Simple and Scalable Frictional Contacts for Thin Nodal Objects

GILLES DAVIET, Weta Digital, New Zealand

Fig. 1. A virtual character going through a running cycle, letting his hair impact repeatedly the back and neck of his shirt. The groom consists of 54, 450 Discrete Elastic Rods, totalling $1.2M$ degrees of freedom, while the shirt mesh contains about 27, 000 vertices. This scene induces as much as $4.5M$ contacts, which are solved implicitly and with nonlinear Coulomb friction thanks to our proposed algorithm. © Weta Digital.

Frictional contacts are the primary way by which physical bodies interact, yet they pose many numerical challenges. Previous works have devised robust methods for handling collisions in elastic bodies, cloth, or fiber assemblies such as hair, but the performance of many of those algorithms degrades when applied to objects with different topologies or constitutive models, or simply cannot scale to high-enough numbers of contacting points.

In this work we propose a unified approach, able to handle a large class of dynamical objects, that can solve for millions of contacts with unbiased Coulomb friction while keeping computation time and memory usage reasonable. Our method allows seamless coupling between the various simulated components that comprise virtual characters and their environment. Furthermore, our proposed approach is simple to implement and can be easily integrated in popular time integrators such as Projected Newton or ADMM.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Contact dynamics, Coulomb friction

ACM Reference Format:
Gilles Daviet. 2020. Simple and Scalable Frictional Contacts for Thin Nodal Objects. *ACM Trans. Graph.* 39, 4, Article 61 (July 2020), 16 pages. https://doi.org/10.1145/3386569.3392439

1 INTRODUCTION

The continuously increasing demand for visual richness of virtual environments has prompted the use of physical simulation to generate an ever larger portion of the final rendered frames. While the dynamics of the various objects that compose virtual environments can in themselves be of great interest, a large part of the visual complexity of their shape and motion can be attributed to the interactions that they have with surrounding components. In many cases, these interactions take the form of contact with dry friction; consider examples as diverse as a granular material, the leaves of a tree gently colliding under a breeze, or hair strands subtly entangled with neighbouring fibers, themselves lying upon layers of cloth and eventually skin.

Yet, simulation of dry friction has always proved numerically challenging, and robust approaches capable of efficiently handling these various object topologies and interactions in a unified manner are still scarce. Many previous works have focused on either methods capable of treating collisions between a very high number of "simple" components, such as granular materials, rigid bodies, or fiber assemblies, or at the other end of the spectrum a more reasonable number of large deformable bodies, such as flesh or cloth. Another dichotomy has also often been made between volumetric and thin objects, as slight amounts of inter-penetration can be tolerated for the former, but would immediately lead to hard-to-recover-from artefacts for the latter.

In this work, we propose a novel, simple yet scalable approach for the numerical simulation of hard contacts with nonlinear Coulomb friction for a large class of dynamical objects. Our method is especially compelling for assemblies of thin objects where the ratio of degrees of freedom to contact points is unfavorable, as illustrated in Fig. 1 where the garments of a character and the tens of thousands of fibers that comprise their hairstyle are simulated together, amounting to millions of degrees of freedom and contact points.

2 RELATED WORK

Due to the ubiquity of frictional contacts in physical and virtual environments, treatment of collisions in numerical simulations has been the subject of a vast amount of literature over the last decades.

2.1 Penalty forces

Inspired by molecular dynamics, the first methods developed in mechanics [Cundall and Strack 1979] and Computer Graphics [Moore and Wilhelms 1988] suggested to resolve contacts through the application of penalties, repulsive forces of intensity proportional to the inter-penetration depth of the colliding bodies. While Baraff and Witkin [1998] proposed an implicit integration scheme for elastic and collision energies that alleviated the drastic timestep restrictions

Author's address: Gilles Daviet, Weta Digital, New Zealand, gdaviet@wetafx.co.nz.

0730-0301/2020/7-ART61 $15.00
https://doi.org/10.1145/3386569.3392439

imposed by the explicit application stiff penalties, their approach suffered from excessive numerical dissipation and sticking artefacts; recently Michels et al. [2017] fixed this issue by using a stiffly accurate integrator. Traditionally, the frictional component of penalty forces is defined as proportional to the sliding velocity; this does not allow modeling dry friction, where non-zero friction can exist at rest. Yamane and Nakamura [2006] propose to sidetrack this limitation through the use of frictional anchors that are continuously updated as to locally satisfy Coulomb law.

Even with implicit treatment, finite penalties cannot prevent tunnelling for arbitrary colliding momenta. This is especially problematic for thin objects like cloth. To this end, the popular method of Bridson et al. [2002] augments penalties with an end-of-step geometric correction scheme that iteratively moves vertices to resolve collisions, and falls back to the rigid-impact-zone failsafe of Provot [1997] if it cannot find valid positions. Harmon et al. [2008] relaxed this failsafe to allow sliding at impact zones, but still fails to accurately model dry friction. These corrective steps also ignore internal elasticity of the dynamic objects, potentially leading to large stresses after resolution.

2.2 Hard contacts

In parallel, a vast amount of literature has been dedicated to implicit constraint-based contacts and the proper treatment of dry friction. Jean and Moreau [1988] first proposed the use of impulses to resolve impacts in a timestepping integrator. In Computer Graphics, Baraff [1989; 1994] popularized the use of inequality constraints and *linear complementarity* to model unilateral contacts. Kaufman et al. [2008] demonstrated that efficient *quadratic program* solvers can be leveraged to solve the normal and tangential force components in a staggered fashion.

Second-order methods. Alart and Curnier [1991] proposed a first approach to solve 3D contact dynamics with Coulomb friction by applying the Newton method to find the roots of a nonsmooth *complementarity function*. In Computer Graphics, Bertails-Descoubes et al. [2011] applied this algorithm to the simulation of fiber assemblies, but noted that convergence was hard to achieve for high ratios of contacts to degrees of freedom (DoF). Another suitable, slightly smoother complementarity function was devised by Fukushima et al. [2002]. Recently, Macklin et al. [2019] proposed a novel Newton algorithm that constructs successive local quadratizations of complementarity functions, yielding a series of linear systems that can be efficiently solved through the Conjugate Residual method and demonstrating much more robust convergence. Newton algorithms have also been employed inside Interior Point solvers [Acary et al. 2011; Heyn 2013] to solve a convexified version of the Coulomb law, the so-called *associated* friction law. Acary et al. [2011] also proposed a fixed-point algorithm reducing exact Coulomb friction to several instances of the associated law.

While the theoretical quadratic rate of convergence of Newton and Interior Points methods is attractive for medium-scale contact dynamics problems requiring high accuracy, the fact that they require solving a new linear system at each iteration, as well as the difficulty to warmstart interior point solvers, have contributed to the greater popularity of first-order methods for large scale visual applications.

Operator-splitting methods. The idea of sequentially solving contacts one-by-one until convergence is achieved for the whole system has first been proposed by Jean and Moreau [1992], then popularized as the *Non-Smooth Contact Dynamics* method [Jourdan et al. 1998; Jean 1999]. They observed that applying the Newton algorithm of Alart and Curnier [1991] to single contacts multiple times is both more robust and more efficient than applying the method to all contacts at once. Duriez et al. [2004]; Catto [2005]; Erleben [2007] popularized such Gauss–Seidel-like algorithms in Computer Graphics, but treated only the frictionless case or linearized friction laws. Bonnefon and Daviet [2011] proposed an analytical enumerative solver for the one-contact problem with exact Coulomb friction. Daviet et al. [2011] combined this analytical solver with the complementarity function from Fukushima et al. [2002] to solve the dynamics of a few thousands Super-Helices [Bertails et al. 2006]. Kaufman et al. [2014] embedded their algorithm inside a non-linear elasticity solver and were able to robustly simulate frictional contacts in assemblies of up to 64k Discrete Elastic Rods [Bergou et al. 2008, 2010]. Recently Erleben [2017] described a general framework for building operator-splitting algorithms.

Operator-splitting algorithms have demonstrated satisfying scaling behaviour for dynamical systems consisting in many small objects, like hair fibers or rigid bodies. However, they require either explicitly assembling [*e.g,* Jean 1999] or back-solving many times [*e.g,* Erleben 2007] the so-called *Delassus operator*, making them unsuitable for the simulation of larger elastic bodies, like cloth or muscles. For such fully-connected bodies the Delassus operator is indeed generally dense, with a number of rows and columns corresponding to the total number of contacts; this leads to intractable memory and/or computational costs. Otaduy et al. [2009] solved this issue by adding another level of operator-splitting, propagating the elastic strains in a second loop on top of the contact solver. Li et al. [2018] proposed to altogether avoid the use of the Delassus operator by expressing the constrained dynamics on the primal velocity variable rather than the dual forces. However, this transformation is only made possible by placing stringent assumptions on the contact configuration.

First-order descent methods. While robust, operator-splitting solvers suffer from poor (logarithmic) theoretical convergence, and asymptotically faster solvers based on variants the Projected Gradient algorithm have also been developed [Tasora 2013; Mazhar et al. 2015]. Derived from convex optimization theory, these methods target the associated version of the Coulomb law. One major advantage is that they require only solving by the Delassus operator once per iteration, and thus remain tractable for large elastic bodies. Moreover, the constraint projection step is embarrassingly parallel. However, the projection being performed on the force variable means that feasibility of the velocity iterate will not be enforced until full convergence of the algorithm. In contrast, primal-dual descent methods, such as the *Alternating Directions Method of Multipliers* (ADMM) deriving from Augmented Lagrangian formalism [Fortin and Glowinski 1983], can perform the projection on the primal velocity variable, ensuring its feasibility. Such methods have recently

become popular in Computer Graphics [Narain et al. 2016; Inglis et al. 2017; Fang et al. 2019], though their application to contact dynamics remains limited to single vertices [Narain et al. 2016] or suffers from hard-to-tune convergence parameters [Daviet 2016]. Finally, we mention that Krylov-subspace descent methods have also been formulated [Renouf and Alart 2005; Verschoor and Jalba 2019]. While such methods are theoretically limited to linear constraints, Verschoor and Jalba [2019] emulate the true quadratic friction cone by progressively aligning the friction direction with the relative velocity.

2.3 Position-based dynamics

Position-based dynamics (PBD) [Müller et al. 2007] were developed as an attempt to plausibly approximate physics-based animations with robust handling of frictional contacts for real-time applications. Their relative simplicity, stability, and availability in off-the-shelf commercial packages have since then made PBD immensely popular for offline simulations as well. While the behaviour of the original method was highly dependent on non-physical parameters such as number of iterations, Macklin et al. [2016] later introduced X-PBD, a modification that ensures convergence to a proper dynamical equilibrium. However, since constraints are solved in a Gauss-Seidel fashion, strains are only propagated to the neighbouring particles at each iteration. For long chains of particles, achieving convergence would thus require an unacceptably long time. Moreover, as we show in Section 5.2.4, the contact solving approach of PBD and X-PBD may introduce artificial energy into the system.

2.4 Continuum approaches

Tangential to the present article, an alternative line of work proposes to apply continuum methods to the treatment of contact dynamics between elastic bodies — which is especially justified when the typical size of the objects heterogeneities is small compared to the simulation resolution. Hadap and Magnenat-Thalmann [2001] first applied this idea to the simulation of hair dynamics, and more recently Jiang et al. [2017] adapted the granular Drucker–Prager rheology to the simulation of thin 1D and 2D objects. While the background grid used in continuum approaches supplants proximity detection and prevents intersections "for free", precise collision resolution requires a fine grid and thus tight timestep restrictions. To alleviate this issue, hybrid methods augmenting coarse continuum simulations with discrete collision resolution have also been proposed [McAdams et al. 2009; Yue et al. 2018; Han et al. 2019; Fei et al. 2019]. The natural collision resolution granted by the background velocity field also tends to average-out high-frequency velocity gradients, hindering subtle changes in local topology.

3 OVERVIEW

Let us first restate our objectives. We would like a frictional contact solver able to handle at the same time a large amount of small objects like hair strands, and large elastic objects like cloth or muscles, with as few as possible restrictions on the choice of discretization and constitutive models. Furthermore, as we are dealing with thin, two-sided objects, we cannot tolerate tunnelling and must thus enforce strict feasibility of the objects velocities. We also need our method to be highly scalable, up to millions of contact points, and to perform robustly even in the unfavorable DoF -starved regime identified by Bertails-Descoubes et al. [2011]. Conversely, as we target mainly visual applications moderate accuracy of the contact forces is acceptable, keeping first-order methods relevant. That being said, we still need our solver to be precise enough for our simulation to remain stable, and properly capture the characteristic behaviour of Coulomb friction. Finally, we want the visual result to depend as little as possible on non-physical parameters such as iteration numbers.

