinematics Jacobian, which relates the variation of $\boldsymbol{q}$ to the variation of body poses, one can obtain $\frac{\mathrm{d}c}{\mathrm{d}\theta}$.

The frame c intervenes in J_c through a change of frame (the adjoint of the placement). By leveraging spatial algebra[30] (see the appendix for the details), we calculate $\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c}$ and $\frac{\partial J_c \boldsymbol{v}^+}{\partial c}$. We add the

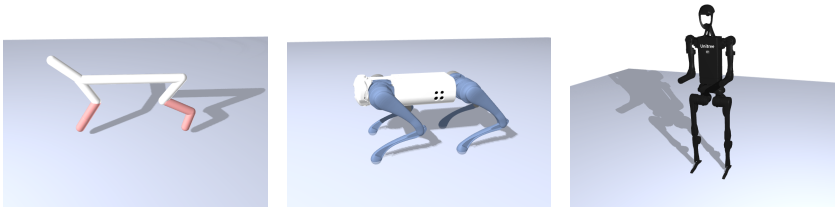

Figure 2: The robotics systems used to evaluate our approach range from simple systems such as MuJoCo's half-cheetah (**Left**) to more complex high-dof robots such as Unitree's Go1 (**Center**) and H1 (**Right**)

	Half-cheetah	Humanoid	UR5	Go1	H1	Atlas
Number of DoFs	12	27	7	18	25	36
Number of geometries	9	20	9	39	25	89
Number of collision pairs	27	161	26	494	255	3399
Number of vertices per mesh	N/A	N/A	200	N/A	700	N/A

Table 2: Details of scenarios for computational timings. For the number of vertices per mesh, N/A indicates a scenario which does not contain any mesh.

first collision term $\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c}\frac{\mathrm{d}c}{\mathrm{d}\theta}$ in $\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\Big|_{\lambda=\lambda^*}$ and the second collision term $\frac{\partial J_c \boldsymbol{v}^+}{\partial c}\frac{\mathrm{d}c}{\mathrm{d}\theta}$ in $\frac{\mathrm{d}J_c\boldsymbol{v}^+}{\mathrm{d}\theta}\Big|_{v=v^+}$ to account for the variation of J_c due to the variation of the contact points. The complete details of the terms and simulation derivative $\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}$ are reported in the appendix.

4 Experiments

In this section, we first demonstrate state-of-the-art computational timings when computing simulation gradients for various robotic systems (Fig. 2). Second, we apply our approach to solve two inverse problems involving contact dynamics: retrieving an initial condition that leads to a target final state and finding a torque that yields a null acceleration on a quadruped. Additional experiments are available in the Appendix.

Implementation details. We have implemented our analytical derivatives in C++ for efficiency. We leverage open-source software of the community: Eigen [39] for efficient linear algebra, Pinocchio [35] for fast rigid body dynamics and their derivatives, and HPP-FCL [40] for high-speed collision detection. The code associated with this paper will be released as open-source as part of the Simple simulator. All the experiments are performed on a single core of an Apple M3 CPU.

4.1 Timings

The computational efficiency of our approach is evaluated by measuring the average time required to compute the full Jacobian of the simulator, i.e., $\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\boldsymbol{q},\boldsymbol{v},\boldsymbol{\tau}}$, along a trajectory. We consider various scenarios ranging from simple systems composed of basic geometry primitives (MuJoCo's half-cheetah and humanoid) to more complex and realistic robots with multiple DoFs and complex geometries (UR5, Unitree Go1, and Boston Dynamics Atlas). Tab. 2 reports the numbers associated with the different robots considered. To stabilize the simulation behaviors, we compute contact collision patches (composed of 4 contact points each), thus substantially increasing the dimensions of the problem to solve.

Tab. 3 demonstrates computational timings for gradient computation that are of the same order of magnitude as simulation and significantly faster than central finite differences. As another point of comparison, Nimble [10] requires 1ms and 16ms on half-cheetah and Atlas, corresponding to an approximate speedup factor of 100 for our method. In Tab. 3, we also compare our approach to MuJoCo MJX GPU simulation pipeline and find it to be competitive even though it operates on a single CPU core. Importantly, our performance gain is obtained although we work on the unrelaxed NCP and with full meshes descriptions (cf. Tab. 1).

	Half-cheetah	Humanoid	UR5	Go1	H1	Atlas	Framework
Simulation	15.8 ± 7.1	42.6 ± 24.7	12.3 ± 6.1	62.6 ± 18.5	139.3 ± 125.4	127.3 ± 32.9	**Ours**
Implicit gradients	14.7 ± 7.0	47.9 ± 23.8	4.1 ± 3.8	93.0 ± 32.0	54.3 ± 34.5	95.2 ± 37.6	Apple M3 CPU
Finite differences	$1.1e3 \pm 0.5e3$	$5.5e3 \pm 3.5e3$	$0.4e3 \pm 0.2e3$	$6.6e3 \pm 1.8e3$	$17.6e3 \pm 14.6e3$	$26.7e3 \pm 6.e3$	
Simulation	5.5 ± 2.0	40.3 ± 40.1	12.3 ± 4.0	15.8 ± 7.0	59.3 ± 31.0	85.1 ± 34.4	**MuJoCo**
Finite differences	$0.34e3 \pm 0.13e3$	$2.9e3 \pm 0.8e3$	$0.39e3 \pm 0.10e3$	$9.9e3 \pm 0.16e3$	$5.9e3 \pm 1.4e3$	$54.3e3 \pm 1.4e3$	Apple M3 CPU
Simulation	1.0 ± 0.0	2.3 ± 0.0	N/A	N/A	N/A	N/A	**MJX**
Autodiff gradients	3.7 ± 0.0	103.2 ± 0.3	N/A	N/A	N/A	N/A	Nvidia A100 GPU

Table 3: Comparative analysis between ours, MuJoCo, and MJX frameworks. Timing statistics (mean and standard deviation in microseconds) for simulation, gradient, and finite-differences computation for one simulation step. For MJX, N/A denotes scenarios where geometries were not supported.

