Efficient and Accurate Collision Response for Elastically Deformable Models

MICKEAL VERSCHOOR, Eindhoven University of Technology and Universidad Rey Juan Carlos
ANDREI C. JALBA, Eindhoven University of Technology

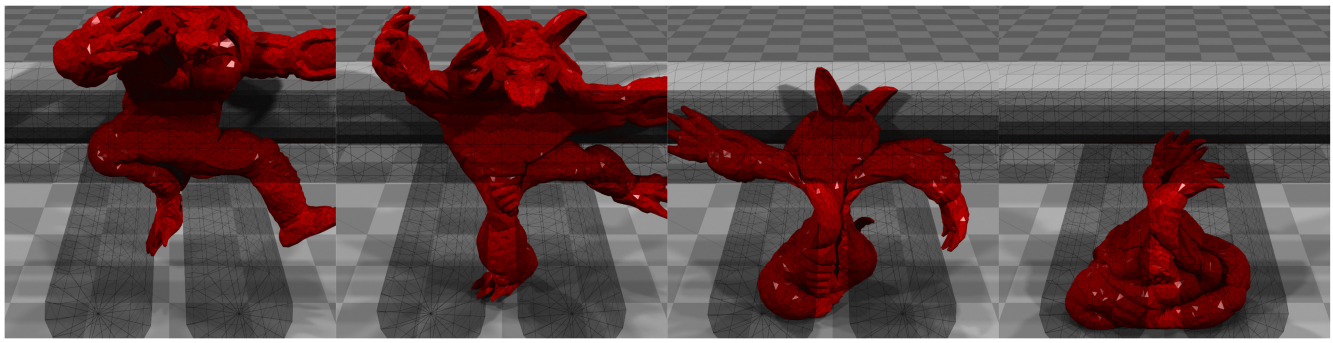


Fig. 1. A hyper-elastic Armadillo pulled through a set of (transparent) rotating cylinders at timesteps 5,000, 7,500, 10,000, and 12,500. The timing results of this experiment for various methods are shown later in Figure 10.

Simulating (elastically) deformable models that can collide with each other and with the environment remains a challenging task. The resulting contact problems can be elegantly approached using Lagrange multipliers to represent the unknown magnitude of the response forces. Typical methods construct and solve a Linear Complementarity Problem (LCP) to obtain the response forces. This requires the inverse of the generalized mass matrix, which is generally hard to obtain for deformable-body problems. In this article, we tackle such contact problems by directly solving the Mixed Linear Complementarity Problem (MLCP) and omitting the construction of an LCP matrix. Since a convex quadratic program with linear constraints is equivalent to an MLCP, we propose to use a Conjugate Residual (CR) solver as the backbone of our collision response system. By dynamically updating the set of active constraints, the MLCP with inequality constraints can be solved efficiently. We also propose a simple yet efficient preconditioner that ensures faster convergence. Finally, our approach is faster than existing methods (at the same accuracy), and it allows accurate treatment of friction.

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

Additional Key Words and Phrases: Collision response, conjugate residual

Authors' addresses: M. Verschoor, Department of Mathematics and Computer Science, Eindoven University of Technology, Eindhoven, P.O. Box 513, 5600 MB, The Netherlands, and Department of Computer Science, Universidad Rey Juan Carlos, Móstoles, Calle Tulipán s/n, 28933, Spain, Spain; email: mickeal.verschoor@gmail.com; A. C. Jalba, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, P.O. Box 513, 5600 MB, The Netherlands; email: a.c.jalba@tue.nl.

0730-0301/2019/03-ART17 $15.00
https://doi.org/10.1145/3209887

ACM Reference format:
Mickeal Verschoor and Andrei C. Jalba. 2019. Efficient and Accurate Collision Response for Elastically Deformable Models. *ACM Trans. Graph.* 38, 2, Article 17 (March 2019), 20 pages.
https://doi.org/10.1145/3209887

1 INTRODUCTION

Physically based models are now widely used in many computer graphics applications, such as animated movies and video games. Typical animations consist of many (complex) objects that can interact with each other and the environment. The behavior of such objects is based on their internal dynamics, modeled through some material simulation (e.g., rigid-body, cloth, or fluid simulation). However, the very motion of each object can also influence the motion of other objects in the scene. Therefore, accurate treatment of collision events dramatically improves the realism of the simulation, delivering rich and visually pleasing animations. In addition to translational and rotational motion, the motion of a deformable object is also affected by shape changes (deformations). Since the deformation is generally unknown, no exact collision time and location can be directly computed. Furthermore, the collision response influences the dynamics of the objects and thus their motion, and ideally, it should also account for accurate friction.

Typical approaches for handling contact problems reformulate them as velocity constraints and use *Lagrange multipliers* to represent the unknown magnitudes of the contact forces. Then, the resulting *Mixed Linear Complementarity Problem* (MLCP) has to be solved, which yields a solution that agrees with both the collision response problem and the model dynamics. The constrained problem is usually tackled by reformulating it as a *Linear Complementarity Problem* (LCP), which typically is solved using

(Projected) Gauss-Seidel methods. The construction of the LCP matrix requires the inverse of the generalized mass matrix, which can be directly computed for rigid-body simulations. However, for deformable bodies, this inverse is not directly available and has to be approximated (e.g., see Otaduy et al. (2009)). Therefore, one often seeks an approximation or solves a system of coupled problems. Alternatively, directly solving the original MLCP constitutes a viable approach for deformable models. Ramage and Wathen (1994) compared the performance of the Conjugate Residual (CR) method for solving coupled indefinite problems, stemming from finite-element discretizations of the Stokes equations, to that of a two-level approach in which two nested Conjugate Gradient (CG) solvers were used. Their results show that the CR method outperforms the coupled CG approach for such equality-constrained problems. Inspired by their results, we shall investigate here how the CR method can be extended for efficiently solving contact problems involving inequality constraints.

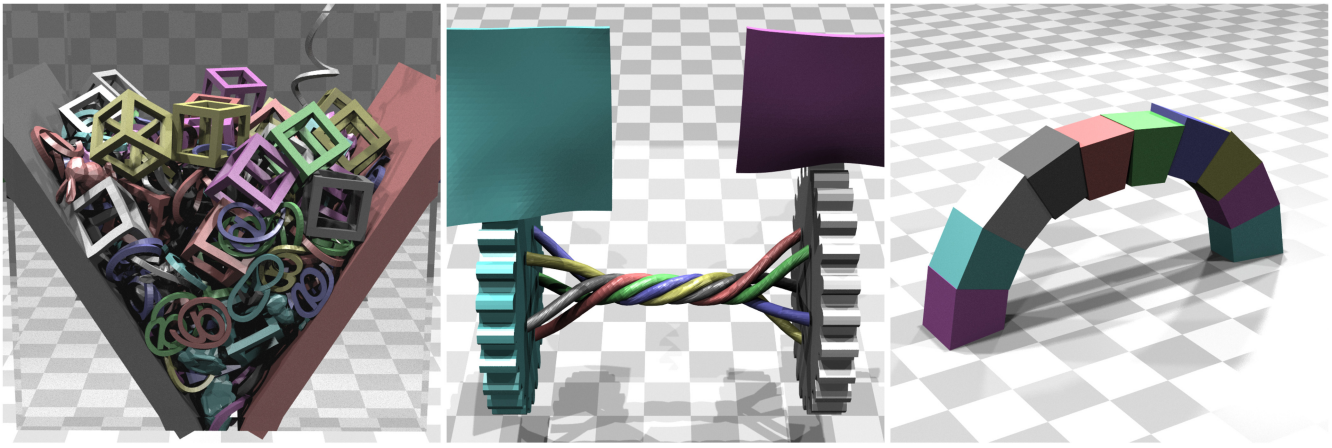
For stable simulations of coupled rigid and deformable bodies, it is important that all collisions are resolved completely at the end of each timestep. However, when a collision is resolved, the resulting deformation may lead to new collisions or changes in contact forces elsewhere. This often results in a state in which contacts do not agree with the actual geometry. If such errors are not corrected, these contacts will eventually introduce energy to the system, resulting in oscillations and instabilities. Additionally, the larger the deformation is, the more likely that such mismatches appear. Such mismatches can occur even for rigid bodies. To minimize these errors, one needs to solve non-linear contact problems.

In this article, we present a method that focuses on solving the contact problem for deformable bodies, without the need to compute the *Delassus operator*. Due to this, contacts can be efficiently linearized, which allows us to solve the non-linear contact problem, resulting in a guaranteed collision-free state at convergence. The method is based on the CR method, which simultaneously provides estimates for both deformation and contact forces. By allowing constraints to change status (switch among active or inactive states), inequality (*non-penetration*) constraints can be handled directly. Furthermore, friction is modeled similarly, such as by switching between static and kinetic friction constraints. By continuously updating the sliding directions of the kinetic friction constraints, Coulomb's friction model is approximated. Additionally, we propose a simple yet effective preconditioner that significantly reduces the number of iterations and computation time. The method allows for large deformations and contact forces, stable stacking of (almost) rigid objects, and an accurate approximation of Coulomb's friction cone (see Figures 1 and 2, and, later, Figure 8). Finally, our approach is faster compared to other methods, with the speedup typically increasing with the complexity of the simulation.

1.1 Previous and Related Work

A tremendous amount of work has been done addressing physical simulations for computer animation, including rigid-body, cloth, deformable-object, and fluid simulations. Here, we shall only give a very brief overview of highly related methods. For background material on contact mechanics, we refer to the work of Wriggers (2002) and references therein.

Rigid bodies. Although penalty methods for collision response (Barbič and James 2007; Moore and Wilhelms 1988) are relatively fast and easy to implement, the computed penalty forces have to compete with all other forces acting on the simulated bodies. Therefore, the forces may fail to avoid inter-penetration. Impulse-based methods (Baraff 1989) model the response of a collision event through the application of contact impulses. These methods apply impulses on objects to resolve collisions but can trigger new ones. This problem can be solved by using shock propagation techniques (Guendelman et al. 2003). Other approaches used in rigid-body simulations often model collision response as a constrained optimization problem (Baraff 1994; Erleben 2007; Müller et al. 2007; Redon et al. 2002). This class of methods also includes approaches based on LCP formulations (Cottle et al. 1992), typically used in rigid-body simulations (Baraff 1996), interactive applications (Catto 2005; Tonge et al. 2012), and cloth simulations (Bridson et al. 2002). LCPs are frequently used in combination with implicit time-integration schemes such that a collision-free state is guaranteed for the next timestep (Stewart and Trinkle 1996). It is this last property that makes such approaches very appealing for solving the contact problem for deformable models as well. Anitescu and Hart (2004) develop fixed-point iteration algorithms that solve convex sub-problems that are guaranteed, for small friction coefficients, to obtain the unique velocity solution of the non-convex friction LCP. Their method converges at a linear rate. A non-smooth non-linear CG method is presented by Silcowitz-Hansen et al. (2010b), which combines Projected Gauss-Seidel (PGS) with a Fletcher-Reeves CG method. Bertails-Descoubes et al. (2011) present a method for simulating contacts between small rods or fibers. The method models exact Coulomb friction and uses a non-smooth Newton solver, which relies on an easy construction of the Delassus operator. Xu et al. (2014) present a method for rigid bodies, which solves a Quadratic-Programming (QP) problem based on contact constraints. A Singular-Value-Decomposition (SVD) solver is used to form the (singular) Schur complement matrix, which is then used in combination with Cholesky decomposition and back-substitution. Friction is modeled using friction anchors. Although a penalty-based method, due to the implicit-integration scheme used, this approach can also be interpreted as a method that solves constraints. Mazhar et al. (2015) introduce the so-called Accelerated Projected Gradient Descent approach to accelerate the simulation of large systems of rigid bodies interacting through normal and frictional contact. The method can easily be parallelized, resulting in speedups of one to two orders of magnitude. Silcowitz-Hansen et al. (2010a) propose a PGS method working on a subspace of the problem. At each iteration, the method first performs an approximation using PGS, followed by a more accurate solve concerning a subset of active constraints having non-clamped multipliers. A detailed overview of numerical methods for solving LCPs is given by Erleben (2013). Finally, an extensive overview of rigid-body simulations is given by Bender et al. (2014). Other related projected Krylov methods, used in rigid many-body/granular-material simulations, are found in Heyn et al. (2012) and Renouf and Alart (2005).



(a) (b) (c)

Fig. 2. Example simulations by our method. (a) Difficult case with many deformable objects in a moving wedge. (b) Large deformations and elastic forces appear when the shown elastic strings are tangled; by releasing external forces, the strings are untangled. (c) Stable stacking of rigid objects, illustrating accurate friction treatment.

Deformable models. Early work on collision handling for deformable models includes the method of Baraff and Witkin (1992), which models the collision response through constraints yielding contact forces. By solving a QP problem, an optimal solution is obtained. Although not directly used for deformable objects, *Constraint Anticipation* (Baraff 1996) is a technique that transforms an MLCP into an LCP. After solving the LCP for the unknown Lagrange multipliers, collision response impulses are applied on the colliding models. When applied to rigid bodies, this technique is very efficient. Otaduy et al. (2009) extend this approach for deformable models as *Iterative Constraint Anticipation* (ICA). ICA computes the multipliers and the collision response using two nested (Projected) Gauss-Seidel solvers that approximate both the multipliers and velocities. Raghupathi and Faure (2006) use an *Active Set* approach for solving the contact response problem: Given a set of active and inactive constraints, a QP problem is solved using the CG method. On convergence, constraints can change state between active and inactive. The Staggered Projections (SP) method (Kaufman et al. 2008) computes an unconstrained velocity that is corrected using two coupled projection steps (each solved using a QP solver). This iterative process continues until the residual is minimized. The method of Allard et al. (2010) constructs volume-based constraints that ensure a zero intersection volume of two colliding objects. The resulting MLCP is solved using the Gauss-Seidel–like method of Duriez et al. (2006). The latter approximates exact Coulomb friction and explicitly constructs the Delassus operator (an LCP matrix), for which some approximations were made. Daviet et al. (2011) present a scalable and robust solver for capturing Coulomb friction in large assemblies of tightly packed fibers, such as hair. The method can handle a few thousand fibers subject to tens of thousands frictional contacts at a reasonable computational cost. Li et al. (2015) present an efficient Gradient Projection method for computing contact responses by decoupling constraints. Unfortunately, their method cannot handle coupled frictional constraints efficiently. Works on collision detection (Teschner et al. 2004), coupling of rigid and deformable bodies (Shinar et al. 2008), deformable bodies (Galoppo et al. 2006; Irving et al. 2007; Pauly et al. 2004; Spillmann et al. 2007), and penalty forces (Tang et al. 2012) are all closely related to the contact problem for deformable models.