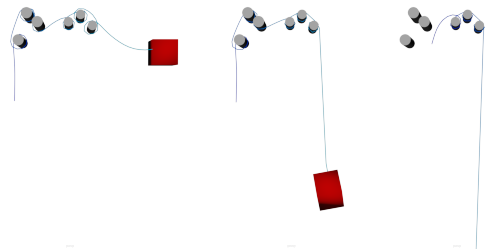


Fig. 2. A long elastic rod is looped around several cylinders and attached to a heavy elastic body (left), which is then released under gravity. Equipped with a high friction coefficient ($\mu = 0.5$), the device is self-locking (middle), while a low friction coefficient ($\mu = 0.2$) will lead to the rod unravelling completely (right). This experiment illustrates the common scenario of a single contact layer, but far-range stretch propagation.

None of the methods presented above satisfy all those criteria, but we will take inspiration from a number of them. Like Otaduy et al. [2009]; Gornowicz and Borac [2015], we will decompose our problem into an elastic relaxation part and a contact projection part, and alternate between solving the two. Similarly to those works, the feasibility constraint will be enforced with a Gauss–Seidel algorithm, but will leverage an analytical local solver on the primal velocity variable, yielding an implementation very similar to PBD [Müller et al. 2007] but with fixed mathematical properties. In contrast to Gornowicz and Borac [2015], we won't compute the actual local contact forces — only the resulting forces at DoF . Moreover, our contact projection scheme takes into account elastic stiffness, and ensures that nonlinear Coulomb friction is satisfied for arbitrary contact points, while Otaduy et al. [2009] uses a faceted approximation and the friction update proposed by Gornowicz and Borac [2015] is only exact for contacts at vertices. For the elastic relaxation step, we will solve penalized linear systems using the Conjugate Gradient method ensuring fast, global propagation of the strain updates. The left-hand-side of these linear systems will remain constant across iterations, permitting the use of an efficient preconditioner. The intuition behind using Gauss–Seidel for contacts but a global linear solve for elastic relaxation is that in our simulations the length of fully-connected DoF chains is typically much greater than the number of densely-packed contacting layers; see *e.g*, Fig. 2. The decomposition of our problem into elastic relaxation and contact projection parts will be achieved through the ADMM formalism [Fortin and Glowinski 1983].

3.1 Contributions

Our contributions are as follow:

- the derivation of an iterative algorithm based on the ADMM framework for **splitting the frictional contact dynamics problem into two sub-problems** that are much easier to solve individually. This algorithm is non-intrusive, can handle a large class of deformable objects, and can be plugged into standard nonlinear integrators such as Projected Newton;
- a **matrix-free Gauss–Seidel solver** for enforcing hard contact constraints with nonlinear Coulomb friction at the velocity level that is both highly scalable and simple to implement;
- various optional extensions for these algorithms, for instance for handling cohesion and contact compliance.

3.2 Outline

Section 4 will present the dynamical simulation context in which we inscribe our contributions, introduce notations, and recall the definition of the Signorini–Coulomb contact law. Section 5 walks through the mathematical derivation of our main contributions and the insights underpinning them. This section assumes some basic knowledge of convex optimization theory, but is not mandatory for practical implementation purposes. We invite the impatient reader to jump ahead the next one. Section 6 summarizes our complete algorithm, and provides a detailed implementation guide. Section 7 validates our algorithm on model examples as well as showcases some large-scale coupled simulations that our method enables. Finally Section 8 discusses benefits and limitations of our approach and potential future work.

4 SIMULATION CONTEXT

4.1 Discretization

Space discretization. Let $\mathbf{q}$ in $\mathbb{R}^m$ denote the concatenated vector of all degrees of freedom of our dynamic objects, and $\mathbf{v} := \dot{\mathbf{q}}$ their time derivative. We now make an assumption on the discretization that will allow for an efficient treatment of collisions.

Assumption 1. *For any point belonging to a dynamic object, we assume that its position $\boldsymbol{x}(t) \in \mathbb{R}^3$ can be expressed as a linear combination of 3D degrees of freedom,*

$$\boldsymbol{x}(t) = \sum_j b_j(\boldsymbol{x})\, \boldsymbol{q}_j(t) + \underline{\boldsymbol{x}}(t),$$

with $b_j(\boldsymbol{x}) \in \mathbb{R}$, $\boldsymbol{q}_j(t) \in \mathbb{R}^3$, and $\underline{\boldsymbol{x}}(t) \in \mathbb{R}^3$ an optional kinematic term.

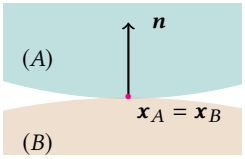
In other terms, objects may have non "position-like" degrees of freedom as long as such DoF do not take part in the computation of the position of potential colliding points.

The class of objects that satisfy Assumption 1 includes nodal systems — whose degrees of freedom are vertices of a mesh; many FEM basis functions; higher-order interpolation schemes such as GMLS [Martin et al. 2010] or frame-based models [Faure et al. 2011]; and even partially-reduced models such as Discrete Elastic Rods [Bergou et al. 2008, 2010], as long as we assume an infinitely thin centerline. Notable exclusions are rigid-bodies (rotation DoF are not allowed in our framework) and most reduced-coordinates models such as Super-Helices [Bertails et al. 2006].

Time discretization. Throughout this paper we will consider a finite timestep $[t, t + \Delta_t]$. Begin-of-step quantities will use the superscript $\cdot^t$ (*e.g,* $\boldsymbol{q}_i^t$). Since we focus on implicit integration, we do not superscript end-of-step values for concise notation. For instance, we will integrate velocities over the timestep as $\boldsymbol{q} = \boldsymbol{q}^t + \Delta_t \boldsymbol{v}$.

4.2 Contacts

Let us consider a contact between point $\boldsymbol{x}_{c,A}$ of body (A) and point $\boldsymbol{x}_{c,B}$ of body (B). We will note $\boldsymbol{n}_c$ the contact normal pointing from (B) to (A), $\boldsymbol{r}_c$ the contact force applied by object (B) onto object (A), and $\boldsymbol{h}_c$ the position gap, $\boldsymbol{h}_c := \boldsymbol{x}_{c,A} - \boldsymbol{x}_{c,B}$. For any 3D vector $\boldsymbol{z}$ defined at the contact point, we will also denote by z_{N} and $\boldsymbol{z}_{\mathrm{T}}$ its *normal* and *tangential* parts, *i.e,* $z_{\mathrm{N}} := \boldsymbol{z} \cdot \boldsymbol{n}_c$ and $\boldsymbol{z}_{\mathrm{T}} := \boldsymbol{z} - z_{\mathrm{N}} \boldsymbol{n}_c$. While the normal should itself be an implicit function of the DoF [Tang et al. 2012], in the following we will considered $\boldsymbol{n}_c$ to be constant over the timestep and rely on multiple collision detection passes to ensure that the end-of-step state is indeed collision-free.



Kinematics. Given Assumption 1, the gap function $\boldsymbol{h}_c$ can be expressed as a linear combination of 3D DoF as

$$\boldsymbol{h}_c(t) = \sum_{j \in \mathbb{J}_c} b_{c,j} \boldsymbol{q}_j(t) + \underline{\boldsymbol{h}}_c(t),$$

where $\mathbb{J}_c$ is the subset of node indices from objects (A) and (B) for which $b_{c,j} \neq 0$ and $\underline{\boldsymbol{h}}_c$ encompasses kinematic components, including for instance the motion of scripted colliders.

For brevity of notation and when there are no ambiguities, in the remainder of this document we will drop the c subscript when referring to quantities defined at any given contact. We will also use upright letters to refer to the concatenated vector of values at all contact points, *e.g,* $\mathbf{u} = (\boldsymbol{u}_c)_{c \geq 0}$. In particular, gathering all $b_{c,j}$ coefficients in the sparse matrix B, we will write the linear relationship between $\mathbf{h}$ and $\mathbf{q}$ as $\mathbf{h} = B\mathbf{q} + \underline{\mathbf{h}}$.

Discrete-time relative velocity. As classically done in contact mechanics [*e.g,* Stewart and Trinkle 1996]), we express the end-of-timestep non-penetration condition $h_{\mathrm{N}} \geq 0$ using a first-order Taylor approximation, $h_{\mathrm{N}} \sim h_{\mathrm{N}}^t + \Delta_t \dot{h}_{\mathrm{N}} \geq 0$. For notational convenience and without loss of generality, we include this normal offset in the definition of our discrete-time relative velocity,

$$\boldsymbol{u} := \dot{\boldsymbol{h}} + \frac{1}{\Delta_t} h_{\mathrm{N}}^t \boldsymbol{n}.$$

This definition allows expressing the velocity-level non-penetration condition compactly as $u_{\mathrm{N}} \geq 0$. Note that the tangential part $\boldsymbol{u}_{\mathrm{T}}$ still coincides with the "physical" relative velocity. Finally, we include this normal offset in an all-encompassing kinematic term $\boldsymbol{k} := \dot{\underline{\boldsymbol{h}}} + \frac{1}{\Delta_t} h_{\mathrm{N}}^t \boldsymbol{n}$, so that our relative velocity $\boldsymbol{u}$ can be expressed as

$$\boldsymbol{u} = \sum_{j \in \mathbb{J}} b_{c,j} \boldsymbol{v}_j + \boldsymbol{k},$$

or more compactly in matrix form as $\mathbf{u} = B\mathbf{v} + \mathbf{k}$.

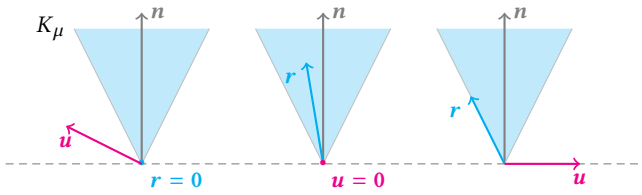


Fig. 3. Take-off (left), sticking (center) and sliding (right) cases of the Signorini–Coulomb conditions.

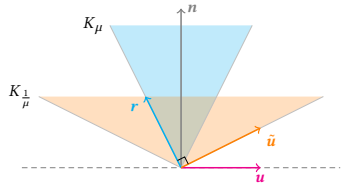


Fig. 4. Feasible velocity vectors $\boldsymbol{u}$ and $\tilde{\boldsymbol{u}}$ for the non-associated (a.k.a. Coulomb; magenta) and associated (orange) flow rules. The associated law $K_\mu \ni \boldsymbol{r} \perp \tilde{\boldsymbol{u}} \in K_{\frac{1}{\mu}}$ requires the velocity $\tilde{\boldsymbol{u}}$ to be orthogonal to the force $\boldsymbol{r}$, which is not the case for the Coulomb law.

4.3 Signorini-Coulomb conditions

The Signorini condition expresses that not only interpenetration must be prevented, *i.e,* $u_\mathrm{N} \geq 0$, but that the contact force needs also be positive and non-zero only if the two points are actually in contact, *i.e,* when $u_\mathrm{N} = 0$. All those conditions can be written concisely using complementarity notation as

$$0 \leq u_\mathrm{N} \perp r_\mathrm{N} \geq 0. \tag{1}$$

Coulomb friction dictates that the friction force $\boldsymbol{r}_\mathrm{T}$ must obey the maximum dissipation principle while being bounded by the friction coefficient μ times the normal contact force r_N. Formally,

$$\begin{cases} \boldsymbol{r} \in K_\mu \\ \boldsymbol{r}_\mathrm{T} = -\mu r_\mathrm{N} \dfrac{\boldsymbol{u}_\mathrm{T}}{\|\boldsymbol{u}_\mathrm{T}\|} & \text{if } \boldsymbol{u}_\mathrm{T} \neq \boldsymbol{0}, \end{cases} \tag{2}$$

where $K_\mu := \left\{ \|\boldsymbol{r}_\mathrm{T}\| \leq \mu r_\mathrm{N},\ \boldsymbol{r} \in \mathbb{R}^3 \right\}$ denotes the second-order cone of aperture μ. The possible solutions of the Signorini-Coulomb conditions (1-2) are summarized in Fig. 3; in the following, we will denote the set such of velocity–force pairs $(\boldsymbol{u}, \boldsymbol{r})$ by C_μ.

4.4 Implicit integration

The unconstrained conservation of momentum written at the end of our timestep reads

$$M\frac{\mathbf{v} - \mathbf{v}^t}{\Delta_t} = -\frac{\partial}{\partial \mathbf{q}}\mathcal{E}(\mathbf{q}, \mathbf{v}, t + \Delta_t) - \frac{1}{\Delta_t}\frac{\partial}{\partial \mathbf{v}}\mathcal{E}(\mathbf{q}, \mathbf{v}, t + \Delta_t),$$

where M is the mass matrix and $\mathcal{E}$ combines elastic and dissipation potentials. Given our Lagrangian advection scheme $\mathbf{q} = \mathbf{q}^t + \Delta_t \mathbf{v}$, we can eliminate $\mathbf{q}$ from the above equation and rewrite it simply as

$$\frac{\partial}{\partial \mathbf{v}}\mathcal{A}(\mathbf{v}) = \mathbf{0}, \tag{3}$$

with the total energy $\mathcal{A}(\mathbf{v}) := \mathcal{E}(\mathbf{q}(\mathbf{v}), \mathbf{v}, t+\Delta_t) + \frac{1}{2}(\mathbf{v}-\mathbf{v}^t)^T M(\mathbf{v}-\mathbf{v}^t)$. Various implicit time integrators have been employed in Computer Graphics to solve the unconstrained optimization problem (3); in the following we will make use of Projected Newton [Teran et al. 2005] and ADMM [Narain et al. 2016].