4.2 Inverse problems

Estimating initial conditions. As a first application of differentiable simulation, we aim at retrieving the initial condition θ (either the initial velocity $\boldsymbol{v}_0$ or an initial impulse $\boldsymbol{\tau}_0$), leading to a target final state $\boldsymbol{q}_T^*$ after T time steps. Here, we consider the case of a cube thrown on the floor evolving in a sliding mode. The problem can be written as:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \left\| \boldsymbol{q}_T(\theta) - \boldsymbol{q}_T^* \right\|_2^2, \tag{12}$$

where $\boldsymbol{q}_T$ is the final configuration and depends on the initial velocity and impulse. Our forward-mode differentiation allows us to efficiently compute the Jacobian of $\boldsymbol{q}_T$ wrt $\boldsymbol{\theta}$. We leverage this feature to implement a Gauss-Newton (GN) approximation of the Hessian of (12). Fig. 3 demonstrates the benefits of using our implicit gradients over finite-differences in order to reach a precise solution of (12). Moreover, exploiting the full Jacobian in a quasi-Newton algorithm also reduces the number of iterations compared to a vanilla gradient descent.

Retrieving the initial impulse on the cube $\boldsymbol{\tau}_0$ is a challenging nonsmooth and nonconvex optimization problem, which can explain the plateau reached by our vanilla Gauss-Newton implementation. Working on dedicated nonsmooth optimization algorithms is a promising research direction that could lead to higher-quality solutions.

Inverse dynamics through contacts. We evaluate our approach on an Inverse Dynamics (ID) task involving contacts. In particular, we aim at finding the torque on actuators $\boldsymbol{\tau}_{\text{act}}$ leading to a null acceleration for a Unitree Go1 quadrupedal robot in a standing position $(\boldsymbol{q}, \boldsymbol{v})$. By denoting S the actuation matrix, the ID problem can be formulated as follows:

$$\min_{\boldsymbol{\tau}_{\text{act}}} \frac{1}{2} \left\| \boldsymbol{v}^+(\boldsymbol{q}, \boldsymbol{v}, S^T \boldsymbol{\tau}_{\text{act}}) - \boldsymbol{v}^* \right\|_2^2, \tag{13}$$

where the initial $\boldsymbol{v}$ and target $\boldsymbol{v}^*$ velocities are null in this example. As previously explained, we use the Jacobians computed by our differentiable simulator with a Gauss-Newton algorithm. As shown by Fig. 4, the problem is solved with high accuracy in only a few iterations (approx. 10 to reach an

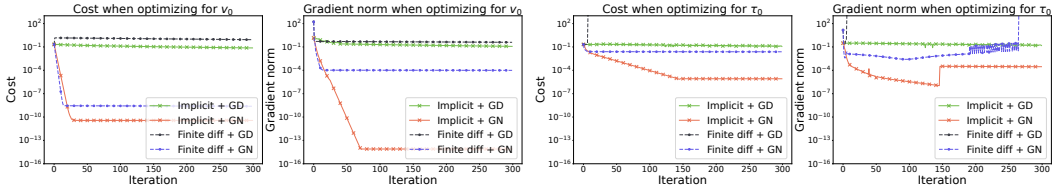

Figure 3: **Estimation of initial conditions.** A Gauss-Newton (GN) algorithm can leverage the efficient implicit differentiation to accurately retrieve the initial velocity $\boldsymbol{v}_0$ and impulse $\boldsymbol{\tau}_0$. On the third and fourth figures, the black curve representing Gradient Descent with finite differences rises due to the excessively large estimated gradients. When at the boundary of a contact mode, the norm of the finite differences gradients becomes inversely proportional to the step size used.

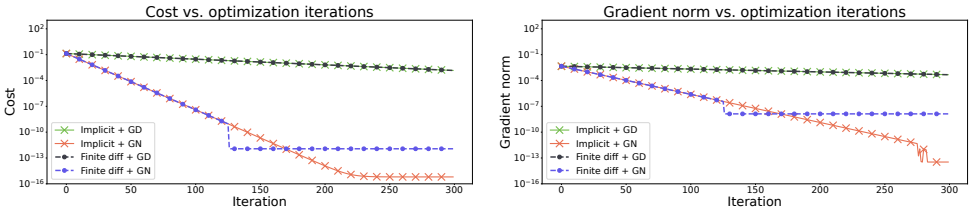

Figure 4: **Contact inverse dynamics** on an underactuated Go1 quadruped can be efficiently performed via a Gauss-Newton algorithm by using Jacobians of our differentiable physics engine.

error of $1e-5$). Just like in the initial conditions estimation setup, implicit gradients allow us to solve the problem with a higher precision than finite differences.

5 Limitations

This paper introduces an end-to-end differentiable physics pipeline for robotics based on the implicit differentiation of the non-relaxed NCP for contacts. By avoiding any relaxation, we prevent the appearance of unphysical simulation artifacts. Moreover, exploiting the sparsity induced by the robot kinematic chains and leveraging the derivatives of rigid body algorithms allows us to achieve state-of-the-art timings, with a speed-up of at least 100 compared to alternative solutions of the state of the art. In an MPC context where the dynamics and its derivatives are evaluated at a high frequency, the gains in physical realism and efficiency could determine the controller's overall performance.

