Received: 28 December 2020 | Revised: 25 March 2021 | Accepted: 4 April 2021

DOI: 10.1002/nme.6693

RESEARCH ARTICLE

WILEY

Solving variational inequalities and cone complementarity problems in nonsmooth dynamics using the alternating direction method of multipliers

Alessandro Tasora | **Dario Mangoni** | **Simone Benatti** | **Rinaldo Garziera**

Department of Engineering and Architecture, University of Parma, Parma, Italy

Correspondence
Alessandro Tasora, Department of Engineering and Architecture, University of Parma, Parco Area delle Scienze, 181/A, 43124 Parma, Italy.
Email: alessandro.tasora@unipr.it

Abstract

This work presents a numerical method for the solution of variational inequalities arising in nonsmooth flexible multibody problems that involve set-valued forces. For the special case of hard frictional contacts, the method solves a second order cone complementarity problem. We ground our algorithm on the Alternating Direction Method of Multipliers (ADMM), an efficient and robust optimization method that draws on few computational primitives. In order to improve computational performance, we reformulated the original ADMM scheme in order to exploit the sparsity of constraint jacobians and we added optimizations such as warm starting and adaptive step scaling. The proposed method can be used in scenarios that pose major difficulties to other methods available in literature for complementarity in contact dynamics, namely when using very stiff finite elements and when simulating articulated mechanisms with odd mass ratios. The method can have applications in the fields of robotics, vehicle dynamics, virtual reality, and multiphysics simulation in general.

KEYWORDS

ADMM, contact, friction, nonsmooth dynamics, simulation

1 | INTRODUCTION

Problems involving frictional contacts can be found in many engineering fields. Their simulation is a challenging task because the discontinuous nature of contact phenomena discourages the adoption of conventional time integration schemes, which assume continuous velocities and accelerations. In fact, if on the one hand it is possible to turn a discontinuous model into a smooth Ordinary Differential Equation by approximating contact forces via spring-damper penalty forces, on the other this usually requires the adoption of extremely small time steps.

In search of better numerical performance and stability, even in case of large time steps, we rather formulate the original nonsmooth dynamical problem as a Measure Differential Inclusion (MDI). MDIs were pioneered by Moreau in the eighties, among others, aiming at a viable approach to the simulation of mechanical systems with hard contacts.[1,2] Being differential inclusions, they can directly embed set-valued force laws, such as the Coulomb friction model, but

they also generalize to the case where velocity is assumed to be a (possibly) discontinuous function of bounded variation, therefore they admit impulsive events.[3]

MDIs can be solved using special time stepping methods that offer high stability and robustness, however this comes at the cost of solving a Complementarity Problem (CP) per each time step.[4] These CPs correspond to the broader class of Variational Inequalities (VI).[5,6]

In general, the solution of VIs or CPs represents the major numerical bottleneck in the entire time stepping process of the MDI, especially when dealing with many parts or many contacts. For instance, similar scenarios can happen when simulating packaging devices, ground–machine interaction, rock dynamics, masonry structures, and granular materials, where the unknowns of the VIs (contact forces) can amount to thousands or millions. This challenge stimulated research on efficient numerical methods in the last three decades.

Among the former strategies for solving this class of problems, we cite the idea of solving a Linear Complementarity Problem (LCP) per each time step.[7] Once cast as a LCP, the VI problem can be solved by direct LCP solvers such as the Lemke algorithm. However, LCP direct solvers often lead to complex subiterations that do not scale well with increasing number of unknowns, hence they are not much used nowadays. Another drawback of casting the problem as a LCP is that they introduce a polyhedral approximation, for example the Coulomb friction cone would be approximated as a faceted pyramid, introducing a numerical source of anisotropy.

At the other end of the spectrum are iterative methods that aim at high robustness and efficiency even at the expense of sacrificing accuracy, since this is often acceptable in fields like real-time simulation, video games and virtual reality.[8] In these fields, the most used approach nowadays consists in fixed point iterations, similarly to Gauss-Seidel, SOR or Jacobi stationary methods, interleaved with simple projections onto the constraint sets at each iteration.[9-11] The major issue of these methods is that they exhibit slow convergence when the system contains long kinematic chains or long sequences of contacts, such as when including high stacks of objects.[12] Their convergence is even more challenged if odd mass ratios are encountered: this often happens when simulating industrial robots, earth-moving machines, complex vehicle suspensions, car drivelines. In these cases the convergence might stall soon, hence if the iterations are truncated in order to meet a time budget in real-time applications, the resulting simulation can be affected by low accuracy—usually detectable in the form of objects that interpenetrate at the point of contact and mechanisms that fall apart. Optimizations like warm starting and substepping can alleviate the drawbacks of these methods.[13]

A more advanced class of solvers is represented by non-smooth Newton methods, such as the one presented in Reference 14, that assume a generic Nonlinear Complementarity Problem (NCP), again a subcase of a VI, and that enforce the NCP complementarity constraints using nonsmooth functions like the Fisher–Burmeister function. In these methods, a generalized nonsmooth Newton method can be used to find the zero of the functions, but this comes at the cost of solving a linear system per each iteration.

Recent efforts pushed forward the idea of casting the VI of nonsmooth dynamics as a convex optimization problem, namely a Quadratic Program (QP) with convex constraints. This requires a simplifying assumption about the Coulomb friction being associative instead of nonassociative. This artificial associativity of the friction model means that the relative sliding velocity in a contact point will be restricted to the dual cone of the Coulomb friction cone, whereas the original model has a more relaxed condition, assuming that such velocity could be even parallel to the surface: the practical consequence is that grazing motion cause an artificial lift-off—however, this artifact can be attenuated thanks to the introduction of a stabilization term.[15] In this setting, the problem is also a convex Cone Complementarity Problem (CCP). The convexity assumption allows the adoption of numerical methods already available for large-scale optimization problems. In Reference 16, the Barzilai–Borwein spectral projected gradient is used to this end, and in Reference 17 we used the Nesterov accelerated projected gradient descend to simulate large-scale contact problems. These are first-order optimization methods, in the sense that they do not require the computation of the Hessian matrix: in fact they rely only on matrix-by-vector multiplications and inner products (similarly to Krylov subspace methods in the field of linear systems), plus a fast projection operator. Although they offer superior convergence with respect to fixed point iterations, they are not easily applicable to problems involving finite elements. In fact, this class of solvers often perform the matrix-by-vector primitive over a Schur complement matrix, a product of constraint jacobians and inverse of the mass matrix. Since the mass matrix is often (block) diagonal, the Schur complement can be quickly factored at the beginning of the iteration, but if finite elements are used, the Schur matrix would also involve to some extent the inverse of element stiffness matrices and damping matrices, a costly and inefficient operation unless special developments are considered.[18] Extending complementarity approaches in order to encompass finite elements is important because recent research has shown that complementarity-based formulations for large deformation dynamics with contacts often show superior performance when compared to penalty-based approaches.[19,20]

Another class of solvers that can draw on the optimization-based approach are Interior Point Methods (IPMs). These solvers share some similarities with the previously mentioned nonsmooth Newton methods, in that they require the solution of a saddle-point linear system at each iteration. IPMs offer the best theoretical convergence for the problem at hand.[21] However, their implementation is complex, and despite the encouraging theoretical properties, in practical scenarios they do not scale well for large problems. One of their drawbacks, also, is that at the present state of knowledge there are no efficient ways to warm-start them.[22] This led us to investigate other methods for constrained optimization problems.

The Alternating Direction Method of Multipliers (ADMM) has been proposed in the 1970s[23,24] as a practical way to solve constrained optimization problems by iterating over the minimization of two (not necessarily differentiable) objectives. ADMM introduces auxiliary variables and casts the original problem as the minimization of a separable objective subject to a linear constraint between the original variables and the auxiliary variables. With a proper choice of auxiliary variables, the minimization of the separable function is obtained iterating on two distinct minimization steps of lower complexity. The method iterates updating primal variables, auxiliary variables and dual variables similarly to a fixed point iteration: as such, they compare less favourably to IPMs if one is interested in convergence to high precision; however, recent developments showed that in practical scenarios they offer superior speed, scalability, and robustness.[25] These positive properties triggered a recent revival of ADMM and similar operator-splitting methods even as alternatives to IPMs, especially in distributed convex optimization with large-scale problems where moderate precision is acceptable.[26]

ADMM methods have been used in many fields, with recent and notable examples in computer graphics,[27] computational fluid dynamics,[28] simulation of deformable structures,[29] and contact dynamics.[30,31]

In the field of robotics, a sequential unconstrained minimization technique (SUMT) method based on augmented Lagrangian has been discussed in References 32,33, to our knowledge for the first time in a context of nonsmooth dynamics: SUMT leads to an iteration with the same structure of ADMM, for the same type of constraints.

Recent developments addressed the possibility of accelerating the ADMM method. In Reference 34 an acceleration scheme has been proposed for special cases of symmetric ADMMs, and assuming that one of the two objectives is differentiable. In Reference 35 the Nesterov acceleration method has been proposed for problems where at least one of the two objectives is quadratic, and under the assumption that both objectives are strongly convex; if the latter assumption is not guaranteed, restarting strategies are needed. In Reference 36 a similar acceleration method has been proposed, with $\mathcal{O}\left(\frac{1}{k^2}\right)$ convergence rate, but without the assumption that one of the objectives must be quadratic. Other types of improvements, based on the Anderson fixed point acceleration, have been put forward in References 27,37, showing high performance in computer graphics applications.

Recent developments also focused on the possibility of parallelizing the ADMM iterations on GPU and parallel architectures.[38]

In Reference 39 a specialized ADMM method for quadratic problems has been presented: building off this method, we develop a solver that can exploit the nature of nonsmooth dynamical problems. An attractive property of the approach is that it requires the solution of a linear system per each iteration, but unlike IPMs, the linear system most often remains unchanged during the iterations, so a factorization can be reused multiple times with a great benefit in terms of speed.

In the approach proposed herein, unknowns are primal variables (velocity measures) and dual variables (impulses in contact points and joints), the convex cone constraints stem from the Coulomb friction law, and the system matrix includes terms from the mass matrices and from the tangent stiffness/damping matrices (hence it is sparse, but not necessarily diagonal).

This paper will present a model for the non-smooth dynamics, it will show how to cast it as a QP with convex conic constraint, it will present the ADMM method to solve it, it will discuss practical implementation details and then it will show benchmarks.

2 | DEFINITIONS

In this section we list some basic definitions that we will use in the rest of the paper.

Definition 1. A *second-order cone*, also Lorentz cone, is a self-dual self-scaled symmetric cone defined as

$$\mathcal{K} = \left\{(x_0, \boldsymbol{x}_1) \in \mathbb{R} \times \mathbb{R}^{p-1} \,:\, \|\boldsymbol{x}_1\|_2 \leq x_0\right\} = -\mathcal{K}^*. \tag{1}$$

Definition 2. The *dual cone* $\mathcal{K}^*$ of a set $\mathcal{K}$ in a vector space equipped with an inner product, is always convex and is defined as

$$\mathcal{K}^* = \left\{ \boldsymbol{y} \in \mathbb{R}^n \,:\, \langle \boldsymbol{y}, \boldsymbol{x} \rangle \geq 0 \quad \forall \boldsymbol{x} \in \mathcal{K} \right\}. \tag{2}$$

Definition 3. The *polar cone* is defined as

$$\mathcal{K}^\circ = \left\{ \boldsymbol{y} \in \mathbb{R}^n \,:\, \langle \boldsymbol{y}, \boldsymbol{x} \rangle \leq 0 \quad \forall \boldsymbol{x} \in \mathcal{K} \right\} = -\mathcal{K}^*. \tag{3}$$

Definition 4. The *normal cone* to a closed convex set $\mathcal{K}$ at the point $\boldsymbol{x} \in \mathcal{K}$ is closed and convex and is defined as

$$\mathcal{N}_{\mathcal{K}}(\boldsymbol{x}) = \{ \boldsymbol{y} \in \mathbb{R}^n \,:\, \langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{z} \rangle \geq 0, \forall \boldsymbol{z} \in \mathcal{K} \}. \tag{4}$$

Definition 5. The *indicator function* of a subset $\mathcal{A} \in \mathcal{E}$ is a scalar function $I \,:\, \mathcal{E} \mapsto \mathbb{R}$ defined as:

$$I_{\mathcal{A}}(\boldsymbol{x}) = \begin{cases} 0 \text{ if } \boldsymbol{x} \in \mathcal{A} \\ \infty \text{ if } \boldsymbol{x} \notin \mathcal{A} \end{cases}. \tag{5}$$

Remark. The indicator function of any closed set is lower semicontinuous.