Comparison to our method. We build on methods that involve velocity-level constraints and compute Lagrange multipliers in combination with implicit time integration. As seen earlier, many methods found in rigid-body applications require the construction of an LCP, Schur complement matrix, or Delassus operator, which relies on the availability of the inverted generalized mass matrix. For rigid-body problems, this inverse can be obtained relatively easily, whereas for Finite Element Method (FEM)-based deformable-body problems, this is generally not immediately possible. Therefore, many of these methods are not efficient for simulating deformable bodies with a large number of degrees of freedom (see Section 6.2.4). Indeed, for deformable-body problems, often coupled solvers (Kaufman et al. 2008; Otaduy et al. 2009) and/or PGS methods are used (Allard et al. 2010; Duriez et al. 2006). Otaduy et al. (2009) iteratively approximate interleaved normal and friction responses, but their method does not satisfy the Maximum Dissipation Principle (MDP). In Kaufman et al. (2008), this principle is guaranteed. This method iteratively solves two coupled QP problems and has been demonstrated for rigid and reduced-order flexible multibody systems. The method of Duriez et al. (2006) models exact Coulomb friction (satisfying the MDP) and solves the problem using a PGS approach.

All methods just mentioned first solve an unconstrained problem, which is corrected later, or first compute collision impulses, which are applied as external forces to objects. Our approach differs from all of these works in that no coupled solvers are used, neither is an LCP, Delassus, or Schur complement matrix constructed or used. We approximate the Coulomb friction model using an additional kinetic friction force, aligned with the sliding velocity, thus satisfying the MDP. The corresponding QP problem (containing the generalized mass matrix of the simulation and its

constraints) is solved using the CR method. By changing the state of the constraints, projections are performed that directly apply to the global problem such that friction and normal responses affect the internal (elastic) energy of the objects. In addition, constraints can be added or re-linearized at convergence to guarantee that a global, collision-free state is obtained, satisfying all constraints. To accelerate the convergence of the method, we derive a preconditioner that significantly reduces the amount of iterations. Finally, in this work, we create constraints per contact point, although we could also use constraints based on a volumetric description of a collision, similar to Allard et al. (2010).

2 BACKGROUND

2.1 Linear and Hyper-Elasticity

The deformation of a body can be described by a mapping $\boldsymbol{\phi}$ from material coordinates $\mathbf{X}$ to world coordinates $\mathbf{x}$ (i.e., $\mathbf{x} = \boldsymbol{\phi}(\mathbf{X})$). The stress $\mathbf{P}$ at a point $\mathbf{X}$ depends solely on the deformation gradient $\mathbf{F} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}$ and is obtained through some energy density function $\boldsymbol{\Psi}(\mathbf{F})$ via $\mathbf{P}(\mathbf{F}) = \frac{\partial \boldsymbol{\Psi}}{\partial \mathbf{F}}$. Since the stress is invariant under rotations, $\mathbf{P}(\mathbf{UF}) = \mathbf{U}\mathbf{P}(\mathbf{F})$ holds for any rotation $\mathbf{U}$. Furthermore, if the material is isotropic, $\mathbf{P}(\mathbf{F}\mathbf{V}^T) = \mathbf{P}(\mathbf{F})\mathbf{V}^T$ holds for any rotation $\mathbf{V}$. By diagonalizing $\mathbf{F}$ using SVD, $\mathbf{F}$ becomes $\mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$, so the stress can be obtained through

$$\mathbf{P}(\mathbf{F}) = \mathbf{U}\mathbf{P}(\hat{\mathbf{F}})\mathbf{V}^T, \tag{1}$$

with $\hat{\mathbf{F}}$ a diagonal matrix containing the singular values of the deformation gradient. Given the (isotropic) constitutional model used in $\boldsymbol{\Psi}$, a wide range of materials can be simulated. If $\boldsymbol{\Psi}$ is a quadratic function, a linear elastic model is obtained, for which the unrotated force gradient is constant; this coincides with co-rotational FEM (Müller and Gross 2004). Once the stress $\mathbf{P}$ is obtained, the nodal force $\mathbf{f}_i$ is obtained by

$$\mathbf{f}_i = -\mathbf{P}(\mathbf{F})\mathbf{b}_i, \tag{2}$$

with $\mathbf{b}_i$ the area-weighted normals of the faces connected to node i for a particular volume and stress $\mathbf{P}$ (see Irving et al. (2004)). By assembling the nodal forces for all volumes, net force $\mathbf{f}$ is obtained. To obtain the force gradient, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \sum_i \frac{\partial \mathbf{f}}{\partial \mathbf{F}_i} \frac{\partial \mathbf{F}_i}{\partial \mathbf{x}}$, the gradients of the SVD in Equation (1) with respect to $\mathbf{F}$ are required (see Sin et al. (2011) for its derivation). When dealing with hyper-elastic materials, the computed force responses can become very large (due to the non-linear nature of the energy model) when the compression becomes large. To robustly simulate such materials, we use the method of Stomakhin et al. (2012), which linearly extrapolates the energy density function after a certain amount of compression is reached.

2.2 Dynamics and Numerical Integration

Given Newton's second law of motion $\mathbf{Ma} = \mathbf{f}$, a first-order Taylor expansion of the net force $\mathbf{f}$ is performed, yielding

$$\mathbf{Ma} = \mathbf{f} + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial t} = \mathbf{f} + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{v} + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \mathbf{a}. \tag{3}$$

Using a first-order forward difference approximation of the acceleration $\mathbf{a}$, the following *semi-implicit* system is obtained,

$$\left(\mathbf{M} + \Delta t \mathbf{C}_d + \Delta t^2 \mathbf{K}\right) \mathbf{v}^{i+1} = \left(\mathbf{M} + \Delta t \mathbf{C}_d\right) \mathbf{v}^i + \Delta t \mathbf{f}, \tag{4}$$

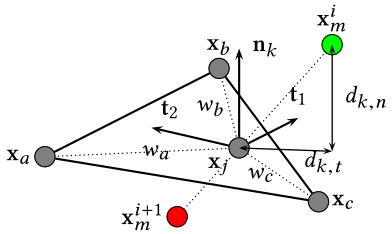


Fig. 3. Vertex-face contact pair. Vertex $\mathbf{x}_m^i$ (green) at iteration i penetrates a face at $i+1$ (red). $\mathbf{x}_j$ is the collision point on the face, w_a, w_b, w_c its barycentric weights, $\mathbf{n}_k$ the contact normal, and $\mathbf{t}_{1,2}$ the tangent vectors; $d_{k,n}$ represents the normal distance at the beginning of the timestep and $d_{k,t}$ the tangential distance.

with $\mathbf{M}$ the mass matrix of the system, $\mathbf{v}$ the velocity, damping matrix $\mathbf{C}_d = -\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$, and stiffness matrix $\mathbf{K} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ (see Baraff and Witkin (1998)). Positions $\mathbf{x}$ are updated using

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \Delta t \mathbf{v}^{i+1}. \tag{5}$$

We discretize Equation (4) in space using the FEM. Accordingly, for each element (tetrahedron), a per-element equivalent of Equation (4) is generated, given mapping $\mathbf{x} = \boldsymbol{\phi}(\mathbf{X})$. By assembling all of these local instances into a global one, the global version of Equation (4) is obtained, now with $\mathbf{M}, \mathbf{C}_d$, and $\mathbf{K}$ the *global* mass-, damping-, and stiffness matrix, and $\mathbf{x}, \mathbf{v}, \mathbf{f}$ now vectors representing positions, velocities, and all internal (Equation (2)) and external forces (e.g., *gravitational forces*) of all nodes in the discretized model. This system is stored using a large yet sparse matrix in the form $\mathbf{A}\mathbf{v}^{i+1} = \mathbf{b}$, with $\mathbf{A} \in \mathbb{R}^{3N_v \times 3N_v}$ the generalized mass matrix $(\mathbf{M} + \Delta t \mathbf{C}_d + \Delta t^2 \mathbf{K})$, $\mathbf{b}$ the right-hand side, and N_v the total number of vertices. Such systems are typically solved using the CG method for the unknown velocities $\mathbf{v}^{i+1}$. Given the new velocities, the positions are updated using Equation (5).

2.3 Non-Penetration Constraints

Two deformable bodies have collided if the signed distance between any two points on their surfaces Γ_1 and Γ_2 is negative. Such collision events are detected by considering *vertex-face* and *edge-edge* contact pairs. For the vertex-face case, let $\mathbf{x}_m^i \in \Gamma_1$ be a (discrete) vertex on the surface of the first object, which penetrates Γ_2 in the next timestep, resulting in a collision point $\mathbf{x}_j$ on a triangular face of Γ_2 (see Figure 3). The (signed) distance $d_{k,n}$ between $\mathbf{x}_j^i$ and $\mathbf{x}_m^i$ is computed using

$$C_k(\mathbf{x}^i) = \left(\mathbf{x}_m^i - \mathbf{x}_j^i\right) \cdot \mathbf{n}_k^i = d_{k,n}, \tag{6}$$

with $\mathbf{n}_k^i$ the *contact normal*, k the constraint identifier, and $\mathbf{x}^i$ a vector containing all vertex positions at timestep i. To avoid penetration, $C_k(\mathbf{x}^i) \geq 0$ must hold for any $\mathbf{x}_j$ and $\mathbf{x}_m$ pair. Clearly, this should also hold *after* the semi-implicit update (i.e., $C_k(\mathbf{x}^i + \Delta t \mathbf{v}^{i+1}) \geq 0$). Since Equation (4) solves for $\mathbf{v}^{i+1}$, Equation (6) is transformed into a velocity constraint using a first-order approximation—that is,

$$\begin{pmatrix} \Delta t \frac{\partial C_k}{\partial \mathbf{x}_j} & \Delta t \frac{\partial C_k}{\partial \mathbf{x}_m} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{v}_j^{i+1} \\ \mathbf{v}_m^{i+1} \end{pmatrix} \geq -d_{k,n} = c_{k,n}, \tag{7}$$

with $\mathbf{v}_j^{i+1}$ and $\mathbf{v}_m^{i+1}$ the new vertex velocities, and $c_{k,n}$ the constraint constant. After collecting all resulting non-penetration constraints and assembling all instances of Equation (7), one obtains

$$\mathbf{J}\mathbf{v}^{i+1} \geq \mathbf{c}_n, \tag{8}$$

with $\mathbf{J} \in \mathbb{R}^{N_c \times 3N_v}$ the Jacobian matrix containing the derivatives of *all* positional constraints, N_c the number of non-penetration constraints, N_v the number of vertices, $\mathbf{v}$ a vector representing all vertex velocities, and $\mathbf{c}$ the right-hand side of Equation (7), for all constraints. Since $\mathbf{x}_j = w_a\mathbf{x}_a + w_b\mathbf{x}_b + w_c\mathbf{x}_c$ and we consider vertex-face and edge-edge pairs, $\mathbf{J}$ contains per contact point a sparse row-vector $\mathbf{j}_k \in \mathbb{R}^{1\times 3N_v}$, *e.g.*, $\mathbf{j}_k = [-w_a\Delta t\mathbf{n}_k^T, -w_b\Delta t\mathbf{n}_k^T, -w_c\Delta t\mathbf{n}_k^T, \Delta t\mathbf{n}_k^T]$ for a vertex-face constraint C_k, with $\mathbf{n}_k$ the contact normal (see Figure 3). At this point, we consider $\mathbf{n}_k$ to be constant within one timestep. In Section 3.3, an extension is proposed that takes the change of the constraints into account. Furthermore, self-collisions in which $\Gamma_1 = \Gamma_2$ and degenerate collisions (vertex-vertex/vertex-edge) are treated similarly. However, the latter will produce duplicate contacts that need special care (see Section 3.4.1).

2.4 Constrained Velocity as a Complementarity Problem

We model collision response using Lagrange multipliers so that normal contact forces are given by $\mathbf{f}_n = \mathbf{J}^T\boldsymbol{\lambda}$, with $\boldsymbol{\lambda}$ the vector of Lagrange multipliers representing the unknown magnitudes of all contact forces. According to Signorini's contact model (see also Duriez et al. (2006)), a complementarity condition is further imposed—to prevent penetration, the contact forces should push the bodies apart, and hence $\boldsymbol{\lambda} \geq \mathbf{0}$. Furthermore, if the non-penetration constraint is not violated ($C_k(\mathbf{x}) > 0$ for some constraint), its corresponding multiplier should be $\lambda_k = 0$. Hence, $(\mathbf{J}\mathbf{v} - \mathbf{c}_n)^T\boldsymbol{\lambda} = (\mathbf{v}^T\mathbf{J}^T - \mathbf{c}_n^T)\boldsymbol{\lambda} = \mathbf{v}^T\mathbf{f}_n - \mathbf{c}_n^T\boldsymbol{\lambda} = 0$, which gives the complementarity condition.