Yet, as the NCP induces inherently non-smooth dynamics, exploiting its gradients requires dedicated algorithms when addressing downstream optimization tasks. Some previous works [41, 42, 37] leverage randomized smoothing techniques that provide smooth gradient estimates from simulation and gradient samples. Alternative solutions relax the physics, either explicitly [2, 1, 43] or implicitly, by leveraging interior-points (IP) methods [13].

Similarly to existing robotic simulators (e.g., MuJoCo, Bullet, DART), this paper models contact interactions as vanilla 3d contact points, while richer but more complex contact models exist. One promising research direction could consider extending this work towards deformable contact interactions to enhance simulator realism, such as in [44].

Conclusion. In future work, we plan to integrate our qualitative gradient approach with recent advancements in control frameworks to tackle more complex robotics tasks. Specifically, this paper introduces an efficient method for computing physics gradients, which paves the way for first-order policy learning algorithms such as SHAC [45]. It is also worth noticing that our approach is not limited to rigid-body robots, but could also be leveraged for soft dynamics in general to design and control soft robots [46] and could be adapted to use implicit integrator as in [47]. The proposed differentiable simulation of an unrelaxed physics model is a crucial step toward reducing the Sim2Real gap [48], and future work may adapt learning algorithms to effectively achieve this goal. Finally, we hope this work will serve as a catalyzer in the robotics and learning communities and motivate the development of new reinforcement learning and trajectory optimization methods leveraging simulation gradients in order to accelerate the discovery of complex robot movements in contact.

Acknowledgments

This work was supported in part by the French government under the management of Agence Nationale de la Recherche (ANR) as part of the "Investissements d'avenir" program, references ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and ANR-22-CE33-0007 (INEXACT), the European project AGIMUS (Grant 101070165), the Louis Vuitton ENS Chair on Artificial Intelligence and the Casino ENS Chair on Algorithmic and Machine Learning. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (ToG)*, 31(4):1–8, 2012.

[2] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[3] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations*, 2019.

[4] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. NeuralSim: Augmenting differentiable simulators with neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL `https://github.com/google-research/tiny-differentiable-simulator`.

[5] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax-a differentiable physics engine for large scale rigid body simulation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[6] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[7] M. Blondel and V. Roulet. The elements of differentiable programming. *arXiv preprint arXiv:2403.14606*, 2024.

[8] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

[9] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[10] 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*, 2021.

[11] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin. Efficient differentiable simulation of articulated bodies. In *ICML*, 2021.

[12] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier. Differentiable simulation for physical system identification. *IEEE Robotics and Automation Letters*, 6(2):3413–3420, 2021.

[13] T. A. Howell, S. Le Cleac'h, J. Bruedigam, J. Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A Differentiable Simulator for Robotics. 2022.

[14] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal. An End-to-End Differentiable Framework for Contact-Aware Robot Design. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi:10.15607/RSS.2021.XVII.008.

[15] Y. D. Zhong, J. Han, and G. O. Brikis. Differentiable physics simulations with contacts: Do they have correct gradients w.r.t. position, velocity and control? *arXiv preprint arXiv:2207.05060*, 2022.

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

[17] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

[18] G. Van den Bergen. Proximity Queries and Penetration Depth Computation on 3D Game Objects. *In Game Developers Conference*, 2001.

[19] X. Wei, M. Liu, Z. Ling, and H. Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

[20] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar. A strictly convex hull for computing proximity distances with continuous gradients. *IEEE Transactions on Robotics*, 30(3):666–678, 2014.

[21] K. Tracy, T. A. Howell, and Z. Manchester. Differentiable collision detection for a set of convex primitives. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3663–3670. IEEE, 2023.

[22] 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)*, pages 3240–3246. IEEE, 2023.

[23] É. Delassus. Mémoire sur la théorie des liaisons finies unilatérales. In *Annales scientifiques de l'École normale supérieure*, volume 34, pages 95–179, 1917.

[24] G. de Saxcé and Z.-Q. Feng. The bipotential method: A constructive approach to design the complete contact law with friction and improved numerical algorithms. *Mathematical and Computer Modelling*, 28(4-8):225–245, Aug. 1998. doi:10.1016/S0895-7177(98)00119-8. URL https://hal.archives-ouvertes.fr/hal-03883288.

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

[26] F. Jourdan, P. Alart, and M. Jean. A gauss-seidel like algorithm to solve frictional contact problems. *Computer methods in applied mechanics and engineering*, 155(1-2):31–47, 1998.

[27] A. Tasora, D. Mangoni, S. Benatti, and R. Garziera. Solving variational inequalities and cone complementarity problems in nonsmooth dynamics using the alternating direction method of multipliers. *International Journal for Numerical Methods in Engineering*, 122(16):4093–4113, 2021.

[28] J. Carpentier, Q. Le Lidec, and L. Montaut. From compliant to rigid contact simulation: a unified and efficient approach. *Robotics: Science and Systems*, 2024.

[29] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff, 1995.

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

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

[32] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

[33] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

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

[35] 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 *IEEE International Symposium on System Integrations (SII)*, 2019.

[36] H. J. Suh, M. Simchowitz, K. Zhang, and R. Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pages 20668–20696. PMLR, 2022.

[37] T. Pang, H. 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*, 2023.

[38] S. Zhang, W. Jin, and Z. Wang. Adaptive barrier smoothing for first-order policy gradient with contact dynamics. In *International Conference on Machine Learning*, pages 41219–41243. PMLR, 2023.

[39] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[40] J. Pan, S. Chitta, J. Pan, D. Manocha, J. Mirabel, J. Carpentier, and L. Montaut. HPP-FCL - An extension of the Flexible Collision Library, Mar. 2024. URL `https://github.com/humanoid-path-planner/hpp-fcl`.

