Contact Models in Robotics: a Comparative Analysis

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

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]

***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), 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 relies 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] 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 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 the computational and, thus, the energetic

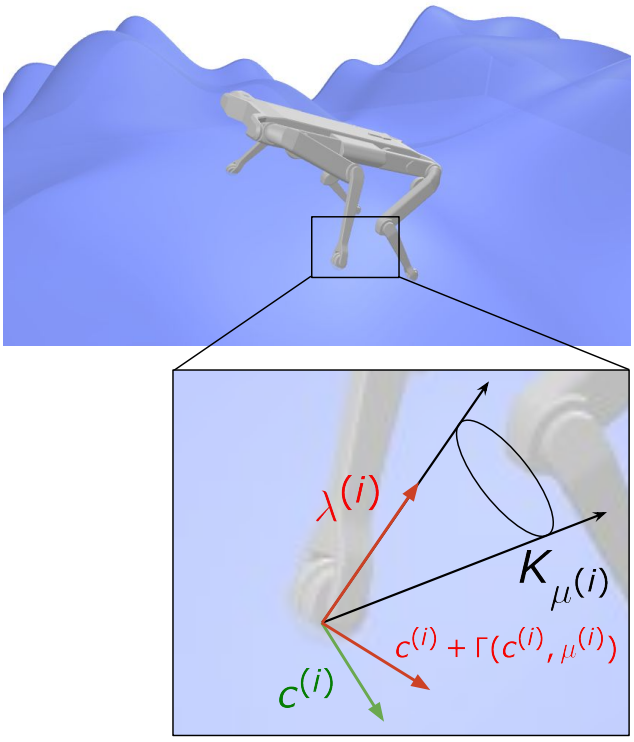

Fig. 1. **Illustration of the dynamics of frictional contacts** between rigid bodies, which are governed by the Signorini condition, Coulomb's law, and the maximum dissipation principle. Combining these three principles leads to the Non-linear Complementarity Problem (15).

[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

cost of learning a control policy. Physical accuracy plays an important role after training as well, as more physically accurate simulations will result in a smaller reality gap to cross for the learned policy to transfer to a real robot [14].

Many manipulation tasks can be tackled by assuming quasi-staticity and considering only a restricted variety of contact events [17], [18]. The recent robotics efforts, highlighted, for instance, by the athletic motions performed by the humanoid robots of Boston Dynamics [19], focus on very dynamic tasks for which these simplification hypotheses cannot hold. In fact, tasks like agile locomotion or dexterous manipulation require the robot to quickly plan and finely exploit, at best, the contact interactions with its environment to shape the movements [20], [21], [22]. In this respect, the ability to handle impacts and friction, physical phenomena at the core of contact interactions, becomes fundamental for robotic simulators.

Physics simulation is often considered a solved problem with several well-known simulators that are available off the shelf. However, simulating a physical system raises several complex issues that are usually circumvented at the cost of approximations or costly computation. When simulating a system evolving freely, rigid body dynamics algorithms [23], [24] are now established as the way to go due to their high efficiency. For robotics, one has to consider interactions through contact between the robot and its environment, thus constraining the movement. However, due to the creation of the breaking of contacts along a trajectory, the dynamics switch from one mode to the other, making the problem of simulating a system with contacts and friction highly non-smooth [25], [26], [27]. Numerical integration schemes for non-smooth systems fall into two main categories: event-driven and time-stepping methods [28]. Most modern robotics simulators are part of the latter category because predicting collisions is intractable due to the complexity of the scenes. Therefore, we will restrict our study to this type of method.

More precisely, contact dynamics between rigid objects are governed by three main principles: the Signorini condition specifies the unilaterality nature of contact interactions, while Coulomb's law of friction and the maximum dissipation principle (MDP) of Moreau state that friction force should lie inside a second-order cone and oppose the movement. Altogether, these three principles correspond to a so-called nonlinear complementarity problem (NCP). The complementarity constraints define a non-convex set while being non-smooth, this problem is difficult to solve in general [28].

Historically, the Open Dynamic Engine (ODE) [29] is one of the first open-source simulators with a large impact on the community, which was then followed by Bullet [30]. Both of them, in their original version, relied on maximal coordinates to depict the state of the objects, and kinematic constraints imposed by the articulations are tackled explicitly. Such a choice leads to large-dimensional problems to solve, impacting *de facto* the computational performances. To lower the computational burden, alternative simulators rooted in generalized coordinates, like DART [31] and MuJoCo [32], appeared shortly after. Since then, Bullet also made this choice the default one. In practice, these simulators are rarely used to tackle engineering problems but rather as physics engines for graphical purposes (Bullet) or research in the RL community (MuJoCo). More recently, RaiSim [33] and Drake [34] were developed as robotic-driven software. RaiSim [33] emerged as one of the first simulators enabling RL policies to transfer to real quadrupedal robots. Its implementation being closed source, we provide what constitutes, to the best of our knowledge, the first in-depth study and open-source re-implementation of this contact solver. Drake also demonstrated some promising results on challenging manipulation [35] as regards the sim-to-real requirements. Still today, the number of alternative algorithms available is growing fast, in an effort to improve the properties of the existing ones, in terms of accuracy and robustness [36], [37], [38], [6], [39], [35], [8]. In a parallel line of work, Isaac Gym [40] and Brax [41] simulators use elementary contact models and rather focus on exploiting the parallelization abilities from GPUs or TPUs for batch computation.

In general, these simulators differ at their very core: one should be aware of the contact modeling embedded in the simulator they are using and how it can impact the applications they aim at. Some high-level benchmarks of simulators exist [42], evaluating the whole simulation pipeline and its multiple internal routines, *e.g.*, rigid-body dynamics algorithms, collision detection, and contact problem-solving. Our work closely relates to [43]. It separately assesses the various contact models and their associated algorithms. We achieve this by decoupling the contact models from their implementations and re-implemented the solvers with a unique back-end based on the Pinocchio toolbox [24], [44] for evaluating the dynamic quantities and on HPP-FCL [45], [46], [47] for computing the collisions. We pursue the effort of [43] by studying recent algorithms and adding advanced evaluation criteria. Our experiments are done in both illustrative and realistic robotics setups.

We make the following contributions:

- ↪ we make a detailed survey of contact models and their associated algorithms, including established and more recent robotics simulators;
- ↪ we expose the main limitations of existing simulators by inspecting both the physical approximations and the numerical methods that are at work;
- ↪ we develop an open source and generic implementation of the main robotic contact solvers in C++;
- ↪ based on our implementation and the theoretical study, we propose quantitative criteria that allow performing an in-depth evaluation of both physical and computational aspects of contact models and solvers.
- ↪ we explore the impacts of the simulation choices on the practical application of MPC for quadruped locomotion.

The article is organized as follows: we first recall the background of contact simulation: the physical principles behind contact modeling (Sec. II) and the numerical algorithms allowing us to solve the resulting equations (Sec. III). In the experimental part (Sec. IV), we propose an exhaustive empirical evaluation of the various existing contact models and solvers to assess both their physicality (Sec. IV-A), self-consistency (Sec. IV-B) and computational efficiency (Sec. IV-C). At last,

Sec. IV-D investigates the consequences of the contact models in the context of quadruped locomotion. It is finally worth mentioning that the authors are linked to the Pinocchio and HPP-FCL open-source projects.

II. Rigid contact modelling

We start by stating the physical principles commonly admitted for rigid body simulation with point contact. If these principles remain hypothetical and can still be discussed, they have been, in general, empirically tested and are arguably better than their relaxations. Once the modeling is done, we transcribe these physical laws into a numerical problem, which should be solved via optimization-based techniques to simulate a system with contacts and frictions. We also present the various open-source tools that allow computing all the intermediate quantities necessary to build a physics simulator.

In this paper, we describe the state of a system with its generalized coordinates $q \in \mathcal{Q} \cong \mathbb{R}^{n_q}$. We denote by $v \in \mathcal{T}_q\mathcal{Q} = \mathbb{R}^{n_v}$ the joint velocity, where $\mathcal{T}_q\mathcal{Q}$ is the tangent space of $\mathcal{Q}$.

Free motion. The principle of least constraint [48], [49], [50] induces the celebrated equations of motion:

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

where $M \in R^{n_v \times n_v}$ represents the joint space inertia matrix of the system, $b(q, v)$ accounts for the centrifugal and Coriolis effects, and for the generalized gravity. This Lagrangian equation of motion naturally accounts for the kinematic constraints induced by the articulations of the rigid-body dynamical system. When applied to a robot, *i.e.*, a system of multiple rigid bodies, the inertia matrix M becomes sparse. Rigid body dynamic algorithms exploit this sparsity at best [23], [24] making it possible to compute the free acceleration in a few microseconds on modern CPUs for robots as complex as a 36-dof humanoid. As done by time-stepping approaches [28], we will express the problem in terms of velocities rather than accelerations, thus discretizing (1) into:

$$M(q^t)v^{t+1} = M(q^t)v^t + (\tau^t - b(q^t, v^t))\Delta t \tag{2}$$

which corresponds to a semi-implicit Euler integration scheme [51]. More advanced implicit integrators [32], [35], [8] come with stability guarantees even in the presence of stiff forces. However, as time-stepping schemes, their order of integration is inherently degraded due to the non-smoothness of the dynamics [52]. For this reason, we restrict our study to a simple scheme, as integrators are not the main focus of this work. In the following, we often drop the instant at which quantities are evaluated for readability purposes. We denote the free velocity v^f, which is defined as the solution of (2).

Bilateral contact. When the system is subject to constraints, *e.g.* kinematic loop closures or anchor points, it is convenient to represent them implicitly:

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

where $\Phi : \mathbb{R}^{n_q} \mapsto \mathbb{R}^m$ is an holonomic constraint function of dimension m, which depends on the nature of the constraint.

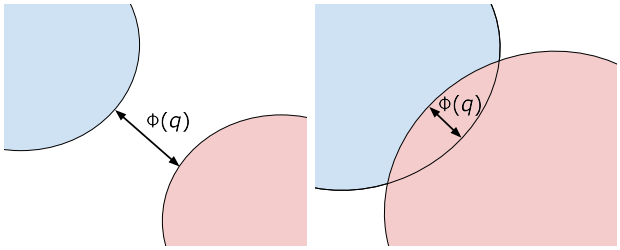

Fig. 2. **The separation vector** Φ allows formulating the non-penetration constraint, which leads to the *Signorini condition* (8). This vector is computed by the GJK or EPA algorithms, which are internal blocks of the simulator. We refer to [56] for a tutorial introduction on the topic.

For solving, it is more practical to proceed to an index reduction [53] by differentiating (3) w.r.t. time, in order to express it as a constraint on joint velocities:

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

where $c = J(q^t)v^{t+1} \in \mathbb{R}^m$ is the constraint velocity, $J = {}^{\partial\Phi}/_{\partial q}$ is the constraint Jacobian explicitly formed at time t, which can be computed efficiently via rigid body dynamics algorithms [23], [54]; and c^* is the reference velocity which stabilizes the constraint. Such a constraint (4) is enforced by the action of the environment on the system via the contact vector impulse $\lambda \in \mathbb{R}^m$. These considerations lead to Gauss's principle of least constraint [55], [48]. By duality, the contact impulses are spanned by the transpose of the constraint Jacobian and should be incorporated in the Lagrangian equations (2) via:

$$Mv^{t+1} = Mv_f + J^\top\lambda \tag{5}$$

Regarding bilateral contacts, the contact efforts, corresponding to the Lagrange multipliers associated with the constraint (3), are unconstrained. If a bilateral constraint is well suited to model kinematic closures, it is not to model interactions between the robot and its environment, which are better represented by unilateral contacts. This paper focuses on the latter, for which we provide a more detailed presentation.