Constrained dynamics. Following Jean and Moreau [1988], we integrate contact forces over the timestep and plug them into Eq. 3 through Lagrange multipliers, yielding our incremental problem

$$\begin{cases} \dfrac{\partial}{\partial \mathbf{v}}\mathcal{A}(\mathbf{v}) = B^T \mathbf{r} \\ \qquad \mathbf{u} = B\mathbf{v} + \mathbf{k} \\ (\mathbf{u}, \mathbf{r}) \in C_\mu \qquad \forall 1 \leq c \leq n. \end{cases} \tag{4}$$

Iterative quasi-Newton methods such as Projected Newton can be easily adapted to solve the constrained incremental problem (4), as exemplified by Jean [1999]. At each iteration k, one constructs a symmetric operator A^k approximating the Hessian of $\mathcal{A}$ at $\mathbf{v}^k$, and proceeds to compute the next iterate $\mathbf{v}^{k+1}$ as the solution of a Discrete Coulomb Friction Problem (DFCP),

$$\begin{cases} A^k \mathbf{v}^{k+1} = \mathbf{f}^k + B^T \mathbf{r}, \quad \mathbf{f}^k := A^k \mathbf{v}^k - \dfrac{\partial}{\partial \mathbf{v}}\mathcal{A}(\mathbf{v}^k) \\ \qquad \mathbf{u} = B\mathbf{v}^{k+1} + \mathbf{k} \\ (\mathbf{u}, \mathbf{r}) \in C_\mu \qquad \forall 1 \leq c \leq n. \end{cases} \tag{5}$$

Note that Problem (5) is itself a special case of Problem (4) where $\mathcal{A}$ happens to be quadratic. Several methods have in turn be proposed to solve this DFCP , however their efficiency will be largely determined by the structure of the matrix A^k. For general elastic bodies, A^k will have a dense inverse, which renders methods requiring explicit assembly or many multiplications by the Schur complement $BA^{k^{-1}}B^T$ intractable.

In the next section we will show how we can apply the ADMM optimization scheme to solve efficiently the incremental problem (4) or its linearized variant (5). Our goal is to give an intuition about the method, but we invite any reader purely interested in the implementation details to jump ahead to Section 6.

5 MATHEMATICAL DERIVATION

5.1 Alternating Direction Method of Multipliers

Proxy convex friction law. To leverage the convex optimization formalism, we would need our friction law to derive from a convex potential. Coulomb friction does not satisfy this condition, but exhibits strong similarities with the *associated* friction law (Fig. 4), which itself is convex; the two laws even coincide in the sticking and frictionless cases. We will thus temporarily use the associated friction law as a convex proxy to guide our derivations before switching back to the actual Signorini-Coulomb conditions. This process will not preserve the convergence guarantees granted by convex optimization theory, but we may expect the practical behaviour to be similar enough. Replacing the Signorini–Coulomb conditions with the associated law in the incremental problem (4), we recognize the optimality conditions of the convex minimization problem

$$\min_{\mathbf{v} \in \mathbb{R}^m,\ B\mathbf{v}+\mathbf{k} \in K_{\frac{1}{\mu}}} \mathcal{A}(\mathbf{v}); \tag{6}$$

see Appendix A for more details.

ADMM. Let us now introduce an auxiliary variable $\mathbf{p} \in \mathbb{R}^m$ that we will constrain to be equal to $\mathbf{v}$, and recall the definition of the characteristic function C associated to the conical constraint,

$$C : \mathbb{R}^m \to \mathbb{R},\ \mathbf{p} \mapsto \begin{cases} 0 & \text{if } B\mathbf{p} + \mathbf{k} \in K_{\frac{1}{\mu}} \\ +\infty & \text{otherwise.} \end{cases}$$

We can split our optimization problem (6) as

$$\min_{\mathbf{p}=\mathbf{v},\ (\mathbf{v},\mathbf{p})\in\mathbb{R}^{2m}} \mathcal{A}(\mathbf{v}) + C(\mathbf{p}),$$

and write it under Augmented Lagrangian form[1],

$$\max_{\tilde{\boldsymbol{\lambda}}\in\mathbb{R}^m} \quad \min_{(\mathbf{v},\mathbf{p})\in\mathbb{R}^{2m}} \mathcal{L}\left(\mathbf{v},\mathbf{p},\tilde{\boldsymbol{\lambda}}\right),$$
$$\mathcal{L}\left(\mathbf{v},\mathbf{p},\tilde{\boldsymbol{\lambda}}\right) := \mathcal{A}(\mathbf{v}) + C(\mathbf{p}) + \tilde{\boldsymbol{\lambda}}^T\sqrt{W}(\mathbf{p}-\mathbf{v}) + \frac{1}{2}\|\sqrt{W}(\mathbf{p}-\mathbf{v})\|^2.$$

Here W is a diagonal matrix assigning a weight $w_j > 0$ to each 3D DoF constraint $\boldsymbol{p}_j = \boldsymbol{v}_j$. Note that this does not mean using a soft constraint – the constraint weights are affecting the convergence of the algorithm, but not its final result, where strict equality should hold. We will come back on how to choose these weights in practice in Section 6.2.6.

The ADMM algorithm then proceeds to iteratively optimize over each variable of the Augmented Lagrangian, assuming the other two fixed; at iteration l,

(i) $\mathbf{v}^{l+1} \leftarrow \arg\min_{\mathbf{v}\in\mathbb{R}^m} \mathcal{L}(\mathbf{v}, \mathbf{p}^l, \tilde{\boldsymbol{\lambda}}^l)$;
(ii) $\mathbf{p}^{l+1} \leftarrow \arg\min_{\mathbf{p}\in\mathbb{R}^m} \mathcal{L}(\mathbf{v}^{l+1}, \mathbf{p}, \tilde{\boldsymbol{\lambda}}^l)$;
(iii) $\tilde{\boldsymbol{\lambda}}^{l+1} \leftarrow \tilde{\boldsymbol{\lambda}}^l + \sqrt{W}(\mathbf{p}^{l+1} - \mathbf{v}^{l+1})$.

As suggested by Narain et al. [2016], to simplify notations we introduce the change of variable $\boldsymbol{\lambda} := \sqrt{W}^{-1}\tilde{\boldsymbol{\lambda}}$. Our ADMM iteration becomes:

(i) $\mathbf{v}^{l+1} \leftarrow \arg\min_{\mathbf{v}\in\mathbb{R}^m} \mathcal{A}(\mathbf{v}) + \frac{1}{2}\|\sqrt{W}\left[\mathbf{v} - (\mathbf{p}^l + \boldsymbol{\lambda}^l)\right]\|^2$;
(ii) $\mathbf{p}^{l+1} \leftarrow \arg\min_{\mathbf{p}\in\mathbb{R}^m} C(\mathbf{p}) + \frac{1}{2}\|\sqrt{W}\left[\mathbf{p} - (\mathbf{v}^{l+1} - \boldsymbol{\lambda}^l)\right]\|^2$;
(iii) $\boldsymbol{\lambda}^{l+1} \leftarrow \boldsymbol{\lambda}^l + \mathbf{p}^{l+1} - \mathbf{v}^{l+1}$.

Step (iii) is trivial, and Steps (i) and (ii) can be recognized as evaluating proximal operators for $\mathcal{A}$ and C under the W-norm. In particular, if $\mathcal{A}$ is quadratic, *i.e,* $\mathcal{A}(\mathbf{v}) := \frac{1}{2}\mathbf{v}^T A\mathbf{v} + \mathbf{f}\mathbf{v}$, then Step (i) reduces to a linear solve, $(A + W)\mathbf{v}^{l+1} = \mathbf{f} + W\left[\mathbf{p}^l + \boldsymbol{\lambda}^l\right]$. Note that the $A + W$ matrix is constant, so it can be prefactored at the beginning of the algorithm. Finally, Step (ii) amounts to projecting the point $\mathbf{v}^{l+1} - \boldsymbol{\lambda}^l$ onto the set $\{\mathbf{p} \in \mathbb{R}^{3n},\ B\mathbf{p} + \boldsymbol{k} \in K_\mu\}$ orthogonally under the W-norm.

Back to Coulomb law. The orthogonal projection in Step (ii) can be recognized as having the same structure as Problem (6). We can thus perform the reverse transform of going from Problem (4) to Problem (6), switching back from the proxy convex law to our original Coulomb friction law. We obtain another DFCP ,

$$\begin{cases} W\mathbf{p} = W\left[\mathbf{v}^{l+1} - \boldsymbol{\lambda}^l\right] + B^T\mathbf{r} \\ \mathbf{u} = B\mathbf{p} + \mathbf{k} \\ (\mathbf{u},\mathbf{r}) \in C_\mu. \end{cases} \tag{7}$$

We can easily check that any fixed-point of our ADMM algorithm where Step (ii) is achieved by solving the DFCP (7) is indeed a solution to our initial incremental problem (4). That is, even though we used an alternative friction law to derive our algorithm, we do end up solving the right problem. Indeed, given Step (iii) any fixed point must satisfy $\mathbf{p} = \mathbf{v}$, where $\mathbf{v}$ is also a solution of Step (i), meaning $W\boldsymbol{\lambda} = \nabla\mathcal{A}$. Plugging these two equalities into the first line of Problem (7) we recognize the incremental problem (4).

5.2 Global projection

One may legitimately wonder whether we gained much by going through this process, as we got from having to solve a series of DFCP (5) to having to solve another series of DFCP (7). However, notice that the stiffness matrix of Problem (7), W, is diagonal by construction, which will enable us to solve the problem much more efficiently. In essence, we have split our problem into a series of elastic relaxation problems — Step (i) — and pure frictional contact projections — Step (ii).

Since the Delassus operator of the DFCP (7) is now easy to assemble, we could solve Step (ii) using any dual-based algorithm such as the solver from Daviet et al. [2011]. However, the restriction that we placed on our degrees of freedom, Assumption 1, allows for a much more efficient local solver, which we detail below.

5.2.1 Isotropic local problem. As classically done in dual-based methods, we can formally eliminate $\mathbf{p}$ from Problem (7) by introducing the Delassus operator (a.k.a Schur complement) $S := B^T W^{-1} B$. The local problem that must be solved for each contact within the Gauss-Seidel loop is

$$\begin{cases} \boldsymbol{u}_c^k = S_{cc}\boldsymbol{r}_c^k + \boldsymbol{u}^* \\ (\boldsymbol{u}_c^k, \boldsymbol{r}_c^k) \in C_\mu, \end{cases} \tag{8}$$

where $\mathbf{u}^*$ is the locally constant part of the relative velocity,

$$\boldsymbol{u}^* := \sum_{c\neq d} S_{cd}\boldsymbol{r}_d^{k-1} + \boldsymbol{k}_c,$$

and the 3×3 blocks S_{cd} are computed as $S_{cd} = B_{c\star}W^{-1}B_{d\star}^T$, with $B_{c\star}$ denoting the 3 rows of B corresponding to contact c. Now recall from Section 4.2 that the B matrix consists only of scaled 3×3 identity blocks, $B_{cj} = b_{c,j}\mathbb{I}_3$, and that W is diagonal, consisting again only of 3×3 identity blocks, $W_{jj} = w_j\mathbb{I}_3$. This means that the diagonal blocks of the Delassus operator follow $S_{cc} = s_c\mathbb{I}_3$ with $s_c = \sum_{j\in\mathbb{I}_c} \frac{1}{w_j} b_{c,j}^2$. In other terms, our local problem is *isotropic*.

We can also see that the constraint weights w_c being strictly positive, s_c is strictly positive as well. We can thus express $\boldsymbol{r}_c^k$ from $\boldsymbol{u}_c^k$ as $\boldsymbol{r}_c^k = \frac{1}{s_c}(\boldsymbol{u}_c^k - \boldsymbol{u}_c^*)$. Since the Signorini-Coulomb conditions are insensitive to a strictly positive scaling on the force, our local problem boils down to

$$\left(\boldsymbol{u}_c^k, \boldsymbol{u}_c^k - \boldsymbol{u}_c^*\right) \in C_\mu. \tag{9}$$

[1] Augmented Lagrangian formulations also typically feature a positive multiplier γ for the quadratic penalization term. However, we have found $\gamma \neq 1$ to not perform well in practice, so have chosen to omit it from the formulation.