[41] H. J. T. Suh, T. Pang, and R. Tedrake. Bundled gradients through contact via randomized smoothing. *IEEE Robotics and Automation Letters*, 7(2):4000–4007, 2022.

[42] Q. Le Lidec, F. Schramm, L. Montaut, C. Schmid, I. Laptev, and J. Carpentier. Leveraging randomized smoothing for optimal control of nonsmooth dynamical systems. *Nonlinear Analysis: Hybrid Systems*, 52:101468, 2024.

[43] G. Kim, D. Kang, J.-H. Kim, S. Hong, and H.-W. Park. Contact-implicit mpc: Controlling diverse quadruped motions without pre-planned contact modes or trajectories. *arXiv preprint arXiv:2312.08961*, 2023.

[44] R. Elandt, E. Drumwright, M. Sherman, and A. Ruina. A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8238–8245. IEEE, 2019.

[45] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2021.

[46] C. Della Santina, C. Duriez, and D. Rus. Model-based control of soft robots: A survey of the state of the art and open challenges. *IEEE Control Systems Magazine*, 43(3):30–65, 2023.

[47] A. M. Castro, F. N. Permenter, and X. Han. An unconstrained convex formulation of compliant contact. *IEEE Transactions on Robotics*, 39(2):1301–1320, 2023. doi:10.1109/TRO.2022.3209077.

[48] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White. Sim2real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2):398–400, 2021. doi:10.1109/TASE.2021.3064065.

Appendix

This appendix document reports the details associated with the submission entitled *End-to-End and Highly-Efficient Differentiable Simulation for Robotics*. In particular, we provide detailed equations related to the three modes of the NCP implicit differentiation, the contributions of collision detection, and the detailed expression of multibody derivatives accounting for contact and collision derivatives.

Appendix A - Sliding mode

In this subsection, we detail the equations (Eq. (9) of the paper) of the sensitivity analysis of the contact force in the case of a sliding mode ($\|\boldsymbol{\sigma}_T\| > 0$).

In sliding mode, both the contact forces and the contact point velocity are on the border of their cone. Thus, from the NCP we have:

$$\|\boldsymbol{\lambda}_T\| = \mu \boldsymbol{\lambda}_N \tag{14a}$$
$$\boldsymbol{\sigma}_N = 0 \tag{14b}$$
$$\boldsymbol{\lambda}^\top \big(\boldsymbol{\sigma} + \Gamma_\mu(\boldsymbol{\sigma})\big) = 0 \tag{14c}$$
$$\boldsymbol{\sigma} = G\boldsymbol{\lambda} + g, \tag{14d}$$

which is equivalent to:

$$\boldsymbol{\lambda}_T = -\mu \boldsymbol{\lambda}_N \frac{\boldsymbol{\sigma}_T}{\|\boldsymbol{\sigma}_T\|} \tag{15a}$$
$$\boldsymbol{\sigma}_T = G_T\boldsymbol{\lambda} + g_T \tag{15b}$$
$$G_N\boldsymbol{\lambda} + g_N = 0. \tag{15c}$$

By differentiating, we get the following system on the derivatives:

$$A\mathrm{d}\boldsymbol{\lambda} = -\frac{1}{\alpha} H d\boldsymbol{\sigma}_T \tag{16a}$$
$$d\boldsymbol{\sigma}_T = G_T d\boldsymbol{\lambda} + (dG_T\boldsymbol{\lambda} + dg_T) \tag{16b}$$
$$G_N\mathrm{d}\boldsymbol{\lambda} = -\left(\mathrm{d}G_N\boldsymbol{\lambda} + \mathrm{d}g_N\right), \tag{16c}$$

where $K = (\mathrm{Id}_2 \quad \mu \boldsymbol{u}_T) \in \mathbb{R}^{2\times 3}$ and $H = \left(\mathrm{Id} - \boldsymbol{u}_T\boldsymbol{u}_T^\top\right) \in \mathbb{R}^{2\times 2}$ with $\boldsymbol{u}_T = \frac{\boldsymbol{\sigma}_T}{\|\boldsymbol{\sigma}_T\|}$ and $\alpha = \frac{\|\boldsymbol{\sigma}_T\|}{\mu\boldsymbol{\lambda}_N}$. We rewrite as

$$\left(\frac{1}{\alpha} H G_T + K\right)\mathrm{d}\boldsymbol{\lambda} = -\frac{1}{\alpha} H\left(\mathrm{d}G_T\boldsymbol{\lambda} + \mathrm{d}g_T\right) \tag{17a}$$
$$G_N\mathrm{d}\boldsymbol{\lambda} = -\left(\mathrm{d}G_N\boldsymbol{\lambda} + \mathrm{d}g_N\right). \tag{17b}$$

Stacking the two equations yields:

$$(PG + \tilde{K})\mathrm{d}\boldsymbol{\lambda} = -P(\mathrm{d}G\boldsymbol{\lambda} + \mathrm{d}g), \tag{18}$$

where we introduce $P = \begin{pmatrix} \frac{1}{\alpha}H & 0_{2\times 1} \\ 0_{1\times 2} & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3}$ and $\tilde{K} = \begin{pmatrix} K \\ 0_{1\times 3} \end{pmatrix} \in \mathbb{R}^{3\times 3}$.