Unilateral contact. When a system is in contact with its environment, the non-penetration constraint enforces the signed distance between the two objects to be non-negative [57]. Defining the separation vector as the vector of minimum norm separating two shapes in contact [58], [56](Fig. 2), the signed distance function corresponds to its normal component. By overloading the notation of the bilateral case, the constraint function Φ now maps to the separation vector (Fig. 2) and describes a unilateral constraint:

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

where $\Phi(q) \in \mathbb{R}^{3n_c}$, n_c is the number of contacts; the subscripts N and T respectively account for the normal and tangential components. In practice, Φ can be computed efficiently via the Gilbert-Johnson-Keerthi (GJK) [59], [47] and the Expanding Polytope Algorithm (EPA) algorithms[60]. GJK operates on convex shapes, but non-convex shapes can also be handled by proceeding to decomposition into convex sub-shapes [61] during an offline preprocessing step. To ease the solving, one

can write (6) in terms of velocities, and supposing that shapes are in contact, *i.e.* $\Phi(q^t)_N \leq 0$, the Taylor expansion of the condition (6) leads to:

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

where $c = J(q^t)v^{t+1} \in \mathbb{R}^{3n_c}$ is the velocity of contact points. It should be noted that J is evaluated at q^t as it avoids computing Φ and its Jacobian several times when solving for q^{t+1} and v^{t+1} which significantly decreases the computational burden. We explain later how c_N^* is set to model physical effects or improve the numerical accuracy of the solutions. As in the bilateral case, the transpose of the contact Jacobian J spans the contact forces, which leads again to (5) the constrained equations of motion. Unlike the bilateral case, unilateral contacts constrain the possible contact impulses λ. In a frictionless situation, the tangential forces are null, which implies that $\lambda_T = 0$. In addition, the contact forces λ can only be repulsive *i.e.*, they should not act in a glue-like fashion (the environment can only push and not pull on the feet of a legged robot) and, thus, are forced to be non-negative. An impulse cannot occur when an object takes off, *i.e.*, the normal velocity and impulse cannot be non-null simultaneously. Combining these conditions, we obtain the so called *Signorini condition* [62] at the velocity level [25]:

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

where $a \perp b$ for vectors a and b means $a^\top b = 0$. However, such a condition does not define a mapping between λ_N and c_N, *i.e.*, the contact forces are not a function of the penetration error. Indeed, their representation is an infinitely steep graph that may be relaxed into a mapping via a spring damper accounting for local deformation of the materials (see Fig. 3). Substituting, v^{t+1} by its expression from the Lagrangian equations (5), we obtain a Linear Complementarity Problem (LCP) [63]:

$$0 \leq \lambda_N \perp (G\lambda + g)_N - c_N^* \geq 0 \tag{9}$$

where $G = JM^{-1}J^\top$ is the so-called Delassus matrix, and $g = Jv^f$ is the free velocity of contact points (the velocity of the contact points in the unconstrained cases). It is worth mentioning at this stage that several approaches [23], [64], [54] have been developed in the computational dynamics and robotics literature to efficiently evaluate the Delassus matrix.

In the case of rigid bodies — purely inelastic impacts, and exact collision detection, *i.e.* $\Phi(q)_N = 0$ — the reference velocity c_N^* can be set to 0. However, setting this velocity to a non-null quantity may be useful to improve modeling on both physical and numerical aspects. A first benefit is the possibility of accounting for impacts that may occur when two objects collide with non-null relative normal velocity. The most common impact law stipulates $c^* = -ec^t$ where e is the restitution coefficient, which adjusts the quantity of energy dissipated during the collision. When time-stepping methods are employed, one cannot avoid penetration errors, *i.e.* $\Phi(q)_N < 0$, without using stabilization by reprojection techniques [65] which are computationally expensive to use in robotics due to the cost of detecting a collision. However, it is still possible to prevent these errors from dramatically growing over time via a Baumgarte correction [66] which sets $c^* = k_B \max(0, -\Phi(q^t)_N)$ where the Baumgarte coefficient k_B is set to be proportional to $\frac{1}{\Delta t}$.

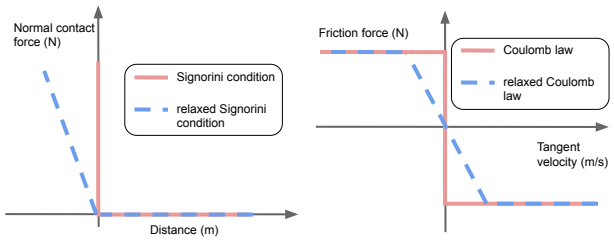

Fig. 3. Both the *Signorini condition* (**Left**) and Coulomb's law (**Right**) induce infinitely steep graphs, which make the contact problem hard to solve.

In addition, in many cases in robotics, Delassus' matrix G is rank deficient. Such physical systems are said to be hyperstatic, and because $\mathrm{rank}(J) > n_v$, several λ values may lead to the same trajectory. This under-determination can be circumvented by relaxing the rigid-body hypothesis, *e.g.* the *Signorini condition*, and considering compliant contacts via a reference velocity linearly depending on λ as represented in Fig. 3. Indeed, with $c^* = -R\lambda$ where R is a diagonal matrix with non-null and positive elements only on the normal components, called compliance and whose value is a property of the material, the original Delassus matrix G is replaced by the damped matrix $\tilde{G} = G + R$ which is full rank. At this stage, one should note that the physical compliance acts on the conditioning in an equivalent way to a numerical regularization.

Friction phenomena are at the core of contact modeling, as they precisely enable manipulation or locomotion tasks. Coulomb's law for dry friction represents the most common way to model friction forces. This phenomenological law states that the maximum friction forces $\|\lambda_T\|$ should be proportional to the normal contact forces λ_N and the friction coefficient μ. Mathematically, this suggests that contact forces should lie inside an ice cream cone whose aperture is set by the coefficient of friction μ:

$$\lambda \in K_\mu = \prod_{i=1}^{n_c} K_{\mu^{(i)}} \tag{10}$$

where the product is Cartesian, the superscript (i) refers to the i^{th} contact point and $K_{\mu^{(i)}} = \left\{\lambda | \lambda \in \mathbb{R}^3,\ \lambda_N \geq 0, \|\lambda_T\|_2 \leq \mu^{(i)}\lambda_N\right\}$. Additionally, when sliding occurs, the maximum dissipation principle formulated by Jean-Jacques Moreau [25] implies that the frictional forces should maximize the dissipated power:

$$\forall i,\ \lambda_T^{(i)} = \underset{\|\gamma_T\| \leq \mu^{(i)}\lambda_N^{(i)}}{\arg\min}\ \gamma_T^\top c_T^{(i)} \tag{11}$$

whose optimality conditions yield the following equation in the sliding case:

$$\forall i,\ \lambda_T^{(i)} = -\mu^{(i)}\,\lambda_N^{(i)}\,\frac{c_T^{(i)}}{\|c_T^{(i)}\|}, \text{if } \|c_T^{(i)}\| > 0. \tag{12}$$

As for the *Signorini condition*, Coulomb's law does not describe a mapping but an infinitely steep graph (Fig. 3). Relaxing this law via viscous frictions, *i.e.*, assuming the tangent contact

forces to be proportional to the tangent velocities, allows defining a mapping between λ_T and c_T.

Combining the Coulomb's law for friction with the *Signorini condition* evoked earlier, we finally get three distinct cases corresponding to a sticking contact point (13a), a sliding contact point (13c) or a take-off (13b):

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 & \text{(13a)} \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 & \text{(13b)} \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} & \text{(13c)} \end{cases}$$

where ∂K indicates the boundary of the cone. The equations (13) are referred to as the disjunctive formulation of the contact problem. However, such a formulation is unsuitable in practice for solving, as the switching condition depends on the contact point velocity c. As this quantity is an unknown of the problem, one cannot know in which case of (13) one is standing. For this reason, the problem is often reformulated as a nonlinear complementarity problem (NCP). Indeed, using de Saxcé's bipotential function [67] defined as:

$$\Gamma : (c, \mu) \in \mathbb{R}^3 \times \mathbb{R} \mapsto [0, 0, \mu \|c_T\|_2] \tag{14}$$

one can show that (13) is equivalent to the following [27], [68] (Fig. 1):

$$\forall i, K_{\mu^{(i)}} \ni \lambda^{(i)} \perp c^{(i)} + \Gamma\left(c^{(i)}, \mu^{(i)}\right) \in K^*_{\mu^{(i)}}. \tag{15}$$

In (15), K^*_μ refers to the dual cone of K_μ, such that if $\lambda \in K_\mu$ and $c \in K^*_\mu$, then $\langle \lambda, c \rangle \geq 0$, where $\langle \cdot, \cdot \rangle$ is the canonical scalar product. It is worth noting that the relation $K^*_\mu = K_{1/\mu}$ stands for second-order cones.

Eq. (15) allows defining, for each contact $i \in [\![1, n_c]\!]$, the primal and dual residuals as $\epsilon_{\text{p}}^{(i)} = \text{dist}_{K_{\mu^{(i)}}}\left(\lambda^{(i)}\right)$ and $\epsilon_{\text{d}}^{(i)} = \text{dist}_{K^*_{\mu^{(i)}}}\left(c^{(i)} + \Gamma\left(c^{(i)}, \mu^{(i)}\right)\right)$ respectively, where $\text{dist}_{\mathcal{C}}$ is the distance function w.r.t. a convex set $\mathcal{C}$. It also induces a contact complementarity criterion $\epsilon_{\text{c}}^{(i)} = \left|\langle \lambda^{(i)}, c^{(i)} + \Gamma\left(c^{(i)}, \mu^{(i)}\right)\rangle\right|$. From these per-contact criteria, it is then possible to introduce a well-posed absolute convergence criterion ϵ_{abs} for (15), as the maximum of $\epsilon_{\text{p}}^{(i)}$,$\epsilon_{\text{d}}^{(i)}$ and $\epsilon_{\text{c}}^{(i)}$ for all i. We use this criterion as a stopping criterion in our implementation of NCP solvers, but also as a measure of physical accuracy in our experiments of Section IV. All the previous derivations were made with λ being an impulse which causes it, and thus the criteria $\epsilon_{\text{p,c}}$, to be proportional to the time step Δt. However, it is preferable from the user-side to have ϵ_{c} and ϵ_{p} not correlated to Δt so the precision threshold of the simulation ϵ_{abs} can be set independently of the time-step. In practice, before solving we operate a change of variable to directly work on the equivalent contact forces $\frac{\lambda}{\Delta t}$. This is done by replacing g and c^* by their scaled counterpart $\frac{g}{\Delta t}$ and $\frac{c^*}{\Delta t}$ in the formulation of (15). For readability purposes, equations are still written in impulse in what follows.