Contact forces are simply included in the motion equations of the model (see Section 2.2), resulting in

$$\mathbf{0} \leq \begin{pmatrix} \mathbf{A} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \end{pmatrix} - \begin{pmatrix} \mathbf{b} \\ \mathbf{c}_n \end{pmatrix} \perp \begin{pmatrix} \mathbf{1} \\ \boldsymbol{\lambda} \end{pmatrix} \geq \mathbf{0} \tag{9}$$

(see Cottle et al. (1992)), which is an MLCP for which the Karush-Kuhn-Tucker (KKT) optimality conditions (Kuhn and Tucker 1950) apply.

3 COLLISION RESPONSE THROUGH THE CR METHOD

MLCPs can be solved by transforming them into LCPs (Baraff 1996), provided that matrix $\mathbf{A}$ can be easily inverted. They can also be solved by any algorithm for strictly convex QP problems with linear constraints. Given the MLCP from Equation (9), the corresponding QP problem is

$$\begin{aligned} \min f(\mathbf{v}) &= \frac{1}{2}\mathbf{v}^T\mathbf{A}\mathbf{v} - \mathbf{v}^T\mathbf{b} \\ \text{subject to } &\mathbf{J}\mathbf{v} \geq \mathbf{c}_n. \end{aligned} \tag{10}$$

A sufficient condition to guarantee strict convexity is for matrix $\mathbf{A}$ to be positive definite. In this case, $f(\mathbf{v})$ is a strictly convex function, and if the constraints also are linear, it has a unique global minimizer (Boyd and Vandenberghe 2004). Since according to Equation (4), $\mathbf{A}$ is given by the linear combination of the (positive-definite) mass and (positive semi-definite) stiffness and damping matrices, $\mathbf{A}$ is positive definite. If the material simulation does not produce a positive semi-definite matrix, this property must be enforced, for example, by removing the negative eigenvalues from the element stiffness matrices.

The quadratic function in Equation (10), but subject to $\mathbf{J}\mathbf{v} = \mathbf{c}_n$, is equivalent to

$$\underbrace{\begin{pmatrix} \mathbf{A} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix}}_{\mathbf{B}}\underbrace{\begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} \mathbf{b} \\ \mathbf{c}_n \end{pmatrix}}_{\mathbf{d}}, \tag{11}$$

or $\mathbf{B}\mathbf{y} = \mathbf{d}$ in short. Since matrix $\mathbf{B} \in \mathbb{R}^{(3N_v+N_c)\times(3N_v+N_c)}$ is not positive definite, the system in Equation (11) cannot be solved by the CG method. The CR method (Luenberger 1970) is a Krylov subspace method, similar to the (more popular) CG method, and minimizes $\|\mathbf{B}\mathbf{y} - \mathbf{d}\|^2$ and thus solves Equation (11). In the following sections, we describe how the CR method is used to solve the contact problem, including friction. Additionally, we describe how constraints are re-linearized to obtain a collision-free state.

3.1 CR Method With Non-Penetration Constraints

Luenberger (1973) proposed a method for solving constrained non-linear programming problems by treating inequality constraints as equality ones that can regularly change states. The method uses a *descent step* to improve the current approximation. When the problem is a QP with linear constraints, the CR method can be used to perform these descent steps. In the following, we describe how the method in Luenberger (1973) is applied in case of contact and friction and how it is used in combination with the CR method in Algorithm 1.

For now, let j denote the loop index inside the CR method. Furthermore, recall that each constraint C_k is represented by a sparse row-vector $\mathbf{j}_k$ in Jacobian $\mathbf{J}$, its corresponding Lagrange multiplier λ_k, and some constant $c_{k,n}$ (see Section 2.3). The complementarity condition in Equation (9) states that if $\lambda_k > 0$ is true, then $\mathbf{j}_k\mathbf{v} = c_{k,n}$ also holds for some constraint C_k, with $\mathbf{v}$ the global velocity. Conversely, if $\lambda_k = 0$ is true, then $\mathbf{j}_k\mathbf{v} \geq c_{k,n}$ and $(\mathbf{j}_k\mathbf{v} - c_{k,n})^T\lambda_k = \mathbf{0}$ also hold. Therefore, constraint C_k does not contribute to the computation of $\mathbf{v}$ and can be *deactivated* in line 13 or line 23. Due to this mechanism, summarized in Algorithms 2 and 3, the CR method can solve the problem in Equation (10). In other words, each constraint is *active* or *inactive* throughout the solver iterations, and its state is regularly updated. The method converges when $\|\mathbf{r}_{j+1}\| < \epsilon_1$ and $\|\mathbf{U}\mathbf{r}_{j+1}\|_\infty < \epsilon_2$, with ϵ_1 and ϵ_2 a relative and absolute tolerance, respectively, and $\mathbf{U}$ a matrix that selects the entries from $\mathbf{r}$ that involve active constraints. This ensures a maximum constraint error less than ϵ_2. The second convergence criterion is used to detect local minima (see Section 4.3). Furthermore, when the CR method converges, new constraints can be added and inactive constraints are removed. However, the main difference with active set methods is that constraints' state changes are allowed before the solver converges. Therefore, the problem is to decide *when* to activate or deactivate constraints *prior* to convergence. Note that when a constraint is inactive, multiplications $\mathbf{B}\mathbf{p}_j$ and $\mathbf{B}\mathbf{y}_{j+1}$ are performed such that the corresponding entries in $\mathbf{J}$ are neglected. This also applies for the corresponding terms of the preconditioner $\mathbf{C}$ (see Section 3.4 for details about

ALGORITHM 1: Pseudo-code of our collision handling system.

```
1  Discretize computational domain;
2  Initialize BVH and additional data structures;
3  while Simulating do
4      Update matrix A using FEM (see Equation (4));
5      Find all potential collision candidates (Section 5.1);
6      Linearize and check all active constraints;
7      r_0 = d − By_0; p_0 = C^{-1} r_0, j = 0;
8      while Not converged do
9          α = (r_j^T C^{-1} B C^{-1} r_j) / (p_j^T B C^{-1} B p_j);
10         y_{j+1} = y_j + α p_j;
11         r_{j+1} = r_j − α B p_j;
12         if α > 0 then
13             EvaluateConstraints(‖r_{j+1}‖, j, ε_1, ε_2);
14         if No constraints changed then
15             β = −(r_{j+1}^T C^{-1} B C^{-1} r_{j+1}) / (r_j^T C^{-1} B C^{-1} r_j);
16             p_{j+1} = C^{-1} r_{j+1} − β p_j;
17             Bp_{j+1} = BC^{-1} r_{j+1} − β Bp_j;
18         else
19             r_{j+1} = d − By_{j+1};
20             p_{j+1} = C^{-1} r_{j+1};
21             Bp_{j+1} = BC^{-1} r_{j+1};
22         if (‖r_{j+1}‖ < ε_1 ∧ ‖U r_{j+1}‖_∞ < ε_2) ∨ ‖BC^{-1} r_{j+1}‖ < ε_1
           then
23             EvaluateConstraints(‖r_{j+1}‖, 0, ε_1, ε_2);
24             if No constraint states have changed then
25                 Check all candidates for new collisions
                   (Section 5.2);
26                 if No new constraints are added then
27                     Re-linearize constraints (Section 3.3 and
                       Equation (21));
28                     if No constraints are updated then
29                         Advance simulation;
30                         break;
31             Recompute residual (see lines 19–21);
32         j = j + 1;
```

ALGORITHM 2: Pseudo-code for constraint evaluations.

```
1  Function EvaluateConstraints(r, i, ε_1, ε_2):
2      interval = max(1, ⌊ log^2(r/ε_1) / 3 ⌋);
3      if i mod interval == 0 ∨ i == 0 then
4          foreach contact k do
5              if Non-penetration constraint active then
6                  if j_k v − c_{k,n} ≥ 0 and λ_k ≤ 0 then
7                      Deactivate constraint, λ_k = 0;
8              else if j_k v − c_{k,n} ≤ 0 then
9                  Activate constraint, set friction to kinetic friction;
10             if Non-penetration constraint active then
11                 if Friction constraint in static friction mode then
12                     if ‖γ_k‖ > μλ_k and (j_{k,t} v − c_{k,t})^T γ_k > 0
                       then
13                         Switch to kinetic friction;
14                         δ̂_k = γ̂_k, λ'_k = ‖γ_k‖/μ, γ_k = 0;
15                 else if (j_{k,t} v − c_{k,t})^T δ̂_k ≤ 0 then
16                     Switch to static friction;
17                     γ_k = γ'_k, γ'_k = 0;
18                 UpdateKineticFriction();
19             if Z^{-1} S_d [Δλ_k, Δγ_k^T + Δγ'^T_k]^T > ε_2 then
20                 Use updated constraint k and report change to
                   solver;
21             else
22                 Discard changes for constraint k;
23         d = [(b − J_t γ')^T, c_n^T, c_t^T]^T /*Update RHS*/;
```

preconditioning). Furthermore, when the residual is recomputed for inactive constraints, the values in $\mathbf{d}$ corresponding to inactive constraints are also neglected.

Constraint activation. The activation of a constraint C_k depends on $c_{k,n}$, $\mathbf{j}_k$ and $\mathbf{v}$ at solver iteration j. Distance $d_{k,n}$ is computed using Equation (6) and stored in $c_{k,n}$ when the constraint is added to the system (when a collision is detected), or when the constraint is re-linearized. At the same moment, $\mathbf{j}_k$ is computed and included in $\mathbf{J}$ (see Section 5 for more details). To determine when a constraint should be activated (assuming it is inactive), the penetration depth is evaluated regularly. If

$$\mathbf{j}_k\mathbf{v} - c_{k,n} \leq 0 \tag{12}$$

holds, the constraint is activated and the corresponding Lagrange multiplier will be computed, which would eventually resolve the collision (see line 8 of Algorithm 2).

Constraint deactivation. Since our solver allows for activation and deactivation of constraints at any time, a criterion is needed to decide when to deactivate them. It may seem logical to deactivate a constraint C_k if $\mathbf{j}_k\mathbf{v} - c_{k,n} \geq 0$. However, since $\mathbf{v}$ does not reflect the final velocity, this criterion cannot *solely* be used. When both $\mathbf{j}_k\mathbf{v}$ and λ_k are used instead, the criterion can be formulated as follows: if

$$\mathbf{j}_k\mathbf{v} - c_{k,n} \geq 0 \text{ and } \lambda_k \leq 0 \tag{13}$$

hold, then there is no collision and the constraint is attracting the objects—that is, it tries to enforce the equality $\mathbf{j}_k\mathbf{v} = c_{k,n}$. Therefore, the constraint is deactivated (see line 6 of Algorithm 2). This is thus the only state that allows deactivation of a non-penetration constraint. For all other combinations, the constraint is kept active since there is still a collision ($\mathbf{j}_k\mathbf{v} - c_{k,n} < 0$) or objects are repelled ($\lambda_k > 0$).

Unlike Luenberger (1973), our CR-based solver for contact problems updates *only* the active set at dynamic intervals determined by the residual norm (see line 2 of Algorithm 2). When the residual is small, constraints are evaluated more frequently. Conversely, for a larger residual norm, constraints are evaluated less frequently, which reduces the number of residual evaluations (each requiring a multiplication $\mathbf{By}$). This makes our method more efficient than

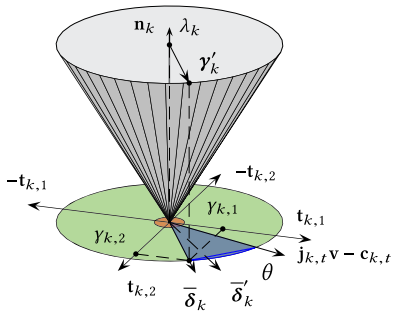


Fig. 4. A representation of the friction cone in which sliding velocity $\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t}$ is misaligned by angle θ (blue segment) with the friction vector $\overline{\boldsymbol{\delta}}_k$. To address this, we compute the new friction direction $\overline{\boldsymbol{\delta}}'_k$ that is more aligned with the sliding velocity (see Algorithm 3). If the magnitude of the sliding velocity is below the solver tolerance (red area), no update of $\overline{\boldsymbol{\delta}}_k$ is performed.

the CR solver in Luenberger (1973). Additionally, our method supports a left preconditioner (matrix $\mathbf{C}^{-1}$ in Algorithm 1).

3.2 Incorporating Friction Constraints

Friction is also treated as a constraint on the velocity. According to Coulomb's friction model, friction can be described by a cone that bounds the friction force $\mathbf{f}_f$ in all directions, given the magnitude of the normal force $\mathbf{f}_n$ and a friction coefficient μ (see Figure 4). When $\|\mathbf{f}_f\| < \mu\|\mathbf{f}_n\|$, the friction force is not bounded and the sliding velocity is zero. When $\|\mathbf{f}_f\| = \mu\|\mathbf{f}_n\|$, the friction force is bounded and is not large enough to keep the sliding velocity at zero. Equation (9) can therefore be extended with additional complementarity conditions, resulting in the following problem:

$$0 \leq \begin{pmatrix} \mathbf{A} & \mathbf{J}^T & \mathbf{J}_t^T & \mathbf{0} \\ \mathbf{J} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_t & \mathbf{0} & \mathbf{0} & -\mathbf{e}^T \\ \mathbf{0} & \boldsymbol{\mu} & -\mathbf{e} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix} - \begin{pmatrix} \mathbf{b} \\ \mathbf{c}_n \\ \mathbf{c}_t \\ 0 \end{pmatrix} \perp \begin{pmatrix} \mathbf{1} \\ \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix} \geq \mathbf{0}, \tag{14}$$

with $\mathbf{J}_t^T\boldsymbol{\gamma} = \mathbf{f}_f$ the applied friction force, $\mathbf{J}_t \in \mathbb{R}^{N_p N_c \times 3N_v}$ a matrix containing tangent vectors for all friction constraints, $\boldsymbol{\gamma}$ a vector of unknown multipliers representing the magnitude of these vectors, and $\mathbf{c}_t$ containing all tangential distances for all constraints. Per constraint, the tangential distances $\mathbf{c}_{k,t}$ (see Figure 3) are computed similar to Equation (6), albeit the normal vector is replaced by a tangent one. Please note that $\mathbf{c}_{k,t}$ represents the distance from configuration $\mathbf{x}_m^i$ to the first moment of impact at $\mathbf{x}_j$. For persistent contacts, $\mathbf{c}_{k,t} = \mathbf{0}$ since $\mathbf{x}_m^i$ is already in contact. Matrix $\mathbf{J}_t$ usually contains, per contact point, a number N_p of scaled instances of tangent vectors, each corresponding to one facet of the discretized friction cone. Furthermore, $\mathbf{e}$ and $\boldsymbol{\mu}$ are matrices used to couple all multipliers in $\boldsymbol{\lambda}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. The latter representing the unknown sliding velocities. If β_k is positive for some friction constraint C_k, a positive sliding velocity exists and the friction force is bounded, resulting in dissipation of energy. If β_k is zero, the friction force is not bounded and no sliding occurs. Unfortunately, Equation (14) is not symmetric and non-convex, and is therefore not equivalent to a QP problem. In the next section, a modification is proposed that restores this relation.