5.2.2 Enumerative solver. Following Bonnefon and Daviet [2011], we can look for a solution $\boldsymbol{u}^k$ in each possible case of the Coulomb law. However, compared to [Bonnefon and Daviet 2011], the isotropy of our local problem makes this process a lot simpler.

(i) If $u^*_{\mathrm{N}} \geq 0$, then the contact is naturally separating, and $\boldsymbol{u}^k = \boldsymbol{u}^*$ is a zero-force solution of Eq (9).
(ii) If $\boldsymbol{u}^* \in -K_\mu$, then $\boldsymbol{u}^k = \mathbf{0}$ is a *sticking* solution of Eq (9).
(iii) Otherwise, $\|\boldsymbol{u}^*_{\mathrm{T}}\| > 0$, and we can directly construct a *sliding* solution by zeroing-out the normal relative displacement and projecting the tangential force onto the cone boundary, *i.e,* $u^k_{\mathrm{N}} = 0$ and $\boldsymbol{u}^k_{\mathrm{T}} = \boldsymbol{u}^*_{\mathrm{T}} + \mu u^*_{\mathrm{N}} \frac{1}{\|\boldsymbol{u}^*_{\mathrm{T}}\|} \boldsymbol{u}^*_{\mathrm{T}}$.

See Appendix B for more thorough justification. Note that in contrast to [Bonnefon and Daviet 2011], existence of solution to the local problem is guaranteed; this follows from S_{cc} being full-rank.

5.2.3 Matrix-free algorithm. Explicitly assembling the Delassus operator S would be quite memory-consuming, and would make adding new contacts during the solve expensive, forbidding regularly checking for new continuous-time collisions.

Instead, we choose to follow Erleben [2007] and incorporate our local solver into a matrix-free Gauss-Seidel algorithm. Compared to the classical variant, we continuously update the $\mathbf{p}$ vector using the newly updated force after solving each contact as $\mathbf{p} \leftarrow \mathbf{p} + W^{-1}B^T_{c\star}(\boldsymbol{r}^k - \boldsymbol{r}^{k-1})$. We can thus compute $\boldsymbol{u}^*_c$ directly from the displacement vector by subtracting the previously applied contact contribution, *i.e,* $\boldsymbol{u}^*_c = \boldsymbol{u}_c - B_{c\star}W^{-1}B^T_{c\star}\boldsymbol{r}^{k-1}_c = B_{c\star}\mathbf{p} + \mathbf{k} - S_{cc}\boldsymbol{r}^{k-1}_c$. Our matrix-free Gauss-Seidel scheme is summarized in Algorithm 1.

ALGORITHM 1: Matrix-free Gauss-Seidel projection

for $k \geq 1$ **do** // Gauss-Seidel iterations
  for $c \geq 1$ **do** each contact
    // Compute current relative velocity
    $\boldsymbol{u}^{k-1}_c \leftarrow \sum_{j \in \mathbb{J}_c} b_{c,j}\boldsymbol{p}_j + \boldsymbol{k}_c$;
    // Subtract previously applied force
    $\boldsymbol{u}^*_c \leftarrow \boldsymbol{u}^{k-1}_c - s_c\boldsymbol{r}^{k-1}_c$;
    Compute $\boldsymbol{u}^k_c$ from $\boldsymbol{u}^*_c$ by enumerating cases (Section 5.2.2);
    $\boldsymbol{r}^k_c \leftarrow \boldsymbol{r}^{k-1}_c + \frac{1}{s_c}\left(\boldsymbol{u}^k_c - \boldsymbol{u}^{k-1}_c\right)$; // Update force
    // Report DoF velocity update
    for $j \in \mathbb{J}_c$ **do**
      $\boldsymbol{p}_j \leftarrow \boldsymbol{p}_j + \frac{b_{c,j}}{w_j}\left(\boldsymbol{r}^k_c - \boldsymbol{r}^{k-1}_c\right)$;
    end
  end
end

5.2.4 Similarities with other algorithms. Algorithm 1 is actually very similar to the usual PBD contact displacement update. The main difference is that we are solving the Signorini-Coulomb conditions with $\boldsymbol{u}^*_c$, while PBD would be using $\boldsymbol{u}^{k-1}_c$ directly. Tracking and subtracting the previously applied force is actually necessary for properly enforcing the complementarity of the contact force and relative velocity variables, which PBD may violate when two contacts are "pushing" in slightly offset directions, as illustrated in Fig. 5. On the other hand, adapting an existing PBD contact solver to use our algorithm is straightforward, and the two additional operations (tracking and subtracting $\boldsymbol{r}_c$) are both cheap.

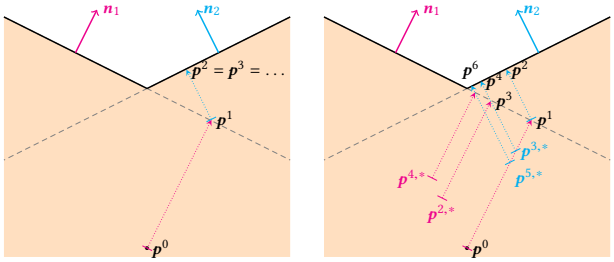


Fig. 5. Comparison of frictionless PBD contact resolution (left) with our scheme (right), for two contacts with normals $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ involving a single degree of freedom, $\boldsymbol{u}_1 := \boldsymbol{u}_2 := \boldsymbol{p}$. Starting from $\boldsymbol{p}^0$, the PBD iterations stop as soon as the two contacts are not penetrating with both $\boldsymbol{p}^k.\boldsymbol{n}_1 > 0$ and $(\boldsymbol{p}^k - \boldsymbol{p}^0).\boldsymbol{n}_1 > 0$, thus violating the orthogonality of the Signorini condition and inducing strictly positive work. In contrast, our projection scheme, which first subtracts the force previously applied by each contact, correctly converges to the orthogonal projection of $\boldsymbol{p}^0$ onto the feasible set.

Up to a change of variable we can also remark the Algorithm 1 is performing similar iterations to the PROX solver of Erleben [2017] when using their "blocked" r-Factor strategy. However, the isotropy of our S_{cc} blocks is ensuring exact enforcement of the local Signorini–Coulomb conditions after each contact solve, and our updates involve only scalar multipliers rather than 3×3 matrices. This grants efficiency to our more specialized approach.

We now have all the pieces to write our contact solver; we summarize our implementation in the next section.

6 PRACTICAL IMPLEMENTATION

6.1 Algorithm outline

Algorithm 2 outlines our contact solver applied to a linearized dynamics equation $A\mathbf{v} = \mathbf{f}$, corresponding for instance to one step of a Projected Newton integrator. The contact projection step described throughout Section 5.2 is summarized in Algorithm 3. Note that we applied a few transformations to Algorithm 1 for performance reasons, like precomputing the scaling coefficients of the DoF update and storing a scaled version of the contact force, $\bar{\boldsymbol{r}}_c := s_c\boldsymbol{r}_c$, to avoid superfluous divisions. As the DFCP that must be solved at subsequent ADMM iterations are usually quite similar, it is important to be able to warmstart the projection process; Algorithm 3 shows how this is done in our approach. This allows us to truncate the number of Gauss-Seidel iterations that are run at each ADMM iteration heavily.

Use with an ADMM implicit time integrator. While Algorithm 2 focuses on commonly used Projected-Newton-like integrators, it is not surprising that our ADMM based contact solver can just as easily be included in an ADMM integrator:

- Just like any other constraint, the weights matrix W must be added to the left-hand-side of the global step linear system, and the force $W(\mathbf{p} + \boldsymbol{\lambda})$ to the right-hand-side.
- The "local" step of our contact constraint consists in running the projection algorithm 3.

ALGORITHM 2: Contact solver for linearized dynamics

Input: Dynamics linearized as $A\mathbf{v} = \mathbf{f}$
Input: Tolerance ϵ_∞
Input: Maximum numbers of ADMM and Gauss-Seidel iterations
Warm start velocity $\mathbf{v}^0$ and forces $\boldsymbol{\lambda}^0$ (Section 6.2.1);
Do proximity and continuous-time collision detection (Section 6.2.3);
Compute diagonal approximation W of A (Section 6.2.6);
Compute linear solver preconditioner $\mathcal{P}(A + W)$ (Section 6.2.5) ;
Set $\boldsymbol{p} \leftarrow \mathbf{v}^0$;
for $l \geq 1$ **do** // Do ADMM iterations
  // Update DoF velocities
  Solve $(A + W)\mathbf{v}^l = \mathbf{f} + W(\mathbf{p} + \boldsymbol{\lambda})$ with preconditioner $\mathcal{P}$;
  Periodically update continuous-time collision detection ;
  // Update feasible projection
  $\mathbf{p} \leftarrow \mathbf{v}^l - \boldsymbol{\lambda}$;
  Project $\mathbf{p}$ onto feasible set using Algorithm 3 ;
  // Check exit criterion (Section 6.2.4)
  $\delta_\infty \leftarrow \Delta_t \left(\|\mathbf{v}^l - \mathbf{v}^{l-1}\|_\infty + \|\mathbf{v}^l - \mathbf{p}\|_\infty \right)$;
  if $\delta_\infty < \epsilon_\infty$ *or max iterations reached* **then break**;
  // Update dual forces
  $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + (\mathbf{p} - \mathbf{v}^l)$
end

ALGORITHM 3: Projection onto feasible set

```
// (Only when initializing contacts)
// Precompute scaling coefficients
for c ≥ 1 do // loop over new contacts
    s_c ← Σ_{j∈J_c} (1/w_j) b_{j,c}^2;
    for j ∈ J_c do
        γ_{c,j} ← b_{c,j} / (s_c w_j);
    end
end
// Warm-start DoF velocities from previous forces
for c ≥ 1 do // loop over all contacts
    for j ∈ J_c do
(1)     p_j ← p_j + γ_{c,j} r̄_c;
    end
end
// Run Gauss-Seidel loop
for 1 ≤ k ≤ maxGSIterations do
    for c ≥ 1 do // loop over all contacts
        // Compute current relative velocity
        u ← Σ_{j∈J_c} b_{c,j} p_j + k_c ;
        // Subtract previously applied force
        u* ← u − r̄_c ;
        // Make u* satisfy Coulomb law
        if u*_N < 0 then
            τ ← ‖u*_T‖ ;
            α ← −μ u*_N;
            u*_N ← 0 ;
            if τ ≤ α then u*_T ← 0 ;
            else u*_T ← (1 − α/τ) u*_T ;
        end
        r̄_c ← r̄_c + (u* − u) ;                    // Update force
        // Report DoF velocity update
        for j ∈ J_c do
            p_j ← p_j + γ_{c,j} (u* − u);
        end
    end
end
```

Note that this strategy is similar to the one proposed by Narain et al. [2016], but extends to self-collisions and frictional contacts.

6.2 Implementation details

6.2.1 Warmstarting velocities. Consistently with Otaduy et al. [2009], we found that warmstarting our solver with the unconstrained velocity is giving the best results: intuitively, the final solution is often close to the projection of the unconstrained velocity onto the feasible set, which is precisely what is computed at the first iteration of Algorithm 2 when starting from the unconstrained velocity. Note that in the case of a non-linear solver such as Projected Newton, this warmstarting strategy is only helpful for the first iteration; for all subsequent iterations, we use directly the last iterate of our contact solver, avoiding the cost of computing the unconstrained velocity.

6.2.2 Warmstarting forces. As the dual force vector $\boldsymbol{\lambda}$ is defined on the DoF rather than contact points, it is very easy to track across timesteps. Thus we always start our solver with $\boldsymbol{\lambda}$ from the previous timestep or non-linear iteration.

6.2.3 Collision detection. Our collision detection use both proximity and continuous-time edge-edge and point-triangle queries, accelerated by bounding volume hierarchies [Provot 1997]. Over the course of ADMM iterations, forces applied to DoF may introduce new collisions, so we need to check periodically for new continuous-time collisions — in practice we do so every 5 iterations. Newly detected collisions are then appended to the list of existing ones.

6.2.4 Stopping criterion. Thanks to the solver working directly on primal quantities (velocities or displacements), the error metric δ_∞ defined in Algorithm 2 has a direct geometric meaning, combining the displacement between successive iterates and the offset from the feasible set; in all our large-scale examples we set the tolerance ϵ_∞ to 0.01mm. As in practice this tolerance is not always be reached for the infinity-norm, we enforce a maximum number of ADMM iterations as well (usually 25 in our large-scale examples).

6.2.5 Linear solver. To solve the linear system at each ADMM iteration we follow the approach from Fei et al. [2017]. At the beginning of the algorithm we build an incomplete Cholesky preconditioner $\mathcal{P}(A+W)$ by factorizing independently the blocks corresponding to individual dynamic objects. Then, we use the Conjugate Gradient method to solve the successive linear systems. Note that if there are no soft constraints between dynamic objects, the preconditioner is exact and converges in a single iteration. If that is not the case, we truncate the convergence of the linear solver by setting its tolerance to a fraction of the current ADMM residual δ_∞.