Definition 6. The *subdifferential* $\partial f(\boldsymbol{x}_0)$ at $\boldsymbol{x}_0$ of a convex, possibly nondifferentiable scalar function $f \,:\, \mathbb{R}^n \mapsto \mathbb{R}$, is the closed convex set of all subgradients $\boldsymbol{g}$ at $\boldsymbol{x}_0$:

$$\partial f(\boldsymbol{x}_0) = \{ \boldsymbol{g} \,:\, f(\boldsymbol{x}) - f(\boldsymbol{x}_0) \geq \langle \boldsymbol{g}, (\boldsymbol{x} - \boldsymbol{x}_0) \rangle \;\; \forall \boldsymbol{x} \in \mathcal{E} \}. \tag{6}$$

Remark. The subdifferential is a set-valued function $\partial f(\boldsymbol{x}) \,:\, \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ in general, but as a special case, if $f(\boldsymbol{x})$ is differentiable, $\partial f(\boldsymbol{x}) = \{ \nabla f(\boldsymbol{x}) \}$.

Remark. The subdifferential of an indicator function of a convex set is also its normal cone:

$$\partial I_{\mathcal{K}}(\boldsymbol{x}) = \mathcal{N}_{\mathcal{K}}(\boldsymbol{x}). \tag{7}$$

Definition 7. The *projection* of $\boldsymbol{y}$ on a nonempty closed convex set $\mathcal{E}$ is a $\mathbb{R}^n \to \mathbb{R}^n$ mapping defined as:

$$\Pi_{\mathcal{E}}(\boldsymbol{y}) = \operatorname{argmin}_{\boldsymbol{x} \in \mathcal{E}} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2. \tag{8}$$

Remark. If $f(\boldsymbol{y})$ is the indicator function $I_{\mathcal{E}}(\boldsymbol{y})$ to a nonempty closed convex set $\mathcal{E}$, defined as in (5), then $\operatorname{prox}_f(\boldsymbol{y})$ is the orthogonal projection onto that set:

$$\operatorname{prox}_{I_{\mathcal{E}}}(\boldsymbol{y}) = \Pi_{\mathcal{E}}(\boldsymbol{y}). \tag{9}$$

3 | THE NONSMOOTH MULTIBODY MODEL

Different time stepping schemes have been proposed for MDIs, here we refer to the one discussed in Reference 12 without lack of generality. After regularization and convexification and discretization, the MDIs leads to a major numerical bottleneck to be solved at each time step: a (mixed) CCP with unknowns $\boldsymbol{v}$ and $\boldsymbol{\gamma}_\epsilon$:

$$\begin{cases} H\boldsymbol{v} - \boldsymbol{k} - D_\epsilon \boldsymbol{\gamma}_\epsilon = 0 & \text{(10a)} \\ \quad D_\epsilon^T \boldsymbol{v} + \boldsymbol{b}_\epsilon = \boldsymbol{u}_\epsilon & \text{(10b)} \\ -\Upsilon^\circ \ni \boldsymbol{u}_\epsilon \quad \perp \quad \boldsymbol{\gamma}_\epsilon \in \Upsilon & \text{(10b)} \end{cases}$$

where

- unknown $\boldsymbol{v} \in \mathbb{R}^{n_v}$ is the speed at the end of the time step h,
- unknown $\boldsymbol{\gamma}_\epsilon \in \mathbb{R}^{n_c}$ is the reaction in contacts and bilateral joints, a vector-signed measure with Lebesgue decomposition in atomic parts (impacts) and continuous parts (continuous reactions),
- $H \in \mathbb{R}^{n_v \times n_v}$ is a positive definite matrix containing M, the block-diagonal matrix of the masses and inertia tensors of the bodies; if stiff loads are added too (ex. when finite elements come into play) it becomes a block-sparse matrix including the tangent stiffness and damping matrices, for instance $H = M - h^2 \nabla_q \boldsymbol{f} - h \nabla_v \boldsymbol{f}$ in case of the first-order time steppers in References 12,18,

- $D_\epsilon \in \mathbb{R}^{n_v \times n_c}$ is a sparse matrix, the transpose jacobian of all constraints,
- $\boldsymbol{k} \in \mathbb{R}^{n_v}$ is a vector containing terms proportional to applied forces $\boldsymbol{f}$ and to the last-known speed,
- $\boldsymbol{b} \in \mathbb{R}^{n_c}$ is a vector containing constraint stabilization terms and rheonomic terms,
- Υ is the Cartesian product of all cones of admissible constraint forces, $\Upsilon = \bigtimes_j \Upsilon_j$, in detail:
- if a frictional contact is added, $\Upsilon_j \subset \mathbb{R}^3$ is a second-order Lorentz cone (see Definition 1) with aperture proportional to friction coefficient,
- if a bilateral constraint is added, $\Upsilon_j = \mathbb{R}$ and $\Upsilon_j^\circ = \{0\}$,
- if a unilateral constraint is added, $\Upsilon_j = \mathbb{R}^+$ and $\Upsilon_j^\circ = \mathbb{R}^-$,
- Υ° is the polar cone, see Definition 3.

For more details on the model we refer to Reference 40.
Since H is positive definite, from (10a) one also gets:

$$\boldsymbol{v} = H^{-1}(\boldsymbol{k} + D_\epsilon \boldsymbol{\gamma}_\epsilon), \tag{11}$$

and Equation (10b) can be written as

$$\boldsymbol{u}_\epsilon = N\boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon,$$

where we introduced the Schur complement*

$$N = D_\epsilon^T H^{-1} D_\epsilon, \tag{12}$$

and the vector

$$\boldsymbol{r}_\epsilon = D_\epsilon^T H^{-1} \boldsymbol{k} + \boldsymbol{b}_\epsilon. \tag{13}$$

In this way, one can also reformulate the problem as the following CCP:

$$-\Upsilon^\circ \ni N\boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon \quad \perp \quad \boldsymbol{\gamma}_\epsilon \in \Upsilon. \tag{14}$$

This CCP corresponds exactly to a first-order optimality condition of a convex quadratic program:

$$\begin{aligned} &\min \quad \frac{1}{2}\boldsymbol{\gamma}_\epsilon^T N \boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon^T \boldsymbol{\gamma}_\epsilon && (15a)\\ &\text{s.t.} \quad \boldsymbol{\gamma}_\epsilon \in \Upsilon, && (15b) \end{aligned}$$

and in fact, the CCP (14) can be written in the more conventional language of the Karush–Kuhn–Tucker optimality conditions on *dual variables* $\boldsymbol{y}$ multipliers, for $\boldsymbol{y} = -\boldsymbol{u}_\epsilon$, and *primal variables* $\boldsymbol{\gamma}_\epsilon$:

$$N\boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon + I\boldsymbol{y} = \boldsymbol{0} \tag{16a}$$

$$\boldsymbol{\gamma}_\epsilon = \boldsymbol{z} \tag{16b}$$

$$\Upsilon \ni \boldsymbol{z} \quad \perp \quad \boldsymbol{y} \in \Upsilon^\circ \tag{16c}$$

After the convex program (15) is solved by ADMM, one can compute $\boldsymbol{v}$ using (11) with an immediate step.
The ADMM method in Reference 39 can be used to solve problems in the form

$$P\boldsymbol{x} + \boldsymbol{q} + A^T\boldsymbol{y} = \boldsymbol{0} \tag{17a}$$

$$A\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{z} \tag{17b}$$

$$\mathcal{C} \ni \boldsymbol{z}, \quad \boldsymbol{y} \in \mathcal{N}_C(\boldsymbol{z}), \tag{17c}$$

*The N matrix, which is the Schur complement of the matrix of the linear system Equations (10a) and (10b), is also called *Delassus operator* in the contact dynamics community.

and recalling Definitions (3) and (4), one can see that (16) is a special case of (17) where $A = I, P = N, C = \Upsilon, \boldsymbol{q} = \boldsymbol{r}, \boldsymbol{x} = \boldsymbol{\gamma}_\epsilon$, $\boldsymbol{b} = \boldsymbol{0}$, where some optimizations can take place because of the structure of our problem.

We remark that the ADMM method just makes the assumption of Υ being convex, so the problem can be generalized to $\boldsymbol{\gamma}_\epsilon \in \mathcal{C}$ where $\mathcal{C}$ is a generic convex set, and at the same time we assume an associated[†] flow $\boldsymbol{u}_\epsilon \in -\mathcal{N}_{\mathcal{C}}(\boldsymbol{\gamma}_\epsilon)$. For instance, $\mathcal{C}$ could be a capped friction cone to represent plasticization of contacts, or a cone translated downward to represent possible adhesion in contact up to a threshold, or the Von Mises yield region if $\boldsymbol{\gamma}_\epsilon$ represent stresses in finite elements undergoing plasticization.

Also, aiming at highest generality, in presence of finite elements one might need to use tangent stiffness matrices K and damping matrices D to accommodate an implicit integration scheme for stiff elements, and this means that the Schur complement would be computed as $N = D_\epsilon^T H^{-1} D_\epsilon$. Here H is a linear combination of M, K, D (depending on the integration scheme) and in general is not block-diagonal anymore.

4 | THE ADMM SOLVER

For variables $(\boldsymbol{x}, \boldsymbol{z}) \in \mathbb{R}^m \times \mathbb{R}^n$, the ADMM methods solve constrained optimization problems with separable structure

$$\min \quad f(\boldsymbol{x}) + g(\boldsymbol{z}) \tag{18a}$$

$$\text{s.t.} \quad A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{b} = \boldsymbol{0}, \tag{18b}$$

where $f : \mathbb{R}^{n_x} \to \overline{\mathbb{R}}$ and $g : \mathbb{R}^{n_z} \to \overline{\mathbb{R}}$ are proper convex lower-semi-continuous functions, and $A \in \mathbb{R}^{n_x \times n_y}, B \in \mathbb{R}^{n_z \times n_y}$ are linear operators.

The optimality conditions require that primal feasibility

$$A\boldsymbol{x}^\star + B\boldsymbol{z}^\star - \boldsymbol{b} = \boldsymbol{0}, \tag{19}$$

and dual feasibility

$$0 \in \partial f(\boldsymbol{x}^\star) + A^T \boldsymbol{y}^\star \tag{20a}$$

$$0 \in \partial g(\boldsymbol{z}^\star) + B^T \boldsymbol{y}^\star, \tag{20b}$$

must hold at the solution $(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{y}^\star)$, saddle point of the Lagrangian with dual variables $\boldsymbol{y} \in \mathbb{R}^{n_y}$.

By introducing an augmented Lagrangian

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{y}^T(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{b}) + \frac{\rho}{2} \left\| A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{b} \right\|_2^2. \tag{21}$$

the ADMM method converges to the solution $(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{y}^\star)$ by iterating over two distinct minimization problems as in the following loop:

$$\boldsymbol{x}^{k+1} \in \arg\min_{\boldsymbol{x}} \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{y}^k). \tag{22}$$

$$\boldsymbol{z}^{k+1} \in \arg\min_{\boldsymbol{z}} \mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{y}^k). \tag{23}$$

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \rho(A\boldsymbol{x}^{k+1} + B\boldsymbol{z}^{k+1} - \boldsymbol{b}). \tag{24}$$

The convergence of such method has rate $\mathcal{O}\left(\frac{1}{k}\right)$. Although the convergence is fast in the first iterations, it tends to deteriorate later as in most fixed-point iterations; however in practical scenarios where loose tolerances can be accepted, even this basic formulation of ADMM proves to be efficient and robust.