ALGORITHM 3: Pseudo-code for kinetic friction update.

```
1  Function UpdateKineticFriction():
2      if Constraint k in kinetic friction mode then
3          if ||γ'_k|| ≠ μλ_k then
4              Compute Simple Moving Average λ_{k,a};
5              λ'_k = λ_{k,a};
6          if 0 < δ̄_k^T (j_{k,t}v − c_{k,t}) / ||j_{k,t}v − c_{k,t}|| < cos(ε_θ) /*If angle θ > ε_θ*/ then
7              Δδ_k = (j_{k,t}v − c_{k,t}) / ||j_{k,t}v − c_{k,t}|| − δ̄_k /*Compute difference*/;
8              δ̄_k = δ̄_k + α_θ Δδ_k /*Decreases θ*/;
9          γ'_k = μλ'_k δ̄_k /*Update friction force*/;
```

3.2.1 Modified Friction Model. Given the model described in Equation (14), we now proceed by adapting this model. Instead of using a discretized friction cone, a continuous and bounded kinetic friction force is modeled. This approach is summarized in Algorithms 2 and 3, and discussed here. Per constraint C_k, the friction force is decomposed using *only* two perpendicular tangent vectors $\mathbf{t}_{k,1}$ and $\mathbf{t}_{k,2}$, which appear in $\mathbf{j}_{k,t} \in \mathbb{R}^{2\times 3N_v}$ as part of $\mathbf{J}_t$ (see Figure 4). Due to this, $\boldsymbol{\gamma}$ contains, per constraint C_k, two Lagrange multipliers $(\gamma_{k,1}, \gamma_{k,2})^T = \boldsymbol{\gamma}_k$, which can be both *positive and negative*. For clarity of exposure, we assume for now that all friction constraints in the system are either in the static or kinetic state. To also keep the presentation brief, we may not make a clear distinction between model and method.

Static friction. For static friction, the sliding velocity is zero ($\boldsymbol{\beta} = \mathbf{0}$), so Equation (14) can be written as

$$0 \leq \begin{pmatrix} \mathbf{A} & \mathbf{J}^T & \mathbf{J}_t^T \\ \mathbf{J} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_t & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \end{pmatrix} - \begin{pmatrix} \mathbf{b} \\ \mathbf{c}_n \\ \mathbf{c}_t \end{pmatrix} \perp \begin{pmatrix} \mathbf{1} \\ \boldsymbol{\lambda} \\ |\boldsymbol{\gamma}| \end{pmatrix} \geq \mathbf{0}, \tag{15}$$

with $|\boldsymbol{\gamma}|$ the componentwise absolute values of $\boldsymbol{\gamma}$, and the additional condition that $\mathbf{0} \leq \boldsymbol{e\gamma} \leq \boldsymbol{\mu\lambda}$, which for a constraint C_k means that $0 \leq \|\boldsymbol{\gamma}_k\| \leq \mu\lambda_k$. The second inequality should hold for the static case, because the friction force is not bounded by the friction cone, and the solver computes a friction force that satisfies $\mathbf{j}_{k,t}\mathbf{v} = \mathbf{c}_{k,t}$. Conversely, if both

$$\|\boldsymbol{\gamma}_k\| > \mu\lambda_k \text{ and } (\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t})^T\boldsymbol{\gamma}_k > 0 \tag{16}$$

hold, the applied friction force is too large and acts in the direction of the sliding velocity; thus, the conditions for static friction are violated and the constraint is switched to kinetic friction (see line 12 of Algorithm 2).

Kinetic friction. For kinetic friction, the sliding velocity is positive ($\boldsymbol{\beta} > \mathbf{0}$), which makes Equation (14) non-symmetric. Such a system can be solved as described in Bender et al. (2014). Alternatively, this system can be symmetrized by removing slack vector $\boldsymbol{\beta}$ from the equation. This has the advantage that solutions always exist and can be found by Lemke's algorithm (Stewart and Trinkle 1996). Unfortunately, dropping $\boldsymbol{\beta}$ results in a violation of the MDP, as $\boldsymbol{\beta}$ selects a discrete vector in $\mathbf{J}_t$ that ensures this principle. In our method, the MDP is enforced differently.

Our approach for handling kinetic friction could be interpreted as a *Bounded Linear Complementarity Problem (BLCP)* (Judice and Pires 1994) reformulation of Equation (14), where the alignment

of the discretized friction cone depends on the state of the system. The relation in the last row of Equation (14) is enforced separately. This means that the kinetic friction force will never be overestimated, as it could happen in the BLCP model.

For kinetic friction, $\boldsymbol{\beta} > \mathbf{0}$, thus $\boldsymbol{\mu\lambda} = \mathbf{e}\boldsymbol{\gamma}$, which for a particular constraint C_k means that $\mu\lambda_k = \|\boldsymbol{\gamma}_k\|$. The sliding velocity is given by $\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t} = \mathbf{e}^T\boldsymbol{\beta} > \mathbf{0}$. The basic idea is now to estimate both the magnitude and direction of $\boldsymbol{\gamma}_k$.

Let $\boldsymbol{\gamma}'_k = (\mu\lambda'_k\delta_{k,1}, \mu\lambda'_k\delta_{k,2})^T$ be such an estimate, with λ'_k an approximation of λ_k, and $\overline{\boldsymbol{\delta}}_k = \overline{\delta_{k,1}\mathbf{t}_{k,1} + \delta_{k,2}\mathbf{t}_{k,2}}$ a unit vector, representing the direction of the kinetic friction force in the tangent plane, for example, at $\mathbf{x}_j$ (see Figure 4). Next, we eliminate from Equation (14) all rows and columns corresponding to $\boldsymbol{\beta}$ and move $\mathbf{J}_t^T\boldsymbol{\gamma}'$ to the right-hand side so that our approximation can be expressed in matrix form as

$$\mathbf{0} \leq \begin{pmatrix} \mathbf{A} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \end{pmatrix} - \begin{pmatrix} \mathbf{b} - \mathbf{J}_t^T\boldsymbol{\gamma}' \\ \mathbf{c}_n \end{pmatrix} \perp \begin{pmatrix} \mathbf{1} \\ \boldsymbol{\lambda} \end{pmatrix} \geq \mathbf{0}, \tag{17}$$

supplemented per constraint by the condition $(\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t})^T\overline{\boldsymbol{\delta}}_k > 0$. This condition is violated when the sliding velocity changes direction due to a too large friction force—that is, when

$$(\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t})^T\overline{\boldsymbol{\delta}}_k \leq 0. \tag{18}$$

By then, switching the constraint to static friction, the solver will estimate $\boldsymbol{\gamma}_k$, which eventually satisfies $\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t} = \mathbf{0}$ (see line 15 of Algorithm 2).

Kinetic friction update. Earlier, a correct approximation of $\boldsymbol{\gamma}'_k$ was assumed to be available; here, we show how this vector is updated such that the friction force and the sliding velocity are aligned, and that the magnitude of the friction force is bounded. Given the preceding conditions, a violation occurs if

$$\mu\lambda_k \neq \|\boldsymbol{\gamma}'_k\| = \| \left(\mu\lambda'_k\delta_{k,1}, \mu\lambda'_k\delta_{k,2}\right)^T \|. \tag{19}$$

If this happens, *Simple Moving Average* (SMA) $\lambda_{k,a}$ is updated by adding the current λ_k to its history of n values ($n = 2$ in our implementation; see line 4 of Algorithm 3). Next, λ'_k is set to the SMA of $\lambda_{k,a}$ at line 5. To impose the MDP, the kinetic friction direction $\overline{\boldsymbol{\delta}}_k$ and the sliding velocity have to be kept aligned. When the angle θ (see Figure 4) between $\overline{\boldsymbol{\delta}}_k$ and the current sliding velocity is larger than a given threshold ($\epsilon_\theta = 1.2$ degrees in our implementation), the difference $\Delta\overline{\boldsymbol{\delta}}_k$ between both vectors is computed (line 7). Next, $\overline{\boldsymbol{\delta}}_k$ is updated using $\overline{\boldsymbol{\delta}}_k + \alpha_\theta\Delta\overline{\boldsymbol{\delta}}_k$ and then normalized, with α_θ a small step size (0.01 in our implementation; see line 8 of Algorithm 3). Thus, $\overline{\boldsymbol{\delta}}_k$ is gradually updated to realign it with the current sliding velocity. Finally, $\boldsymbol{\gamma}'_k$ is approximated (see line 9 of Algorithm 3). Using this strategy, eventually all kinetic friction vectors approach their final configuration and satisfy the conditions specified in Equation (17). By using this strategy, a stabilization mechanism is introduced that suppresses undesired oscillations while solving the contact problem. Finally, in a real simulation, the system contains both static and kinetic friction constraints, so the problem can be described in matrix form as

$$\mathbf{0} \leq \begin{pmatrix} \mathbf{A} & \mathbf{J}^T & \mathbf{J}_t^T \\ \mathbf{J} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_t & \mathbf{0} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \end{pmatrix} - \begin{pmatrix} \mathbf{b} - \mathbf{J}_t^T\boldsymbol{\gamma}' \\ \mathbf{c}_n \\ \mathbf{c}_t \end{pmatrix} \perp \begin{pmatrix} \mathbf{1} \\ \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \end{pmatrix} \geq \mathbf{0}, \tag{20}$$

where, for static constraints, entries in $\boldsymbol{\gamma}'$ are set to zero, whereas for kinetic constraints, all entries in $\boldsymbol{\gamma}$, $\mathbf{c}_t$ and those of matrix $\mathbf{B}$ corresponding to $\mathbf{J}_t, \mathbf{J}_t^T$ are similarly set to zero.

3.3 The Non-Linear Contact Problem

The problem solved in Equation (20) assumes that $\mathbf{J}$ and $\mathbf{J}_t$ are constant during the timestep, which generally is not true. Hence, $C(\mathbf{x}^{i+1}) \perp \boldsymbol{\lambda}$ may not hold at the end of the timestep. To minimize such errors, the complementarity conditions must also hold after updating the geometry. To solve this non-linear problem, $C(\mathbf{x})$ is re-linearized using an interpolated approximation of $\mathbf{x}$ based on its previous and current approximations, followed by a solve of Equation (20). Instead of interpolating $C(\mathbf{x})$ globally, one can re-linearize each individual $C_k(\mathbf{x})$—that is,

$$C_k^{l+1'}(\mathbf{x}_{i+1}) = wC_k^{l+1}(\mathbf{x}_{i+1}) + (1-w)C_k^l(\mathbf{x}_i), \tag{21}$$

with $0 < w < 1$ a weight such that the change in the contact normal is less than, say, 15 degrees, and C_k^l indicates the last, C_k^{l+1} the current, and $C_k^{l+1'}$ the new interpolated configuration of the constraining geometry. The re-linearization of the constraints results in an updated $\mathbf{J}, \mathbf{J}_t$, and $\mathbf{c}$ (line 27 of Algorithm 1). Then, the method recomputes the preconditioner and the residual vector, and continues until it converges again for the linearized problem. This procedure continues until, for each active constraint, $0 < C_k(\mathbf{x}) < \epsilon_2$ holds. When $w = 0$, this procedure is exactly Newton's method in which the constraints are re-linearized followed by a solve of the updated Equation (20). Since line 25 of Algorithm 1 updates the set of potential collisions, a collision-free state is eventually obtained. Additionally, when a contact point slides off a face or edge, the corresponding constraint must be removed from the system. The main advantage of Algorithm 1 is that updating $\mathbf{J}, \mathbf{J}_t$, and $\mathbf{c}$ does not require an update of the LCP matrix or Delassus operator. This allows us to efficiently solve the non-linear contact problem for large systems.

3.4 Preconditioning

In this section, we derive a preconditioner that is easy to compute, yet it significantly reduces the amount of iterations compared to a simple diagonal preconditioner. A common approach for improving the convergence rate of Krylov-subspace methods is to use a preconditioner matrix $\mathbf{C}^{-1}$. Many methods exist for approximating preconditioner matrices, such as LU and Cholesky factorizations (Golub and Van Loan 1996). Unfortunately, these methods assume that the matrix is positive definite, and therefore these preconditioners are not useful for our indefinite problem from Equation (20).