6.2.6 Diagonal approximation. Many previous works have noted that the rate of convergence of *ADMM* severely depends on the

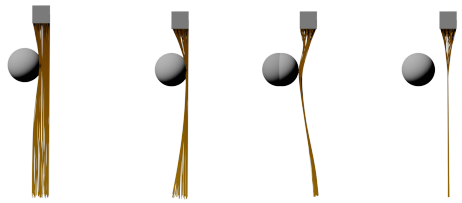


Fig. 6. An animated sphere colliding with a patch of 400 sticky elastic rods, leading to the emergence of a clumped configuration.

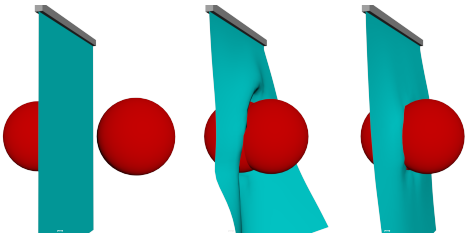


Fig. 7. A piece of cloth being pinched between two kinematic spheres. Without compliance (middle) the cloth quickly becomes unstable, while finite compliance (right) greatly dampens this effect.

chosen constraint weights [*e.g,* Narain et al. 2016; Fang et al. 2019]; it is thus important that we construct the diagonal approximation W of A wisely. Intuitively, W should reflect how hard it is to move a given DoF . In practice, for each 3D DoF j, we extract the corresponding 3×3 block from A, A_{jj}, and compute its minimum eigenvalue η_j. It would be possible to define directly w_j from η_j, however this may overestimate the DoF weight, as the full stiffness matrix may have eigenvalues much smaller than any given diagonal block (consider for instance two light 3D particles stiffly constrained to keep the same position). To overcome this effect, we take also into account the mass or inertia associated to the DoF , m_j. We then define the weight w_j using the expression $w_j := \max\left(\sigma\eta_j, \min\left(\beta m_j, \eta_j\right)\right)$. We found this heuristic to be rather robust, and used consistently $\sigma = 0.001$ and $\beta = 25$ in our examples.

6.2.7 Geometric correction. At each ADMM iteration two primal quantities are updated, $\mathbf{v}$ and $\mathbf{p}$. They should coincide upon convergence of the algorithm, but in practice will differ ever slightly. One may thus choose either $\mathbf{v}$ or $\mathbf{p}$ as the final step velocities. Using $\mathbf{v}$ will usually yield a slightly lower elastic energy (as it results from a minimization of $\mathcal{A}$), while using $\mathbf{p}$ ensures that the solution is feasible, as it is the direct result of a projection onto the contact manifold. In a sense, the latter amounts to applying a final geometric correction step. As our examples include thin, two-sided objects, we value resolving collisions more that solving elasticity at very high accuracy, so we use $\mathbf{p}$ as the final velocities.

6.2.8 Parallelization. Most of the steps in our approach can be easily parallelized, with the exception of the matrix-free Gauss–Seidel algorithm 3. To ensure deterministic results — *i.e,* independent of the thread scheduling, we must ensure that threads running concurrently to not operate on the same DoF . As it is usually impossible to split all contacts into fully independent sets, we also decompose the contact projection into several sequential stages, and use a greedy coloring algorithm to assign contacts to the different threads at each stage. We limit the number of stages to 8 and solve all remaining contacts sequentially.

6.3 Extensions

We now describe optional modifications to the solver presented in Section 6.1 that bring new features or improve performance.

6.3.1 Cohesion. Cohesion is typically modeled through the application of a constant, negative normal force $-\xi_c \boldsymbol{n}_c$ at each contact point [Raous et al. 1999; Gascón et al. 2010]. This supplemental force can easily be incorporated in our framework, as illustrated in Fig. 6. We simply need to modify the initialization of the Gauss–Seidel solver from Algorithm 3; rather than simply adding back the saved force $\bar{\mathbf{r}}_c$ from the previous iteration, we apply the cohesion force by replacing line (1) with

$$\boldsymbol{p}_j \leftarrow \boldsymbol{p}_j + \gamma_{c,j}\left(\bar{\boldsymbol{r}}_c - s_c \xi_c \boldsymbol{n}_c\right).$$

Note that this is equivalent to shifting the Signorini condition by ξ,

$$-\xi \leq r_{\mathrm{N}} \perp u_{\mathrm{N}} \geq 0.$$

6.3.2 Compliance. Real-world animations regularly contain contradicting constraints which cannot be all simultaneously resolved, causing unnatural strains to the simulated objects (Fig. 7). In some situations a slight violation of the contact constraints may be preferable to such deformations. Fortunately, we can easily mark some contacts as compliant within our solver. For each contact c that we want to make compliant, we simply append a virtual 3D slack DoF $\boldsymbol{p}_{j_c}$ to the vector $\boldsymbol{p}$, and insert a new coefficient $b_{c,j_c} = 1$ into the B matrix. Then the compliance of the contact can be chosen by tuning the w_{j_c} coefficient of the diagonal matrix W – determining the "inertia" of the virtual DoF . Setting $w_{j_c} = +\infty$ amounts to leaving the contact as non-compliant, but giving it a finite value will shift part of the contact-induced deformation to the slack DoF , the exact amount being determined by the ratio $\gamma_{c,j_c} = \frac{1}{s_c w_{j_c}}$.

On a more theoretical note, we can remark that adding a finite compliance to all contacts for which $k_{\mathrm{N}} < 0$ is sufficient — but not necessary — to ensure existence of a solution to the DFCP (this follows from the existence criterion given by Acary et al. [2011]).

6.3.3 Sleeping heuristics. Most of the time, we observe that the majority of contacts have converged well before the end of the ADMM algorithm. This means that keeping solving them at each iteration of the Gauss-Seidel projection is wasteful. To focus the computational power onto the regions that are still being actively solved, we adopt the contact-freezing heuristics from Daviet et al. [2011]; if the update $\|\mathbf{u} - \mathbf{u}^*\|$ is smaller than some threshold, we simply stop solving the contact for a fixed number of iterations. In practice we set this threshold to $0.01\delta_\infty$, where δ_∞ is the current ADMM residual, and unfreeze the contact at the beginning of Algorithm 3.

6.3.4 Accelerated algorithm. Inspired by Nesterov momentum for gradient descent, many works have proposed accelerated variants of the ADMM algorithm. We had success adapting the algorithm from

Table 1. Performance comparison with the nodal solver from Li et al. [2018] and a variant of ICA [Otaduy et al. 2009] solving nonlinear Coulomb friction. Average contact solver time per timestep (ms) for each model problem on a quad-core Intel® Core™ i5-8259U processor.

	Cards				
Solver	Belt	$\mu = 0.2$	$\mu = 0.6$	Dragging	Box & Cone
Ours	45	2.4	92	0.4	275
Nodal	28	7.6	144	1.6	292
NL-ICA	189	8.7	495	8.2	490

Goldstein et al. [2014], though the practical performance improvement remains limited. For the sake of completeness, we outline the necessary modifications in Appendix C, Algorithm 4.

7 RESULTS

7.1 Validation on model cloth problems

In order to validate the accuracy of our method, we leverage the examples devised by Li et al. [2018] that have been open-sourced alongside their *Argus* solver. To this end, we implemented Algorithm 2 as an alternative friction contact solver in their framework, with the following alterations:

- To be consistent with other solvers in Argus, we do not update collision detection during the ADMM iterations;
- Linear systems are solved using a diagonal-preconditioned Conjugate Gradient;
- As the frictional problem interface in Argus provides the stiffness matrix A but not the mass matrix M, we simply define the diagonal weighting matrix W as $w_j = 0.1\eta_j$;

We then proceed to reproduce examples from Li et al. [2018]: houses of cards with varying friction coefficients, a stretched conveyor belt, a piece of cloth dragged on a plane, and another cloth draping a box and a cone (Fig. 8; see also the accompanying video). For all those examples our results are qualitatively similar to that of Li et al. [2018]. Table 1 and Fig. 9 compare performance of our method against that of the nodal solver from Li et al. [2018], and NL-ICA, a modified version of the nested-relaxation (ICA) solver from Otaduy et al. [2009] that solves the nonlinear friction law leveraging the local solver from Daviet et al. [2011] and is also implemented in Argus. Generally, the nodal solver performs well when the contact configuration is simple, but its performance degrades in situations involving multiple layers, where it is required to duplicate many vertices — see for instance the house of card example with low friction, or the later stage of the "Box & Cone" simulation. In contrast, the performance of our algorithm is much less affected. Fig. 9 also features the original (faceted) ICA solver from Otaduy et al. [2009]. While the simulations diverge quickly due to solving different friction laws, as can be seen from the evolution of the number of contacts in time, our algorithm still stands out as competitive across the whole time range.

7.2 Large assemblies

We now switch back to our primary implementation that follows Section 6 strictly. Except for the ADMM implicit integrator example (Section 7.4), all following simulations are run using a Projected Newton integrator, with the sub-blocks in the incomplete Cholesky preconditioner being factorized leveraging Intel®MKL band-diagonal routine (for fibers) and Pardiso (for other objects). Physical and numerical parameters are summarized in Tables 4 and 5.

Hairy balls. Inspired by Kaufman et al. [2014], we evaluate the scaling of our solver using hairy balls of increasing hair density. We use a setup roughly matching the one from Kaufman et al. [2014]: human-hair-like, slightly frizzy fibers are grown on top of one hemisphere of a ball of diameter 18cm. They are discretized as Discrete Elastic Rods (DER) with 30 elements per rod and a radius of 0.037mm. The balls are then subjected to the sequence of rotations around their three axis described in [Kaufman et al. 2014]. We used balls with $16k$, $32k$, $64k$ and $127k$ rods, and ran simulations at a rate of 24 frames per second, using four timesteps per frame (*i.e*, $\Delta_t \sim 10$ms). The friction coefficient is set to 0.2. The highest-density ball contains $4.2M$ 3D DoF and induced a maximum of $5.7M$ contact points[2]; performance numbers are reported in Table 2. While our hardware and simulation setup does not exactly match that of Kaufman et al. [2014] — in our implementation the proximity and physical radii always match and the geometry of our strands may differ slightly – our method exhibits nicer scalability as the rod count increases. This follows from our method avoiding the time- and memory-intensive assembly of the Delassus operator. We were thus able to simulate a hairy ball with density and curve count comparable to that of a human head of hair while staying well within the memory budget of a modern workstation.

To further evaluate the performance impact of the local solver, we modified our matrix-free contact projection Algorithm 3 to handle non-isotropic local problems by storing full 3×3 blocks and plugging-in the local solver of Daviet et al. [2011] used by Kaufman et al. [2014]. Building on this we also implemented NL-ICA by replacing our ADMM outer loop with the nested relaxation algorithm. Table 3 compares costs to reach the 0.01mm tolerance and to perform a fixed number of 25 iterations on contact configurations obtained by initializing the solver with the geometry of 5 representative frames at various resolutions. Note that we did not succeed running complete hairy ball simulations with NL-ICA due to the solver regularly diverging. We explain the higher robustness of our method as the result of two phenomenons, NL-ICA's block-Gauss–Seidel having trouble handling the staggered DER twist DoF and the proximal penalization term providing stability to our approach.

120 bunnies. While our algorithm is mainly designed to address the challenges stemming from the simulation of thin objects, we evaluated its performance on volumetric bodies by reproducing the "120 bunnies" examples from Verschoor and Jalba [2019], which features a high DoF -to-contacts ratio. Figure 11 compares the performance of our solver with NL-ICA using a timestep of 1ms; our approach systematically required less iterations to reach the prescribed tolerance of 0.01mm. The outer surface of each bunny was discretized using 2498 faces and embedded in a non-conforming

[2]Supplemental collisions detected during the ADMM iterations are not accounted for in the reported contact numbers.

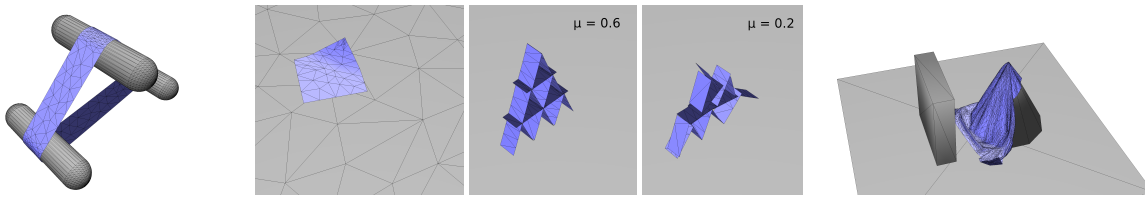


Fig. 8. Representative images from our reproduction of Li et al. [2018] model problems using our proposed contact solver. From left to right, high-tension belt, cloth drag, houses of cards with high and low friction, "box and cone". The resulting animations are consistent with [Li et al. 2018].