The performance of ADMM depends on the efficiency of the updates in (22) and (23). We will design the method such that the step (23) will correspond to a very efficient projection onto a cone with separable structure,

[†]Not all dissipative constitutive models are associated. For example, in some computational plasticity models, the plastic flow might deviate with respect to the normal to the yield surface. Moreover, the Coulomb contact model itself is not associated—in fact the contact velocity $\boldsymbol{u}$ can range into a wider cone than the polar of the friction cone, but our MDI model compensates for such effect via a stabilization term. We will assume associated models heretofore, and we do not deal with non-associated models to avoid additional complexity for the moment.

whereas the only bottleneck will be (22), corresponding to a quadratic optimization to be solved with a linear system.

In order to rewrite (15) as a sum of two functions as in (20), we introduce $\boldsymbol{\gamma}_\epsilon \in \mathbb{R}^{n_c}$ as primal variables, $\boldsymbol{z} \in \mathbb{R}^{n_c}$ as auxiliary variables, we add $\boldsymbol{\gamma}_\epsilon - \boldsymbol{z} = \boldsymbol{0}$ as the linear constraint, we reformulate the $\boldsymbol{\gamma}_\epsilon \in \Upsilon$ constraint by adding a nonsmooth penalty function $I_\Upsilon(\boldsymbol{z})$ given by an indicator function (see Definition 5), and finally we have

$$\min \quad \frac{1}{2}\boldsymbol{\gamma}_\epsilon^T N \boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon^T \boldsymbol{\gamma}_\epsilon + I_\Upsilon(\boldsymbol{z}) \tag{25a}$$

$$\text{s.t.} \quad \boldsymbol{\gamma}_\epsilon = \boldsymbol{z}. \tag{25b}$$

We can write the augmented Lagrangian introducing a step size parameter ρ and a vector of dual variables $\boldsymbol{y}$, obtaining:

$$\begin{aligned}\mathcal{L}_\rho\left(\boldsymbol{\gamma}_\epsilon, \boldsymbol{z}, \boldsymbol{y}\right) = &\frac{1}{2}\boldsymbol{\gamma}_\epsilon^T N \boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon^T \boldsymbol{\gamma}_\epsilon + I_\Upsilon(\boldsymbol{z}) \\ &+ \boldsymbol{y}^T(\boldsymbol{\gamma}_\epsilon - \boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{\gamma}_\epsilon - \boldsymbol{z}\right\|_2^2,\end{aligned} \tag{26}$$

that is also, using the property $\boldsymbol{y}^T\boldsymbol{r} + \frac{\rho}{2}\|\boldsymbol{r}\|_2^2 = \frac{\rho}{2}\left\|\boldsymbol{r} + \frac{1}{\rho}\boldsymbol{y}\right\|_2^2 - \frac{1}{2\rho}\|\boldsymbol{y}\|_2^2$, the following

$$\begin{aligned}\mathcal{L}_\rho\left(\boldsymbol{\gamma}_\epsilon, \boldsymbol{z}, \boldsymbol{y}\right) = &\frac{1}{2}\boldsymbol{\gamma}_\epsilon^T N \boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon^T \boldsymbol{\gamma}_\epsilon + I_\Upsilon(\boldsymbol{z}) \\ &+ \frac{\rho}{2}\left\|\boldsymbol{\gamma}_\epsilon - \boldsymbol{z} + \frac{1}{\rho}\boldsymbol{y}\right\|_2^2 - \frac{1}{2\rho}\|\boldsymbol{y}\|_2^2.\end{aligned} \tag{27}$$

The first step of ADMM requires to compute

$$\boldsymbol{\gamma}_\epsilon^{k+1} = \arg\min_{\boldsymbol{\gamma}_\epsilon} \mathcal{L}_\rho\left(\boldsymbol{\gamma}_\epsilon, \boldsymbol{z}^k, \boldsymbol{y}^k\right). \tag{28}$$

$$= \arg\min_{\boldsymbol{\gamma}_\epsilon} \frac{1}{2}\boldsymbol{\gamma}_\epsilon^T N \boldsymbol{\gamma}_\epsilon + \boldsymbol{r}_\epsilon^T \boldsymbol{\gamma}_\epsilon + \frac{\rho}{2}(\boldsymbol{\gamma}_\epsilon - \boldsymbol{z}^k + \frac{1}{\rho}\boldsymbol{y}^k)^T(\boldsymbol{\gamma}_\epsilon - \boldsymbol{z}^k + \frac{1}{\rho}\boldsymbol{y}^k) - \frac{1}{2\rho}\left\|\boldsymbol{y}^k\right\|_2^2. \tag{29}$$

$$= \arg\min_{\boldsymbol{\gamma}_\epsilon} \frac{1}{2}\boldsymbol{\gamma}_\epsilon^T (N + \rho I)\boldsymbol{\gamma}_\epsilon + (\boldsymbol{r}_\epsilon - \rho\boldsymbol{z}^k + \boldsymbol{y}^k)^T \boldsymbol{\gamma}_\epsilon + C^k. \tag{30}$$

This is an unconstrained convex quadratic program whose optimality conditions lead to the following linear problem:

$$[N + \rho I]\,\boldsymbol{\gamma}_\epsilon^{k+1} = \rho\boldsymbol{z}^k - \boldsymbol{r} - \boldsymbol{y}^k. \tag{31}$$

The linear system (31) can be solved as it is, but in our case it would be better to exploit the fact that the Schur matrix N is a product $D_\epsilon^T H^{-1} D_\epsilon$. We would like to avoid computing H^{-1}, even storing a precomputed inverse for all iterations would be unpractical because often dense (except when H is a diagonal mass matrix M). So we propose to replace (31) with an equivalent saddle point problem:

Theorem 1. *Let $\boldsymbol{\gamma}_\epsilon^{k+1}$ be a solution to (31), then it is also a solution to*

$$\begin{bmatrix} H & D_\epsilon \\ D_\epsilon^T & -\rho I \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^{k+1} \\ -\boldsymbol{\gamma}_\epsilon^{k+1} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{k} \\ -\boldsymbol{b}_\epsilon + \rho\boldsymbol{z}^k - \boldsymbol{y}^k \end{Bmatrix}. \tag{32}$$

Proof. Performing the multiplication of the upper part of the saddle point matrix one has $H\boldsymbol{v}^{k+1} - D_\epsilon\boldsymbol{\gamma}_\epsilon^{k+1} = \boldsymbol{k}$, pre-multiplying by H^{-1} one gets $\boldsymbol{v}^{k+1} = H^{-1}D_\epsilon\boldsymbol{\gamma}_\epsilon^{k+1} + H^{-1}\boldsymbol{k}$. The multiplication of the lower part gives $D_\epsilon^T\boldsymbol{v}^{k+1} + \rho I\boldsymbol{\gamma}_\epsilon^{k+1} = -\boldsymbol{b}_\epsilon + \rho\boldsymbol{z}^k - \boldsymbol{y}^k$, hence after substitution of $\boldsymbol{v}^{k+1}$ one gets $D_\epsilon^T H^{-1} D_\epsilon\boldsymbol{\gamma}_\epsilon^{k+1} + D_\epsilon^T H^{-1}\boldsymbol{k} + \rho I\boldsymbol{\gamma}_\epsilon^{k+1} = -\boldsymbol{b}_\epsilon + \rho\boldsymbol{z}^k - \boldsymbol{y}^k$. Recalling the definition of N in (12) and the definition of $\boldsymbol{r}_\epsilon$ in (13), this can be written also as $(N + \rho I)\boldsymbol{\gamma}_\epsilon^{k+1} = \rho\boldsymbol{z}^k - \boldsymbol{r} - \boldsymbol{y}^k$. ∎

Corollary 1. *The linear system (32) introduces an auxiliary variable $\boldsymbol{v} \in \mathbb{R}^{n_v}$, however, the matrix is very sparse and does not require storing or computing any H^{-1} matrix as we should do if using (31).*

Corollary 2. *A positive side effect of this approach is that its auxiliary variable $\boldsymbol{v}$ is also the velocity term in the original mixed CCP (10), so one does not need to compute it as $\boldsymbol{v} = H^{-1}(\boldsymbol{k} + D_\epsilon\boldsymbol{\gamma}_\epsilon)$ after the iteration converged because it is already a byproduct of the linear solver in (32).*

Corollary 3. *If constraint compliance is added to the MDI, a compliance matrix C_ϵ can be present in the CCP, thus leading to a modified version of (10b):*

$$D_\epsilon^T \boldsymbol{v} + C_\epsilon \boldsymbol{\gamma}_\epsilon + \boldsymbol{b}_\epsilon = \boldsymbol{u}_\epsilon.$$

In such case one would have $N = D_\epsilon^T H^{-1} D_\epsilon + C_\epsilon$, and (32) would change into:

$$\begin{bmatrix} H & D_\epsilon \\ D_\epsilon^T & -\rho I - C_\epsilon \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^{k+1} \\ -\boldsymbol{\gamma}_\epsilon^{k+1} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{k} \\ -\boldsymbol{b}_\epsilon + \rho \boldsymbol{z}^k - \boldsymbol{y}^k \end{Bmatrix}. \tag{33a}$$

The second step of ADMM requires to compute

$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} \mathcal{L}_\rho \left(\boldsymbol{\gamma}_\epsilon^{k+1}, \boldsymbol{z}, \boldsymbol{y}^k \right).$$

This is a minimization problem too, but it can be rephrased in terms of a projection on the Υ set, which has a separable structure. In fact, recalling (27) and removing constant terms:

$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} I_\Upsilon(\boldsymbol{z}) + \frac{\rho}{2} \left\| \boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z} + \frac{1}{\rho} \boldsymbol{y}^k \right\|_2^2. \tag{34}$$

Using the Definition 7, this leads to:

$$\boldsymbol{z}^{k+1} = \Pi_\Upsilon \left(\boldsymbol{\gamma}_\epsilon^{k+1} + \frac{1}{\rho} \boldsymbol{y}^k \right). \tag{35}$$

Note that the projection Π_Υ can be performed efficiently in our case, because Υ is the Cartesian product of n_c Coulomb second-order friction cones of lower dimension, assuming a set of n_c contacts:

$$\Upsilon = \Upsilon_1 \times \Upsilon_2 \times \ldots \times \Upsilon_{n_c}, \quad \Upsilon_i \subset \mathbb{R}^3.$$

The separable structure allows the projection to be performed as a tuple of simpler projections:

$$\Pi_\Upsilon(\boldsymbol{p}) = \left\{ \Pi_{\Upsilon_1}(\boldsymbol{p}_1), \Pi_{\Upsilon_2}(\boldsymbol{p}_2), \ \ldots \ , \Pi_{\Upsilon_{n_c}}(\boldsymbol{p}_{n_c}) \right\}, \quad \boldsymbol{p}_i \in \mathrm{R}^3.$$

In a more general setting, where $\boldsymbol{\gamma}_\epsilon$ contains both contact reactions and reactions in bilateral and unilateral joints, the projection operator for the ith bilateral reaction is simply $\Pi_{\mathbb{R}}(p_i) = p_i$ and the projection operator for the ith unilateral reaction is $\Pi_{\mathbb{R}^+}(p_i) = \max(0, p_i)$.

Note that the entire Π_Υ projection requires an inexpensive and parallelizable operation.