Algorithm 1 shows the pseudo-code of our preconditioned CR solver with inequality constraints. Matrix $\mathbf{B}$ in the solver represents the matrix from Equation (20) and $\mathbf{C}^{-1}$ the preconditioner matrix such that $\mathbf{C}^{-1} \approx \mathbf{B}^{-1}$. The blockwise inverse of $\mathbf{B}$ is

$$\mathbf{B}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{L}^T\mathbf{S}^{-1}\mathbf{L}\mathbf{A}^{-1} & (\mathbf{S}^{-1}\mathbf{L}\mathbf{A}^{-1})^T \\ \mathbf{S}^{-1}\mathbf{L}\mathbf{A}^{-1} & -\mathbf{S}^{-1} \end{pmatrix}, \tag{22}$$

with $\mathbf{L}^T = (\mathbf{J}^T\ \mathbf{J}_t^T) \in \mathbb{R}^{3N_v \times 3N_c}$ the combined constraint matrix and $\mathbf{S} = \mathbf{L}\mathbf{A}^{-1}\mathbf{L}^T$. Next, we approximate matrices $\mathbf{A}$ and $\mathbf{S}$ by

$\mathbf{A}_d = \text{diag}(\mathbf{A})$ and $\mathbf{S}_d = \text{diag}(\mathbf{Z}\mathbf{L}\mathbf{A}_d^{-1}\mathbf{L}^T)$, respectively, with $\text{diag}(\cdot)$ extracting the diagonal part of its matrix argument and $\mathbf{Z}$ a diagonal scaling matrix (see the following). Thus, the computations of $\mathbf{A}_d^{-1}$ and $\mathbf{S}_d^{-1}$ become trivial. Additionally, $\mathbf{S}_d^{-1}\mathbf{L}\mathbf{A}_d^{-1}$ is easy to evaluate and yields the same sparsity structure as that of $\mathbf{L}$. Based on this, we construct the following preconditioners:

Full: $\mathbf{C}^{-1}$ is obtained from Equation (22) by replacing $\mathbf{A}$ and $\mathbf{S}$ by $\mathbf{A}_d$ and $\mathbf{S}_d$, respectively;

Diag: $\mathbf{C}^{-1} = \begin{pmatrix} \mathbf{A}_d^{-1} & 0 \\ 0 & \mathbf{S}_d^{-1} \end{pmatrix}$.

When the preconditioner is applied, only those parts corresponding to active non-penetration and static friction constraints are considered. To construct the *Full* preconditioner, we first compute the quantities $\mathbf{S}_d^{-1}, \mathbf{S}_d^{-1}\mathbf{L}\mathbf{A}_d^{-1}, \mathbf{A}_d^{-1}\mathbf{L}^T$ for each constraint and use them when evaluating the blocks of the preconditioner (see Equation (22)). Furthermore, when constraints are added, removed, or re-linearized, these precomputed quantities are also updated.

If a positive-definite preconditioner is used, like our Diag preconditioner, then the norm of the preconditioned residual will decrease monotonically when the states of the constraints are not updated. As we will discuss in Section 4.2, updating constraint states does not always result in a decrease of $\|\mathbf{r}\|$ (see Algorithm 1), but the approximation will always be closer to the new optimum. When an indefinite preconditioner is used, the evolution of $\|\mathbf{r}\|$ is more chaotic. More importantly, α can also be negative, which indicates that the current optimum for the preconditioned problem lies in the opposite direction of $\mathbf{p}$. Such an update does improve the preconditioned problem ($\|\mathbf{r}^T\mathbf{C}^{-1}\mathbf{r}\|$), although generally not the unpreconditioned problem ($\|\mathbf{r}^T\mathbf{r}\|$). Since the constraint updates are based on the unpreconditioned values $\mathbf{x}$ and $\boldsymbol{\lambda}$, an update is only performed if a step toward the unpreconditioned optimum is taken (i.e., when $\alpha > 0$).

3.4.1 Preconditioner Scaling. The Full preconditioner approximated by Equation (22) significantly improves the convergence rate of the method. Unfortunately, this preconditioner will have large deviations in the upper-left block when $\mathbf{B}$ becomes singular or ill conditioned due to (nearly) duplicate constraints in $\mathbf{L}$. When $\mathbf{L}$ is full rank, both the inverse and pseudo-inverse of $\mathbf{B}$ are equal. By adding duplicate rows to $\mathbf{L}$, $\mathbf{B}$ and thus $\mathbf{S}$ become singular. In our approximation, $\mathbf{S}_d$ does not become singular. Computing then the approximated block inverse for our preconditioner using $\mathbf{S}_d$ yields a change in the upper-left block of the preconditioner compared to the non-singular case. If this error becomes too large, the CR method converges significantly slower. In contrast, the pseudo-inverse of $\mathbf{B}$ does not introduce such a change.

To keep the upper-left block invariant for duplicates in $\mathbf{L}$, it is sufficient to see that duplicate rows and columns *can* be added together. As a result, the diagonal values in $\mathbf{S}_d$ *would* have been scaled up by the number of duplicates for their corresponding constraint. Since we do not add duplicates together, the same scaling must be introduced to $\mathbf{S}_d$ using $\mathbf{Z}$, which contains, per constraint in $\mathbf{L}$, the number of similar or duplicate instances. When computing the upper-left block, duplicate constraints are added implicitly through the multiplication $\mathbf{L}^T\mathbf{S}_d^{-1}\mathbf{L}$. Since this multiplication also involves $\mathbf{S}_d^{-1}$ and thus $\mathbf{Z}^{-1}$, the difference introduced by the duplicates is compensated by $\mathbf{Z}^{-1}$.

Similar or identical constraints occur, for example, when an edge intersects other edges close to one of their shared vertices. The barycentric coordinate for the shared vertex is relatively large for all constraints. The other involved vertices have a small contribution, which makes the constraints almost identical to each other. For constraints involving deformable bodies, it is sufficient to compute, for each pair of constraints a and b, a weight based on the differences between their barycentric coordinates (i.e., $(1 - |w_{a,i} - w_{b,i}|)$, with i a vertex id). The similarity of a pair is the product of these factors for each vertex that a and b have in common. Eventually, the total scaling factor for one constraint is the sum of all of these products. For contacts between rigid bodies (Figure 2(c)), many constraints *can* act on the same degrees of freedom of the rigid objects. Here the amount of duplicates, and thus the error in the preconditioner, can be potentially large. In this case, it is sufficient to scale all constraints working on the same two bodies by the total number of constraints between those bodies. This is in some way similar to the method of Tonge et al. (2012).

3.5 Optimizations

The convergence rate of the method described in Algorithm 1 is improved by allowing constraints to update their status at dynamic intervals. In other words, the closer the method is to convergence, the smaller the intervals are between successive constraint evaluations (see line 2 of Algorithm 2). Eventually, constraints are evaluated every iteration when the residual is small. If we *always* allow constraints to update their status every iteration, in the worst case, the solver would need to recompute the residual vector as in line 19 every iteration, instead of using the recurrence from line 15. The performance would then be comparable to that of a *Gradient Descent* method, which can converge slowly due to *zigzagging*. Conversely, when constraints are *only* allowed to update their status at convergence, one has an *Active Set* method. In this case, the number of iterations between successive constraint evaluations *can be* relatively large. Due to this, the solver *can* enter a loop in which constraints continuously change their status. This *constraint cycling* drastically affects solver convergence and should be avoided.

By evaluating constraints at dynamic intervals, the solver responds fast enough to situations in which constraints *should* change states, although also slow enough for the solver to exploit the conjugacy of the residual vectors. A positive side effect of this strategy is that the number of residual re-evaluations (see line 19) is significantly reduced. We have run a large number of experiments while tuning the relation between the computed interval and the residual norm. We have found that this strategy works very well for efficiently solving contact-response problems. This approach gave good results in all test cases but are not optimal for all cases.

Within our method, the possibility still exists that constraints change their status a large number of times, decreasing the convergence rate. To prevent this, one approach is to keep (at some point) constraints in a fixed state, as mentioned in Raghupathi and Faure (2006). Unfortunately this results in a violation of the conditions specified in Sections 3.1 and 3.2, which could introduce additional energy to the system. However, even in such cases, the

current approximation is able to move toward the optimum, meaning that the method is still able to converge (see Luenberger (1973)).

When a constraint is about to change (see line 19 of Algorithm 2), the change is postponed if the change in distance is less than ϵ_2. In this case, the effect of performing the update is negligible. This strategy also eliminates undesired oscillations of constraint states (constraint cycling) that have no effect on the final result. This usually occurs when the solution for a particular constraint is located close to a discontinuity in the derivatives (i.e., when a vertex is about to slide or when two entities are barely touching).

Each time constraints change states, vectors $\mathbf{r}$, $\mathbf{p}$, and $\mathbf{Bp}$ must be recomputed to restore the conjugacy of the residual vector. Since the computation of the residual requires a full multiplication with matrix $\mathbf{B}$, this is not very efficient. However, since each residual vector can be decomposed as $\mathbf{r} = \mathbf{r}_u + \mathbf{r}_c$, with $\mathbf{r}_u = \mathbf{b} - \mathbf{Av}$ the unconstrained residual and $\mathbf{r}_c$ the remaining constraint residual, a multiplication with matrix $\mathbf{A}$ can therefore be completely omitted since $\mathbf{r}_u$ is not affected by a state change. Indeed, only $\mathbf{r}_c$ needs to be recomputed, which significantly improves performance. Please note that this decomposition requires that other intermediate vectors and computations are decomposed similarly.

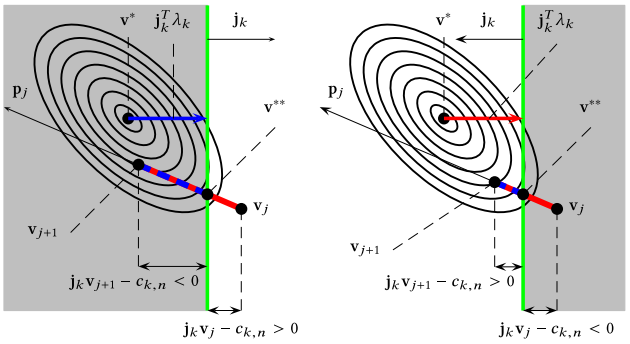
4 CONVERGENCE

In this section, we discuss the convergence behavior of our method, given the criteria described in Section 3 for switching constraints. First, we show that a minimal residual, in combination with all constraints satisfied, implies that a global solution is found. Next, we show that successive updates of constraints lead to a state in which all constraints are satisfied. Finally, we show that when all constraints are satisfied but the residual does not reach the minimum, its gradient will reach the minimum instead.

4.1 Global Convergence

The Delassus operator $(\mathbf{LA}^{-1}\mathbf{L}^T)$, where $\mathbf{L}^T = (\mathbf{J}^T \quad \mathbf{J}_t^T)$, describes, for *each pair of constraints* that share the *same* object in the inverted generalized mass matrix $\mathbf{A}^{-1}$, their influence on each other. The *inverted* Delassus operator can therefore be seen as the "solution" to the contact problem and describes the influence of *each* pair of constraints onto each other, no matter how many objects and constraints are present in the chain between the two constraints. Hence, this solution is a global one. In our method, the Delassus operator is not used. However, the CR method minimizes $\|\mathbf{By} - \mathbf{d}\|$, with $\mathbf{y} = \mathbf{B}^{-1}\mathbf{d}$ its minimum. Given the blockwise inverse of $\mathbf{B}$ (see Equation (22)), minimizing $\|\mathbf{By} - \mathbf{d}\|$ also implicitly applies the Delassus operator. This is equivalent to solving both $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ through the Delassus operator, and applying these forces to the bodies to obtain the unknown velocity $\mathbf{v}$.

Within our method, the states of the constraints are actively switched when they do not agree with the currently computed $\mathbf{v}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\lambda}$. When some constraints change states, the method *must* reset vectors $\mathbf{r}, \mathbf{p}$, and thus $\mathbf{Bp}$ (i.e., a state change influences the residual energy of the system). This reset allows the method to search in a direction that minimizes the energy according to the new state. Since all constraints and the deformation problem in sub-matrix $\mathbf{A}$ contribute to the residual, a sufficiently small residual should imply that the problem is solved. However, the solution can only be the global minimizer if all constraints are in their final state at convergence . If not, then constraint states are changed, the residual will be influenced, and thus the current solution cannot be a global minimizer for the problem. The method then continues until all constraints are satisfied.



(a) Constraint activation (b) Constraint deactivation

Fig. 5. Activation and deactivation of a constraint.

Assuming that the constraint states are known and only active constraints are considered, a global minimum exists if the constraints are convex and $\mathbf{A}$ and the second derivatives of the constraints (which vanish for linear constraints) are symmetric and positive definite (see Luenberger (1973)). At this global minimum, all multipliers of all active constraints are strictly positive (or not clamped in case of friction constraints) and their distances are zero. When inactive constraints are considered as well, their multipliers are zero and their distances are strictly positive. Hence, the complementarity conditions are satisfied, and the complementarity error vanishes (see Section 6.3). Let us define such a residual that, once minimized, solves Equation (20)—that is,

$$\mathbf{r}_g = \begin{pmatrix} \mathbf{Av} - \mathbf{b} + \mathbf{J}_A^T(\boldsymbol{\lambda})^+ + \mathbf{J}_{t,S}^T(\boldsymbol{\gamma})^{\mu\lambda} + \mathbf{J}_{t,K}^T(\boldsymbol{\gamma}')^{\mu\lambda} \\ (\mathbf{J}_A\mathbf{v} - \mathbf{c}_n) + (\mathbf{J}_I\mathbf{v} - \mathbf{c}_n)^- \\ \overline{\boldsymbol{\gamma}}^T(\mathbf{J}_{t,S}\mathbf{v} - \mathbf{c}_t) + \left(\overline{\boldsymbol{\delta}}^T(\mathbf{J}_{t,K}\mathbf{v} - \mathbf{c}_t)\right)^- \end{pmatrix}, \tag{23}$$

with $\mathbf{J}_A$ and $\mathbf{J}_I$ corresponding to all active and inactive non-penetration constraints; similarly, $\mathbf{J}_{t,S}$ and $\mathbf{J}_{t,K}$ correspond to all static and kinetic friction constraints, respectively. Furthermore, the multipliers and distances are clamped by operators $(\cdot)^-, (\cdot)^+, (\cdot)^{\mu\lambda}$ between $(-\infty, 0), (0, \infty)$, and $(-\mu\lambda, \mu\lambda)$, respectively. At the global minimum, $\|\mathbf{r}_g\| = 0 \Rightarrow \|\mathbf{r}\| = 0$. Conversely, $\|\mathbf{r}\| = 0 \Rightarrow \|\mathbf{r}_g\| = 0$ if and only if no constraints are violated. Unfortunately, the final constraint states for which this relation holds are not known *a priori*.