At this point, it is worth mentioning that the problem (15), which we refer to as NCP, does not derive from a convex optimization problem, thus making its solving complex. Alternatively, one can see the frictional contact problem as two interleaved convex optimization problems [69], [70], [36], [71], [6] whose unknowns, λ and v, appear in both. Other formulations exist and we refer to [68] for a more complete review on the NCP. Practically, the non-convexity can induce the existence of multiple, or even an infinite number of contact forces satisfying (15). As mentioned earlier, this can be due to normal forces, but tangential components can also cause under-determination (Fig. 4). In this situation, it would be preferable for a simulator to provide the minimum norm solution in forces. This property can prevent a simulator from exhibiting internal friction forces compressing or stretching the objects (Fig. 4, right). Indeed, such forces would not coincide with the forces observed by force sensors and would rather correspond to some internal deformations of the objects, which should thus be considered soft and no more rigid. In the following, we will use the term "internal forces" to denote the force component deviating from the minimum norm solution. Additionally, these internal forces might also be problematic as it is impossible to characterize them. This may induce inconsistent derivatives, which become critical in the context of differentiable simulation.

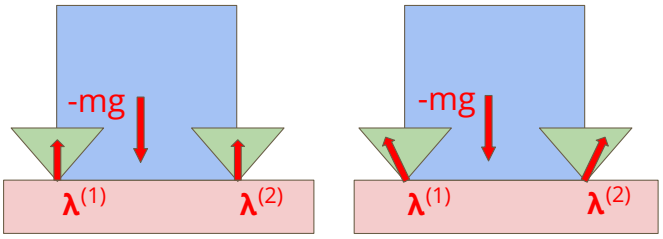

Fig. 4. **Underdetermined contact problem.** The left and right contact forces are solutions of the NCP (15) and lead to the same system velocity. Such an undetermined problem can also occur on normal forces.

Open-source frameworks for contact simulation. To conclude this section, we propose to review the open-source software that is popular in the robotics community and that can be used for simulating contact. Simulating contact interactions, as illustrated in Fig. 5, involves two main stages, corresponding to the collision detection step (which objects are in contact) and the collision resolution (which contact forces are applied through the contact interaction). These frameworks are enumerated in Tab. I.

More precisely, at each time step, a simulator must first detect which geometries are colliding and compute their separation vector Φ. The GJK and EPA algorithms are widely adopted for their low computational cost. HPP-FCL [45], [46], [47], an extension of the Flexible Collision Library (FCL) [72] and libccd [73] implement them efficiently. Some simulators such as Bullet [30] or ODE [29] also re-implement the same algorithm as an internal routine.

Once collisions are evaluated, one still requires the contact points free velocity v_f and Jacobians J to formulate (15). These two quantities are efficiently computed via rigid body algorithms [23]. The RBDL [74] or the Pinocchio library [24] provide efficient implementations to evaluate them. In addition, Pinocchio proposes a direct and robust way to compute the Cholesky decomposition of the Delassus matrix G [54]. These

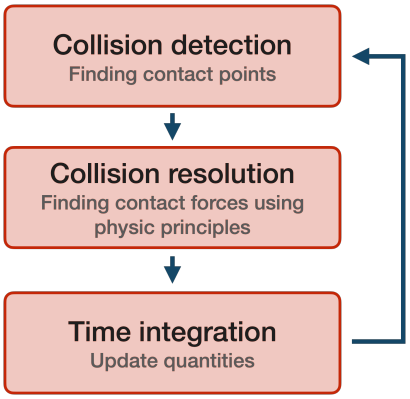

Fig. 5. **Simulation routines.** When simulating rigid bodies with frictional contacts, a physics engine goes through a sequence of potentially challenging sub-problems: collision detection, collision resolution, and integration time step.

TABLE I
OPEN SOURCE TOOLS FOR PHYSICS SIMULATION IN ROBOTICS.

	License	API	Used by
Collision detection			
FCL [72]	BSD	C++	DART, Drake
libccd [73]	BSD	C++, Python	MuJoCo, Drake, FCL, Bullet, ODE
HPP-FCL [45]	BSD	C++, Python	Pinocchio
Bullet [30]	BSD	C++, Python	DART
ODE [29]	BSD/GPL	C++, Python	DART
Rigid body dynamics algorithms			
Pinocchio [24]	BSD	C++, Python	
RBDL [74]	zlib	C++, Python	
Collision resolution			
MuJoCo [32]	Apache 2.0	C++, Python	
DART [31]	BSD 2	C++, Python	
Bullet [30]	BSD	C++, Python	
Drake [34]	BSD 3	C++, Python	
ODE [34]	BSD/GPL	C++, Python	

algorithms are also embedded as internal routines in various simulators such as MuJoCo [32], DART [31], Drake [34], Bullet [30] or ODE [29], but they often are only partially exposed to the user.

Eventually, when all quantities necessary to formulate the NCP (15) are computed, the simulator has to call a solver. Every simulator, *i.e.* MuJoCo [32], DART [31], Bullet [30], Drake [34] and ODE [29], proposes its own implementation. This procedure varies greatly depending on the physics engine, as each has its own physical and numerical choices. In the next section, we detail the existing algorithms.

TABLE II
CHARACTERISTICS OF VARIOUS CONTACT MODELS.

	Signorini	Coulomb	MDP
LCP	✓		
CCP		✓	✓
RaiSim [33]	✓	✓	
NCP	✓	✓	✓

III. ALGORITHMIC VARIATIONS OF THE CONTACT PROBLEM

As explained in the previous section, the nonlinear complementarity problem (15) does not derive from a variational principle but can be formulated as variational inequalities [68]. Thus, classical numerical optimization solvers cannot be used straightforwardly to solve it. This section studies the various approximations and algorithmic techniques in the literature to tackle this problem. As summarized in Tab. III, this section is organized into subsections describing the four contact models most commonly used in robotics, namely the linear complementarity problem (LCP), the cone complementary problem (CCP), RaiSim, and the nonlinear complementarity problem (NCP). For each contact model, we also report the related algorithmic variants. If each tick in Tab. III represents a positive point for the concerned algorithm, Sec. IV shows that even one missing tick may be prohibitive and can cause a solver to be unusable in practice. Finally, we also mention a set of useful implementation tricks that can be used to build an efficient simulator.

A. Linear Complementarity Problem

A first way to simplify the solving of problem (15) is to linearize the NCP problem by approximating the second-order cone constraint from Coulomb's law with a pyramid, typically composed of four facets. This is done by replacing $K_{\mu^{(i)}}$ by $\tilde{K}_{\mu^{(i)}} = \left\{\lambda | \lambda_N \geq 0, \|\lambda_T\|_\infty \leq \mu^{(i)} \lambda_N\right\}$. Doing so allows retrieving a linear complementarity problem (LCP), often easier to solve [63]. Such a problem is more standard and better-studied than its nonlinear counterpart as it already has a long history of applications to frictional contacts [77], [78], [79], [80].

Direct methods for LCP date back to the 1960s and are available options in well-known simulators such as ODE [29] and Bullet [30] which implement respectively the Lemke's [81] and Dantzig's [82] algorithms. Under specific circumstances [83], the algorithm is guaranteed to find a solution.

Projected Gauss-Seidel. Due to its easy implementation and its early stopping property, the projected Gauss-Seidel (PGS) algorithm (Alg. 1) algorithm represents an attractive alternative and was widely adopted as the default solver by many physics engines, such as in Bullet [30], PhysX [75], ODE [29], and DART [31], [7] simulators. This iterative algorithm loops on contact points and successively updates the normal and tangent contact forces. Because PGS works separately on each contact point, the update compensates for the current errors due to the estimated forces from other contact points. Yet, as illustrated in the experimental section IV, this process induces the emergence of internal forces during the solving. Moreover, Gauss-Seidel-based approaches are similar to what is also known as block coordinate descent in the optimization literature. As first-order algorithms, they do not benefit from improved convergence rates or robustness with respect to their conditioning, unlike second-order algorithms. In parallel, the linearization of the second-order cone causes the loss of the isotropy for friction, as stated by Coulomb's law. By choosing the axes for the facets

TABLE III
CHARACTERISTICS OF NUMERICAL ALGORITHMS.

	Hard contacts	No internal forces	Robust	Convergence guarantees
LCP				
PGS [30], [29], [75], [31]	✓			
Staggered projections [36]	✓	✓	✓	
CCP				
PGS [76]	✓			✓
MuJoCo [32]		✓	✓	✓
ADMM (Alg. 3)	✓	✓	✓	✓
Drake [35]		✓	✓	✓
RaiSim [33]	✓			
NCP				
PGS	✓			
Staggered projections [6]	✓	✓	✓	

Algorithm 1: Pseudocode of the projected Gauss-Seidel (PGS) algorithm for solving LCPs.

Input: Delassus matrix: G, free velocity: g, friction cones: K_μ
Output: Contact forces: λ

for $k = 1$ **to** n_{iter} **do**
  for $i = 1$ **to** n_c **do**
    $\lambda_N^{(i)} \leftarrow \lambda_N^{(i)} - \frac{1}{G_{NN}^{(ii)}} (G\lambda + g)_N^{(i)}$;
    $\lambda_N^{(i)} \leftarrow \max(0, \lambda_N^{(i)})$;
    $\lambda_T^{(i)} \leftarrow \lambda_T^{(i)} - \frac{1}{\min(G_{T_xT_x}^{(ii)}, G_{T_yT_y}^{(ii)})} (G\lambda + g)_T^{(i)}$;
    $\lambda_T^{(i)} \leftarrow \text{clamp}(\lambda_T^{(i)}, \mu_i \lambda_N)$;
  end
end

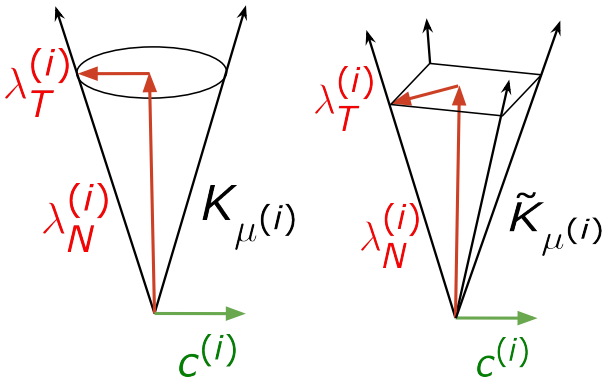

Fig. 6. **Cone linear approximation.** Linearizing the friction cone induces a bias in the direction of friction forces. The MDP tends to push tangential forces toward the corners of the pyramid.

of the pyramid and due to the maximum dissipation principle, it is established that one incidentally biases the friction forces towards the corners [84], [85], as illustrated in Fig. 6. This error is sometimes mitigated by increasing the number of facets, which also comes at the cost of more computations.

B. Cone Complementarity Problem.

An alternative approach consists of approximating the NCP problem in order to transform it into a more classical convex optimization problem. By relaxing the complementarity constraint from (15), one can obtain a Cone Complementarity Problem (CCP) [86]:

$$K_\mu \ni \lambda \perp c \in K_\mu^* \tag{16}$$

If this relaxation preserves the Maximum Dissipation Principle (MDP) and the second-order friction cone, it loses the *Signorini condition* (8). Indeed, re-writing the complementarity of (16) yields:

$$\forall i,\ \lambda_N^{(i)} c_N^{(i)} + {\lambda_T^{(i)}}^\top c_T^{(i)} = 0, \tag{17}$$

and if the i$^{\text{th}}$ contact point is sliding, the MDP (12) leads to:

$$\lambda_N^{(i)} c_N^{(i)} - \mu^{(i)} \lambda_N^{(i)} \|c_T^{(i)}\|_2 = 0, \tag{18}$$

which is equivalent to the following complementarity condition:

$$\lambda_N^{(i)} \perp (c_N^{(i)} - \mu \|c_T^{(i)}\|_2). \tag{19}$$

(19) indicates that the CCP approximation allows for simultaneous normal velocity and forces, contrary to (8). In practice, this results in objects interacting at distance when contact points are sliding. In its seminal work [86], Anitescu shows the interaction distance to be $\Delta t \mu \|c_T^{(i)}\|$. It is worth insisting on the fact that such an artifact only emerges in the case of a sliding contact and can be mitigated, and even controlled, with smaller time steps and sliding velocities. Moreover, it is still under debate to determine if this behavior is prohibitive for robotics applications.