A basic form of ADMM will then iterate over these updates:

$$\begin{bmatrix} H & D_\epsilon \\ D_\epsilon^T & -\rho I - C \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^{k+1} \\ -\boldsymbol{\gamma}_\epsilon^{k+1} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{k} \\ -\boldsymbol{b}_\epsilon + \rho \boldsymbol{z}^k - \boldsymbol{y}^k \end{Bmatrix}. \tag{36}$$

$$\boldsymbol{z}^{k+1} = \Pi_\Upsilon \left(\boldsymbol{\gamma}_\epsilon^{k+1} + \frac{1}{\rho} \boldsymbol{y}^k \right). \tag{37}$$

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \rho \left(\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1} \right). \tag{38}$$

Before being able to apply this iteration in real problems, we should introduce some optimizations that will make a big difference for the performance of the method, and that will be discussed in the next section.

5 | IMPLEMENTATION

5.1 | Residuals and termination

In order to monitor the convergence of the method, a measure of residual quantities must be introduced. Given the definitions (19), (20a), (20b), one can define a primal and a dual residual as:

$$\boldsymbol{r}_{\text{primal}}^{k+1} = \boldsymbol{\gamma}_{\epsilon}^{k+1} - \boldsymbol{z}^{k+1}. \tag{39}$$

$$\boldsymbol{r}_{\text{dual}}^{k+1} = N\boldsymbol{\gamma}_{\epsilon}^{k+1} + \boldsymbol{r} + \boldsymbol{y}^{k+1}. \tag{40}$$

An alternative way to compute $\boldsymbol{r}_{\text{dual}}^{k+1}$, that does not need to use the N matrix, is

$$\boldsymbol{r}_{\text{dual}}^{k+1} = D_{\epsilon}^{T}\boldsymbol{v}^{k+1} + C_{\epsilon}\boldsymbol{\gamma}_{\epsilon}^{k+1} + \boldsymbol{r} + \boldsymbol{y}^{k+1}.$$

Another option (see Reference 25) is to compute it as

$$\boldsymbol{r}_{\text{dual}}^{k+1} = \rho(\boldsymbol{z}^{k+1} - \boldsymbol{z}^{k}),$$

which is even faster, at the expense of storing the previous value of $\boldsymbol{z}$.

The iteration is terminated when both $\boldsymbol{r}_{\text{primal}}$ and $\boldsymbol{r}_{\text{dual}}$ fall under prescribed tolerances.

Here primal and dual residuals have an interesting physical interpretation: a large primal residual means that the solution provides reaction forces that do not satisfy the set inclusions (e.g., the friction might be overestimated), whereas a large dual residual means that the speed (in the contact point metric) is wrong. That is, in the discussed time stepping scheme, the measuring units of $\boldsymbol{r}_{\text{primal}}$ is the same of reaction impulses, e.g. N.s, whereas the measuring units of $\boldsymbol{r}_{\text{primal}}$ can be considered the same of speeds, for example, m/s.

We remark that, depending on the measuring units, the two residual errors can have different importance: this means that it is useful to set two distinct tolerances ϵ_{primal} and ϵ_{dual}. Different heuristics can be used to define relative tolerances too, as shown in Reference 39.

5.2 | Step size selection

The basic method outlined in (36)–(38) uses a fixed step ρ.

In general, small values of ρ cause a fast reduction of the dual residual, but a slow reduction of the primal residual. This has a physical interpretation: truncating iterations based on too small ρ most often lead to solutions where parts do not interpenetrate, but some contact forces do not satisfy the inclusion in the Coulomb cone. That is, some parts might stick like contacts were glued.

On the other hand, large values of ρ cause a fast reduction of the primal residual, but a slow reduction of the dual residual. This means that truncating iterations with too large ρ most often lead to solutions where contact forces satisfy very well the inclusion in the Coulomb friction cones, but parts might interpenetrate.

Clearly, this calls for the introduction of a scheme that adaptively changes the step size ρ in order to achieve a balanced reduction of both residuals. In most cases one can use heuristics and adjust ρ only one or two times during the iterations, hence allowing to reuse the factorization of the linear system (36) as much as possible. In fact, the most time-consuming step is the factorization of the matrix in (36), but this happens only at the first iteration and at all iterations where ρ changes, otherwise one can reuse the last factorization because the matrix is the same, and perform only the back solve step with the different right-hand side. On average, in our tests, the back solve operation took less than 1/10 of the time needed for the factorization.

We tested different policies for scaling the ρ step in our algorithm. All policies share the following concept: starting from an initial value ρ_0, the step is scaled once each n_s iterations such that the primal and dual residuals are well balanced,[41] that is:

$$\rho^{k+1} = \rho^{k}\eta_{\rho}. \tag{41}$$

We define the *balanced* scaling policy as

$$\eta_{\rho} = \frac{\left\| \boldsymbol{r}_{\text{primal}}^{k} \right\|_{\infty}}{\left\| \boldsymbol{r}_{\text{dual}}^{k} \right\|_{\infty}}. \tag{42}$$

We define the *balanced normalized* scaling policy as

$$\eta_\rho = \left(\frac{\left\| \boldsymbol{r}^k_{\text{primal}} \right\|_\infty}{\max\left(\left\| \boldsymbol{z}^k \right\|_\infty, \left\| \boldsymbol{\gamma}^k_\epsilon \right\|_\infty \right)} \right) \left(\frac{\left\| \boldsymbol{r}^k_{\text{dual}} \right\|_\infty}{\left\| \boldsymbol{y}^k \right\|_\infty} \right)^{-1}. \tag{43}$$

Another idea is presented in Reference 42, where the the step is changed using the Barzilai–Borwein spectral formula. This is referenced as the *spectral* scaling policy in the following.

In all the cases above, one needs safeguards to avoid excessive scaling, for example we limit the scaling factor η_ρ by clamping it in the $\left[\frac{1}{50}, 50\right]$ interval by default. Also, in order to avoid an excessive number of factorizations, one could force the scaling factor to $\eta_\rho = 1$ (hence no change in ρ) if it is not too far from the unity anyway—for example, by default we use a tolerance interval $\left[\frac{1}{2}, 2\right]$, and if the scaling falls inside that interval, it is forced to $\eta_\rho = 1$.

5.3 | Custom treatment of bilateral joints

We experimented with the idea of using a ρ value that changes on a per-constraint basis. If one could know in advance which contact will be active at the solution, one could split ρ_i values between very high or very low values for the fastest convergence; however, this is not possible because the set of active constraints is known only at the solution. However, we still can use a simple heuristics consisting in forcing a small value $\rho_b = 1 \times 10^{-9}$ for constraints belonging to the set C_B of bilateral constraints, where one knows in advance that the corresponding dual variable y_i would be null anyway. This produces better convergence results in systems containing both contacts and many bilateral joints, such as when simulating articulated robots that grasp some objects.

This means that, instead of using a scalar ρ^k, in our code we use a diagonal matrix Θ^k defined as:

$$\Theta^k = \Theta(\rho^k), \quad \Theta^k_{i,j} = \begin{cases} 0 & \text{if} \quad i \neq j \\ \rho_b & \text{if} \quad i = j \text{ and } i \in C_B \\ \rho^k & \text{if} \quad i = j \text{ and } i \notin C_B \end{cases} \tag{44}$$

6 | FINAL ALGORITHM

Using the above results, one can finally obtain the ADDM algorithm for solving the CCP of the nonsmooth dynamics timestepper:

The algorithm above contains a single computational bottleneck, that is the solution of the linear system. However, if a direct solver is used, a LU decomposition[‡] could be computed at the beginning and then recomputed only when the step size is adjusted, otherwise in all other iterations with $\rho^{k+1} = \rho^k$ the decomposition is unaltered and only a relatively inexpensive back solve is required.

6.1 | Preconditioning

We experienced that the convergence of Algorithm 1 is negatively affected by the presence of odd mass ratios in the mechanical system. The immediate effect of uneven mass ratio is a broadening of the spectrum of the N matrix. Note that this can be also a consequence of using different measuring units for different constraint multipliers γ_i, for instance even a system with odd mass ratios can have a badly conditioned N matrix if some constraint reactions are assumed in newtons, other in kilo newtons—clearly, as the choice of measuring units is arbitrary, one should find a way to make the solver as much insensitive as possible to the scaling of the constraints.

[‡]In many cases the H matrix is hermitian, for example when just block-diagonal mass matrices are used, hence the decomposition of the symmetric indefinite saddle point matrix can be done via a LDL decomposition, which is faster than the LU decomposition. However there are cased where un-symmetric tangent stiffness matrices might be added to H, as happens in some finite element problems, precluding this approach.

Algorithm 1. ADMM for solving CCPs in nonsmooth dynamics

Input: Initial approximations $\boldsymbol{y}^0, \boldsymbol{z}^0$

Outputs: Solution $\boldsymbol{\gamma}_\epsilon^\star, \boldsymbol{v}^\star$, optionally also $\boldsymbol{y}^\star, \boldsymbol{z}^\star$

$\Theta^0 = \Theta(\rho^0)$

while $k \leq n_{\text{maxiters}}$ **do**

$$\begin{bmatrix} H & D_\epsilon \\ D^T & -\Theta^k - C_\epsilon \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^{k+1} \\ -\boldsymbol{\gamma}_\epsilon^{k+1} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{k} \\ -\boldsymbol{b}_\epsilon + \Theta^k \boldsymbol{z}^k - \boldsymbol{y}^k \end{Bmatrix} \quad \triangleright \text{ Solve for } \boldsymbol{v}^{k+1}, \boldsymbol{\gamma}_\epsilon^{k+1}$$

$\boldsymbol{z}^{k+1} = \Pi_\Upsilon \left(\boldsymbol{\gamma}_\epsilon^{k+1} + (\Theta^k)^{-1} \boldsymbol{y}^k \right)$ ▷ Project onto Υ

$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \Theta^k \left(\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1} \right)$

$\boldsymbol{r}_{\text{primal}}^{k+1} = \boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1}$

$\boldsymbol{r}_{\text{dual}}^{k+1} = \Theta^k (\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)$

if $\left\| \boldsymbol{r}_{\text{primal}}^{k+1} \right\| < \epsilon_{\text{primal}}$ and $\left\| \boldsymbol{r}_{\text{dual}}^{k+1} \right\| < \epsilon_{\text{dual}}$ **then**

break

end if

$\rho^{k+1} = \text{StepAdjustPolicy}(\rho^k)$

$\Theta^{k+1} = \Theta(\rho^{k+1})$

end while

A possible remedy for this difficulty is to perform a simplified form of preconditioning by doing a diagonal scaling of N and $\boldsymbol{r}_\epsilon$ before starting the iteration, thus operating on the scaled form of the problem:

$$\min \quad \frac{1}{2} \breve{\boldsymbol{\gamma}}_\epsilon^T \breve{N} \breve{\boldsymbol{\gamma}}_\epsilon + \breve{\boldsymbol{r}}_\epsilon^T \breve{\boldsymbol{\gamma}}_\epsilon, \tag{45}$$

$$\text{s.t.} \quad \breve{\boldsymbol{\gamma}}_\epsilon \in \breve{\Upsilon}, \tag{46}$$

with $\breve{\boldsymbol{\gamma}}_\epsilon = S^{-1} \boldsymbol{\gamma}_\epsilon$, $\breve{N} = SNS$, $\breve{\boldsymbol{r}}_\epsilon^T = S \boldsymbol{r}_\epsilon^T$. We set $S_{ii} = \sqrt{\frac{1}{N_{ii}}}$. Note that other more advanced (and CPU intensive) preconditioners could be used, for example in Reference 39 a modified Ruitz equilibration is proposed. After the solution is computed, one can quickly recover $\boldsymbol{\gamma}_\epsilon = S \breve{\boldsymbol{\gamma}}_\epsilon$.