Table 2. Performance results for our medium and large scale simulations. Number of contacts and timings are averaged over the whole simulation, with maximum values given inside parentheses. Simulation times are given for a full 24fps frame, and are defined as follow: t_{frame} is the total elapsed time for the frame, including dynamics, collision detection, and contact resolution; t_{solver} is the total time spent inside the contact solver (Algorithm 2 excluding contact detection); $\%_{\text{linear}}$ is the percentage of time spent solving linear systems; $\%_{\text{proj}}$ is the percentage of time spent in the contact projection (Algorithm 3). "# steps" indicates the number of timesteps per 24fps frame; we used a fixed number of 2 Projected Newton iterations per substep. "Peak GB" indicates the peak memory consumption of the process, including the host application. Simulations were allocated a subset of cores of two Intel®Xeon®E5-2680v3 processors.

Scene	# Δ_t (ms)	# hairs	# DoF	# contacts	$t_{\text{frame}}(s)$	$t_{\text{solver}}(s)$	$\%_{\text{linear}}$	$\%_{\text{proj}}$	# cores	Peak GB
Hairy ball 16k	10.4	15565	498k	212k (337k)	84 (109)	39 (48)	53	15	8	7.7
Hairy ball 32k	10.4	31765	1.0M	555k (902k)	193 (271)	86 (123)	51	17	8	13.3
Hairy ball 64k	10.4	63529	2.0M	1.4M (2.3M)	393 (542)	177 (235)	56	13	16	27.0
Hairy ball 127k	10.4	127058	4.1M	3.9M (5.8M)	618 (835)	298 (339)	61	17	24	51.3
Hair band 5k	6.9	5445	257k	161k (219k)	104 (124)	56 (70)	41	32	10	5.8
Hair band 13k	6.9	13308	638k	569k (684k)	242 (285)	127 (141)	38	34	20	12.9
Hair band 27k	6.9	26615	1.3M	2.0M (2.4M)	614 (718)	331 (373)	38	34	20	26.54
Shirt only	6.9	0	27k	2026 (3228)	62 (74)	38 (45)	87	0.4	8	2.7
Long hair only	6.9	5445	120k	150k (260k)	134 (283)	37 (78)	21	55	8	11.9
Long hair+Shirt 5k	6.9	5445	147k	176k (340k)	294 (442)	116(174)	45	36	8	11.9
Long hair+Shirt 27k	6.9	27225	632k	1.14M (1.49M)	839 (1967)	216 (310)	35	34	18	20
Long hair+Shirt 54k	6.9	54450	1.2M	3.3M (4.4M)	1329 (3163)	451 (721)	31	49	24	35
Furry Bunny	6.9	29k	217k	78k (183k)	350 (550)	211 (364)	60	23	14	8.1
Palm tree	20.8	0	128k	43k (44k)	36.5 (44.9)	13 (18)	96	0	24	6

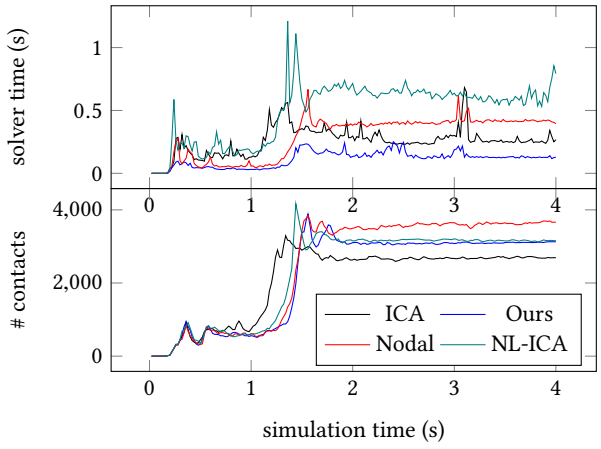


Fig. 9. Comparison of contact solver time and number of contacts (bottom) per timestep (top) for the box-and-cone simulation of Li et al. [2018] using their nodal solver, the ICA algorithm from Otaduy et al. [2009], and our approach.

Table 3. Ablation study on contact configurations extracted from 5 "hairy ball" frames at each resolution. "tol" indicates the tolerance (in mm) reached and contact projection time after 25 iterations, while "iters" indicates the number of iterations and contact projection time required to reach the prescribed 0.01mm tolerance. "No Sleep" disable contact sleeping heuristics and "Anisotropic" does not assume isotropy of local problems. Run on a 6-core Intel®Xeon®E5-1650v4.

	16k		32k		64k	
	tol	iters	tol	iters	tol	iters
Ours	0.04	31	0.02	31	0.01	29
	0.5s	0.6s	1.3s	1.7s	3.1s	3.8s
No Sleep	0.04	30	0.03	37	0.02	28
	1.2s	1.5s	3.1s	4.8s	7.7s	9.0s
Anisotropic	0.03	30	0.02	31	0.01	29
	2.8s	3.3s	7.4s	9.3s	17.4s	20.4s
NL-ICA	0.09	286	0.15	798	0.17	1578
	4s	27s	11.7s	163s	27.7s	715s

Fig. 10. A hairy ball comprising 127, 058 elastic rods going through the successive rotations described in [Kaufman et al. 2014].

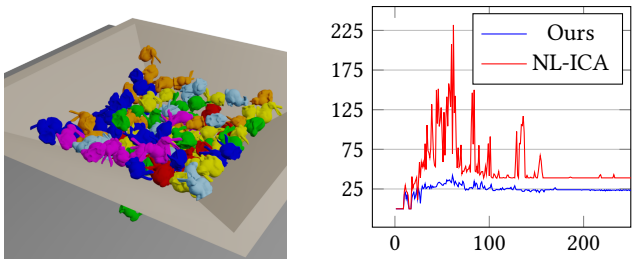


Fig. 11. Our reproduction of the "Bunnies" example from Verschoor and Jalba [2019]. Right: Per-frame average of the number of iterations required to reach the prescribed 0.01mm tolerance for our method and NL-ICA.

tet-mesh containing 2132 elements. Our simulation ran at an average of 5.5 per timestep on an Intel®Xeon®E5-1650v4 3.6GHz (16.9 seconds per timestep single-threaded) while always reaching the prescribed tolerance, 24% of this time being spent for the contact solver.

7.3 Coupled simulations

While the previous sections have shown that the performance of our method is already competitive with more specialized solvers, the main motivation behind our approach was the ability to couple together dynamical objects of different topologies; we now proceed to run such tests on medium-to-large scale scenes. For artistic convenience the friction coefficients are set per-object and the effective coefficient is computed using geometric mean.

Furry bunny. Our first test consists in equipping a soft elastic bunny with a porcupine groom comprising 29k short elastic rods, letting it fall on an inclined plane and roll through a piece of hanging cloth (Fig. 12). The bunny is embedded in a non-conforming

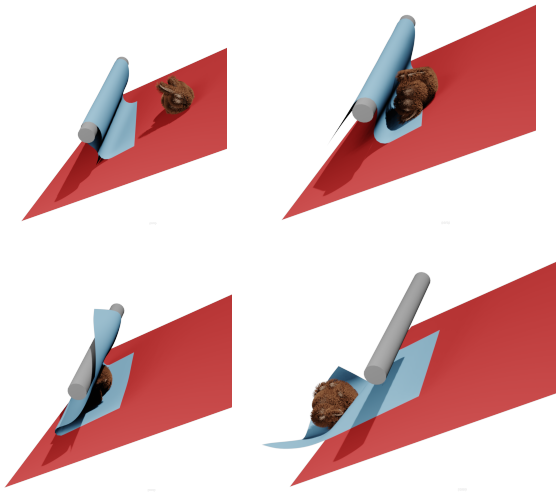


Fig. 12. An elastic bunny with 29k fur strands is impacting an inclined plane before crashing through a hanging piece of cloth, stress-testing the coupling capabilities of our contact solver.

regular tet-mesh containing $4.8k$ vertices and subject to fixed corotated elasticity [Stomakhin et al. 2012], while the cloth is using the Discrete Shell model [Grinspun et al. 2003] and is meshed with $1.7k$ vertices. The friction coefficients are set to 1.0 for the ground plane, 0.5 for the dynamical objects, and 0 for the cylinder. The high-velocity impact of the bunny on the ground, the spikiness of the strands colliding frontally with the thin layers of cloth (2mm thick), and the stiff coupling between the rods and the elastic bunny all contribute to make this scene highly challenging for frictional contact solvers.

Hair band. Out next test illustrates a typical scenario which would be challenging to make look right leveraging solely one-way coupled setups, but that can be simulated seamlessly with our method: the tightly coupled dynamics of a stretched elastic hair band holding a medium-length curly groom in place (Fig. 13). Note that the band is maintained in place solely by contact and friction with the underlying hairs and head; no additional constraints are used. The friction coefficients are set to 0.2 for the hair, and 0.5 for the elastic band and character body.

Our solver manages to robustly handle this complex contact configuration, even as the character goes through a running motion. Timings for varying densities of hair curves are reported in Table 2.

Long hair on shirt. Here we equip our character with a shirt and a long hairstyle, and make them go through the running motion once again. The friction coefficients are set once again to 0.2 for the hair, and 0.5 for the shirt and character body. To reduce wake drag from the fast motion, a narrow-band of air is also simulated around the hair [Stomakhin et al. 2020]. This running sequence is inducing a swinging motion of the hair, which repeatedly impacts the simulated shirt. Again, simulating the scene with staggered one-way coupled

Fig. 13. A stretched elastic band is maintaining the $27k$ strands of the character's hairstyle in place, thanks to frictional contacts. © Weta Digital.

Fig. 14. Leaves of a palm tree colliding under a gentle breeze. © Weta Digital.

Fig. 15. Efficiently relaxing a wavy hairstyle comprising over 5M vertices with frictional contacts using the ADMM implicit integrator. © Weta Digital.

solves would not be straightforward; the full mass of hair is heavier than the shirt, but individual strands are much lighter.

We also ran this scene with varying numbers of hair strands, and in order to estimate the cost of coupling, with either just the hair or just the shirt. From the statistics reported on Table 2, we notice that this scene is harder on the solver; for a similar problem size, the solve times are significantly higher than for the hairy balls, and proportionally a larger amount of time is spent inside the contact projection step. Indeed, the long, straight hairstyle is prone to create packed strand assemblies, and as a result a very high number of contacts. The simulation thus suffers from a scarcity of DoF as can be evaluated using the criterion from Bertails-Descoubes et al. [2011] based on the ratio $\frac{\mu n}{m}$. Intuitively, this means a denser Delassus operator. We also notice that running the simulation with both hair and shirt is slightly more expensive than running the two elements separately; this likely follows from the supplemental contacts induced by collisions between hair and shirt.

7.4 Extensions

Cohesion. The simulation depicted in Fig. 6 involves a patch of 401 30cm-long hairs being clumped tightly under the effect of cohesion, and sticking to an animated ball collider. In this example we used a simple cohesion model (linear fall-off of the attractive force between neighbouring primitives), but more involved ones [Fei et al. 2017, 2019] could be used as well.

Frame-based models. We demonstrate that our algorithm can also be applied to frame-based models [Martin et al. 2010; Faure et al. 2011] by discretizing a palm tree using a one-dimensional wireframe for the trunk and two-dimensional surfaces for the leaves, both using Linear Blend Skinning shape functions and co-rotated elasticity. The motion of this tree under a fresh breeze that induces many leaf-leaf and leaf-branch collisions is depicted in Fig. 14.

ADMM implicit time integrator. Last, we show how the ADMM implicit integrator can be combined with our contact solver to provide an efficient way of relaxing a wavy hairstyle under gravity (Fig. 15). Our naive grooming attempt consists in starting from a porcupine initial geometry, first applying a backwards acceleration, then switching back to normal gravity. The 53507 strands with 100 vertices each are simulated as Discrete Elastic Rods with quasistatic twist relaxation, leading to cheap, decoupled tridiagonal linear systems having to be solved at each global step of the algorithm. Due to the slow relative velocities, collision detection is performed using proximity queries only. This approach is very memory-efficient, with our simulation only requiring $15GB$ despite containing $5.4M$ DoF and over $11.9M$ contacts in the later stages. The total runtime on a 6-core Intel®Xeon®E5-1650v4 was 9.6 hours for 200 frames, averaging to $174s$ per frame.