Because CCP (16) approximates the NCP (15), the convergence is checked via a different criterion. In fact, in the same way, the De Saxcé correction was ignored in (16), a convergence criterion is obtained by removing this term from the dual convergence criterion ϵ_{d} of the NCP introduced previously.

PGS. The PGS algorithm can be directly adapted to handle the CCP problem [76] (Alg. 2). In the light of what follows, the algorithm even becomes equivalent to a projected gradient descent which is a classical constrained optimization technique.

Algorithm 2: Projected Gauss-Seidel (PGS) algorithm for the dual Cone Complementarity Problem (CCP)

Input: Delassus matrix: G, free velocity: g, friction cones: K_μ
Output: Contact forces: λ

for $k = 1$ **to** n_{iter} **do**
  for $i = 1$ **to** n_c **do**
    $\lambda^{(i)} \leftarrow \lambda^{(i)} - \frac{3}{G^{(ii)}_{NN}+G^{(ii)}_{T_xT_x}+G^{(ii)}_{T_yT_y}} (G\lambda + g)^{(i)}$;
    $\lambda^{(i)} \leftarrow \text{proj}_{K_{\mu_i}}(\lambda^{(i)})$;
  end
end

Algorithm 3: ADMM algorithm for the dual Cone Complementarity Problem (CCP)

Input: Delassus matrix: G, free velocity: g, friction cones: K_μ
Output: Contact forces: λ

$\tilde{G}^{-1} \leftarrow (G + \rho \text{Id})^{-1}$;
for $k = 1$ **to** n_{iter} **do**
  $\lambda \leftarrow -\tilde{G}^{-1}(g - \rho z + \gamma)$;
  $z \leftarrow \text{proj}_{K_\mu}(\lambda + \frac{\gamma}{\rho})$;
  $\gamma \leftarrow \gamma + \rho(\lambda - z)$;
end

Optimization on the dual. The problem (16) can, in fact, be viewed as the Karush-Kuhn-Tucker conditions of an equivalent Quadratically Constrained Quadratic Programming (QCQP) problem:

$$\min_{\lambda \in K_\mu} \frac{1}{2}\lambda^\top G\lambda + g^\top \lambda \tag{20}$$

Once the contact problem is formulated as an optimization problem, any optimization algorithms can be employed to solve it and classical optimization theory provides convergence guarantees. Here, we propose to study an ADMM algorithm [87] to solve (20), but Interior Point algorithms [88] could also be used [32]. A benefit of using the family of proximal algorithms like ADMM is their natural ability to handle the numerical issues coming from ill-conditioned and hyper-static cases [89], [90]. This property makes it possible to accurately simulate hard contacts, *i.e.*, without any shift due to compliance $R = 0$, and is reported in Tab. III by the "hard contacts" column. Another by-product of such methods is the implicit regularization they induce on the found solution, which removes the potential internal forces. This last property is an empirical observation resulting from the experimental section IV and, to our knowledge, has not yet been proven by the literature of proximal optimization. Therefore, it remains to be confirmed by subsequent work.

One may argue that such algorithms require to compute G^{-1} (Alg. 3, line 3) while per contact approaches repeatedly solve for each contact point individually, and thus only require the cheap inverse of diagonal blocks from G (Alg. 1, lines 3,5, Alg. 2, line 3, Alg. 5, line 4, Alg. 6, lines 3,5). However, the recent progress [54] demonstrated the Cholesky decomposition of G can be computed efficiently and robustly. We detail this point later when discussing implementation tricks, at the end of this section. Exploiting the knowledge of G^{-1} and not the block components as in the "per-contact" approaches mentioned earlier allows us to capture the coupling between all contact points.

Optimization on the primal. By reverse engineering, it is possible to form an optimization problem on joint velocities v whose dual would be (20). This approach is adopted in both MuJoCo[32] and Drake[35] and results in the following optimization problem:

$$\begin{aligned} \min_{v,y} \ & \frac{1}{2}\left\|v - v^f\right\|^2_M + \frac{1}{2}\|y - c^*\|^2_{R^{-1}} \\ & s.t.\ Jv - y \in K^*_\mu \end{aligned} \tag{21}$$

where $\|x\|_X = \sqrt{x^\top X x}$ with $X \succ 0$. Working on the equations, this problem can be formulated as an unconstrained optimization problem:

$$\min_v \ l_p(v) = \frac{1}{2}\left\|v - v^f\right\|^2_M + \frac{1}{2}\|\mathcal{P}^R_{\mathcal{K}_\mu}(y(Jv))\|^2_R \tag{22}$$

where $\mathcal{P}^R_{\mathcal{K}_\mu}(y) = \arg\min_{\gamma \in \mathcal{K}_\mu} \|\gamma - y\|^2_R$; $y(c) = -R^{-1}(c - c^*)$; and which is viable only when R is non-null. The latter condition makes it impossible to model hard contacts. As evoked earlier, this is equivalent to replacing G by $\tilde{G} = G + R$ in the quadratic part of (20), which is justified by a compliant contact hypothesis. Indeed, R corresponds to a compliance, which should be a material property of the objects involved in the collision. However, MuJoCo arbitrarily sets this to the diagonal of αG, where $\alpha \in [0, 1]$ is close to 0. This choice has no physical justification (at least, without making strong assumptions that are not met in practice), and its only intent is to improve the conditioning of the problem to ease the solving and artificially make the solution unique. Moreover, R has non-null tangential components and thus may also introduce some tangential "compliance" which corresponds to the relaxation of Coulomb's law (Fig. 3). In fact, this should instead be interpreted as a Tikhonov regularization term enforcing the strict convexity of the problem to facilitate the numerics and the existence of both the forward and inverse dynamics computation at the cost of shifting, even more, the solution. Drake's algorithm [35] improves this point by providing a more physical way of setting R.

Both Drake and MuJoCo use a Newton solver to tackle (22) (Alg. 4). Due to the non-linearity of the second term of (22), this approach requires updating the inverse of the Hessian at every iteration (Alg. 4,line 3). As proposed in [35], the use of advanced algebra routines allows to reduce the computational burden of each step. In this work, we provide an implementation of the Newton algorithm with an Armijo backtracking linesearch (using parameters from [35]), which is close to what is done in the MuJoCo and Drake simulators.

Algorithm 4: Newton algorithm for the primal Cone Complementarity Problem (CCP)

Input: Inertia matrix: M, Jacobian of contacts: J, compliance: R, free velocity: v^f, friction cones: K_μ

Output: Joint velocity: v

1 **for** $k = 1$ **to** n_{iter} **do**
2 $\quad \nabla_v l_p \leftarrow M(v - v^f) - J^\top \mathcal{P}^R_{\mathcal{K}_\mu}(y(Jv))$;
3 $\quad H \leftarrow M + J^\top \nabla_v \mathcal{P}^R_{\mathcal{K}_\mu}(y(Jv))J$;
4 $\quad \Delta v \leftarrow -H^{-1}\nabla_v l_p$;
5 $\quad \alpha \leftarrow \arg\min_\beta l_p(v + \beta\Delta v)$;
6 $\quad v \leftarrow v + \alpha\Delta v$;
7 **end**

C. Raisim contact model

A contact model introduced in [91] and implemented in the RaiSim simulator [33] aims at partially correcting the drawbacks from the CCP contact model exploited in MuJoCo [32] and Drake [34]. As explained earlier, the CCP formulation relaxes the Signorini condition for sliding contacts, leading to positive power from normal contact forces. The contact model proposed in [91] fixes this by explicitly enforcing the Signorini condition by constraining $\lambda^{(i)}$ to remain in the null normal velocity hyper-plane $V_N^{(i)} = \{\lambda | G_N^{(ii)}\lambda + \tilde{g}_N^{(i)} = 0\}$ where $\tilde{g}^{(i)} = g^{(i)} + \sum_{j\neq i} G^{(ij)}\lambda^{(j)}$ is the i$^{\text{th}}$ contact point velocity as if it were free. Here, we generalize the use of the subscript and the superscript introduced previously to matrices, where a second superscript (or subscript) corresponds to a slicing operation on the columns *e.g.* $G^{(ij)} \in \mathbb{R}^{3\times 3}$ denotes the sub-block of G whose rows are associated to the i$^{\text{th}}$ contact and columns to the j$^{\text{th}}$ contact. For a sliding contact point, the problem (20) becomes:

$$\min_{\lambda \in K_{\mu^{(i)}} \cap V_N^{(i)}} \frac{1}{2}\lambda^\top G^{(i)}\lambda + \tilde{g}^{(i)\top}\lambda \tag{23}$$

The new problem (23) remains a QCQP and [33] leverages the analytical formula of the ellipse $K_{\mu^{(i)}} \cap V_N^{(i)}$ in polar coordinates to tackle it as a 1D problem via the bisection algorithm [92] (Alg. 5, line 10). We refer to the original publication for a more detailed description of the bisection routine [33].

This approach implies several drawbacks. Indeed, it requires knowing whether a contact point is sliding, which cannot be known in advance as the contact point velocity depends on the contact forces. Thus, some heuristics, based on the disjunctive formulation of the contact problem (13), are introduced to try to guess the type of contact which will occur, *i.e.* take-off (Alg. 5, line 5), sticking (Alg. 5, line 7) or sliding (Alg. 5, line 9). Such heuristics may be wrong, which may cause the algorithms to get stuck and lose convergence guarantees. This effect is strengthened by the caveats of the per-contact loop, which additionally make RaiSim not robust to conditioning and prone to internal forces. Eventually, if adding the constraint $\lambda^{(i)} \in V_N^{(i)}$ allows retrieving the Signorini condition from the CCP model, it also induces the loss of the maximum dissipation

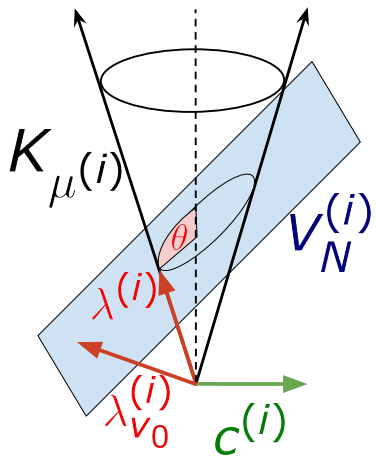

Fig. 7. **Bisection algorithm.** When the contact point is sliding, $\lambda_{v_0}^{(i)}$ (Alg. 5, line 4) lies outside the friction cone $K_{\mu^{(i)}}$, leading to a non-null tangential contact velocity $c^{(i)}$. In this case, RaiSim solves (23). This is equivalent to finding the $\lambda \in K_{\mu^{(i)}} \cap V_N^{(i)}$ which is the closest to $\lambda_{v_0}^{(i)}$ under the metric defined by $G^{(ii)}$. The constraint set being an ellipse, the problem boils down to a 1D problem on θ using polar coordinates. This figure is inspired from Fig. 2 of [33].

principle. Writing the Karush Kuhn Tucker (KKT) conditions of the problem (23) and some algebra manipulations yields:

$$c_T^{(i)} \propto -\lambda_T^{(i)} - \frac{{\mu^{(i)}}^2 \lambda_N^{(i)}}{G_{NN}^{(ii)}} G_{NT}^{(ii)} \tag{24}$$

which contradicts (12).