Note that in search of high performance we never build explicitly $\breve{N}$ and $\breve{\boldsymbol{r}}_\epsilon^T$: what we need is just a scaling of the saddle point matrix involved in (36), that will simply include the scaled jacobians $\breve{D}_\epsilon = D_\epsilon S$, the scaled compliance $\breve{C}_\epsilon = S C_\epsilon S$, the scaled term $\breve{\boldsymbol{b}}_\epsilon = S \boldsymbol{b}_\epsilon$, the scaled set $\breve{\Upsilon} = \left\{ \breve{\boldsymbol{\gamma}} : S \breve{\boldsymbol{\gamma}}_\epsilon \in \Upsilon \right\}$.

The scaled form of the problem would require a projection $\Pi_{\breve{\Upsilon}}$, however this can be simplified we use the same S_{ii} values for each triplet corresponding to the jth friction cone constraint, for uniform scaling of the jth cone, thus $\Pi_{\breve{\Upsilon}}(\breve{\boldsymbol{z}}) = \Pi_\Upsilon(\breve{\boldsymbol{z}})$. More in general, for a convex set $\mathcal{C}$, again assuming a uniform scaling for each $\mathcal{C}_j$ set, one can use the simplification

$$\Pi_{\breve{\mathcal{C}}}(\breve{\boldsymbol{z}}) = \Pi_{\breve{\mathcal{C}}}(S^{-1} \boldsymbol{z}) = S^{-1} \Pi_{\mathcal{C}}(\boldsymbol{z}) = S^{-1} \Pi_{\mathcal{C}}(S \breve{\boldsymbol{z}}).$$

The preconditioned form of the ADMM algorithm, that we call P-ADMM heretofore, is not reported here for compactness: it is enough to replace the above scaled quantities in Algorithm 1, where $\breve{\boldsymbol{\gamma}}_\epsilon, \breve{\boldsymbol{y}}, \breve{\boldsymbol{z}}$ will replace $\boldsymbol{\gamma}_\epsilon, \boldsymbol{y}, \boldsymbol{z}$, then a final step is added after convergence, to recover the unscaled solution: $\boldsymbol{\gamma}_\epsilon = S \breve{\boldsymbol{\gamma}}_\epsilon$. Moreover, if one is interested in the residuals in the not-scaled original form, for instance to verify convergence, these can be computed as

$$\boldsymbol{r}_{\text{primal}}^{k+1} = S \left(\breve{\boldsymbol{\gamma}}_\epsilon^{k+1} - \breve{\boldsymbol{z}}^{k+1} \right)$$
$$\boldsymbol{r}_{\text{dual}}^{k+1} = S^{-1} \Theta^k (\breve{\boldsymbol{z}}^{k+1} - \breve{\boldsymbol{z}}^k).$$

As an alternative, one can see that the preconditioned algorithm can be written with the original variables $\boldsymbol{\gamma}_\epsilon, \boldsymbol{y}, \boldsymbol{z}$ if one substitutes the scaled variables and performs some algebraic simplifications, obtaining the following steps:

$$\begin{bmatrix} H & D_\epsilon \\ D^T & -\Theta^k S^{-2} - C_\epsilon \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^{k+1} \\ -\boldsymbol{\gamma}_\epsilon^{k+1} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{k} \\ -\boldsymbol{b}_\epsilon + \Theta^k S^{-2} \boldsymbol{z}^k - \boldsymbol{y}^k \end{Bmatrix}. \tag{47a}$$

$$\boldsymbol{z}^{k+1} = \Pi_\Upsilon \left(\boldsymbol{\gamma}_\epsilon^{k+1} + (\Theta^k)^{-1} S^2 \boldsymbol{y}^k \right). \tag{47b}$$

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \Theta^k S^{-2} \left(\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1} \right). \tag{47c}$$

The method above corresponds to using the original ADMM algorithm with a nonuniform step $\Theta^k S^{-2}$, with $(S^{-2})_{ii} = N_{ii}$, hence it avoids the necessity of scaling all variables and matrices. However, when one needs to compute the residuals (for instance for termination criteria or for step scaling policies), in this simplified algorithm (47) those must be computed as:

$$\begin{aligned} \boldsymbol{r}_{\text{primal}}^{k+1} &= \left(\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1} \right) \\ \boldsymbol{r}_{\text{dual}}^{k+1} &= \Theta^k S^{-2} (\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) \\ \breve{\boldsymbol{r}}_{\text{primal}}^{k+1} &= S^{-1} \left(\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1} \right) \\ \breve{\boldsymbol{r}}_{\text{dual}}^{k+1} &= \Theta^k S^{-1} (\boldsymbol{z}^{k+1} - \boldsymbol{z}^k). \end{aligned}$$

6.2 | Warm starting

One of the nice properties of ADMM methods is that they can be easily warm started, something that for example is difficult to do with IPMs. This is a relevant feature in our context, because we must solve the problem (14) at each time step, where one can expect some degree of temporal coherency in the values of $\boldsymbol{\gamma}_\epsilon$ between each step. This is especially true for simulations involving stacked objects, like when simulating environments for robots, granular flows or masonry buildings, because the contact forces often show limited changes over time, and the amount of contacts that change state can be limited. If so, one can reuse the last computed values of $\boldsymbol{\gamma}_\epsilon$ in the previous time step to warm-start the ADMM solver at the next time step.

This poses two difficulties. First of all, one must develop algorithms and data structures that are able to convey information from the contact at time t_A to a time $t_B = t_A + h$. In fact, after ADMM computes the contact forces at step t_A, those could be saved into the contact data structures; however, those contacts would be recomputed by the collision detection engine at the next time step t_B: since the objects move, also the contact manifold changes, thus it is not trivial to compute which contacts are persistent between the two steps. By using tolerances, one can detect if some contacts at time t_B are the same contacts used at time t_A, if so, their last reaction forces $\boldsymbol{\gamma}_\epsilon$ can be copied to the new contacts and used for the warm starting, whereas completely new contacts are initialized with $\boldsymbol{\gamma}_\epsilon = 0$.

Furthermore, one can see that the ADDM Algorithm 1 is a fixed point $\{\boldsymbol{y}^{k+1}, \boldsymbol{z}^{k+1}\} = \boldsymbol{g}\left(\{\boldsymbol{y}^k, \boldsymbol{z}^k\}\right)$, thus one needs $\{\boldsymbol{y}^0, \boldsymbol{z}^0\}$ to warm start it, rather than $\boldsymbol{\gamma}_\epsilon^0$. This leads to two options: store both $\boldsymbol{y}$ and $\boldsymbol{z}$ values in the contact data (wasting some memory in case of large scale simulations, and complicating both data structures and bookkeeping) otherwise assume that the last ADMM run converged exactly up to $\boldsymbol{r}_{\text{primal}} = 0$, so one could just store $\boldsymbol{\gamma}_\epsilon$ values in contacts, and then compute

$$\begin{aligned} \boldsymbol{v}^0 &= H^{-1}(\boldsymbol{k} + D_\epsilon \boldsymbol{\gamma}_\epsilon^0) \\ \boldsymbol{y}^0 &= -(D_\epsilon^T \boldsymbol{v}^0 + C \boldsymbol{\gamma}_\epsilon^0 + \boldsymbol{b}_\epsilon) \\ \boldsymbol{z}^0 &= \boldsymbol{\gamma}_\epsilon^0. \end{aligned}$$

This idea, implemented in Algorithm 2, provides a speedup of 2×–10× in our tests.

Note that also the last ρ^k value from the previous time step can be used as the initial ρ^0 for the next step, adding a further optimization. Statistically, given some degree of temporal coherency in the mechanical system, such pretuned value would be better than a default value like $\rho^0 = 0.1$.

Algorithm 2. Warm-started ADMM

Input: Initial approximation $\boldsymbol{\gamma}_\epsilon^0$, optional: $\boldsymbol{v}^0$
Output: Solution $\boldsymbol{\gamma}_\epsilon^\star, \boldsymbol{v}^\star$, optionally also $\boldsymbol{y}^\star, \boldsymbol{z}^\star$

if $\boldsymbol{v}^0$ is not provided **then**
  $\boldsymbol{v}^0 = H^{-1}(\boldsymbol{k} + D_\epsilon \boldsymbol{\gamma}_\epsilon^0)$
end if
$\boldsymbol{y}^0 = -(D_\epsilon^T \boldsymbol{v}^0 + C\boldsymbol{\gamma}_\epsilon^0 + \boldsymbol{b}_\epsilon)$
$\boldsymbol{z}^0 = \boldsymbol{\gamma}_\epsilon^0$
$\boldsymbol{\gamma}_\epsilon^\star, \boldsymbol{v}^\star = \mathrm{ADMM}(\boldsymbol{y}^0, \boldsymbol{z}^0)$

7 | ACCELERATED ADMM

Following the idea presented in Reference 35, one can improve the convergence of ADMM by introducing a predictor-corrector step based on the Nesterov acceleration, with minimal computational overhead. Heretofore we will call this variant as FADDM (Fast ADMM). The convergence of the original Nesterov-accelerated ADMM requires the assumption that both $f(\boldsymbol{x})$ and $g(\boldsymbol{z})$ in (20) are strongly convex.

Definition 8. A function $f : \mathbb{R}^n \to \mathbb{R}$, possibly nondifferentiable, is *β-strongly convex* if for all points $\boldsymbol{x}, \boldsymbol{y}$ in its domain it holds:

$$f(\boldsymbol{x}) - f(\boldsymbol{y}) \geq \langle \boldsymbol{u}, \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \quad \forall \boldsymbol{u} \in \partial f(\boldsymbol{y}). \tag{48}$$

In our case, $f(\boldsymbol{x})$ is a quadratic function, hence strongly convex, but $g(\boldsymbol{z})$ is an indicator function, hence convex but not strongly convex. In the case where just one of the two functions is strongly convex, a restart scheme can be applied as in Reference 35, where the method reverts to the original ADMM if the combined residual $r_{\text{comb}}^{k+1} = \Theta^{-1}\left\|\boldsymbol{y}^{k+1} - \boldsymbol{y}^k\right\|_2 + \Theta\left\|\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{\gamma}_\epsilon^k\right\|_2$ is not monotonically decreasing. The monotone decrease is assessed with a factor $\epsilon_r < 1$, as a default value we use $\epsilon_r = 0.999$ to avoid too frequent restarts. Using the same performance optimization strategies that we used to develop Algorithm 1, we finally obtain Algorithm 3.

We note in passing that the Algorithm 3 is based on a $\boldsymbol{z} \to \boldsymbol{\gamma}_\epsilon \to \boldsymbol{y}$ update ordering whereas the Algorithm 1 is based on a $\boldsymbol{\gamma}_\epsilon \to \boldsymbol{z} \to \boldsymbol{y}$ ordering, leading to a slightly modified warm starting method.

Also, a preconditioned version with diagonal scaling can be obtained with the already discussed transformation of (46), leading to a method that we call P-FADMM in the following.

8 | BENCHMARKS

In the literature there are many examples of iterative solvers for nonsmooth dynamics, but many of them are targeting mostly computer-graphics applications, where efficiency and robustness are more important than fidelity and precision. For instance, many benchmarks in computer graphics involve objects with low stiffness (e.g., tissues, ropes, plastics), whereas we need to test our solver with structures of engineering interest, such as steel structures. Also, the proposed method allows an accurate retrieval of the contact stresses, something that is particularly important for mechanics-related applications and that can be neglected in the computer graphics field. On the basis of these requirements we designed the benchmarks that follow.

At each time step the ADMM solver requires an updated set of contacts with the corresponding D_ϵ jacobians: to this end a contact detection algorithm provides an updated set of pairs of potential colliding points between the geometric primitives (edges, faces, curved surfaces) if such points are within a distance threshold. When a deformable structure is present, the contact detection algorithm finds the contact points using the triangles of a tessellated surface that covers the outer skin of the finite elements.