Because $\boldsymbol{\lambda}$ is constrained to stay on the boundary of the cone, its variations $\mathrm{d}\boldsymbol{\lambda}$ live in the tangent 2D plane whose basis is $R = \left(\frac{\boldsymbol{\lambda}}{\|\boldsymbol{\lambda}\|} \quad \boldsymbol{e}_z \times \frac{\boldsymbol{\sigma}_T}{\|\boldsymbol{\sigma}_T\|}\right) \in \mathbb{R}^{3\times 2}$. Applying the change of variable $\mathrm{d}\boldsymbol{\lambda} = R\mathrm{d}\tilde{\boldsymbol{\lambda}}$ and multiplying the previous system 18 by $R^\top$ allows getting a linear system of reduced dimension:

$$\tilde{G}\mathrm{d}\tilde{\boldsymbol{\lambda}} = -R^\top P\left(\mathrm{d}G\boldsymbol{\lambda} + \mathrm{d}g\right), \tag{19}$$

where $\tilde{G} = R^\top PGR + Q \in \mathbb{R}^{2\times 2}$ with $Q = R^\top \tilde{K} R = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{2\times 2}$. Hence the final expression when taking derivative w.r.t θ in the optimal force $\boldsymbol{\lambda}^*$

$$\tilde{G}\frac{\mathrm{d}\tilde{\boldsymbol{\lambda}}}{\mathrm{d}\theta} = -R^\top P\left(\frac{\mathrm{d}G}{\mathrm{d}\theta}\boldsymbol{\lambda}^* + \frac{\mathrm{d}g}{\mathrm{d}\theta}\right). \tag{20}$$

Appendix B - Implicit NCP gradient system

In this subsection, we detail the final system solved to compute the gradients of the NCP (equation (10) of the paper).

Following the paper notations, we denote $\mathcal{A}_{\text{brk}}$, $\mathcal{A}_{\text{stk}}$, and $\mathcal{A}_{\text{sld}}$ as the sets of contact indices corresponding to the breaking, sticking, and sliding contacts respectively and n_{brk}, n_{stk}, and n_{sld} their cardinals and $n = n_{\text{brk}} + n_{\text{stk}} + n_{\text{sld}}$ the total number of contacts. For the implicit gradients system, we removed the $\mathrm{d}\boldsymbol{\lambda}$ variables associated with contacts in $\mathcal{A}_{brk}$ and sorted the remaining ones by putting first the components associated with the contacts in $\mathcal{A}_{stk}$ before the reduced ones $\mathrm{d}\tilde{\boldsymbol{\lambda}}$ of $\mathcal{A}_{sld}$. Then the total variation of $\boldsymbol{\lambda}$ is $\mathrm{d}\boldsymbol{\lambda} = CX \in \mathbb{R}^{3n}$ with

$$X = \begin{pmatrix} \mathrm{d}\boldsymbol{\lambda}^{(1)} \\ \vdots \\ \mathrm{d}\boldsymbol{\lambda}^{(n_{\text{stk}})} \\ \mathrm{d}\tilde{\boldsymbol{\lambda}}^{(1)} \\ \vdots \\ \mathrm{d}\tilde{\boldsymbol{\lambda}}^{(n_{\text{sld}})} \end{pmatrix} \in \mathbb{R}^{3n_{stk}+2n_{sld}} \tag{21}$$

$$C = \left(\begin{array}{c|c} 0_{3n_{\text{brk}},3n_{\text{stk}}} & 0_{3n_{\text{brk}},2n_{\text{sld}}} \\ \hline \mathrm{Id}_{3n_{\text{stk}},3n_{\text{stk}}} & 0_{3n_{\text{stk}},2n_{\text{sld}}} \\ \hline & R^{(1)} \\ 0_{3n_{\text{sld}},3n_{\text{stk}}} & \ddots \\ & R^{(n_{\text{sld}})} \end{array}\right) \in \mathbb{R}^{3n\times(3n_{stk}+2n_{sld})} \tag{22}$$

Note that in the sticking case, the right-hand side is $\mathrm{d}G\boldsymbol{\lambda} + \mathrm{d}g$ and the left-hand side is $G\mathrm{d}\lambda$ and for the sliding mode, the right-hand side is multiplied by $R^\top P$ and the right-hand side is composed by $R^\top P$ and R plus an additional term due to Q. So if we introduce B and A as

$$B = \left(\begin{array}{c|c|c} 0_{3n_{\text{stk}},3n_{\text{brk}}} & \mathrm{Id}_{3n_{stk},3n_{stk}} & 0_{3n_{\text{stk}},3n_{\text{sld}}} \\ \hline & & R^{(1)\top}P^{(1)} \\ 0_{2n_{\text{sld}},3n_{\text{brk}}} & 0_{2n_{\text{sld}},3n_{stk}} & \ddots \\ & & R^{(n_{\text{sld}})\top}P^{(n_{\text{sld}})} \end{array}\right) \in \mathbb{R}^{(3n_{stk}+2n_{sld})\times 3n} \tag{23}$$

$$A = BGC + \left(\begin{array}{c|c} 0_{3n_{\text{stk}},3n_{\text{stk}}} & 0_{3n_{\text{brk}},2n_{\text{sld}}} \\ \hline & Q \\ 0_{2n_{\text{sld}},3n_{\text{brk}}} & \ddots \\ & Q \end{array}\right) \in \mathbb{R}^{(3n_{stk}+2n_{sld})\times(3n_{stk}+2n_{sld})} \tag{24}$$

where G is the complete Delassus matrix that induces coupling between the different contacts. We recover the linear system of the implicit gradient

$$AX = -B(\mathrm{d}G\lambda + \mathrm{d}g), \tag{25}$$

and, finally, the derivative w.r.t θ of the force taken in the optimal force λ^* as

$$\frac{\mathrm{d}\boldsymbol{\lambda}^*}{\mathrm{d}\theta} = -CA^{-1}B\left(\frac{\mathrm{d}G}{\mathrm{d}\theta}\boldsymbol{\lambda}^* + \frac{\mathrm{d}g}{\mathrm{d}\theta}\right). \tag{26}$$

Details on implementation. In practice, the matrix B and C are not computed, but we directly work on G and $(\mathrm{d}G\lambda + \mathrm{d}g)$ by discarding the right lines and modifying groups of columns and lines to exploit the sparsity of B and C. We compute the matrix A and its inverse using a QR decomposition.