4.2 Constraint Updates

In the following, we reason about the implications of updating constraint states, as illustrated by Figure 5. We consider only non-penetration and kinetic friction constraints, as static friction can be treated similar to non-penetration constraints.

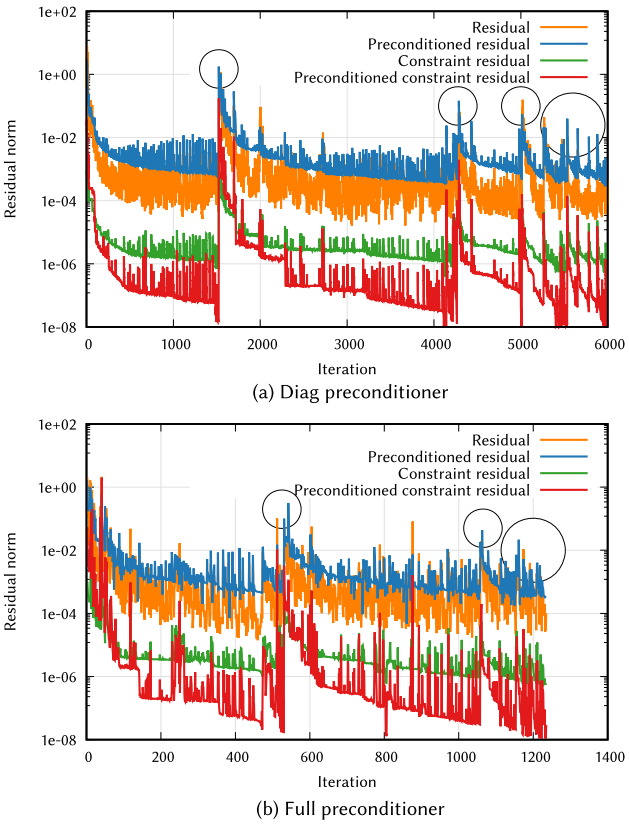
Activation of non-penetration constraints. Figure 5(a) shows the activation of an inactive constraint C_k at iteration j. Its

distance $\mathbf{j}_k\mathbf{v}_j - c_{k,n}$ is positive, thus not contributing to $\mathbf{r}_g$ nor to $\mathbf{r}$. By minimizing $\|\mathbf{r}\|$ in the direction of $\mathbf{p}_j$, a new approximation $\mathbf{v}_{j+1}$ is obtained, which is closer to the current minimum $\mathbf{v}^*$ (Luenberger 1970). If distance $\mathbf{j}_k\mathbf{v}_{j+1} - c_{k,n}$ becomes negative, the constraint is activated. Due to this active constraint, the *new* minimum $\mathbf{v}^{**}$ along $\mathbf{p}_j$ now lies on the (green) plane defined by the constraint, which intersects the path between $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$. Since $\mathbf{v}_{j+1}$ is now closer to the new optimum $\mathbf{v}^{**}$ along $\mathbf{p}_j$ than $\mathbf{v}_j$ was to the previous optimum, this constraint switch results in an improvement of the approximation. However, the distance between $\mathbf{v}_{j+1}$ and $\mathbf{v}^{**}$ appears as an additional residual term via $\mathbf{j}_k\mathbf{v}_{j+1} - c_{k,n} = \mathbf{j}_k(\mathbf{v}_{j+1} - \mathbf{v}^{**})$, through the first term in the second row of Equation (23). Depending on the used preconditioner, $\|\mathbf{r}\|$ might or might not decrease. However, since approximation $\mathbf{v}_{j+1}$ will now be closer to the *new* optimum, this step including the constraint activation is an improvement.

Deactivation of non-penetration constraints. Figure 5(b) shows the deactivation of a constraint C_k. At iteration j, distance $\mathbf{j}_k\mathbf{v}_j - c_{k,n}$ is negative. Next, a new step in direction $\mathbf{p}_j$ is performed, with $\mathbf{v}_{j+1}$ the new approximation. When both $\lambda_k < 0$ and $\mathbf{j}_k\mathbf{v}_{j+1} - c_{k,n} > 0$ hold, the constraint must be deactivated. The negative multiplier in combination with a positive distance indicates that the minimum lies on the positive side of the constraint and not on its plane, and hence the constraint is working in the wrong direction and can be deactivated. Since $\mathbf{j}_k\mathbf{v} - c_{k,n}$ is zero somewhere between $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$, and the new minimum $\mathbf{v}^*$ lies on the positive side of the constraint, $\mathbf{v}_{j+1}$ is closer to $\mathbf{v}^*$ than $\mathbf{v}_j$ was. Hence, the state is improved. Since the positive distance $\mathbf{j}_k\mathbf{v}_{j+1} - c_{k,n}$ and negative multiplier are clamped in Equation (23), $\|\mathbf{r}_g\|$ decreases. The actual $\|\mathbf{r}\|$ again might or might not decrease, because it is based on active constraints and possible negative multipliers.

Delayed evaluation. When constraints are evaluated and switched at fixed intervals, n optimization steps are performed that move the current approximation closer to the current optimum. This can also be considered as a single optimization step. If constraints do change their state after n steps, a new optimum is obtained and the method proceeds in that direction. This only demonstrates that the method eventually converges to an optimum. The choice of n largely affects the performance of the method. In general, the larger n is, the larger is the possibility that the method *overshoots*, which results in a smaller overall improvement per n optimization steps. Therefore, we keep n large for large residuals and decrease n with the residual norm (see line 2 of Algorithm 2).

Kinetic friction constraint update. The update of a kinetic friction constraint changes the magnitude λ'_k and direction $\overline{\boldsymbol{\delta}}_k$ of the kinetic friction force. Since the kinetic friction force is placed on the right-hand side of Equation (20), any update of $\overline{\boldsymbol{\delta}}_k$ and λ'_k results in a change of $\|\mathbf{r}\|$. When directions $\overline{\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t}}$ and $\overline{\boldsymbol{\delta}}_k$ are misaligned, $\overline{\boldsymbol{\delta}}_k$ is updated as described in Section 3.2. Since each updated $\overline{\boldsymbol{\delta}}_k$ is somewhere between the previous $\overline{\boldsymbol{\delta}}_k$ and $\overline{\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t}}$ due to *moving average* $\boldsymbol{\delta}_{k,a}$, the angle between these vectors becomes smaller (see Figure 4). Due to this, also the change in $\mathbf{j}_{k,t}^T\overline{\boldsymbol{\delta}}_k\mu\lambda'$ (the kinetic friction force; see the right-hand side of Equation (20)) becomes smaller. Therefore, successive increases of $\|\mathbf{r}\|$ due to these updates (through the last term in the first row of Equation (23)) become smaller and eventually vanish when $\overline{\boldsymbol{\delta}}_k$ and $\overline{\mathbf{j}_{k,t}\mathbf{v} - \mathbf{c}_{k,t}}$ are parallel.



(a) Diag preconditioner

(b) Full preconditioner

Fig. 6. Plots of several residual norms for preconditioners Diag and Full of the simulation timestep shown in Figure 7. The non-linear updates of the constraints are indicated by circles. The preconditioned residual norm is computed as $\|\mathbf{C}^{-1/2}\mathbf{r}\|$, the constraint residual norm as $\|\mathbf{U}\mathbf{r}\|_\infty$, and the preconditioned constraint residual norm as $\|\mathbf{U}\mathbf{C}^{-1/2}\mathbf{r}\|_\infty$. Due to the infinity norm, the shown (preconditioned) constraint residuals represent an upper bound of the actual constraint residual.

Convergence plots. Figure 6 depicts the convergence behavior of our method for both preconditioners for the simulation timestep shown in Figure 7. Figure 6(a) shows the residual norms obtained with the Diag preconditioner. The residual shows a similar behavior compared to the *preconditioned residual* and *(preconditioned) constraint residuals*. Although the residual does not provide much information about the convergence, the preconditioned residual $\|\mathbf{C}^{-1/2}\mathbf{r}\|$ shows a decreasing behavior on the lower bound. A state change of a constraint results in an increase of the residuals. After continuing for a few iterations, the preconditioned residual is (in most cases) smaller compared to its norm before the state update. Additionally, the (preconditioned) constraint residual shows an overall decreasing trend on the upper bound of the error, indicating that, in general, the update of the constraint states eventually results in a better approximation. This behavior also explains the effect of the optimization discussed in Section 3.5 in which the

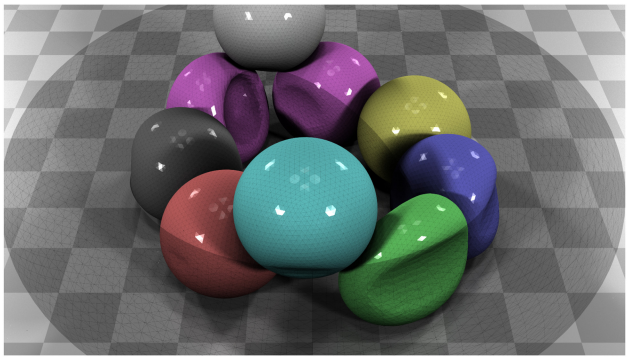


Fig. 7. Test case in which nine deformable spheres roll between a (transparent) rigid torus and a rotating spherical object. Figure 6 depicts the residual norms at a certain simulation iteration.

constraints are evaluated at a certain rate depending on the residual norm. This allows the method proceed toward the new optimum for a few iterations until constraints are re-evaluated. The large peaks in the plots correspond to the non-linear updates of the constraints. The intervals between successive non-linear updates become smaller as the approximation approaches the final solution (see Section 3.3). Figure 6(b) shows the (preconditioned) residuals obtained with the Full preconditioner, which shows less correspondence with the (preconditioned) constraint residuals. However, the constraint residuals show the same behavior compared to the Diag preconditioned case, albeit the Full preconditioned case results in a much faster convergence. The constraint tolerance for both experiments were $\epsilon_2 = 10^{-6}$ and a relative tolerance for the residual of $\epsilon_1 = 10^{-5}$.

4.3 Local Minima

Whenever the problem is over-constrained, no unique solution exists. Hence, the problem has local minima. At such a local minimum, the gradient vanishes. Since the CR method minimizes $\|\mathbf{C}^{-1/2}\mathbf{r}\|^2 = |\mathbf{r}^T\mathbf{C}^{-1}\mathbf{r}|$, the gradient of the minimized function $(2\mathbf{B}\mathbf{C}^{-1}\mathbf{r})$ becomes zero at a local minimum. When this happens, α becomes zero in line 9 of Algorithm 1, so the approximation cannot be further improved. Next, the computation of β will result in a division by zero, causing the method to break down. To prevent this, the norm of the gradient function (second test in line 22 of Algorithm 1) is also inspected, as proposed by Hayami (2002). If this norm drops below ϵ_1, the method has computed a least-square approximation for the current local minimum. This allows the method to perform the checks in lines 23 through 25 and to update the problem.

5 SYSTEM OVERVIEW

Non-penetration and friction constraints act on subsets of vertices of the colliding models. To constrain the movement globally, all *vertex-face* and *edge-edge* collision pairs should be detected, collected, and assembled in (global) matrices $\mathbf{J}$ and $\mathbf{J}_t$. Since the solution of Equation (20) is collision free, a *standard* collision detection system would not find any collisions in the beginning of each simulation timestep. This suggests that all collisions would have to be detected inside the internal loop of the CR method. However, doing so would be prohibitively expensive in terms of computations.

Algorithm 1 gives the overall pseudo-code of our collision handling system, which is further described in the following sections. Within our approach, we search once for all possible *candidate* collision pairs at the beginning of each simulation timestep (see line 5 of Algorithm 1). Then, all existing constraints are updated. When the solver converges, a collision check is performed (see line 25). If no new collisions are found, then all active constraints are checked if their configuration and state agree with the actual geometry and velocity; if necessary, constraints are re-linearized or removed from the system (see line 27). Finally, if the system is not updated, the state of the simulation is guaranteed to be collision free, and the error made by each constraint is guaranteed to be less than the desired tolerance.

5.1 Collision Detection

When the simulation is initialized, a *Bounding Volume Hierarchy* (BVH) is constructed, containing axis-aligned bounding volumes enclosing all surface faces (triangles) of the simulated bodies (which coincides with the simulation meshes; see line 2 of Algorithm 1); our hierarchical data structure is a binary tree. The BVH is used to quickly find potential face-face collisions by testing for intersection of their bounding volumes. Since deformable objects can have self-collisions, all faces of an object must also be tested for collisions with all other faces of the same object. Since bounding volumes of adjacent faces will intersect, one approach to discard false positives is to terminate the tree traversal when both faces belong to the same (self-collision free) surface patch (see Volino and Thalmann (1994)). When there is a possibility that both faces can collide at the end of the timestep, both faces must be included in further collision checks to guarantee a collision-free state at the end of the timestep.

To detect nearby faces of a given face, we extend each bounding box by $3\Delta t\mathbf{v}$ in the directions of its vertex velocities. For each face represented in the BVH, a set is created that stores all nearby and potentially colliding faces. All faces whose *extended bounding volumes* intersect the one of the current face are selected by traversing the tree. These face-face pairs are then used to create so-called candidate lists. The candidate lists contain, for each vertex/edge, a list of all nearby faces/edges (the candidates) that can potentially collide; duplicate edge-edge pairs are discarded.