All benchmarks have been computed on a Intel® Core® i7-8750H CPU, clocked at 2.20 GHz, with six physical cores and 16 GB of RAM. The ADMM method has been implemented in C++ within the Chrono open-source library,[43] using the Eigen library v.3.37 for dense and sparse linear algebra, and Intel® MKL Pardiso[44] as a direct solver for sparse linear systems.

Algorithm 3. FADMM for solving CCPs in nonsmooth dynamics

Input: Initial approximations $\boldsymbol{\gamma}_\epsilon^0, \boldsymbol{y}^0$
Output: Solution $\boldsymbol{\gamma}_\epsilon^\star, \boldsymbol{v}^\star$, optionally also $\boldsymbol{y}^\star, \boldsymbol{z}^\star$

$\alpha^0 = 1, \quad \Theta = \Theta(\rho), \quad r_{\text{comb}}^0 = \infty$
while $k \leq n_{\text{max iters}}$ **do**
  $\boldsymbol{z}^{k+1} = \Pi_\Upsilon \left(\boldsymbol{\gamma}_\epsilon^k + (\Theta^k)^{-1} \boldsymbol{y}^k \right)$ ▷ Project onto Υ
  $\begin{bmatrix} H & D_\epsilon \\ D^T & -\Theta - C_\epsilon \end{bmatrix} \left\{ \begin{matrix} \boldsymbol{v}^{k+1} \\ -\boldsymbol{\gamma}_\epsilon^{k+1} \end{matrix} \right\} = \left\{ \begin{matrix} \boldsymbol{k} \\ -\boldsymbol{b}_\epsilon + \Theta \boldsymbol{z}^{k+1} - \boldsymbol{y}^k \end{matrix} \right\}$ ▷ Solve for $\boldsymbol{v}^{k+1}, \boldsymbol{\gamma}_\epsilon^{k+1}$
  $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \Theta \left(\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1} \right)$
  $\boldsymbol{r}_{\text{primal}}^{k+1} = \boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{z}^{k+1}$
  $\boldsymbol{r}_{\text{dual}}^{k+1} = \Theta(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)$
  if $\left\| \boldsymbol{r}_{\text{primal}}^{k+1} \right\| < \epsilon_{\text{primal}}$ and $\left\| \boldsymbol{r}_{\text{dual}}^{k+1} \right\| < \epsilon_{\text{dual}}$ **then**
    break
  end if
  $r_{\text{comb}}^{k+1} = \Theta^{-1} \left\| \boldsymbol{y}^{k+1} - \boldsymbol{y}^k \right\|_2 + \Theta \left\| \boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{\gamma}_\epsilon^k \right\|_2$
  if $r_{\text{comb}}^{k+1} < \epsilon_r r_{\text{comb}}^k$ **then**
    $\alpha^{k+1} = \frac{1+\sqrt{1+4(\alpha^k)^2}}{2}$
    $\boldsymbol{\gamma}_\epsilon^{k+1} = \boldsymbol{\gamma}_\epsilon^k + \frac{\alpha^k - 1}{\alpha^{k+1}} (\boldsymbol{\gamma}_\epsilon^{k+1} - \boldsymbol{\gamma}_\epsilon^k)$
    $\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \frac{\alpha^k - 1}{\alpha^{k+1}} (\boldsymbol{v}^{k+1} - \boldsymbol{v}^k)$
    $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \frac{\alpha^k - 1}{\alpha^{k+1}} (\boldsymbol{y}^{k+1} - \boldsymbol{y}^k)$
  else
    $\alpha^{k+1} = 1$
    $r_{\text{comb}}^{k+1} = \frac{1}{\epsilon_r} r_{\text{comb}}^k$
  end if
end while

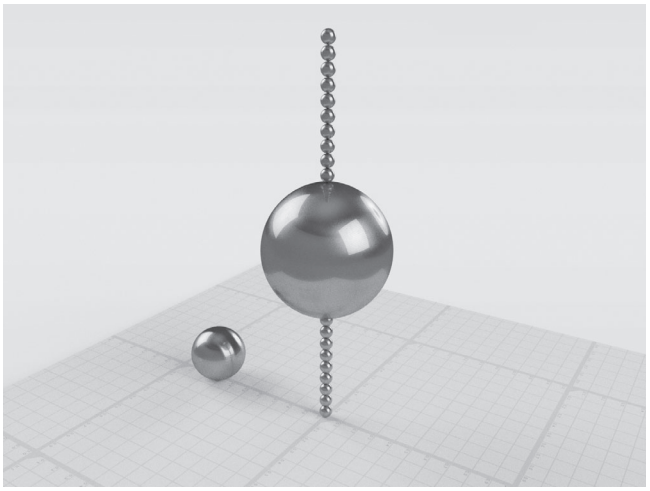

FIGURE 1 Setup of TEST 1

8.1 | High stack with odd mass ratio

This benchmark represents a typical worst case scenario in non-smooth dynamics, where a heavy object sits on a high stack of rigid bodies of much lower mass, as shown in Figure 1. The stable stacking is already a demanding case for contact dynamics, but the presence of extremely odd mass ratios add a further complication.

To this end we designed two subcases: TEST 1.A and TEST 1.B. Both feature a vertical stack of 20 spheres with mass $m = 10$ kg each, except the 10th sphere that has a mass of $m = 10,000$ kg. A further sphere with mass $m = 1000$ kg is placed

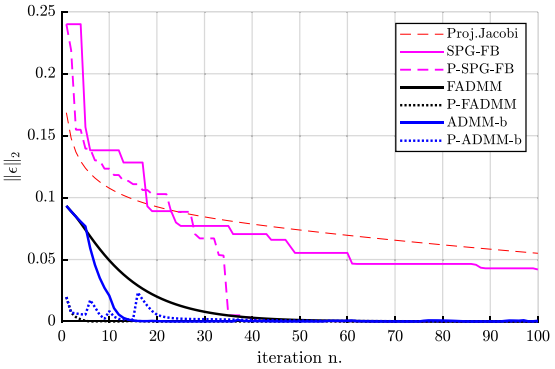

FIGURE 2 Convergence of alternating direction method of multipliers methods compared to the Jacobi projected fixed point iteration, for TEST 1.B The plot shows the violation of unilateral constraints in terms of penetrating residual velocity

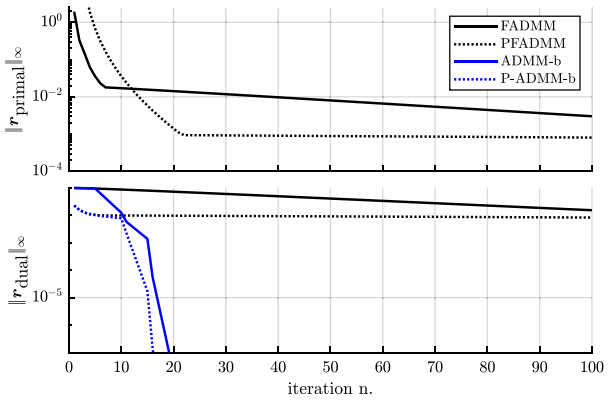

FIGURE 3 Convergence of primal and dual residuals in TEST 1A

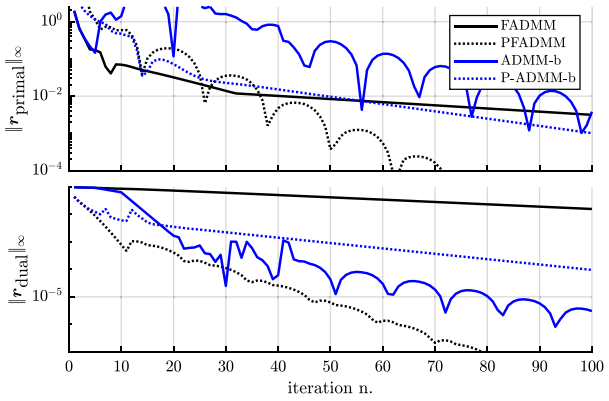

FIGURE 4 Convergence of primal and dual residuals in TEST 1.B

on the floor. All rigid bodies are subject to a vertical gravitational field $g = 9.8$ m/s^2. For TEST 1.A one has the analytical solution (all contacts are active). The TEST 1.B differs from TEST 1.A in that the heavy sphere is pulled upward by a force twice its weight, hence causing a separation at the contact below it.

As expected, in both TEST 1.A and TEST 1.B the projected Jacobi iteration, one of the most common methods in non-smooth dynamics, has a very bad convergence rate[§]. As shown in Figure 2, the ADMM method presented in this paper behave much better than the projected Jacobi iteration and, interesting enough, it also converges faster than the Preconditioned Spectral Projected Gradient with Fall-Back (P-SPG-FB), an efficient method for non-smooth problems presented in Reference 17.

For TEST 1.A, Figure 3 shows that the ADMM method using the *balanced* adaptive step size can converge to the exact solution in less than 20 iterations (the primal residual is not visible in the semi-logarithmic plot as it is immediately zero from the first iteration) (Figures 4–6).

[§]It is a known fact that the simulation of high stacks of objects is a major difficulty in non-smooth dynamics, especially in real-time simulators and in video games where the projected fixed-point iterations are truncated to keep the computational time within a limited frame budget. In those cases, the limited precision of the solution will lead to underestimated contact forces, undesired slow interpenetration of the parts and, ultimately, to unnatural collapses of stacks.

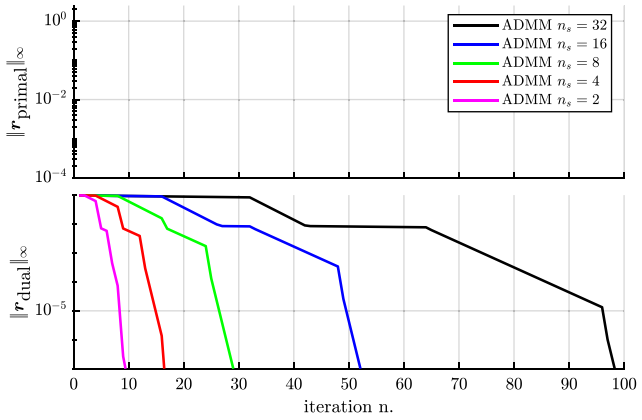

FIGURE 5 Convergence of primal and dual residuals in TEST 1.A for varying frequency n_s of automatic updates to the step size, in the ADMM algorithm

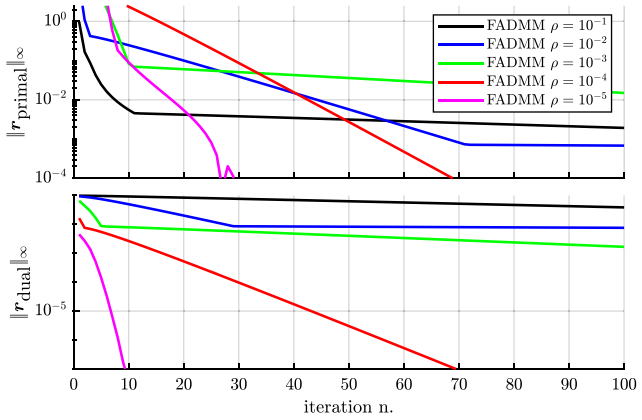

FIGURE 6 Convergence of primal and dual residuals in TEST 1.A for varying values of the step size ρ in the FADMM algorithm

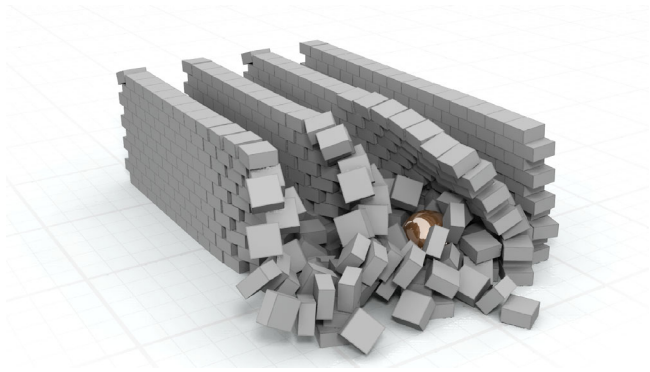

FIGURE 7 Snapshot from TEST 2, the wrecking ball benchmark (600 bricks in four walls)

Note also that we force the update of the step every n_s steps because each update of ρ corresponds of a costly factorization of a large matrix. Figure 7 shows that, in this benchmark, the convergence improves for denser updates, meeting the tolerance in the same amount of updates: this would suggest using a small n_s anyway. However, for more complex and randomized scenarios, we found that a good tradeoff is $n_s = 5$. This tradeoff value can change depending on the speed of solver used for the factorization: the higher the performance for the back solve respect to the factorization, the higher is the optimal n_s.