Appendix C - Collision detection contribution

Given one contact frame c between body 1 and body 2, the contact Jacobian is

$$J_c = E({}^cX_1J_1 - {}^cX_2J_2), \tag{27}$$

where $E = \begin{pmatrix} \mathrm{Id}_3 & 0_{3,3} \\ 0_{3,3} & 0_{3,3} \end{pmatrix}$ is an operator that allows the extraction of the linear part, and J_1, J_2 are the kinematic Jacobian of the bodies 1 and 2

Here, we present the terms $\frac{\partial J_c \boldsymbol{v}^+}{\partial c}\frac{\mathrm{d}c}{\mathrm{d}\theta}$ and $\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c}\frac{\mathrm{d}c}{\mathrm{d}\theta}$ from Section 3.3 using c to be agnostic of the choice of representation of the contact frame c. In practice, we use HPP-FCL to get the placement of c relative to the world given the placement of bodies 1 ($^0M_1(\boldsymbol{q})$) and 2 ($^0M_2(\boldsymbol{q})$) relative to the world. In general, we can write:

$$^0M_c(\boldsymbol{q}) = \mathrm{CD}(^0M_1(\boldsymbol{q}), ^0M_2(\boldsymbol{q})), \tag{28}$$

where CD is an acronym for *collision detection*. We are interested in the derivatives of $J_c\boldsymbol{v}$ when J_1 and J_2 are considered constant because their derivation is already considered in the other terms. In this setting, we rewrite (27) apply to a joint velocity $\boldsymbol{v}$ as

$$J_c\boldsymbol{v} = E(\mathrm{Ad}_{^0M_c^{-1}(\boldsymbol{q})^0M_1(\boldsymbol{q})}\, J_1\boldsymbol{v} - \mathrm{Ad}_{^0M_c^{-1}(\boldsymbol{q})^0M_2(\boldsymbol{q})}\, J_2\boldsymbol{v}), \tag{29}$$

where Ad denotes the adjoint operator on $\mathrm{SE}(3)$ to explicitly show the dependency in the variables. We note that only $\boldsymbol{q}$ appears in (29).

First, we have

$$\frac{\mathrm{d}^0M_c}{\mathrm{d}\theta} = \frac{\partial \mathrm{CD}}{\partial ^0M_1}J_1\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\theta} + \frac{\partial \mathrm{CD}}{\partial ^0M_2}J_2\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\theta}, \tag{30}$$

which can be computed using the randomized smoothed derivatives presented in *Differentiable collision detection: a randomized smoothing approach* [16] in the main paper.

Second, with the rules of spatial algebra, we have for a vector x of the lie algebra, and placement M, M_a, M_b in $\mathrm{SE}(3)$

$$\mathrm{d}(\mathrm{Ad}_M\, x) = \mathrm{Ad}_M\, \mathrm{ad}_{\mathrm{d}M}\, x = -\,\mathrm{ad}_{\mathrm{Ad}_M\, x}\, \mathrm{Ad}_M\, \mathrm{d}M \tag{31a}$$

$$\mathrm{d}(M_a^{-1}M_b) = -\,\mathrm{Ad}_{M_b^{-1}M_a}\, \mathrm{d}M_a + \mathrm{d}M_b, \tag{31b}$$

with ad the small adjoint on the Lie algebra and by the chain rule

$$\mathrm{d}(\mathrm{Ad}_{M_a^{-1}M_b}\, x) = \mathrm{ad}_{\mathrm{Ad}_{M_a^{-1}M_b}\, x}\, \mathrm{d}M_a - \mathrm{Ad}_{M_a^{-1}M_b}\, \mathrm{ad}_x\, \mathrm{d}M_b, \tag{32}$$

so we have

$$\begin{aligned}\mathrm{d}(J_c\boldsymbol{v}) &= E(\mathrm{ad}_{\mathrm{Ad}_{c_{M_1}}\, J_1\boldsymbol{v}}\, \mathrm{d}^0M_c - \mathrm{Ad}_{c_{M_1}}\, \mathrm{ad}_{J_1\boldsymbol{v}}\, J_1 d\boldsymbol{q} - \mathrm{ad}_{\mathrm{Ad}_{c_{M_2}}\, J_2\boldsymbol{v}}\, \mathrm{d}^0M_c + \mathrm{Ad}_{c_{M_2}}\, \mathrm{ad}_{J_2\boldsymbol{v}}\, J_2 d\boldsymbol{q}) \\ &= E(\mathrm{ad}_{J_c\boldsymbol{v}}\, \mathrm{d}^0M_c - \mathrm{Ad}_{c_{M_1}}\, \mathrm{ad}_{J_1\boldsymbol{v}}\, J_1 d\boldsymbol{q} + \mathrm{Ad}_{c_{M_2}}\, \mathrm{ad}_{J_2\boldsymbol{v}}\, J_2 d\boldsymbol{q}),\end{aligned} \tag{33}$$

using the linearity of ad in the index variable. And finally, we obtain the term

$$\boxed{\frac{\partial J_c\boldsymbol{v}^+}{\partial c}\frac{\mathrm{d}c}{\mathrm{d}\theta} = E\left[\left(\mathrm{ad}_{J_c\boldsymbol{v}^+}\frac{\partial \mathrm{CD}}{\partial ^0M_1} - {}^cX_1\,\mathrm{ad}_{J_1\boldsymbol{v}^+}\right)J_1 - \left(\mathrm{ad}_{J_c\boldsymbol{v}^+}\frac{\partial \mathrm{CD}}{\partial ^0M_2} - {}^cX_2\,\mathrm{ad}_{J_2\boldsymbol{v}^+}\right)J_2\right]\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\theta}.}$$