8 DISCUSSION

8.1 Limitations and future work

Assumption on discretization. By design and through Assumption 1, we limited our approach to specific kinds of DoF . While our results show that we are still able to simulate a large class of objects, our method is not applicable to rigid bodies, for instance; the closest approximation in our framework would be to use a 12-DoF elaston equipped with a stiff elastic material. Actually, extending our method to handle arbitrary DoF would be relatively straightforward; the use of ADMM to decompose the incremental problem into an elastic relaxation part and a simpler contact projection would still be valid, and we could use more general-purpose approaches [*e.g.*, Daviet et al. 2011; Erleben 2017] to solve the DCFP with diagonal

Table 4. Summary of physical parameters used in our simulations, abbreviated as follow: D: thickness/diameter (in mm); ρ: volumetric mass in g.cm^{-3}; E_b and E_s: bending and stretching moduli (in MPa); ν: Poisson ratio; μ: friction coefficient; ξ: elastic damping (in ms). We use the Discrete Elastic Rods and Discrete Shells models for 1D and 2D objects, and either co-rotated or fixed co-rotated elasticity models for volumetric bodies.

Object	D	ρ	E_b	E_s	ν	μ	ξ
Hairy ball	0.07	1.3	4000	100	0.48	0.2	1
Long hair	0.08	1.3	4000	100	0.48	0.2	1
Shirt	2	1	0.001	1	0.4	0.5	5
Curly hair	0.08	1.3	4000	100	0.48	0.2	1
Hair band	5	0.4	0.001	1	0.4	0.5	5
Bunny, fur	0.05	1.3	400	100	0.48	0.2	1
Bunny, cloth	2	1	0.05	1	0.4	0.5	5
Bunny, body	-	1	0.025		0.2	1	0
Palm Tree	-	0.2	0.25		0.3	0.5	10
120 Bunnies	-	1	5		0.4	0.5	0

Table 5. Summary of non-physical parameters used in our simulations, abbreviated as follow: N_{PN}: fixed number of Projected Newton iterations; N_{ADMM}: maximum number of ADMM iterations; N_{GS}: fixed number of Gauss–Seidel iterations; ϵ_∞: infinity-norm ADMM tolerance in mm; τ_{LS}: linear solver tolerance as ratio of δ_∞ when the preconditioner is inexact; β, σ : parameters for building our diagonal approximation.

Newton	N_{PN}	N_{ADMM}	N_{GS}	ϵ_∞	τ_{LS}	β	σ
Hairy ball	2	25	5	0.01	-	25	0.001
Hair & Shirt	2	25	10	0.01	-	25	0.001
Hair & Band	2	25	10	0.01	-	25	0.001
Furry Bunny	2	50	25	0.01	10^{-4}	25	0.001
Palm Tree	2	25	5	0.01	-	25	0.001
120 Bunnies	1	250	10	0.01	-	25	0.001

stiffness matrix. However, that would imply a significantly higher per-iteration cost and implementation complexity, which is antagonistic to our initial goals.

Alternative friction laws. Our solver is designed to be very efficient at solving the isotropic Signorini–Coulomb conditions; it is unclear whether it could be easily adapted to solve similar contact laws, such as anisotropic Coulomb friction [Erleben et al. 2019]. In the future we would like to explore a broader variety of contact laws, and examine whether we could derive appropriate local solvers.

Choice of diagonal approximation. As noted by several works from the literature [*e.g,* Narain et al. 2016; Fang et al. 2019], the choice of weights for the constraint in the Augmented Lagrangian formulation has a severe impact on the convergence behaviour and resulting performance of the method. The heuristic that we propose in Section 6.2.6, based on the DoF inertia and the minimum eigenvalue of associated diagonal block is not particularly elegant and can seem rather arbitrary at first; Fig. 16 shows that the choice of β, the relative inertia weight, does affect the convergence, though

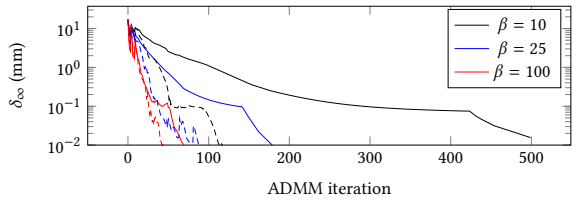


Fig. 16. Number of iterations required to reach our prescribed tolerance $\delta_\infty = 0.01$mm for various choices of ADMM constraint weight, on a problem extracted from a timestep of Fig. 2. Dashed lines correspond to the accelerated variant of the algorithm (Appendix C).

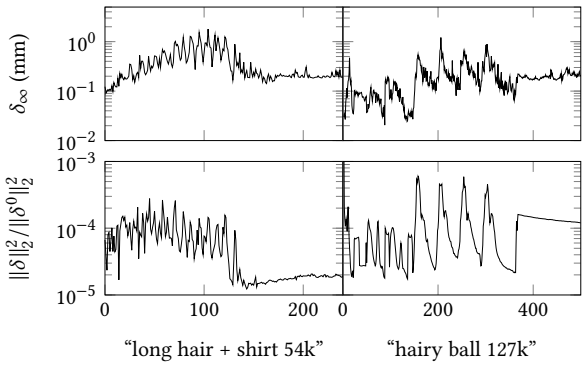


Fig. 17. Infinity-norm error (top) and ratio of final to initial squared residual ℓ_2 norm averaged over each frame of our two largest simulations.

using the accelerated variant reduces this effect. However, in practice we found using $\beta = 25$ to be rather robust, and did not have to tune this parameter for any of the examples presented above, despite the variety in size, shape and topology of our dynamical objects. Still, deriving the choice of constraints weights from a more grounded analysis would be an interesting future line of research.

Non-physical parameters. One of our initial goals was to limit the number of non-physical parameters that would need to be tuned by an artist to produce a simulation as much as possible; we list the values chosen for our examples in Table 5. Aside from the constraint weight discussed above, our method requires defining a maximum number of iterations for the Gauss–Seidel and ADMM loops, and optionally a tolerance for the iterative linear solver. As shown in Fig. 17, for our largest simulations the prescribed infinity-norm tolerance is infrequently reached, and the solver terminates due to reaching the maximum number of ADMM iterations. In practice we rarely adjusted this number, doubling it solely for the "Furry Bunny" example. We also note that when not reaching the infinity-norm tolerance still manages to significantly lower the error residual, as is exhibited by the decrease in ℓ_2 norm; deriving a more perceptual exit criterion, less sensitive to problematic contact configurations, would help reduce the effect of the maximum number of iterations.

Performance. ADMM being a first-order method, our algorithm needs to run for many iterations before yielding satisfying results. Despite being competitive with existing approaches, the runtime of our method on full-resolution scenes can thus still verge on the side of intractability. However, the ability of our scheme to two-way

couple various dynamical elements should help reduce the number of iterations required to simulate a complex animation sequence. We would also like to explore adapting our algorithm to massively parallel architectures. The main difficulty of such an endeavour would be the Gauss–Seidel contact projection, and it would be interesting to assess whether ideas from the literature [*e.g,* Tonge et al. 2012; Fratarcangeli and Pellacini 2015] could be applicable.

8.2 Conclusion

Leveraging a loose assumption on the degrees of freedom of dynamical objects, we presented a novel approach for solving nonlinear dynamics of elastic bodies with frictional contact and exact Coulomb friction. While simple to implement, our approach derives from a sound theoretical basis; it was shown to scale to millions of contacts and degrees of freedom and allows seamless coupling of fiber assemblies, shells, and volumetric bodies, unlocking the simulation of virtual characters and their environment with an unprecedented amount of details.

ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their insightful comments; Andrew Moffat, Carlos Lin, Nicholas Wilson, Nick Grace, Vincent Bonnet for their help devising relevant examples; Marie-Lena Eckert for her many valuable suggestions; the Weta Digital Simulation group for the fruitful discussions and for building the platform underpinning this contribution, as well as Argus' authors for making their code and examples publicly available.

REFERENCES

V. Acary, F. Cadoux, C. Lemaréchal, and J. Malick. 2011. A formulation of the linear discrete Coulomb friction problem via convex optimization. *ZAMM — J. of App. Math. and Mech.* 91, 2 (2011).

P. Alart and A. Curnier. 1991. A Mixed Formulation for Frictional Contact Problems Prone to Newton like Solution Methods. *Comput. Methods Appl. Mech. Eng.* 92, 3 (1991).

D. Baraff. 1989. Analytical Methods for Dynamic Simulation of Non-Penetrating Rigid Bodies. In *Proc. of ACM SIGGRAPH (SIGGRAPH '89)*. ACM.

D. Baraff. 1994. Fast Contact Force Computation for Nonpenetrating Rigid Bodies. In *Proc. of ACM SIGGRAPH (SIGGRAPH '94)*. ACM.

D. Baraff and A. Witkin. 1998. Large Steps in Cloth Simulation. In *Proc. of ACM SIGGRAPH (SIGGRAPH '98)*. ACM.

M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, and E. Grinspun. 2010. Discrete Viscous Threads. *ACM Trans. Graph.* 29, 4 (2010).

M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. 2008. Discrete Elastic Rods. *ACM Trans. Graph.* 27, 3 (2008).

F. Bertails, B. Audoly, M-P. Cani, B. Querleux, F. Leroy, and J-L. Lévque. 2006. Super-Helices for Predicting the Dynamics of Natural Hair. *ACM Trans. Graph.* 25, 3 (2006).

F. Bertails-Descoubes, F. Cadoux, G. Daviet, and V. Acary. 2011. A Nonsmooth Newton Solver for Capturing Exact Coulomb Friction in Fiber Assemblies. *ACM Trans. Graph.* 30, 1 (2011).

O. Bonnefon and G. Daviet. 2011. *Quartic formulation of Coulomb 3D frictional contact.* Technical Report RT-0400. INRIA.

R. Bridson, R. Fedkiw, and J. Anderson. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. In *ACM Trans. Graph. (SIGGRAPH '02)*. ACM.

E. Catto. 2005. Iterative dynamics with temporal coherence. *GDC '05* (2005).

P. A. Cundall and O. D. L. Strack. 1979. A discrete numerical model for granular assemblies. *Géotechnique* 29, 1 (1979).

G. Daviet. 2016. *Modeling and simulating complex materials subject to frictional contact : application to fibrous and granular media.* Ph.D. Dissertation.

G. Daviet, F. Bertails-Descoubes, and L. Boissieux. 2011. A Hybrid Iterative Solver for Robustly Capturing Coulomb Friction in Hair Dynamics. *ACM Trans. Graph.* 30, 6 (2011).

C. Duriez, C. Andriot, and A. Kheddar. 2004. Signorini's contact model for deformable objects in haptic simulations. In *Int. Conf. on Intelligent Robots and Systems*, Vol. 4.

K. Erleben. 2007. Velocity-Based Shock Propagation for Multibody Dynamics Animation. *ACM Trans. Graph.* 26, 2 (2007).

K. Erleben. 2017. Rigid Body Contact Problems Using Proximal Operators. In *Proc. of Symp. Comp. Anim. (SCA '17)*. ACM.

K. Erleben, M. Macklin, .S Andrews, and P.G. Kry. 2019. The Matchstick model for anisotropic friction cones. In *Computer Graphics Forum*. Wiley.

Y. Fang, M. Li, M. Gao, and C. Jiang. 2019. Silly Rubber: An Implicit Material Point Method for Simulating Non-Equilibrated Viscoelastic and Elastoplastic Solids. *ACM Trans. Graph.* 38, 4 (2019).

F. Faure, B. Gilles, G. Bousquet, and D.K. Pai. 2011. Sparse Meshless Models of Complex Deformable Solids. *ACM Trans. Graph.* 30, 4 (2011).

Y. Fei, H.T. Maia, C. Batty, C. Zheng, and E. Grinspun. 2017. A Multi-scale Model for Simulating Liquid-hair Interactions. *ACM Trans. Graph.* 36, 4 (2017).

Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2019. A Multi-Scale Model for Coupling Strands with Shear-Dependent Liquid. *ACM Trans. Graph.* 38, 6 (2019).

M. Fortin and R. Glowinski. 1983. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems.* Elsevier Science Ltd.

M. Fratarcangeli and F. Pellacini. 2015. Scalable partitioning for parallel position based dynamics. In *Computer Graphics Forum*, Vol. 34. Wiley.

M. Fukushima, Z-Q. Luo, and P. Tseng. 2002. Smoothing Functions for Second-Order-Cone Complementarity Problems. *SIAM J. on Optimization* 12, 2 (2002).

J. Gascón, J.S. Zurdo, and M.A. Otaduy. 2010. Constraint-Based Simulation of Adhesive Contact. In *Proc. of Symp. Comp. Anim. (SCA '10)*. Eurographics Association.

T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. 2014. Fast Alternating Direction Optimization Methods. *SIAM J. on Imaging Sciences* 7, 3 (2014).

G. Gornowicz and S. Borac. 2015. Efficient and Stable Approach to Elasticity and Collisions for Hair Animation. In *Symp. on Digital Production (DigiPro '15)*. ACM.

E. Grinspun, A.N. Hirani, M. Desbrun, and P. Schröder. 2003. Discrete Shells. In *Proc. of Symp. Comp. Anim. (SCA '03)*. Eurographics Association.