5.2 Raw Collision Detection

At convergence, all candidate pairs (vertex-face and edge-edge) are checked for collisions given the current velocity (see line 25). First, each collision pair is updated (time integrated) using the current velocity. If the pair has intersected, a root-finding method, such as the Brent-Dekker method (Press et al. 1992), is used to find the collision point, contact normal, and barycentric coordinates, given the current and initial geometry state (at the beginning of the timestep). Then, the corresponding non-penetration constraint is initialized, activated, and stored in matrix $\mathbf{J}$ (see Section 2.3). Likewise, the corresponding friction constraints are initialized and stored in $\mathbf{J}_t$. Since this collision check is performed when the solver converges, new constraints are added as long as the state is not

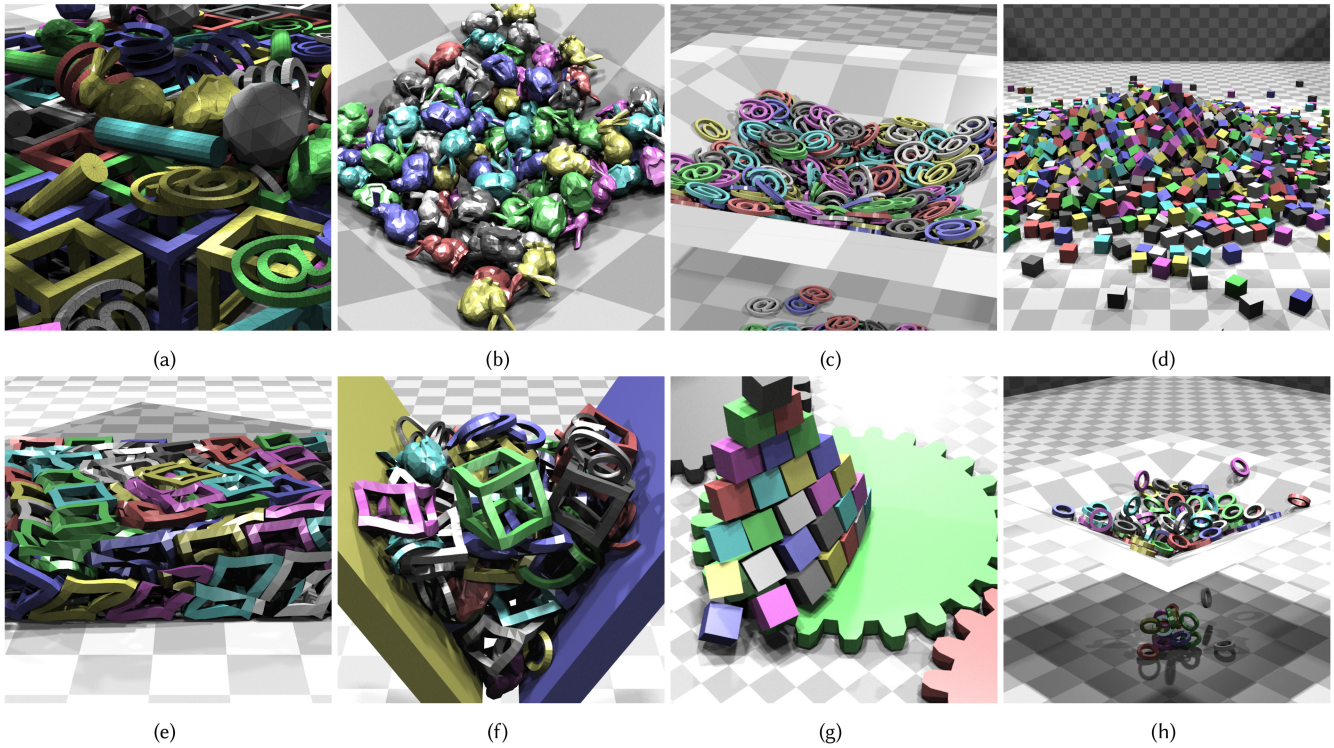


Fig. 8. Simulations of multiple elastic objects. The collision response is efficiently computed through our method, which can handle complex situations such as tens of thousands of constraints, large impact forces, and complex interactions (see the text and accompanying video).

collision free. This is similar to the Constraint Manifold Refinement (CMR) method of Otaduy et al. (2009), to ensure that no collisions/constraints are missed. Please note that the collision check relies on the *candidate* pairs so that (multiple) tree traversals are avoided. In some exceptional cases, a vertex can move out of its extended bounding volume such that collisions might be missed. If such a case is detected at convergence (line 25), using a simple vertex-in-box check, we need to search locally for additional missed collisions. This search is done by first computing the extended bounding box and rechecking the BVH for collisions with the updated box. If any new box-box intersections are found, the corresponding candidate lists are updated, and a collision check is performed on the updated candidate lists. The overhead of this additional check is relatively small—up to 5% of the total collision detection time.

6 RESULTS

In this section, we validate our approach by providing quantitative and qualitative results, and comparing these to those of existing methods. An overview of example results is shown in Figure 8.

6.1 Varying Parameters

The plot in Figure 9 shows the average number of solver iterations during the first second of the experiment shown in Figure 8(e) for different values of Δt. In general, a larger Δt requires more iterations per simulation step, but fewer simulation steps are needed for advancing the simulation to a given timestamp. However, in all cases, the total amount of solver iterations is comparable. We

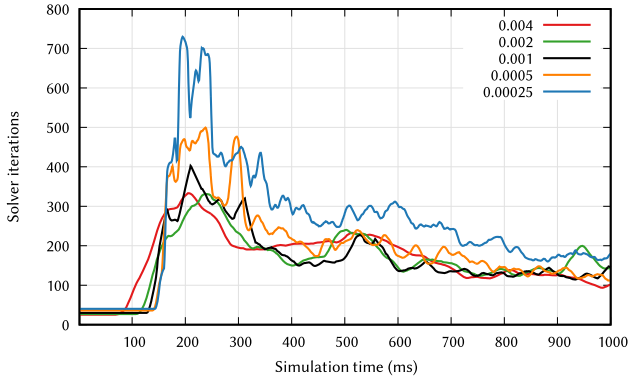


Fig. 9. Number of solver iterations per simulation step for different values of $\Delta t(s)$ and the experiment shown in Figure 8(e), with $E = 5 \times 10^6 Pa$.

observed that simulations with smaller timesteps reveal subtle motions of objects but require more iterations; however, such details are not visible when larger timesteps are used.

We have also changed the stiffness parameter E while keeping Δt fixed at 0.001s. A smaller stiffness parameter generally results in faster convergence for unconstrained problems. When the objects are less stiff, we observed that the method requires more Newton steps due to larger deformations, and this often requires more iterations. For instance, the setup shown in Figure 8(e) is simulated using varying stiffness parameters starting from $5 \times 10^6 Pa$ to $3.125 \times 10^5 Pa$ by dividing E by two for every simulation. Only for the case $E = 3.125 \times 10^5 Pa$ was the total amount of iterations

about 1.5 times larger than for $E = 5 \times 10^6 Pa$. In all other cases, these numbers were similar. Please note that it is difficult to quantify these results because a different stiffness parameter results in a different behavior of the material (and simulation). In case of a collision, the time that objects have contact is larger for flexible objects, which influences the behavior of the method.

6.2 Comparative Results

In this section, we compare various versions of our method against ICA (Otaduy et al. 2009) and SP (Kaufman et al. 2008). For this comparison we have simulated three setups with varying difficulty degrees (see Figures 8(h), (e), and (b)). All methods used double-precision arithmetic and converged to the same absolute error ($\epsilon_1 = 5 \times 10^{-5}$). Additionally, our method uses a threshold for updating vectors $\overline{\boldsymbol{\delta}}_k$ of $\epsilon_\theta = 1.2$ degrees and used a constraint tolerance/safety distance $\epsilon_2 = 5 \times 10^{-6}$. All objects had the following material properties: $\nu = 0.2$ (Poisson's ratio), $E = 5 \times 10^5 Pa$ (Young's modulus), and $\rho = 1{,}000\text{kg/m}^3$ (density). Furthermore, a gravitational acceleration of $g = 9.81\text{m/s}^2$ was used. The FEM machinery is based on co-rotated finite elements (Müller and Gross 2004), and no additional damping was used. Our test machine was equipped with a 4GHz AMD FX-8350 CPU (using one core) and 16GB of RAM memory. All methods use a warm start (reuse values from the previous simulation iteration), the same collision detection system, the same optimized routines for vector-vector and matrix-vector operations, and the same tolerances/precision in an attempt to make a fair comparison.

The first experiment from Figure 8(h) contains 120 randomly placed elastic rings, simulated with a timestep $\Delta t = 0.001$s. The total number of contact points grew to more than 1,500. The second experiment (Figure 8(e)) contains 140 objects, which are aligned in a pyramid. For Δt, we used 0.0005. The experiment shown in Figure 8(b) uses complex objects in which many internal elements exist. For this experiment, we randomly placed 120 bunnies with a Young's modulus E of $5 \times 10^6 Pa$ and simulated with $\Delta t = 0.001s$. The friction coefficient was set to 0.5 for all simulations. Table 1 shows, for each method, the average (and median) time spent on collision detection, contact resolution, solving the whole contact problem, and the average number of iterations performed by all methods. Furthermore, additional details about the simulations are provided. The application of the Full preconditioner required about 15% of the total time, whereas its construction time was negligible—less than 0.1%.

6.2.1 Hyper-Elastic Materials. In addition to linear FEM, we also made a comparison using highly deformable, hyper-elastic models. The stiffness of hyper-elastic materials changes with the amount of compression. For a neo-Hookean material, the stress will approach infinity when the volume of the object approaches zero. Due to this, numerical methods have to deal with large contact forces. Unfortunately, directly using these materials *can* cause problems, as small changes in deformation *could* result in excessively large forces. To deal with this, the underlying energy density function is linearly extrapolated, as described in Stomakhin et al. (2012). Furthermore, since more deformation is involved, all methods need to re-linearize constraints more often. For ICA, this implies that the LCP matrix also needs to be re-computed. On

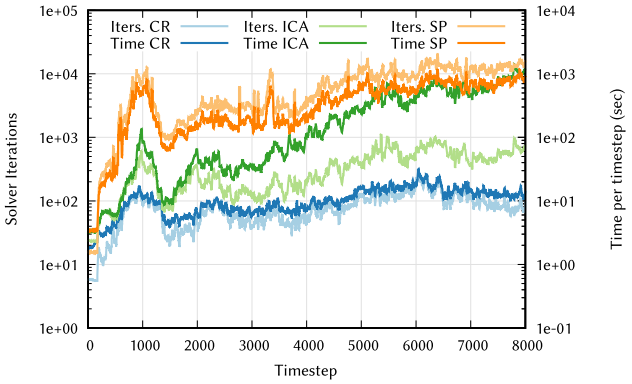


Fig. 10. Total runtime and solver iterations for a hyper-elastic object (see Figure 1 and the Armadillo case in Table 1). The total running-time includes all tasks performed during the process, including collision detection.

average, constraints are re-linearized 5 times per timestep, with peaks approaching 10 times. In our test case (see Figure 1), a neo-Hookean energy density function is used and extrapolated when the volume of an element drops below 70% of its initial volume. Furthermore, the Lamé parameters were set using Young's modulus of elasticity $E = 5 \times 10^5 Pa$ and Poisson ratio $\nu = 0.2$. An Armadillo consisting of roughly 200K degrees of freedom is pulled through a set of rotating cylinders, resulting in large compressions, deformation, and contact forces. The total number of iterations and total computation time are presented in Figure 10. For all methods, the total number of iterations and computation time increase with the amount of compression. This test also faces the underlying collision detection with numerous challenging cases, which are all resolved correctly.

6.2.2 Comparison to ICA. We compared our approach to the elegant and flexible ICA method of Otaduy et al. (2009). We chose this method for comparison because it constructs and solves complete LCPs (instead of an approximation (Duriez et al. 2006)) and takes the full generalized mass matrix into account when computing collision responses. Both of these aspects are essential when dealing with FEM-based deformable bodies. Furthermore, the computation of non-penetration and friction multipliers are combined, making the method very efficient. Please note that we needed to compute the unconstrained velocity with a slightly lower tolerance to guarantee that the final approximation is accurate enough.

In Figure 11 and Table 1, a comparison is shown between our method and ICA. We compare the average numbers of iterations and average time per timestep. ICA performs, per iteration, up to two Sparse Matrix-Vector (SpMV) multiplications: one for performing a (block) Gauss-Seidel step in the outer loop and one (block) Jacobi step for setting up the right-hand side of the LCPs. Additionally, the iterations spent on computing the unconstrained velocity are considered. We did not explicitly count the number of iterations used for solving the LCPs, as these matrices generally are very sparse, and the LCPs converge quite quickly.

For both methods, we have observed that computing the multipliers converged quickly compared to the global problem. Within

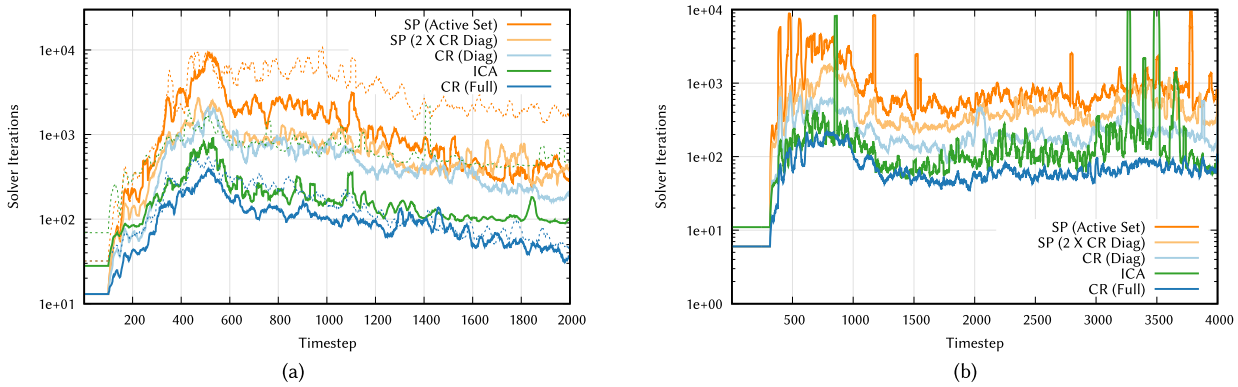


Fig. 11. Number of solver iterations per timestep for different methods, performing the simulation shown in Figure 8(h) (left) and Figure 8(e) (right). CR (Full) represents our complete method including the Full preconditioner. CR (Diag) represents our method with the diagonal preconditioner, in which the friction cone is replaced by a friction pyramid. SP (Active Set) is an implementation of the SP method, in which the QP problems are solved using two CR-based Active Set methods. SP (2xCR) represents an implementation of the SP method, in which the QP problems are solved by a modified version of our diagonally preconditioned CR method. All methods converged to the same absolute error $\epsilon_1 = 5 \times 10^{-5}$, and used the same constraint tolerance $\epsilon_2 = 10^{-6}$ and double-precision arithmetic. The dashed plots represent test case "Rings*," in which the stiffness was increased.