To compute the $\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c}\frac{\mathrm{d}c}{\mathrm{d}\theta}$ we use the previous term and the duality stating that for any $\boldsymbol{\lambda}$ and $\boldsymbol{v}$ we have $\langle J_c^\top\boldsymbol{\lambda}, \boldsymbol{v}\rangle = \langle \boldsymbol{\lambda}, J_c\boldsymbol{v}\rangle$. Taking derivatives we have

$$\begin{aligned}\langle \partial(J_c^\top\boldsymbol{\lambda})\mathrm{d}\boldsymbol{q}, \boldsymbol{v}\rangle &= \langle \lambda, \partial(J_c\boldsymbol{v})\mathrm{d}\boldsymbol{q}\rangle \\ &= \langle \boldsymbol{\lambda}, L\boldsymbol{v}\rangle \\ &= \langle L^\top\boldsymbol{\lambda}, \boldsymbol{v}\rangle,\end{aligned} \tag{34}$$

and because it is true for any $\boldsymbol{v}$ we have $\partial(J_c\boldsymbol{v})\mathrm{d}\boldsymbol{q} = L^\top\boldsymbol{\lambda}$. We calculate:

$$\begin{aligned}\partial(J_c\boldsymbol{v})\mathrm{d}\boldsymbol{q} &= E\left[\left(\mathrm{ad}_{J_c\boldsymbol{v}}\frac{\partial \mathrm{CD}}{\partial ^0M_1} - {}^cX_1\,\mathrm{ad}_{J_1\boldsymbol{v}}\right)J_1 - \left(\mathrm{ad}_{J_c\boldsymbol{v}}\frac{\partial \mathrm{CD}}{\partial ^0M_2} - {}^cX_2\,\mathrm{ad}_{J_2\boldsymbol{v}}\right)J_2\right]\mathrm{d}\boldsymbol{q} \\ &= L\boldsymbol{v} = -E\left[\left(\mathrm{ad}_{\frac{\partial \mathrm{CD}}{\partial ^0M_1}J_1\mathrm{d}\boldsymbol{q}}\, J_c - {}^cX_1\,\mathrm{ad}_{J_1\mathrm{d}\boldsymbol{q}}\, J_1\right) - \left(\mathrm{ad}_{\frac{\partial \mathrm{CD}}{\partial ^0M_2}\mathrm{d}\boldsymbol{q}}\, J_c - {}^cX_2\,\mathrm{ad}_{J_2\mathrm{d}\boldsymbol{q}}\, J_2\right)\right]\boldsymbol{v},\end{aligned} \tag{35}$$

using the anti-commutativity of ad. Then we have

$$\begin{aligned} L^\top \boldsymbol{\lambda} &= -\left[\left(J_c^\top \,\mathrm{ad}^\top_{\frac{\partial \mathrm{CD}}{\partial^0 M_1} J_1 \mathrm{d}\boldsymbol{q}} - J_1^\top \,\mathrm{ad}_{J_1 \mathrm{d}\boldsymbol{q}}{}^c X_1^\top\right) - \left(J_c^\top \,\mathrm{ad}^\top_{\frac{\partial \mathrm{CD}}{\partial^0 M_2} J_2 \mathrm{d}\boldsymbol{q}} - J_2^\top \,\mathrm{ad}_{J_2 \mathrm{d}\boldsymbol{q}}{}^c X_2^\top\right)\right] E\boldsymbol{\lambda} \\ &= -\left[\left(J_c^\top \,\mathrm{P}_{E\boldsymbol{\lambda}} \frac{\partial \mathrm{CD}}{\partial^0 M_1} J_1 - J_1^\top \,\mathrm{P}_{{}^c X_1^\top E\boldsymbol{\lambda}}\, J_1\right) - \left(J_c^\top \,\mathrm{P}_{E\boldsymbol{\lambda}} \frac{\partial \mathrm{CD}}{\partial^0 M_2} J_2 - J_2^\top \,\mathrm{P}_{{}^c X_2^\top E\boldsymbol{\lambda}}\, J_2\right)\right] \mathrm{d}\boldsymbol{q}, \end{aligned} \tag{36}$$

where P is the variable commutation of $\mathrm{ad}^\top$. Precisely for all x in the Lie algebra and y in the dual Lie algebra: $\mathrm{ad}_x^\top\, y = \mathrm{P}_y\, x$. For elements of dual spatial algebra $y = [f, m]$ we have $\mathrm{P}_y = \begin{pmatrix} 0 & f_\times \\ f_\times & m_\times \end{pmatrix}$. And finally, we obtain the second term

$$\boxed{\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta} = \left[\left(J_c^\top \,\mathrm{P}_{E\boldsymbol{\lambda}^*} \frac{\partial \mathrm{CD}}{\partial^0 M_2} - J_2^\top \,\mathrm{P}_{{}^c X_2^\top E\boldsymbol{\lambda}^*}\right) J_2 - \left(J_c^\top \,\mathrm{P}_{E\boldsymbol{\lambda}^*} \frac{\partial \mathrm{CD}}{\partial^0 M_1} - J_1^\top \,\mathrm{P}_{{}^c X_1^\top E\boldsymbol{\lambda}^*}\right) J_1\right] \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\theta}.}$$

Details on implementation. Here, the terms are calculated for one contact. For multiple contacts, $\frac{\partial J_c \boldsymbol{v}^+}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}$ is the concatenation of the terms for individual contacts and $\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}$ is the sum of the terms from each contacts. Note also that similarly to the kinematic Jacobians, the two terms can be computed efficiently by exploiting the sparsity induced by the kinematic structure.