The problem solved by RaiSim depends on the contact mode, *e.g.* (23) is solved only for a sliding contact and would require the computation of the unknown dual variable. Therefore, it is more complex to define a proper convergence criterion than in previous cases (15) and (16). In this respect, either a fix-point criterion *i.e* the distance between two consecutive iterates, or the previously defined NCP criterion (15) can be used to coarsely monitor convergence. We chose the latest in order to have a criterion homogeneous to the ones used for (15) and (16).

D. Tackling the NCP

Despite the non-smooth and non-convex issues described previously, some simulation algorithms aim to directly solve the original NCP problem [79], [68], [37], [8].

PGS. The PGS algorithm exploited for LCP and CCP problems can easily be adapted to the NCP case by changing the clamping step (Alg. 1,line 6) or the normal projection (Alg. 2, line 4) for a horizontal projection on the cone (Alg. 6, line 6). However, it is worth noting that such approaches have fewer convergence guarantees than their relaxed counterpart [27]. As with every Gauss-Seidel approach, the methods inherited from the sensitivity to ill-conditioning and spurious internal forces.

Staggered-projections. The staggered projections (Alg. 7) approach, appearing in [69], [70] and implemented in a simulator in [36], [6], proceeds by rewriting the NCP as two interleaved optimization problems. This interconnection is

Algorithm 5: Per-contact bisection algorithm

Input: Delassus matrix: G, free velocity: g, friction cones: K_μ
Output: Contact forces: λ, velocity: v

for $k = 1$ **to** n_{iter} **do**
  for $i = 1$ **to** n_c **do**
    $\tilde{g}^{(i)} \leftarrow g^{(i)} + \sum_{j \neq i} G^{(ij)} \lambda^{(j)}$;
    $\lambda_{v_0}^{(i)} \leftarrow -{G^{(ii)}}^{-1} \tilde{g}^{(i)}$;
    if $\tilde{g}_N^{(i)} > 0$ **then**
      `// takeoff`
      $\lambda^* \leftarrow 0$;
    else if $\lambda_{v_0}^{(i)} \in K_{\mu^{(i)}}$ **then**
      `// stiction`
      $\lambda^* \leftarrow \lambda_{v_0}^{(i)}$;
    else
      `// sliding`
      $\lambda^* \leftarrow \text{bisection}(G^{(ii)}, \tilde{g}^{(i)}, K_{\mu^{(i)}}, \lambda_{v_0}^{(i)})$;
    end
    $\lambda^{(i)} \leftarrow (1-\alpha)\lambda^{(i)} + \alpha\lambda^*$;
    $\alpha \leftarrow \gamma\alpha + (1-\gamma)\alpha_{min}$;
  end
end

Algorithm 6: Projected Gauss-Seidel (PGS) algorithm for Non-linear Complementarity Problem (NCP)

Input: Delassus matrix: G, free velocity: g, friction cones: K_μ
Output: Contact forces: λ, velocity: v

for $k = 1$ **to** n_{iter} **do**
  for $i = 1$ **to** n_c **do**
    $\lambda_N^{(i)} \leftarrow \lambda_N^{(i)} - \frac{1}{G_{NN}^{(ii)}} (G\lambda + g)_N^{(i)}$;
    $\lambda_N^{(i)} \leftarrow \max(0, \lambda_N^{(i)})$;
    $\lambda_T^{(i)} \leftarrow \lambda_T^{(i)} - \frac{1}{\min(G_{T_x T_x}^{(ii)}, G_{T_y T_y}^{(ii)})} (G\lambda + g)_T^{(i)}$;
    $\lambda_T^{(i)} \leftarrow proj_{\mu_i \lambda_N^{(i)}}(\lambda_T^{(i)})$;
  end
end

solved via a fix-point algorithm that repeatedly injects one problem's solution into the formulation of the other. The staggered projection algorithm has no convergence guarantees but was heavily tested and seems, in practice, to converge most of the time in a few iterations (typically five iterations [6]). Solving a cascade of optimization problems allows the use of robust optimization algorithms (*e.g.*, ADMM), but remains more costly than other approaches.

E. Implementation details

In practice, the performances of contact solvers can be improved by a few simple tricks.

Warm-starting the solver by providing the contact forces from the previous time step allows to greatly reduce the required computation. Indeed, in the case of a persisting contact between two objects, the contact forces are being cached and reused as an initial guess when solving for the contact forces of the next time step. This relies on the ability of the contact solver algorithm to be warm-started. This excludes Interior Point [88] algorithms, as they would only benefit from an initial guess close to the so-called central path [89]. By contrast, the feasible set of contact forces may change from one time step to the other, even in the case of a persisting contact point. On the opposite, ADMM and, more generally, Augmented Lagrangian (AL) methods can naturally be warm started: not only the primal (*i.e.*, contact forces) and dual (*i.e.*, contact velocities) variables, but also the proximal parameter is initialized with the previous values.

Algorithm 7: Staggered projections algorithm

Input: Delassus matrix: G, free velocity: g, friction cones: K_μ
Output: Contact forces: λ, velocity: v

for $k = 1$ **to** n_{iter} **do**
  $\tilde{g_N} \leftarrow g_T + G_{NT}\lambda_T$;
  $\lambda_N \leftarrow \arg\min_{\lambda \geq 0} \frac{1}{2}\lambda^\top G_N \lambda + \tilde{g_N}^\top \lambda$;
  $\tilde{g_T} \leftarrow g_T + G_{TN}\lambda_N$;
  $\lambda_T \leftarrow \arg\min_{\|\lambda^{(i)}\| \leq \mu_i \lambda_N^{(i)}} \frac{1}{2}\lambda^\top G_T \lambda + \tilde{g_T}^\top \lambda$;
end

Cholesky computation. In addition, second-order algorithms can further exploit the recent progress in rigid body algorithms [54]. This work takes advantage of the sparse structure of the kinematic chains in order to efficiently compute the Cholesky decomposition of the Delassus matrix G. This approach is robust enough to handle the case of hyperstatic systems and reduces the cost of the computation of matrix-vector products involving G^{-1} (Alg. 3, line 3). This also indicates that evaluating G from its Cholesky decomposition, as required by per-contact approaches, actually constitutes an additional cost.

Proximal parameter adaptation. In the context of ADMM (Alg. 3), the algorithm from [54] can also be favorably combined with the adaptation of the proximal parameter. Indeed, updating the regularized Cholesky can be done at almost no cost by using [54]. In our implementation, we follow the work from [93] to detect when and how ρ should be adapted. More precisely, whenever the primal residual is significantly greater than the dual one (a threshold for the ratio has to be set, a typical value being 10), ρ should be increased in order to better enforce the constraint of the problem and thus, reduce the primal residual. The proximal parameter ρ is then updated via a spectral rule which multiplies it by $\kappa^{0.05}$, κ being the condition number of the Delassus matrix, defined as the ratio between the largest and the smallest eigenvalues. Conversely, whenever the dual residual dominates the primal one, ρ should be decreased by dividing it with the same factor. Alternatively, we could use a linear update rule for ρ as it is done in OSQP [94] which would be less efficient in the case of ill-conditioned problems.

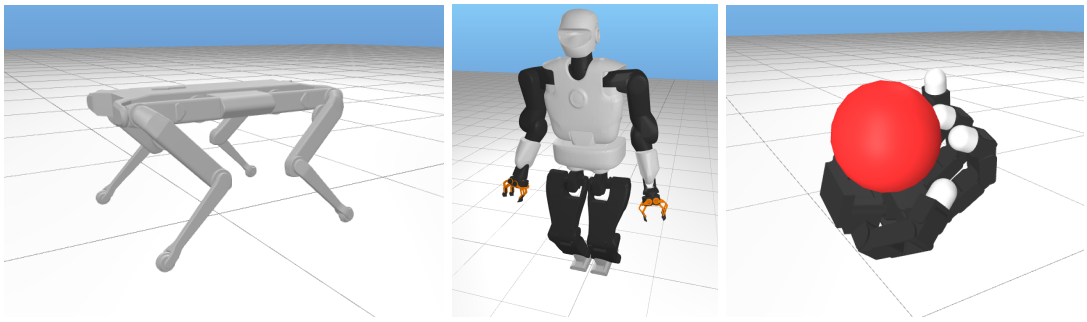

Fig. 8. **Robotics systems used for the experiments**. The Solo-12 quadruped (**Left**), the Talos humanoid (**Center**), and the Allegro hand (**Right**) allow to respectively exhibit locomotion, high-dimensional, and manipulation contact scenario.

Over-relaxation. Additionally, over-relaxation is often employed to accelerate the convergence of both Gauss-Seidel and ADMM algorithms. This technique applies the following update:

$$\lambda \leftarrow \alpha\lambda^{-} + (1-\alpha)\lambda, \tag{25}$$

where $\alpha \in]0, 2[$ and λ^{-} denotes the previous iteration. For $\alpha > 1$, over-relaxing consists of an extrapolation step and should be carefully used, as it may also hinder convergence. Typically, setting α to 0.8 improved convergence of the PGS algorithm.

IV. Experiments

In this section, we evaluate the performances and behaviors of the formulations explained in Sec. III. To fairly compare and benchmark the various algorithmic formulations, we have implemented them in a unified C++ framework called ContactBench. In the following, we denote by RaiSim and Drake our re-implementation of the contact solvers described in the corresponding papers [33], [35]. For Drake, it is worth noting that our implementation uses a backtracking line-search with an Armijo condition instead of the line-search proposed in the original paper [35]. Our framework extensively relies on the Pinocchio library [24] for rigid body algorithms and HPP-FCL [45], [56] implementation of GJK and EPA for collision detection. ContactBench will be released as open-source upon article acceptance.

Several factors may hinder the correctness and accuracy of simulators based on time-stepping methods:

i) the low accuracy of the solver of the contact problem;
ii) the limitation from the contact model itself;
iii) or the numerical integration due to the time discretization scheme.

In this section, we first evaluate the error from sources i) and ii) (Sec. IV-A). The source of error i) is evaluated by measuring the time taken to reach a given accuracy. The errors from ii) are analyzed by measuring the residual for an (approximately) infinite time budget. We further assess i) and iii) by examining the sensitivity of the contact solvers with respect to respectively the stopping criterion value ϵ_{abs} and the time-step Δt (Sec. IV-B). Sec. IV-C evaluates their computational efficiency. Finally, Sec. IV-D explores how the contact models and their implementations can impact the final robotics applications, in the case of the MPC for quadruped locomotion.

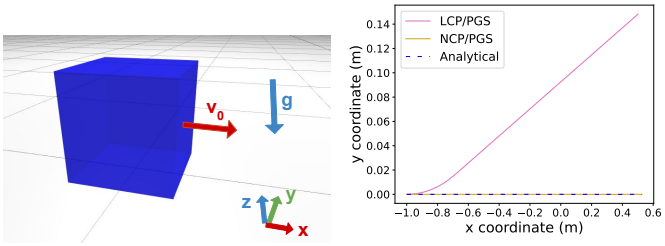

Fig. 9. **Trajectory of a cube sliding on a plane.** The cube is initialized with an initial tangential velocity along the x-axis. **Right:** The bias of friction forces (Fig.6) introduces a tangential velocity along the y-axis, which deviates the cube from the expected straight-line trajectory.