The FADMM method with the default fixed step $\rho = 0.05$, instead, converges slowly. Repeating the FADMM test with different step sizes as shown in Figure 6, however, the convergence is greatly improved, suggesting that the FADMM method is sensitive to the initial choice of the time step.

For TEST 1.B, convergence plots in Figure 4 show that the FADMM is very efficient even if starting from a fixed not optimal step size, although the performance deteriorates if no preconditioning is used. The ADMM method with balanced step adaptivity $n_s = 5$ converges quite well, especially with the help of a preconditioner. We note that ADMM converges even if primal and dual residuals might exhibit oscillatory behavior (in such cases, the combined residual would be monotonically decreasing anyway[35]).

Showing that a solver is capable of handling this class of object-stacking problems is relevant, for instance, to the field of civil engineering, because it shares the same difficulties of simulating tall masonry structures when assessing stability and seismic response via discrete elements.[45]

8.2 | Wrecking ball

We modeled four parallel walls made with 10 rows of 15 bricks each. Bricks length, height, width are, respectively, 3.96, 2, 4 m, and their density is 100 kg/m^3. A wrecking ball with diameter 8 m and density 8000 kg/m^3 impacts horizontally with the 600 bricks. The density of the bricks respect to the ball is deliberately high in order to generate a badly conditioned problem with odd mass ratios. The friction coefficient is $\mu = 0.4$ and the time step is $h = 0.02$ s. Figure 7 shows a snapshot from the simulation. The convergence of the ADMM method, as shown in Figure 8 for a random time step, is noticeably superior to the convergence of fixed point iterations such as projected Jacobi. The method exhibit even better convergence when using the diagonal preconditioning (46), as shown Figure 9, because of the odd scaling of the masses (Figure 10).

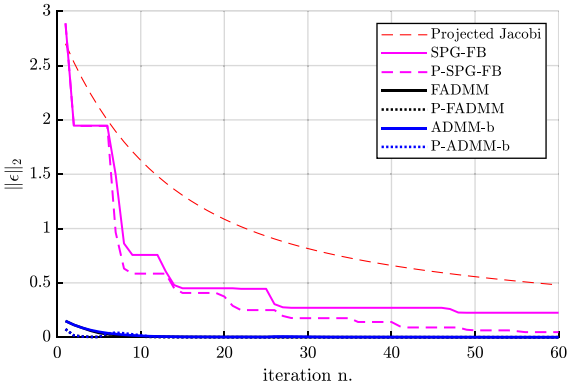

FIGURE 8 Constraint violation compared to fixed point Jacobi iterations and to spectral projected gradient methods, in TEST 2

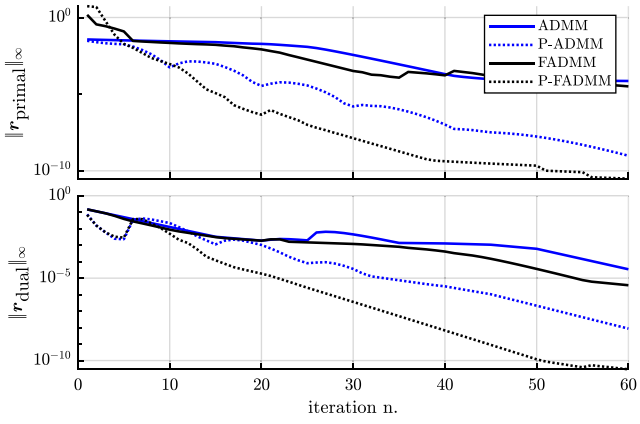

FIGURE 9 Primal-dual convergence of the algorithm in TEST 2

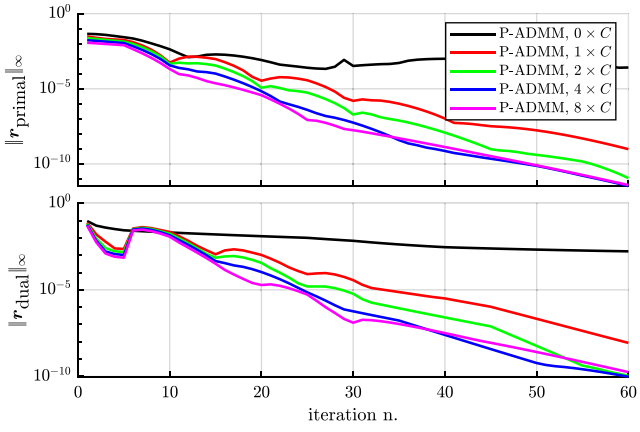

FIGURE 10 Effect of regularization or compliance in contacts in TEST 2

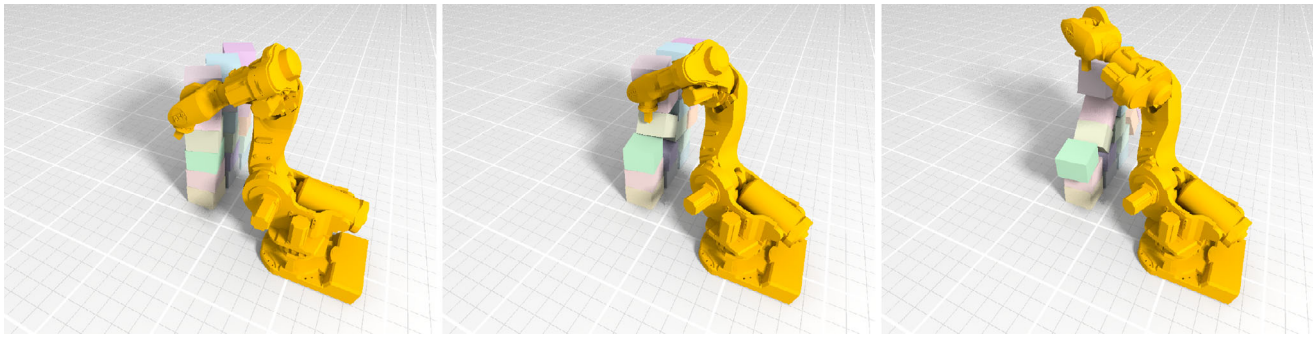

FIGURE 11 TEST 3. Robot interacting with the environment via contacts between the gripper and few boxes. The robot features extremely odd mass ratios, rheonomic constraints and redundant constraints

8.3 | Robotic manipulator

This benchmark features a 6-DOF industrial robot whose end effector interacts with 15 boxes with height 0.3 m and width 0.4 m, moving them in the working area via contacts and collisions, as shown in Figure 11. The friction coefficient between the boxes is $\mu = 0.4$, the time step is $h = 0.01$ s. The difficulty highlighted by this benchmark is the simultaneous presence of nonsmooth contacts and bilateral constraints in a critical articulated mechanism. In fact the simulation of a robot arm actuated via motorized joints is a trivial problem, but here we do the opposite: we drive the end effector via a rheonomic constraint defining an imposed trajectory respect to the base, hence the rest of the arm will naturally move to the configuration prescribed by joint constraints; however, since the mass of the end effector is much lighter than the bicept, since some joints are redundant[¶] and since the robot passes close to a singularity, the simulation of the robot alone is a challenging task. That is, even without contacts, most iterative solvers like SOR or Jacobi would converge too slowly, and even Krylov linear solvers like GMRES, CG, MINRES would require many iterations unless equipped by some preconditioner. In our test, convergence to high precision is achieved in 15 iterations at most, where just one or two of them require a refactorization of the saddle-point matrix. Moreover, when there are no contacts, the solution is provided in exactly one step, as the iteration boils down to a single factorization followed by a forward/back solve just like when simulating the robot via a direct solver in the conventional context of smooth dynamics.

8.4 | Deformable bars

This benchmark aims at estimating the efficiency and robustness of the method in problems involving also finite elements. Few methods already exist in the literature that can simulate deformable structures in the context of nonsmooth dynamics, but most of them target the field of interactive computer graphics, where the stiffness of structures is often very low. In our case, being interested in engineering applications, we require that the method would be able to handle both very deformable structures (ex. rubber-like materials) as well as very stiff structures. In this benchmark, we simulate the fall of 15 deformable bars, each modeled with a mesh of tetrahedral finite elements, for a total of 9360 degrees of freedom. At initial state bars are not touching and are dropped in five groups of three, each group with random yaw rotation and with increasing height in the 0.1 … 0.5 m range. The outer skin of the bars can collide with each other and with rigid bodies—in this case there is just a single rigid body, namely the flat ground. At the contact points there is a friction coefficient $\mu = 0.3$ and a null restitution coefficient, the time step is $h = 0.05$ s. In all simulations the material of the bars has a density of 1×10^3 kg/m^3, a 0.01 Rayleigh damping coefficient and a Poisson ratio $\nu = 0.3$, but in the case of TEST 4.A (see Figure 12) the Young modulus is $E = 1 \times 10^6$ Pa, whereas in TEST 4.B (see Figure 13) the Young modulus is many orders of magnitude higher: $E = 2 \times 10^{11}$ Pa. Tests has shown that our ADMM method converges equally well even in the case of extreme stiffness, converging to the required tolerances with comparable number of iterations. At each time step, both TEST 4.A and TEST 4.B required an average number of three factorizations for the adjustment of the ρ^i step and a variable number of forward/back solves in the range of $4 \div 50$. When bars come into contact, an average

[¶]The robot has a gravity compensator modeled via two revolute joints and one cylindrical joint, leading to an overconstrained loop.

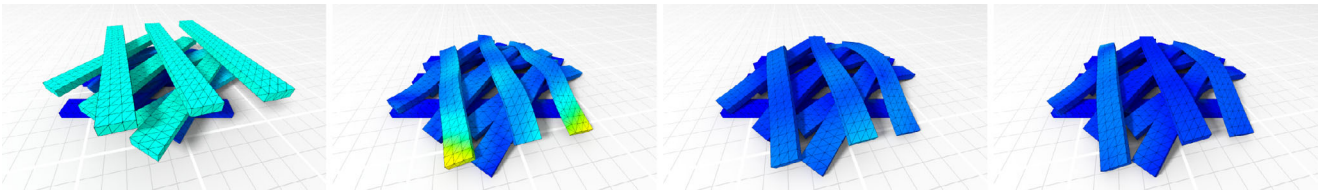

FIGURE 12 TEST 4.A. Frictional contact between deformable bars with low stiffness, at $t = 0.1$ s, $t = 0.2$ s, $t = 0.3$ s, $t = 0.4$ s. False color of the mesh represents the instantaneous norm of the speed of the nodes

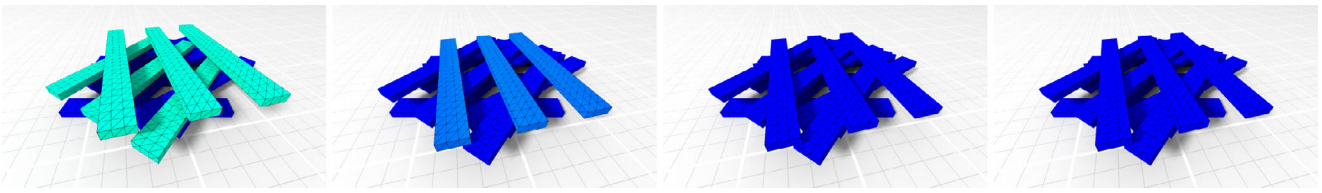

FIGURE 13 TEST 4.B. Frictional contact between deformable bars with high stiffness ($E = 200$ GPa) at $t = 0.1$ s, $t = 0.2$ s, $t = 0.3$ s, $t = 0.4$ s

number of 1650 contacts is found, leading to saddle-point linear problems with about 15 thousands of unknowns that are factorized, on average, in 0.116 s using the sparse direct solver. Each forward/back solve requires 0.0069 s on average. Each time step then required about 1 s of CPU time.