Appendix D - Complete simulation gradients expression

The complete expression of the simulation step derivative is

$$\boxed{\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta} = \left.\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{\lambda=\lambda^*} - \Delta t M^{-1} J_c^\top C A^{-1} B \left(J_c \left.\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{\lambda=\lambda^*} + \left.\frac{\mathrm{d}J_c \boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{v=v^+} + \frac{\partial J_c \boldsymbol{v}^+}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}\right),}$$

with

$$\left.\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{\lambda=\lambda^*} = \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}\theta} + \Delta t \left(\frac{\partial \mathrm{ABA}}{\partial \boldsymbol{q}} \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\theta} + \frac{\partial \mathrm{ABA}}{\partial \boldsymbol{v}} \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}\theta} + \frac{\partial \mathrm{ABA}}{\partial \boldsymbol{\tau}} \frac{\mathrm{d}\boldsymbol{\tau}}{\mathrm{d}\theta} + M^{-1} \frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}\right), \tag{37}$$

$$\left.\frac{\mathrm{d}J_c \boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{v=v^+} = \frac{\partial \mathrm{FKV}(\boldsymbol{q}, \boldsymbol{v}^+, c)}{\partial \boldsymbol{q}} \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\theta}, \tag{38}$$

where $\mathrm{FKV}(\boldsymbol{q}, \boldsymbol{v}^+, c)$ is the forward kinematic velocity that gives the velocity of the origin of the frame c when the system is in configuration $\boldsymbol{q}$ with joint velocity $\boldsymbol{v}^+$. A, B, C are as presented in Appendix B and $\frac{\partial J_c \boldsymbol{v}^+}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}$, $\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}$ are as presented in Appendix C.

Details on implementation. The partial derivatives $\frac{\partial \mathrm{ABA}}{\partial \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\tau}}$, $\frac{\partial \mathrm{FKV}}{\partial \boldsymbol{q}}$ and the term $M^{-1}(\boldsymbol{q}) J_c^\top(\boldsymbol{q})$ can be efficiently computed via rigid-body algorithm as implemented in Pinocchio.

The terms $\frac{\partial J_c \boldsymbol{v}^+}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}$, $\frac{\partial J_c^\top \boldsymbol{\lambda}^*}{\partial c} \frac{\mathrm{d}c}{\mathrm{d}\theta}$, A, B and C can be computed efficiently jointly with the ABA derivatives during forward and backward search of the kinematic tree to exploit its sparsity.

Baumgarte stabilization is often used in practice to prevent penetration errors from growing. The correction is integrated by adding terms to the expression of g

$$g = J_c v_f + \frac{\Phi(q)}{\Delta t} - K_p \left[\frac{\Phi(q)}{\Delta t}\right]_- - K_d J_c v \tag{39}$$

where K_p and K_d are the gains of the corrector. In the case of sticking or sliding contacts we have $(G\boldsymbol{\lambda} + g)_N = 0$ and expanding the expression of g yields

$$(J_c v^+)_N = K_d (J_c v)_N - (1 - K_p) \frac{\Phi(q)}{\Delta t}. \tag{40}$$

Therefore, using a Baumgarte correction affects the derivative of the simulation. In particular, the derivatives of the proportional term involve Φ and thus should be handled when computing the

derivatives of the collision detection. Differentiating the derivative term $K_d J_c v$ is done similarly to the $J_c v^+$ term i.e. via the Forward Kinematics derivatives. In more details, the term in parentheses of (37) becomes

$$\frac{\mathrm{d}G}{\mathrm{d}\theta}+\frac{\mathrm{d}g}{\mathrm{d}\theta} = J_c \left.\frac{\mathrm{d}\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{\lambda=\lambda^*} + \left.\frac{\mathrm{d}J_c\boldsymbol{v}^+}{\mathrm{d}\theta}\right|_{v=v^+} + \frac{\partial J_c\boldsymbol{v}^+}{\partial c}\frac{\mathrm{d}c}{\mathrm{d}\theta}+\frac{(1-K_p)}{\Delta t}\frac{\mathrm{d}\Phi(q)}{\mathrm{d}\theta}-K_d J_c\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}\theta}-K_d\left.\frac{\mathrm{d}J_c\boldsymbol{v}}{\mathrm{d}\theta}\right|_{v=v} \tag{41}$$

Appendix E - Additional experimental support

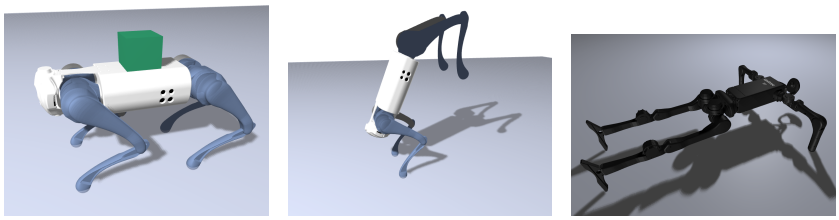

Figure 5: The differentiable simulator is used to find a control torque stabilizing various robotics systems: a Unitree Go1 in a standing position with a 10kg mass on its back (**Left**) or in a "hand-stand" pose (**Center**) and a humanoid Unitree H1 in a "push-up" pose (**Right**).

This subsection provides several additional experiments using our differentiable simulator to solve contact inverse dynamics problems on underactuated robotics systems.

The considered problems, depicted in Fig. 5, are the following:

- A Unitree Go1 is stabilized in a standing position with a 10kg mass put on its back;
- A Unitree Go1 is stabilized in a "hand-stand" pose;
- A Unitree H1 humanoid is stabilized in a "push-up" pose.

In every case, the robots are stabilized by optimizing the torque on the actuators.

We refer to the video attached to this paper for more visualization of the experiments.