Except where expressly indicated, we use the following values for the solvers parameters: $\epsilon_{abs} = 10^{-6}$, $n_{iter} = 10^4$ and $\Delta t = 1ms$. For the ADMM algorithm, the proximal parameter ρ is adapted dynamically as previously detailed (Sec. III-E).

A. Evaluation of physical correctness

LCP relaxation. The linearization of the friction cone loses the isotropy and biases the friction forces towards some specific directions, as shown in Fig. 6. This observation has already been raised in the literature [84], [95], [85], [6], [8]. As expected, the bias on the contact forces significantly impairs the simulation by deviating the trajectory of the simulated system (Fig. 9).

CCP relaxation. As detailed previously, the CCP contact model relaxes the *Signorini condition*. As shown in Fig. 10, this results in non-null normal contact forces and velocities when a contact point is sliding. As a consequence, the contact points start to bounce, which modifies the trajectory of the system (Fig. 10, left), which also impacts the overall dissipated energy (Fig. 10, right). The model adopted by Raisim aims at correcting this undesired phenomenon by enforcing the Signorini condition but still does not match the analytical solution due to its relaxation of the MDP (24) (Fig. 10, right).

Underdetermined contact problems. Underdetermination occurs when infinite combinations of contact forces lead to

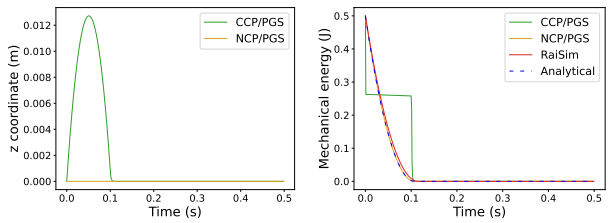

Fig. 10. **A cube is initialized on a plane with a tangential velocity along the x-axis,** similarly to the case studied in Fig. 9. **Left:** The CCP contact model relaxes the Signorini condition, which induces unphysical forces leading to the vertical bouncing of the cube. **Right:** From the MDP, it is possible to determine the evolution of the energy of the system analytically and compare it to what is computed by the various simulation algorithms. The CCP relaxation induces a significant gap with the analytical solution. The RaiSim contact model narrows this gap but dissipates less power than expected, as it does not always enforce the MDP. The NCP formulation, solved using the PGS solver, is the only formulation that closely matches the expected analytical behavior of the system.

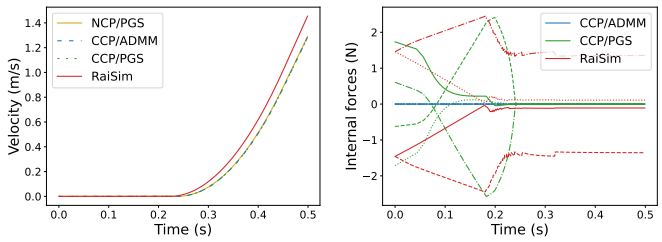

Fig. 11. **Applying a linearly growing force along the x-axis to a cube on a plane. Left:** The cube is at stiction before it starts sliding after approximately 0.25s. The tangential velocity differs depending on the contact model: CCP induces some bounces which occasionally dissipate energy during impacts, while RaiSim violates the MDP leading to contact points sliding faster than in the case of NCP. **Right:** At stiction, multiple combinations of tangential forces may lead to the same trajectory. There are four curves for each contact model, each curve accounting for the y-component from one of the four contact forces on the cube. Gauss-Seidel-like solvers, *e.g.* RaiSim and PGS, exhibit internal forces "stretching" the cube at stiction before the MDP enforces these forces to disappear when the cube starts to slide. RaiSim relaxes the maximum dissipation principle so the friction forces are not opposed to the movements, and internal forces persist when the cube is sliding. Eventually, ADMM avoids injecting spurious internal forces even at stiction.

the same trajectory. These artifacts happen on the normal and tangential components of contact forces, as depicted in Fig. 11. As shown in Fig. 11, the solution found depends on the numerical scheme. We observe that the per-contact approaches (Alg. 2,5 and 6) exhibit spurious internal forces at stiction, values which are not controlled by the algorithms. On the opposite, the algorithms working directly on the global contact problem with a proximal regularization (Alg. 3 and 7) seem to avoid injecting such artifacts in the contact forces (Fig. ,11). As future work, it would be interesting to investigate the theory behind the latter conjecture.

This phenomenon may seem innocuous as forward dynamics are not affected. However, it makes the inverse dynamics ill-posed, as there is no way to predict such numerical artifacts. Additionally, in the context of differentiable physics, we believe these spurious contact forces may catastrophically impact the computation of derivatives, but we also leave this study as future work. Finally, it is worth mentioning that such under-determined cases are ubiquitous in robotics (e.g., legged robots making redundant contact with their environments).

Robustness to ill-conditioned contact problems. More generally, the contact problem becomes challenging when the ratio between the biggest and the smallest eigenvalue of the Delassus matrix grows. The experiment of Fig. 12 exhibits the convergence issues of per-contact approaches when simulating systems with a strong coupling between the different contact points, which causes large off-diagonal terms on the matrix G. In this situation, the latter approaches hit the maximum number of iteration before convergence, leading to unrealistic trajectories. Such a behavior can be expected as supposing the matrix to be diagonally dominant is a classical hypothesis ensuring the convergence of Gauss-Seidel methods. On the contrary, the proximal algorithms account for off-diagonal terms of G, and only rely on a regularized inverse of G (Alg. 3, line 1), and thus robustly converge towards an optimal solution.

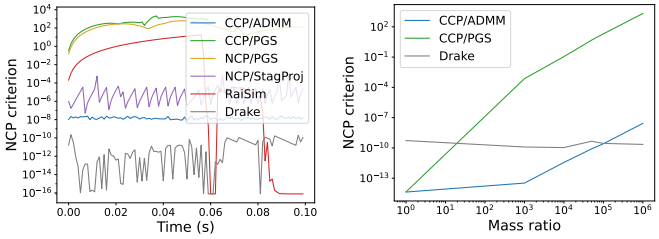

Fig. 12. **Simulation of ill-conditioned systems. Left:** Stacking a heavy cube (10^3kg) on a light one (10^{-3}kg) makes the problem ill-conditioned and, therefore, not solvable via per-contact algorithms (CCP/PGS, NCP/PGS and RaiSim) which results in the violation of the contact complementarity criterion (15). By contrast, the ADMM, staggered projections and Newton approaches appear to be robust in this case. **Right:** The accuracy of the simulators improves when the ratio between the masses of the two cubes gets close to one. The ADMM and Newton algorithms are less affected by this ratio than PGS.

Effects of compliance. As demonstrated by Fig. 13, the normal forces vary linearly with the compliance parameter R. Moreover, adding compliance to the tangential components induces the vanishing of dry friction, resulting in tangential oscillations instead of a null velocity. These compliant effects regularize the infinitely steep graphs due to the Signorini condition and Coulomb's law and replace them with locally linear mapping, which also eases the numerics. Therefore, the compliance added in MuJoCo has no physical purpose and should be considered a numerical trick designed to circumvent the issues due to hyper-staticity or ill-conditioning at the cost of impairing the simulation.

B. Self-consistency of the solvers

The accuracy of simulators can be affected by the numerical resolution induced by two "hyper-parameters": the value of the stopping criterion for the contact solver desired accuracy (ϵ_{abs}) and the time-step value (Δt). We measure their effect on the simulation quality when varying them independently. A simulator is said to be self-consistent when this deviation remains limited.

Time stepping simulators are sensitive to the choice of the

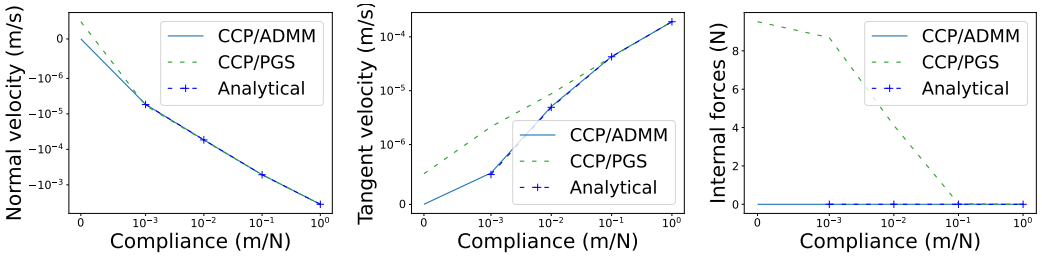

Fig. 13. **Simulation of Solo-12 with varying compliance for the contacts with the floor. Left:** Adding a compliance to the contact model relaxes the Signorini condition. **Center:** This compliance also relaxes the Coulomb's law of friction. **Right:** Compliance also regularizes the problem, removing the spurious internal forces in the under-determined cases.

time-step Δt. Here, we intend to assess the self-consistency of the various contact solvers by examining their deviation when Δt grows. Because time discretization also affects the collision detection process, our study is done on the trajectory of a cube dragged on a plane by a growing tangential force and whose contact points should remain constant. This scenario also allows to asses both sticking and sliding modes. For each simulator, a trajectory $\overline{q}$ obtained by simulating the system with a small time-step ($\Delta t = 10^{-2} ms$) serves as a reference to compute the state consistency error along the trajectories simulated with larger time-steps (Fig. 15). The deviation from the reference trajectory is quantified via the integral consistency error defined as $\sum_{\tau=0}^{T} \|q^\tau - \overline{q}^\tau\| \Delta t$.

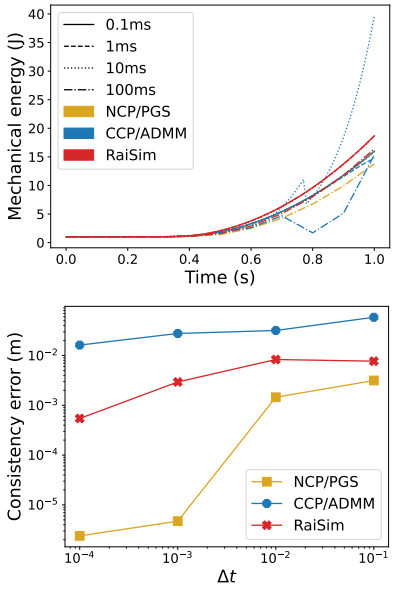

Fig. 14. **Self-consistency w.r.t. time-stepping when simulating a cube dragged on a plane by a growing tangential force.** The CCP contact model appears to be more sensitive to the time step Δt. This sensitivity can also be observed through the evolution of mechanical energy.

Looking at Fig. 14, we observe that the CCP contact model is more sensitive with respect to the time step in the considered scenario. Indeed, because CCP relaxes the *Signorini condition*, the cube slides at a height proportional to Δt. Similarly, as shown by Fig. 14, the energy evolution of the system simulated via NCP and RaiSim models is only a little modified when increasing Δt while CCP leads to a nonphysical and inconsistent behavior.

Stopping criterion. As done in the case of the time-stepping, the sensitivity to the stopping criterion is evaluated by measuring the integral consistency error with respect to a high-resolution ($\epsilon_{abs} = 10^{-9}$) reference trajectory (Fig. 15). As demonstrated by Fig. 15, the solvers appear to have different behaviors when their stopping criteria are modified. The gap between trajectories generated via the NCP contact models stays almost constant to a low value. Trajectories obtained via RaiSim significantly deviates when ϵ_{abs} is increased, while those resulting from CCP grows quickly from a low to a high consistency error. This reveals the importance of setting ϵ_{abs} to control the consistency of simulation with CCP. In addition, Fig. 15 demonstrates that the NCP model robustly simulates the mechanical energy evolution, which is not true for the CCP and RaiSim models. In particular, as the accuracy of RaiSim is relaxed, due to its approximation (24), the error on the MDP becomes uncontained leading to a reduced energy dissipation.