9 | RESULTS AND CONCLUSION

We performed benchmarks involving multibody systems with contacts between multiple parts, showing that the performance of the ADMM method is capable of handling problems that would converge too slowly using conventional projected fixed point methods or first-order spectral methods.

Our ADMM method requires few computational primitives: basically a projection of dual variables on conic sets, a forward/backward solve of a linear system, and its factorization. The latter is a computational bottleneck, but it can be performed only once per run, as the matrix does not change often during the iterations. We noted that ADMM can be successfully applied to problems that exhibit temporal coherence because, unlike IPMs, it supports warm-starting. A good estimation of the ADMM step size proved to be fundamental in achieving good convergence: using adaptive step scaling we obtained an efficient auto-tuning algorithm. Another optimization that allowed superior performance is the adoption of a diagonal preconditioning, with block scaling for the triplets of lagrangian multipliers relative to the conic constraints.

Further research on this topic might address the acceleration of the ADMM method by means of Anderson acceleration and the solution of nonconvex problems with complex and non-associated frictional models that go beyond the standard Coulomb model.

NOMENCLATURE

ADMM	Alternating Direction Method of Multipliers
CCP	Cone Complementarity Problem
CP	Complementarity Problem
FADMM	Fast Alternating Direction Method of Multipliers
LCP	Linear Complementarity Problem
MDI	Measure Differential Inclusion
NCP	Nonlinear Complementarity Problem
QP	Quadratic Program
VI	Variational Inequality

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in `https://github.com/projectchrono/chrono`

ORCID

Alessandro Tasora https://orcid.org/0000-0002-2664-7895

REFERENCES

1. Moreau JJ. Liaisons unilatérales sans frottement et chocs inélastiques. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*. 1983;296:1473-1476.
2. Moreau JJ, Panagiotopoulos PD, Strang G. *Topics in Nonsmooth Mechanics*. Basel, Switzerland: Birkhäuser; 1988.
3. Glocker C. *Set-Valued Force Laws: Dynamics of Non-Smooth Systems*. Vol 1. Berlin, Germany: Springer Science & Business Media; 2013.
4. Acary V, Brogliato B. *Numerical Methods for Nonsmooth Dynamical Systems: Applications in Mechanics and Electronics*. Vol 35. Berlin, Germany: Springer Science & Business Media; 2008.
5. Kinderleher D, Stampacchia G. *An Introduction to Variational Inequalities and Their Application*. New York, NY: Academic Press; 1980.
6. Harker PT, Pang JS. Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. *Math Program*. 1990;48(1):161-220. https://doi.org/10.1007/BF01582255.
7. Stewart DE, Trinkle JC. An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and Coulomb friction. *Int J Numer Methods Eng*. 1996;39:2673-2691.
8. Bender J, Müller M, Otaduy MA, Teschner M, Macklin M. A survey on position-based simulation methods in computer graphics. *Comput Graph Forum*. 2014;33(6):228-251. https://doi.org/10.1111/cgf.12346.
9. Anitescu M, Tasora A. An iterative approach for cone complementarity problems for nonsmooth dynamics. *Comput Optim Appl*. 2010;47(2):207-235. https://doi.org/10.1007/s10589-008-9223-4.
10. Tonge R, Benevolenski F, Voroshilov A. Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans Graph*. 2012;31(4):105.
11. Tasora A, Anitescu M. A complementarity-based rolling friction model for rigid contacts. *Meccanica*. 2013;48(7):1643-1659. https://doi.org/10.1007/s11012-013-9694-y.
12. Tasora A, Anitescu M. A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics. *Comput Methods Appl Mech Eng*. 2011;200(5-8):439-453. https://doi.org/10.1016/j.cma.2010.06.030.
13. Macklin M, Storey K, Lu M, et al. Small steps in physics simulation. Paper presented at: Proceedings of the ACM SIGGRAPH. Association for Computing Machinery SCA '19; 2019:1-7; New York, NY.
14. Macklin M, Erleben K, Müller M, Chentanez N, Jeschke S, Makoviychuk V. Non-smooth Newton methods for deformable multi-body dynamics. *ACM Trans Graph*. 2019;38(5):140:1-140:20. https://doi.org/10.1145/3338695.
15. Anitescu M. Optimization-based simulation of nonsmooth rigid multibody dynamics. *Math Program*. 2006;105(1):113-143. https://doi.org/10.1007/s10107-005-0590-7.
16. Heyn T, Anitescu M, Tasora A, Negrut D. Using Krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation. *IJNME*. 2013;95(7):541-561. https://doi.org/10.1002/nme.4513.
17. Mazhar H, Heyn T, Tasora A, Negrut D. Using Nesterov's method to accelerate multibody dynamics with friction and contact. *ACM Trans Graph*. 2015;34(3):32:1-32:14.
18. Frâncu M, Moldoveanu F. Position based simulation of solids with accurate contact handling. *Comput Graph*. 2017;69:12-23. https://doi.org/10.1016/j.cag.2017.09.004.
19. Yu X, Matikainen MK, Harish AB, Mikkola A. Procedure for non-smooth contact for planar flexible beams with cone complementarity problem. *Proc Inst Mech Eng Part K J Multi-body Dyn*. 2020:1464419320957450. https://doi.org/10.1177/1464419320957450.
20. Bozorgmehri B, Yu X, Matikainen MK, Harish AB, Mikkola A. A study of contact methods in the application of large deformation dynamics in self-contact beam. *Nonlinear Dyn*. 2021;103(1):581-616. https://doi.org/10.1007/s11071-020-05984-x.
21. Potra FA, Wright SJ. Interior-point methods. *J Comput Appl Math*. 2000;124(1):281-302. https://doi.org/10.1016/S0377-0427(00)00433-7.
22. Mangoni D, Tasora A, Garziera R. A primal-dual predictor-corrector interior point method for non-smooth contact dynamics. *Comput Methods Appl Mech Eng*. 2018;330:351-367. https://doi.org/10.1016/j.cma.2017.10.030.
23. Glowinski R, Marrocco A. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problémes de Dirichlet non linéaires. Revue française d'automatique, informatique, recherche opérationnelle. *Analyse numérique*. 1975;9(2):41-76.
24. Gabay D, Mercier B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Comput Math Appl*. 1976;2(1):17-40. https://doi.org/10.1016/0898-1221(76)90003-1.
25. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found Trends Mach Learn*. 2011;3(1):1-122. https://doi.org/10.1561/2200000016.
26. Cannon M, Goulart P, Garstka M. COSMO: a conic operator splitting method for large convex problems. Paper presented at: Proceedings of the European Control Conference, Naples, Italy; 2019.
27. Zhang J, Peng Y, Ouyang W, Deng B. Accelerating ADMM for efficient simulation and optimization. *ACM Trans Graph*. 2019;38(6):163:1-163:21. https://doi.org/10.1145/3355089.3356491.

28. Gregson J, Ihrke I, Thuerey N, Heidrich W. From capture to simulation: connecting forward and inverse problems in fluids. *ACM Trans Graph*. 2014;33(4):139:1-139:11. https://doi.org/10.1145/2601097.2601147.
29. Overby M, Brown GE, Li J, Narain R. ADMM ⊇ projective dynamics: fast simulation of hyperelastic models with dynamic constraints. *IEEE Trans Vis Comput Graph*. 2017;23(10):2222-2234. https://doi.org/10.1109/TVCG.2017.2730875.
30. Daviet G. Simple and scalable frictional contacts for thin nodal objects. *ACM Trans Graph*. 2020;39(4):61:61:1-61:61:16. https://doi.org/10.1145/3386569.3392439.
31. Le Lidec Q, Kalevatykh I, Laptev I, Schmid C, Carpentier J. Differentiable simulation for physical system identification; 2020. Available online (HAL).
32. Yunt K, Glocker C. Trajectory optimization of mechanical hybrid systems using SUMT. Paper presented at: Proceedings of the 9th IEEE International Workshop on Advanced Motion Control, Istanbul, Turkey; 2006:665-671.
33. Yunt K. An augmented Lagrangian based shooting method for the optimal trajectory generation of switching Lagrangian systems. *Dyn Contin Discrete Impuls Syst Ser B Appl Algorithms*. 2011;18(5):615-645.
34. Goldfarb D, Ma S, Scheinberg K. Fast alternating linearization methods for minimizing the sum of two convex functions. *Math Program*. 2013;141(1):349-382. https://doi.org/10.1007/s10107-012-0530-2.
35. Goldstein T, O'Donoghue B, Setzer S, Baraniuk R. Fast alternating direction optimization methods. *SIAM J Imaging Sci*. 2014;7(3):1588-1623. https://doi.org/10.1137/120896219.
36. Kadkhodaie M, Christakopoulou K, Sanjabi M, Banerjee A. Accelerated alternating direction method of multipliers. Paper presented at: Proceedings of the ACM Association for Computing Machinery KDD '15; 2015:497-506; New York, NY.
37. Ouyang W, Peng Y, Yao Y, Zhang J, Deng B. Anderson acceleration for nonconvex ADMM based on Douglas-Rachford splitting. *Comput Graph Forum*. 2020;39(5):221-239. https://doi.org/10.1111/cgf.14081.
38. Schubiger M, Banjac G, Lygeros J. GPU acceleration of ADMM for large-scale quadratic programming. *J Parall Distrib Comput*. 2020;144:55-67. https://doi.org/10.1016/j.jpdc.2020.05.021.
39. Stellato B, Banjac G, Goulart P, Bemporad A, Boyd S. OSQP: an operator splitting solver for quadratic programs. *Math Program Comput*. 2020. https://doi.org/10.1007/s12532-020-00179-2.
40. Negrut D, Serban R, Tasora A. Posing multibody dynamics with friction and contact as a differential complementarity problem. *ASME J Comput Nonlinear Dyn*. 2017;13(1):014503. https://doi.org/10.1115/1.4037415.
41. Wohlberg B. ADMM penalty parameter selection by residual balancing; 2017. arXiv: 1704.06209. arXiv:1704.06209 [cs, eess, math].
42. Xu Z, Figueiredo M, Goldstein T. Adaptive ADMM with spectral penalty parameter selection. Paper presented at: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR, Ft. Lauderdale, FL; 54; 2017:718-727.
43. Tasora A, Serban R, Mazhar H, et al. CHRONO: an open source multi-physics dynamics engine. In: Kozubek T, ed. *High Performance Computing in Science and Engineering – Lecture Notes in Computer Science*. New York, NY: Springer; 2016:19-49.
44. Schenk O, Gärtner K. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generat Comput Syst*. 2004;20(3):475-487. https://doi.org/10.1016/j.future.2003.07.011.
45. Beatini V, Royer-Carfagni G, Tasora A. A regularized non-smooth contact dynamics approach for architectural masonry structures. *Comput Struct*. 2017;187:88-100.

How to cite this article: Tasora A, Mangoni D, Benatti S, Garziera R. Solving variational inequalities and cone complementarity problems in nonsmooth dynamics using the alternating direction method of multipliers. *Int J Numer Methods Eng*. 2021;1–21. https://doi.org/10.1002/nme.6693