S. Hadap and N. Magnenat-Thalmann. 2001. Modeling Dynamic Hair as a Continuum. *Computer Graphics Forum* 20, 3 (2001).

X. Han, T.F. Gast, Q. Guo, S. Wang, C. Jiang, and J. Teran. 2019. A Hybrid Material Point Method for Frictional Contact with Diverse Materials. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2 (2019).

D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun. 2008. Robust Treatment of Simultaneous Collisions. *ACM Trans. Graph.* 27, 3 (2008).

T. Heyn. 2013. *On the modeling, simulation, and visualization of many-body dynamics problems with friction and contact.* Ph.D. Dissertation.

J-B. Hiriart-Urruty and C. Lemaréchal. 1993. *Convex Analysis and Minimization Algorithms.* Springer.

T. Inglis, M-L. Eckert, J. Gregson, and N. Thuerey. 2017. Primal-Dual Optimization for Fluids. In *Computer Graphics Forum*, Vol. 36. Wiley.

M. Jean. 1999. The non-smooth contact dynamics method. *Comp. Meth. Appl. Mech. Engng.* 177, 3 (1999).

M. Jean and J.J. Moreau. 1988. Dynamics in the presence of unilateral contacts and dry friction : a numerical approach. In *Second Meeting on Unilateral Problems in Structural Analysis (Unilateral Problems in Structural Analysis, 2)*. Springer.

M. Jean and J.J. Moreau. 1992. Unilaterality and dry friction in the dynamics of rigid body collections. In *1st Contact Mechanics International Symposium.*

C. Jiang, T. Gast, and J. Teran. 2017. Anisotropic Elastoplasticity for Cloth, Knit and Hair Frictional Contact. *ACM Trans. Graph.* 36, 4 (2017).

F. Jourdan, P. Alart, and M. Jean. 1998. A Gauss-Seidel like algorithm to solve frictional contact problems. *Comp. Meth. Appl. Mech. Engng.* 155, 1 (1998).

D.M. Kaufman, S. Sueda, D.L. James, and D.K. Pai. 2008. Staggered Projections for Frictional Contact in Multibody Systems. *ACM Trans. Graph.* 27, 5 (2008).

D.M. Kaufman, R. Tamstorf, B. Smith, J-M. Aubry, and E. Grinspun. 2014. Adaptive Nonlinearity for Collisions in Complex Rod Assemblies. *ACM Trans. Graph.* 33, 4 (2014).

J. Li, G. Daviet, R. Narain, F. Bertails-Descoubes, M. Overby, G.E. Brown, and L. Boissieux. 2018. An Implicit Frictional Contact Solver for Adaptive Cloth Simulation. *ACM Trans. Graph.* 37, 4 (2018).

M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and V. Makoviychuk. 2019. Non-Smooth Newton Methods for Deformable Multi-Body Dynamics. *ACM Trans. Graph.* 38, 5 (2019).

M. Macklin, M. Müller, and N. Chentanez. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. In *Motion in Games (MIG '16)*. ACM.

S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, and M. Gross. 2010. Unified Simulation of Elastic Rods, Shells, and Solids. *ACM Trans. Graph.* 29, 4 (2010).

H. Mazhar, T. Heyn, D. Negrut, and A. Tasora. 2015. Using Nesterov's Method to Accelerate Multibody Dynamics with Friction and Contact. *ACM Trans. Graph.* 34, 3 (2015).

A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran. 2009. Detail Preserving Continuum Simulation of Straight Hair. *ACM Trans. Graph.* 28, 3 (2009).

D.L. Michels, V.T. Luan, and M. Tokman. 2017. A Stiffly Accurate Integrator for Elastodynamic Problems. *ACM Trans. Graph.* 36, 4 (2017).

M. Moore and J. Wilhelms. 1988. Collision Detection and Response for Computer Animation. In *Proc. of ACM SIGGRAPH (SIGGRAPH '88)*. ACM.
M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. 2007. Position based dynamics. *J. of Visual Communication and Image Representation* 18, 2 (2007).
R. Narain, M. Overby, and G.E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proc. of Symp. Comp. Anim. (SCA '16)*. Eurographics Association.
M.A. Otaduy, R. Tamstorf, D. Steinemann, and M. Gross. 2009. Implicit Contact Handling for Deformable Objects. *Computer Graphics Forum* 28, 2 (2009).
X. Provot. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation '97*, Daniel Thalmann and Michiel van de Panne (Eds.). Springer Vienna.
M. Raous, L. Cangémi, and M. Cocu. 1999. A consistent model coupling adhesion, friction, and unilateral contact. *Comp. Meth. Appl. Mech. Engng.* 177, 3-4 (1999).
M. Renouf and P. Alart. 2005. Conjugate gradient type algorithms for frictional multi-contact problems: applications to granular materials. *Comp. Meth. Appl. Mech. Engng.* 194, 18 (2005).
D.E. Stewart and J.C. Trinkle. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *Int. J. for Num. Meth. in Engng.* 39, 15 (1996).
A. Stomakhin, R. Howes, C. Schroeder, and J.M. Teran. 2012. Energetically Consistent Invertible Elasticity. In *Proc. of Symp. Comp. Anim. (EUROSCA'12)*. Eurographics Association.
A. Stomakhin, J. Wretborn, K. Blom, and G. Daviet. 2020. Underwater bubbles and coupling. In *ACM SIGGRAPH 2020 Talks (SIGGRAPH '20)*.
M. Tang, D. Manocha, M.A. Otaduy, and R. Tong. 2012. Continuous penalty forces. *ACM Trans. Graph.* 31, 4 (2012).
A. Tasora. 2013. Efficient simulation of contacts friction and constraints using a modified spectral projected gradient method. In *roceedings of WSCG 2013*.
J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proc. of Symp. Comp. Anim.*
R. Tonge, F. Benevolenski, and A. Voroshilov. 2012. Mass Splitting for Jitter-Free Parallel Rigid Body Simulation. *ACM Trans. Graph.* 31, 4 (2012).
M. Verschoor and A.C. Jalba. 2019. Efficient and Accurate Collision Response for Elastically Deformable Models. *ACM Trans. Graph.* 38, 2 (2019).
K. Yamane and Y. Nakamura. 2006. Stable penalty-based model of frictional contacts. In *ICRA'06*. IEEE.
Y. Yue, B. Smith, P.Y. Chen, M. Chantharayukhonthorn, K. Kamrin, and E. Grinspun. 2018. Hybrid Grains: Adaptive Coupling of Discrete and Continuum Simulations of Granular Media. *ACM Trans. Graph.* 37, 6 (2018).

A CONVEX OPTIMALITY CONDITIONS

The *normal cone* to a convex set $C \subset \mathbb{R}^n$ at $\boldsymbol{x}$ is defined as the closed convex cone $\mathcal{N}_C(\boldsymbol{x}) := \{\boldsymbol{z} \in \mathbb{R}^n, (\boldsymbol{z}-\boldsymbol{x}).\boldsymbol{y} <= 0 \quad \forall \boldsymbol{y} \in C\}$. The following equivalence [Hiriart-Urruty and Lemaréchal 1993, Theorem VII.1.1.1] provides a convenient way of expressing the optimality conditions of a convex optimization problem:

Theorem A.1. *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be differentiable and convex, then*

$$\tilde{\boldsymbol{x}} = \min_{\boldsymbol{x} \in C} f(\boldsymbol{x}) \iff \nabla f(\tilde{\boldsymbol{x}}) \in -\mathcal{N}_C(\tilde{\boldsymbol{x}}).$$

We also note the following expressions of remarkable normal cones (see *e.g*, [Daviet 2016, Corollary A.3 and Theorem A.4]);

Property 1 (Normal cone to the precomposition by an affine map). *Let $\mathcal{B} : \mathbb{R}^m \mapsto \mathbb{R}^n$, $\boldsymbol{v} \to B\boldsymbol{v} + \boldsymbol{k}$ be an affine map, and let $V := \{\boldsymbol{v}, \mathcal{B}(\boldsymbol{v}) \in C\}$. Then there always holds the inclusion*

$$B^T \mathcal{N}_C(\mathcal{B}(\boldsymbol{v})) \subset \mathcal{N}_V(\boldsymbol{v}),$$

and the equality is achieved under the sufficient condition that the interior of V is non-empty.

Property 2 (Normal cone to the second-order cone K_μ).

$$\boldsymbol{y} \in -\mathcal{N}_{K_\mu}(\boldsymbol{x}) \iff \boldsymbol{x} \ni K_\mu \perp \boldsymbol{y} \in K_{\frac{1}{\mu}} \iff \boldsymbol{x} \in -\mathcal{N}_{K_{\frac{1}{\mu}}}(\boldsymbol{y});$$

Our result follows from combining Properties 2 and 1:

Theorem A.2. *Let $f : \mathbb{R}^m \mapsto \mathbb{R}$ be a differentiable convex function, $\mathcal{B} : \mathbb{R}^m \mapsto \mathbb{R}^n$, $\boldsymbol{v} \to B\boldsymbol{v} + \boldsymbol{k}$ an affine map, and $C := \left\{\boldsymbol{v} \in \mathbb{R}^m, \mathcal{B}(\boldsymbol{v}) \in K_{\frac{1}{\mu}}\right\}$. Then the existence of $\boldsymbol{r} \in \mathbb{R}^m$ such that*

$$\begin{cases} \nabla f(\boldsymbol{v}) = B^T \boldsymbol{r} \\ K_{\frac{1}{\mu}} \ni \mathcal{B}(\boldsymbol{v}) \perp \boldsymbol{r} \in K_\mu \end{cases}$$

implies that $\boldsymbol{v}$ realizes the minimum of f over C, and the reverse implication is granted if the interior of C is non-empty.

B ISOTROPIC LOCAL SOLVER

Let study each case of our proposed local solver and show that all satisfy the Signorini–Coulomb conditions (1–2).

(i) In the separating case, we set $\boldsymbol{u} = \boldsymbol{u}^*$ and $\boldsymbol{r} = \boldsymbol{u} - \boldsymbol{u}^* = \boldsymbol{0}$. Since $u_N \geq 0$ and $r_N = 0$, the Signorini condition is satisfied, and $\boldsymbol{r}_T = \boldsymbol{0}$ always satisfy the Coulomb condition.
(ii) In the sticking case, we set $\boldsymbol{u} = 0$ and thus $\boldsymbol{r} = -\boldsymbol{u}^* \in K_\mu$. Both conditions are again satisfied.
(iii) In the last case, we know that ${u^*}_N < 0$ and $\|\boldsymbol{u^*}_T\| \neq \boldsymbol{0}$. We set $u_N = 0$, thus $r_N = u_N - {u^*}_N \geq 0$, and the Signorini condition is satisfied. Moreover $\boldsymbol{r}_T = \mu \frac{{u^*}_N}{\|\boldsymbol{u^*}_T\|} \boldsymbol{u^*}_T = -\mu r_N \frac{\boldsymbol{u^*}_T}{\|\boldsymbol{u^*}_T\|}$, which also satisfies the Coulomb condition.

C ACCELERATED ADMM ALGORITHM

Algorithm 4 outlines how to modify Algorithm 2 to implement Nesterov acceleration as suggested by Goldstein et al. [2014].

ALGORITHM 4: Accelerated ADMM iterations

$\theta^0 \leftarrow 1$; $\hat{\mathbf{p}} \leftarrow \mathbf{p}^0$; $\hat{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\lambda}^0$;
for $l \geq 1$ **do** `// Do ADMM iterations`
  `// Update DoF velocities`
  Solve $(A+W)\mathbf{v}^l = \mathbf{f} + W\left(\hat{\mathbf{p}} + \hat{\boldsymbol{\lambda}}\right)$;
  `// . . .`
  `// Update feasible projection`
  $\mathbf{p}^l \leftarrow \mathbf{v}^l - \hat{\boldsymbol{\lambda}}$;
  Project $\mathbf{p}^l$ onto feasible set using Algorithm 3 ;
  `// . . .`
  $\boldsymbol{\lambda}^l \leftarrow \hat{\boldsymbol{\lambda}} + (\mathbf{p} - \mathbf{v}^l)$; `// Update dual forces`
  if *residual is increasing* **then**
    $\theta^l = 1$; $\hat{\mathbf{p}} \leftarrow \mathbf{p}^l$; $\hat{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\lambda}^l$; `// Reset acceleration`
  else
    `// Add momentum`
    $\theta^l \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4\left(\theta^{l-1}\right)^2}\right)$;
    $\hat{\mathbf{p}} \leftarrow \mathbf{p}^l + \frac{\theta^{l-1}-1}{\theta^l}\left(\mathbf{p}^l - \mathbf{p}^{l-1}\right)$;
    $\hat{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\lambda}^l + \frac{\theta^{l-1}-1}{\theta^l}\left(\boldsymbol{\lambda}^l - \boldsymbol{\lambda}^{l-1}\right)$;
  end
end

Received January 2020; accepted April 2020