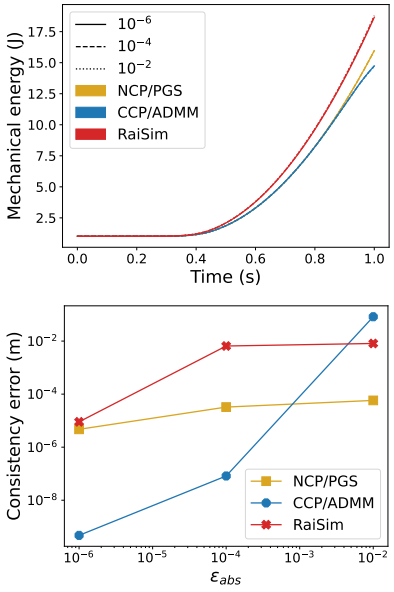

Fig. 15. **Self-consistency w.r.t. stopping criteria when simulating a cube dragged on a plane by a growing tangential force.** Dropping the solver accuracy hardly affects the energy evolution when using NCP/PGS. On the contrary, the energy of trajectories from the CCP/ADMM and RaiSim algorithms is more spectacularly impacted.

C. Performance benchmarks

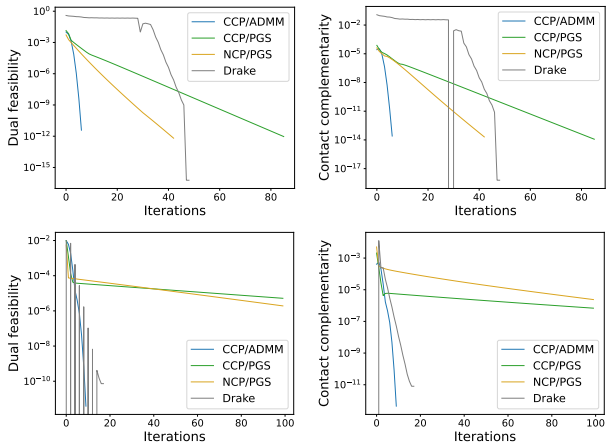

Fig. 16. **Convergence for contact problems on Solo (Top) and Talos (Bottom) robots.** PGS is fast to reach a mild accuracy before saturating, while ADMM and second order algorithms can get to higher precision.

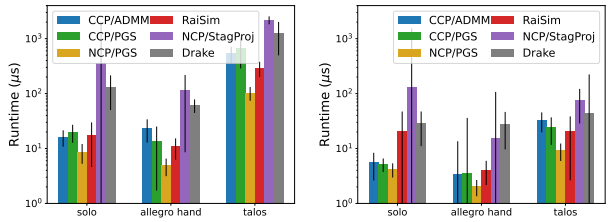

Fig. 17. **Computational timings measured along a trajectory for three robotic systems** (*c.f.* Fig. 8). Solo-12 and Talos are perturbed by applying an external force at their center of mass, while a ball is dropped in the Allegro hand, so the trajectories are not static. The contact solvers are tested in both cold-start (**Left**) and warm-start (**Right**) modes. We simulate the same trajectories to evaluate the benefit of warm-starting, but we use the solution of the previous time step as an initial guess. This leads to significant improvements in the computational timings.

As evoked earlier, in addition to being physically accurate, it is also essential for a simulator to be fast, which, in general, constitutes two adversarial requirements. To evaluate the computational footprint of the various solvers, we measure both the number of iterations and the time taken to reach a fixed accuracy on dynamic trajectories involving robotics systems (Fig. 8).

Looking at the number of iterations required to converge (Fig. 16), PGS approaches appear to be reasonably fast to reach mild accuracy ($\epsilon_{\text{abs}} = 10^{-5}$) while they eventually saturate before reaching high precision in complex scenarii (Fig. 16, bottom). We show later this can be unsufficient for challenging tasks (Fig. 19). On the other hand, ADMM, Newton and Staggered Projections algorithms can find high accuracy solutions using only a few, but more costly, iterations.

The latter analysis does not account for the per-iteration computational cost, so we report a study on final timings in Fig. 17. When the contact solvers are cold started, we observe that the second-order optimization techniques [32], [35] are less efficient than the PGS solvers and their cheap per-contact iterations (Fig. 17, left). The advanced first-order algorithms like ADMM (Alg. 3) working on the dual CCP problem (20) stands in-between as they leverage the very efficient Featherstone algebra [54] for the computation of the Cholesky factorisation of G (Sec. III-E). However, leveraging the solution from the previous time step to warm-start the solvers — a common strategy in practice — allows for significantly reducing this gap (Fig. 17, right). Therefore, regarding the study of Sec. IV-A, a trade-off appears for algorithms like ADMM, which treat all the contact points globally. In practice, they might be slower than their PGS counterpart while they benefit from better behaviors on ill-conditioned problems.

D. MPC for quadruped locomotion

The previous examples already illustrate the differences among the various simulators in terms of both physical accuracy and computational efficiency. However, such scenarios may not represent the richness of contacts in practical robotics situations. For this purpose, we use the implementation of MPC on the Solo-12 system introduced in [96] to generate locomotion trajectories on flat and bumpy terrains. These experiments are designed to involve a wide variety of contacts (*i.e.*, sticking, sliding, and taking-off) and see how the simulation choices impact the final task (*i.e.*, horizontal translation of the robot).

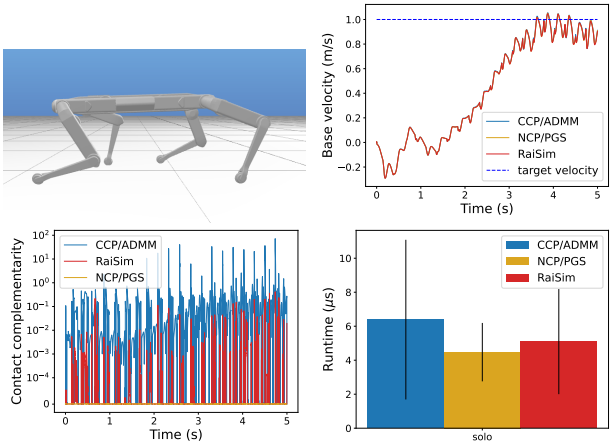

Fig. 18. **MPC for locomotion on a flat terrain** (**Top left**). The target horizontal translation velocity of the base is similarly reached by the controller with the different simulators (**Top right**). However, they do not equally respect the contact complementarity criterion (15) (**Bottom left**). Per-contact approaches, *e.g.* PGS and RaiSim, are more efficient (**Bottom right**).

For a flat and barely slippery ($\mu = 0.9$) ground, we observe that the choice of simulator hardly affects the base velocity tracked by the MPC controller (Fig. 18, top right). In this case, the contacts are mainly sticking, leading to low violation of the NCP criterion (15) (Fig. 18, bottom left).

When the terrain is bumpy (roughness of $10^{-1}m$) and slippery ($\mu = 0.3$), the locomotion velocity generated from the RaiSim and CCP models significantly deviates from the NCP one (Fig. 19, top right). This can be expected, in light of our previous study, as both the RaiSim and CCP contact models make physical approximations when contact points are sliding (Fig. 19, bottom left). However, we occasionally observe that NCP/PGS also violates the NCP criterion (15) (Fig. 19, bottom left) but for a different reason: PGS was not

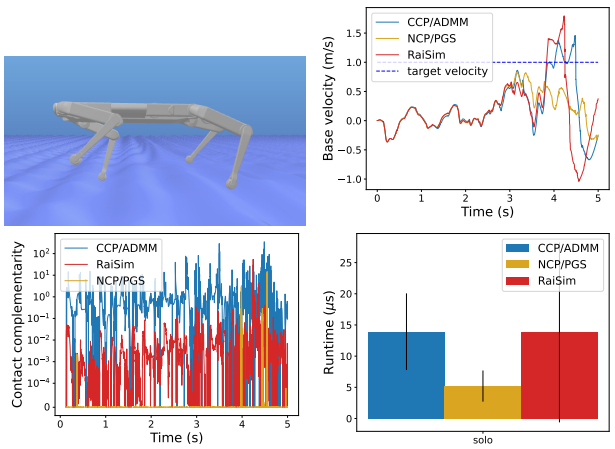

Fig. 19. **MPC for locomotion on a bumpy terrain** (**Top left**). The tracked velocity of the base quickly differs depending on the used simulator (**Top right**). Slippery contact points violate the contact complementarity criterion (15) for the RaiSim and CCP contact modelings (**Bottom left**). The complexity of contacts also hampers the solvers and reduces the gap between per-contact and ADMM approaches (**Bottom right**).

able to converge before the maximum number of iterations was reached. Therefore, Gauss-Seidel like approaches appear to be sufficient for mild conditions (Fig. 18) but are not robust enough to ensure convergence of the simulation when the locomotion tasks becomes more challenging (Fig. 19). This also causes increased computations from the solvers, particularly for RaiSim (Fig. 19, bottom right). These observations indicate that the low-level choices of the contact solver may induce significant differences in the high-level behaviors of locomotion controllers on complex terrains.

V. Discussion and Conclusion

NCP is known to be complex to solve and thus is often relaxed to find approximate solutions. In this article, we report a deeper study on how the various rigid contact models commonly employed in robotics and their associated solvers can impact the resulting simulation. We have notably established and experimentally highlighted that these choices may induce unphysical artifacts, thus widening the reality gap, leading to unrealistic behaviors when the simulator is later used for practical robotics applications. Our experiments show that there is no fully satisfactory approach at the moment, as all existing solutions compromise either accuracy, robustness, or efficiency. This indicates that there may still be room for improvements in contact simulation. It is also worth mentionning that, for robotics, simulation samples of lesser but controlled accuracy are already valuable for many applications, *e.g.* RL and MPC, while a failed simulation represent a waste of ressources. This paper showcases that situations prone to failure of simulation are not only corner cases but can become quite common when adressing challenging tasks such as locomotion. This justifies the emphasis put by modern simulators [32], [37], [35] on robustness when modeling contacts and implementing the associated solvers.

Beyond contact simulation, differentiable physics constitutes an emergent and closely related topic. However, the impact of forward simulation artifacts on gradient computation remains unexplored. In particular, some of the relaxations at work, *e.g.* the artificial compliance added in MuJoCo, result in crucial differences in gradients, which then affect downstream applications like trajectory optimization [97], [98]. We leave the study of the various existing differentiable simulators [5], [6], [7], [99], [8] through this lens as future work.

For all these reasons, we believe it would be highly beneficial for the robotics community to take up such low-level topics around simulation, as they could lead to substantial progress in the field. The work of [43] is an inspiring first step in this direction. With this article, we intend to go further by also providing open-source implementations and benchmarks to the community.

Acknowledgments

We warmly thank Jemin Hwangbo for providing useful details on the algorithm of the RaiSim simulator, Stéphane Caron and Nicolas Mansard for helpful discussions. This work was supported in part by L'Agence d'Innovation Défense, the French government under the management of Agence Nationale de la Recherche through the project INEXACT (ANR-22-CE33-0007-01) and as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute), by the European Union through the AGIMUS project (GA no.101070165) and the Louis Vuitton ENS Chair on Artificial Intelligence. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.," vol. 1, pp. 222–229, 01 2004.
[2] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.
[3] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, IEEE, 2012.
[4] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems (RSS 2018)*, 2018.
[5] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," *Advances in neural information processing systems*, vol. 31, 2018.
[6] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, 2021.
[7] 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.
[8] T. Howell, S. Le Cleac'h, J. Brüdigam, Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable simulator for robotics," *arXiv preprint arXiv:2203.00806*, 2022.
[9] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*, pp. 65–93, Springer, 2006.
[10] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.
[11] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7330–7336, IEEE, 2021.

[12] E. Dantec, M. Taix, and N. Mansard, "First order approximation of model predictive control solutions for high frequency feedback," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4448–4455, 2022.

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

[14] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots.," in *Robotics: Science and Systems*, 2018.

[15] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[16] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[17] M. T. Mason and J. K. Salisbury Jr, *Robot hands and the mechanics of manipulation*. The MIT Press, Cambridge, MA, 1985.

[18] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet, "On computing four-finger equilibrium and force-closure grasps of polyhedral objects," *The International Journal of Robotics Research*, vol. 16, no. 1, pp. 11–35, 1997.

[19] B. Dynamics, "Atlas gets a grip | boston dynamics." https://youtu.be/-e1_QhJ1EhQ.

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

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

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

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

[24] 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 *2019 IEEE/SICE International Symposium on System Integration (SII)*, pp. 614–619, IEEE, 2019.

[25] J. J. Moreau, "Unilateral Contact and Dry Friction in Finite Freedom Dynamics," in *Nonsmooth Mechanics and Applications* (M. J.J. and P. P.D., eds.), vol. 302 of *International Centre for Mechanical Sciences (Courses and Lectures)*, pp. 1–82, Springer, 1988.

[26] M. Jean, "The non-smooth contact dynamics method," *Computer methods in applied mechanics and engineering*, vol. 177, no. 3-4, pp. 235–257, 1999.

[27] V. Acary, F. Cadoux, C. Lemaréchal, and J. Malick, "A formulation of the linear discrete Coulomb friction problem via convex optimization," *Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 91, pp. 155–175, Feb. 2011.

[28] B. Brogliato, A. Ten Dam, L. Paoli, F. Génot, and M. Abadie, "Numerical simulation of finite dimensional multibody nonsmooth mechanical systems," *Appl. Mech. Rev.*, vol. 55, no. 2, pp. 107–150, 2002.

[29] R. Smith, "Open dynamics engine," 2008. http://www.ode.org/.

[30] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." http://pybullet.org, 2016–2021.

[31] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "DART: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, p. 500, Feb 2018.

[32] 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*, pp. 5026–5033, IEEE, 2012.

[33] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[34] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019.

[35] A. M. Castro, F. N. Permenter, and X. Han, "An unconstrained convex formulation of compliant contact," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1301–1320, 2022.

[36] D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai, "Staggered projections for frictional contact in multibody systems," pp. 1–11, 2008.

[37] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and V. Makoviychuk, "Non-smooth newton methods for deformable multi-body dynamics," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–20, 2019.

[38] A. Enzenhöfer, N. Lefebvre, and S. Andrews, "Efficient block pivoting for multibody simulations with contact," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 1–9, 2019.

[39] Z. Ferguson, M. Li, T. Schneider, F. Gil-Ureta, T. Langlois, C. Jiang, D. Zorin, D. M. Kaufman, and D. Panozzo, "Intersection-free rigid body dynamics," *ACM Trans. Graph.*, vol. 40, jul 2021.

[40] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu based physics simulation for robot learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[41] 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.

[42] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4397–4404, 2015.

[43] P. Horak and J. C. Trinkle, "On the similarities and differences among contact models in robot simulation," *IEEE Robotics and Automation Letters*, vol. 4, pp. 493–499, 2019.

[44] J. Carpentier, F. Valenza, N. Mansard, *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems." https://stack-of-tasks.github.io/pinocchio, 2015–2021.

[45] J. Pan, S. Chitta, D. Manocha, F. Lamiraux, J. Mirabel, J. Carpentier, *et al.*, "HPP-FCL: an extension of the Flexible Collision Library." https://github.com/humanoid-path-planner/hpp-fcl, 2015–2022.

[46] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "HPP: A new software for constrained motion planning," in *International Conference on Intelligent Robots and Systems*, 2016.

[47] L. Montaut, Q. Le Lidec, V. Petrík, J. Sivic, and J. Carpentier, "Collision Detection Accelerated: An Optimization Perspective," in *Proceedings of Robotics: Science and Systems*, (New York City, NY, USA), June 2022.

[48] F. E. Udwadia and R. E. Kalaba, "A new perspective on constrained motion," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1906, pp. 407–410, 1992.

[49] H. Bruyninckx and O. Khatib, "Gauss' principle and the dynamics of redundant and constrained manipulators," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3, pp. 2563–2568, IEEE, 2000.

[50] S. Redon, A. Kheddar, and S. Coquillart, "Gauss' least constraints principle and rigid body simulations," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1, (Washington, DC, United States), pp. 517–522, 2002.

[51] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1071–1076, IEEE, 2011.

[52] C. Studer, R. Leine, and C. Glocker, "Step size adjustment and extrapolation for time-stepping schemes in non-smooth dynamics," *International journal for numerical methods in engineering*, vol. 76, no. 11, pp. 1747–1781, 2008.

[53] S. L. Campbell and C. W. Gear, "The index of general nonlinear DAEs," *Numerische Mathematik*, vol. 72, pp. 173–196, 1995.

[54] J. Carpentier, R. Budhiraja, and N. Mansard, "Proximal and sparse resolution of constrained dynamic equations," in *Robotics: Science and Systems*, 2021.

[55] C. F. Gauß, "Über ein neues allgemeines grundgesetz der mechanik.," 1829.

[56] L. Montaut, Q. Le Lidec, V. Petrik, J. Sivic, and J. Carpentier, "Collision detection accelerated: An optimization perspective," in *RSS 2022-Robotics: Science and Systems*, 2022.

[57] G. Stavroulakis, "Multibody dynamics with unilateral contacts by friedrich pfeiffer and christoph glocker, wiley, new york, 1996," *Shock and Vibration*, vol. 5, pp. 371–372, 01 1998.

[58] C. Ericson, *Real-time collision detection*. Crc Press, 2004.

[59] 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*, vol. 4, no. 2, pp. 193–203, 1988.

[60] C. Ericson, *Real-Time Collision Detection*. The Morgan Kaufmann Series, 2004.

[61] K. Mamou and F. Ghorbel, "A simple and efficient approach for 3d mesh approximate convex decomposition," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 3501–3504, 2009.

[62] A. Signorini, "Questioni di elasticità non linearizzata e semilinearizzata," *Rendiconti di Matematica e delle sue applicazioni*, vol. 18, no. 5, pp. 95–139, 1959.

[63] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementarity Problem*. Society for Industrial and Applied Mathematics, 2009.

[64] P. Wensing, R. Featherstone, and D. E. Orin, "A reduced-order recursive algorithm for the computation of the operational-space inertia matrix," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4911–4917, IEEE, 2012.

[65] O. A. Bauchau and A. Laulusa, "Review of contemporary approaches for constraint enforcement in multibody systems," 2008.

[66] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer methods in applied mechanics and engineering*, vol. 1, no. 1, pp. 1–16, 1972.

[67] 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*, vol. 28, pp. 225–245, Aug. 1998.

[68] 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.

[69] H. Barbosa and R. Feijóo, "A numerical algorithm for signorini's problem with coulomb friction," in *Unilateral Problems in Structural Analysis—2: Proceedings of the Second Meeting on Unilateral Problems in Structural Analysis, Prescudin, June 17–20, 1985*, pp. 33–45, Springer, 1987.

[70] M. A. Tzaferopoulos, "On an efficient new numerical method for the frictional contact problem of structures with convex energy density," *Computers & structures*, vol. 48, no. 1, pp. 87–106, 1993.

[71] K. Erleben, "Rigid body contact problems using proximal operators," in *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[72] J. Pan, S. Chitta, and D. Manocha, "FCL: A General Purpose Library for Collision and Proximity Queries," in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012.

[73] D. Fiser, "libccd." https://github.com/danfis/libccd.

[74] M. L. Felis, "Rbdl: an efficient rigid-body dynamics library using recursive algorithms," *Autonomous Robots*, vol. 41, no. 2, pp. 495–511, 2017.

[75] M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller, "Small steps in physics simulation," in *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[76] M. Anitescu and A. Tasora, "An iterative approach for cone complementarity problems for nonsmooth dynamics," *Computational Optimization and Applications*, vol. 47, no. 2, pp. 207–235, 2010.

[77] E. Mitsopoulou and I. Doudoumis, "A contribution to the analysis of unilateral contact problems with friction," *Solid Mechanics Archives*, vol. 12, no. 3, pp. 165–186, 1987.

[78] I. Doudoumis and E. Mitsopoulou, "On the solution of the unilateral contact frictional problem for general static loading conditions," *Computers & structures*, vol. 30, no. 5, pp. 1111–1126, 1988.

[79] F. Jourdan, P. Alart, and M. Jean, "A gauss-seidel like algorithm to solve frictional contact problems," *Computer methods in applied mechanics and engineering*, vol. 155, no. 1-2, pp. 31–47, 1998.

[80] K. Erleben, "Numerical methods for linear complementarity problems in physics-based animation," in *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, (New York, NY, USA), Association for Computing Machinery, 2013.

[81] C. E. Lemke, "Bimatrix equilibrium points and mathematical programming," *Management science*, vol. 11, no. 7, pp. 681–689, 1965.

[82] G. B. Dantzig and R. W. Cottle, "Positive (semi-) definite matrices and mathematical programming," *Report ORC*, vol. 13, pp. 63–18, 1963.

[83] M. Anitescu and F. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, 03 1997.

[84] J. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo, "On dynamic multi-rigid-body contact problems with coulomb friction," *Zeitschrift Angewandte Mathematik und Mechanik*, vol. 77, no. 4, pp. 267–279, 1997.

[85] V. Acary and B. Brogliato, *Numerical methods for nonsmooth dynamical systems: applications in mechanics and electronics*. Springer Science & Business Media, 2008.

[86] M. Anitescu, "Optimization-based simulation of nonsmooth rigid multibody dynamics," *Mathematical Programming*, vol. 105, pp. 113–143, 2006.

[87] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[88] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[89] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

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

[91] T. Preclik, *Models and algorithms for ultrascale simulations of non-smooth granular dynamics*. Friedrich-Alexander-Universitaet Erlangen-Nuernberg (Germany), 2014.

[92] S. Boyd and L. Vandenberghe, "Localization and cutting-plane methods," *From Stanford EE 364b lecture notes*, 2007.

[93] J. Carpentier, Q. Le Lidec, and L. Montaut, "From compliant to rigid contact simulation: a unified and efficient approach," 2024.

[94] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[95] M. Renouf, V. Acary, and G. Dumont, "3d frictional contact and impact multibody dynamics. a comparison of algorithms suitable for real-time applications," in *Mutlibody Dynamics 2005, ECCOMAS Thematic Conference*, 2005.

[96] P.-A. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a Reactive Walking Controller for the New Open-Hardware Quadruped Solo-12," in *2021 IEEE International Conference on Robotics and Automation - ICRA*, (Xi'an, China), May 2021.

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

[98] 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*, vol. 52, p. 101468, 2024.

[99] 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.