ICA, a small LCP is solved per iteration, which converges very rapidly. In our approach, the convergence of the multipliers is slower because each iteration improves the multipliers by just a bit, whereas ICA actually solves an LCP. Nevertheless, the convergence rate of the contact problem is mostly bounded (especially for deformable bodies) by the convergence of the velocity (the deformation part) rather than that of the multipliers. For more complex cases, like the Pyramid setup, a significant amount of time is spent on the LCPs as well. In the hyper-elastic test case (Figures 1 and 10), ICA tends to perform slower over time. First, per timestep, more re-computations of the LCPs are required due to the non-linear motion and deformation. Second, due to the computation of the unconstrained velocity (which releases all stress), more collisions are detected, and more constraints than needed are generated. This results in a much larger overhead spent on constructing and solving the LCPs. Due to the increased stiffness of the problem, the PGS and GS methods converge slower as well. Finally, since the test case in Figure 1 also involves rigid bodies, the constructed LCP contains dense regions, which increases the time per PGS iteration. In Section 7, we shall further discuss the differences between both methods.

6.2.3 Comparison to SP. We also compared our approach to SP (Kaufman et al. 2008). The SP method solves the contact problem by first computing the unconstrained velocity, which is then corrected by separately computing impulses of the non-penetration and friction constraints. This correction procedure first solves a QP problem for non-penetration constraints, provided that some estimate of the friction impulses is available, followed by a QP solve of the friction problem using non-penetration impulse estimates. This process continues until an optimum is found that minimizes both the friction and non-penetration sub-problems. Since a FEM-based discretization results in a large yet sparse matrix $\mathbf{A}$, we have used a diagonally preconditioned MINRES/CR-based Active Set approach for solving the QP problems rather than a dense solver, as in the original work (Kaufman et al. 2008). Each individual solve uses a "warm start." The relative error of the friction impulses is computed by solving a linear problem using the CG method. Since the SP method does not guarantee a collision-free state, we have replaced the CR solver within our framework from Algorithm 1 with the SP method such that at convergence, additional constraints can be added and corrected.

In Figure 11 and Table 1, a comparison between our approach and the SP method is shown. Since SP solves two QP problems per iteration, the total number of QP solver iterations per simulation step is counted. We have also used a modified version of our CR approach as a QP solver for SP. In this solver, the kinetic friction treatment, described in Section 3.2, is replaced by one in which a friction pyramid is used. This is because our approximation of the friction cone is not suitable for use in the SP method.

The main reason for the difference in the number of solver iterations is that SP solves two QP problems, which both influence each other. In our approach, both non-penetration and friction impulses are approximated, and constraints are allowed to update their state more frequently. Due to this, our approach switches the state of a constraint when the current approximation starts to deviate from the (unknown) global minimum. Since SP keeps a part of the constraints fixed, while updating the other set, it can, for instance, omit changing the state of a non-penetration constraint while solving for the friction impulses. This *can* result in a sequence of approximations that is deviating from the (unknown) global minimum for a longer stretch of iterations. In some exceptional cases, we have observed a slow convergence due to cycling between non-penetration and friction constraints. The total speedup of our method compared to the Active Set–based SP method is roughly between 7 and 12 times. Replacing the Active Set solvers by modified versions of our method, an improvement in the iteration count of about 2 times was obtained when a diagonal preconditioner was used. Additionally, we have also applied a modified version of our Full preconditioner (in which either the active non-penetration or static friction constraints were present), but this approach was not successful.

To run the experiment from Figure 8(e) using SP, we needed to reduce Δt to 0.0005s, because for larger timesteps, the method

did not converge. Even with the smaller Δt, solving both QP sub-problems converged properly, but applying each outcome to the other sub-problem resulted in an oscillating and diverging behavior. This problem appeared, for example, when the first two layers of objects collided with a large impact; the effects are visible in Figure 11(b), in which larger iteration counts (spikes) can be seen. A similar observation was done for the hyper-elastic test (see Figures 1 and 10). Here also the method takes a large number of iterations for solving the individual QPs, and then it requires many steps for finding a global optimum. A tighter coupling between the friction and non-penetration constraints seems to be necessary in these kinds of situations. Using our method, we were able to increase Δt even further and even for stiffer models (see Section 6.1). When the stiffness of the models is increased (setup "Rings*"), SP seems to converge much slower compared to both ICA and CR. When stiffer and more complex objects are used (case "Bunnies"), SP did not properly converge, whereas the QP problems did converge. Therefore, these results are not present in Table 1.

6.2.4 Comparison to Reduced-form Methods. Reduced-form methods transform the complete contact problem into a *reduced* one and solve dynamics and contact separately. These forms require the Delassus operator $\mathbf{W} = \mathbf{L}\mathbf{A}^{-1}\mathbf{L}^T$, which is often used in methods in which the number of degrees of freedom per object is relatively small, such as granular materials, rigid bodies, reduced deformable bodies, or fiber simulations. In those cases, the generalized mass matrix $\mathbf{A}$ can be inverted efficiently. For deformable bodies with a large amount of degrees of freedom, the computation of $\mathbf{A}^{-1}$ is more demanding and may bring extra inaccuracies. As suggested in Bertails-Descoubes et al. (2011), $\mathbf{A}^{-1}$ can be found through a Cholesky decomposition followed by forward and backward substitutions. Since $\mathbf{A}^{-1}$ contains large and dense blocks, *and* not all columns of $\mathbf{A}^{-1}$ are required for constructing $\mathbf{W}$, it is preferable to compute $\mathbf{W}$ through m forward and backward substitutions involving $\mathbf{A}$, $\mathbf{L}^T$, and $\mathbf{L}$, with m the number of constraints (see Daviet et al. (2011)). This omits the explicit form of $\mathbf{A}^{-1}$, which can have a very large memory footprint.

The method in Tonge et al. (2012) does not explicitly compute $\mathbf{W}$ but computes, per contact block, each individual $\mathbf{L}\mathbf{A}^{-1}\mathbf{L}^T$ for each constraint and object associated with the contact block and accumulates this in a small matrix. The computational complexity is similar to the method in Daviet et al. (2011), in which only the non-zero blocks of $\mathbf{W}$ are computed. Their running-time analysis for constructing $\mathbf{W}$ assumes that the degrees of freedom per object are bounded, and that the computation time is dominated by the squared number of constraints. Hence, by exploiting the block-diagonal structure of $\mathbf{A}$, $\mathbf{W}$ can be assembled efficiently.

The main problem with deformable bodies is to compute $\mathbf{W}$ in reasonable time. Unfortunately, all mentioned methods assume that $\mathbf{A}^{-1}$ is easy to compute, which is not the case for deformable bodies with a large amount of degrees of freedom. Figure 12 shows the time consumed for computing $\mathbf{W}$ through a Cholesky decomposition (by also taking the block structure of $\mathbf{A}$ into account (Daviet et al. 2011)) and our method for the same number of constraints. This clearly shows that the computation time of $\mathbf{W}$ grows with the number of constraints, where its slope is determined by

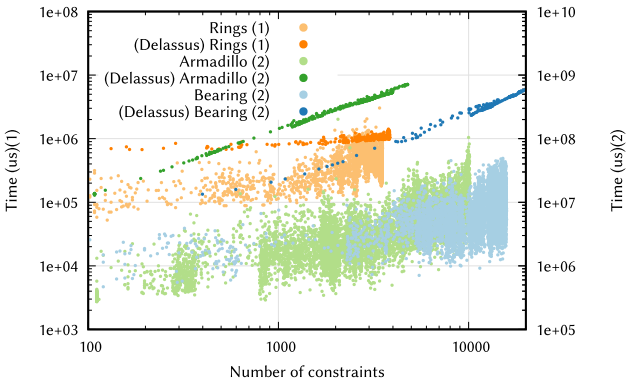


Fig. 12. Timing comparison of our method compared to the time required for constructing the Delassus operator $\mathbf{L}\mathbf{A}^{-1}\mathbf{L}^T$ using the optimized approach of Daviet et al. (2011) for the Rings, Armadillo, and Bearing test cases, respectively, in Table 1 (see also Figures 1, 8(h), and 6). The numbers in the legend refer to the left (1) or right (2) y-axis.

the complexity of computing $\mathbf{A}^{-1}$ (which mainly depends on its structure and/or conditioning). Once all blocks of the Delassus operator are computed, the methods can start solving the problem, typically using a (P)GS method in which $\mathbf{W}$ is used as the iteration matrix (Daviet et al. 2011; Tonge et al. 2012) (both methods explicitly require the diagonal blocks of $\mathbf{W}$), or by using a Newton method in which at each iteration the search direction and step size is found by solving linear systems involving $\mathbf{W}$ (Bertails-Descoubes et al. 2011). Alternatively, in the latter method, the Delassus operator can be applied implicitly using an operator that solves a linear system involving $\mathbf{A}$. However, we experienced convergence issues when solving a linear problem involving this operator for the examples shown in Figures 1 and 8(h). This is likely caused by the bad conditioning of $\mathbf{W}$ due to duplicate constraints or multiple constraints between the same degrees of freedom, which typically occur between rigid and/or deformable objects, or in cases of large compression. In fact, they mention that their method cannot solve these types of problems, including stacking of rigid bodies as shown in Figure 2(c) for our method. Similar issues are mentioned in Daviet et al. (2011) when $\mathbf{L}$ becomes rank deficient. Please note that Figure 12 only considers the time required for constructing the Delassus operator. On top of that, the actual running time of the method must be added. Furthermore, the plot does not take into account additional updates of $\mathbf{W}$ due to re-linearized constraint Jacobians, which boils down to re-computing $\mathbf{W}$ a few times per timestep.

6.2.5 Comparative Timings. Comparing only the iteration count, our approach requires between 10 and 60 times fewer iterations compared to SP. Similarly, our method requires between 2 and 8 times fewer iterations than ICA. When the actual computation time is taken into account, the differences are small for relatively simple models. When the complexity of the models increases (e.g., due to arbitrary shaped tetrahedral elements and many internal vertices), the timing differences become larger. This is due to the increased density and conditioning of matrix $\mathbf{A}$. The conditioning affects the convergence rate, whereas the density affects the time spent per iteration. Furthermore, when a model has many

Table 1. Comparison of Our Method (CR (Full)) Against Various Instances of SP (With Active Set Solvers or a Modified Version of Our Approach) and ICA

Method	Resolution (s)	Collision (s)	Time (s)	Iterations	Constraints	Objects	Elements	Faces	Setup	$\Delta t(s)$	$E(Pa)$
SP (Active Set)	5.6 (3.2)	0.68 (0.72)	6.2 (3.9)	1,546 (780)	2,385 (2,919)	120	12,960	17,280	Rings	0.001	5×10^5
SP (2xCR Diag)	5.0 (4.4)	0.68 (0.72)	5.7 (5.2)	814 (728)	2,340 (2,886)	120	12,960	17,280	Rings	0.001	5×10^5
CR (Diag)	2.8 (2.3)	1.1 (1.0)	3.9 (3.5)	451 (340)	2,435 (2,976)	120	12,960	17,280	Rings	0.001	5×10^5
ICA	0.8 (0.7)	0.85 (0.85)	1.6 (1.5)	180 (134)	2,846 (3,180)	120	12,960	17,280	Rings	0.001	5×10^5
CR (Full)	0.52 (0.44)	0.94 (0.96)	1.46 (1.43)	97 (76)	2,280 (2,466)	120	12,960	17,280	Rings	0.001	5×10^5
SP (Active Set)	20.8 (11.75)	2.9 (2.9)	23.7 (14.9)	1,088 (604)	8,647 (9,651)	140	60,480	73,920	Pyramid	0.0005	5×10^5
SP (2xCR Diag)	15.6 (11.5)	3.7 (3.8)	19.3 (15.4)	463 (332)	8,901 (9,939)	140	60,480	73,920	Pyramid	0.0005	5×10^5
CR (Diag)	7.6 (6.1)	4.4 (4.3)	12.0 (10.8)	229 (179)	8,871 (9,877)	140	60,480	73,920	Pyramid	0.0005	5×10^5
ICA	14.2 (3.8)	3.77 (3.61)	17.95 (7.77)	558 (75)	8,734 (9,765)	140	60,480	73,920	Pyramid	0.0005	5×10^5
CR (Full)	2.3 (1.9)	3.77 (3.70)	6.0 (5.6)	69 (59)	8,565 (9,546)	140	60,480	73,920	Pyramid	0.0005	5×10^5
SP (Active Set)	20.2 (15.65)	0.9 (1.06)	21.7 (16.7)	3,373 (2,423)	2,290 (2,892)	120	12,960	17,280	Rings*	0.001	5×10^6
ICA	2.1 (1.7)	0.78 (0.75)	2.7 (2.5)	655 (550)	2,439 (3,217)	120	12,960	17,280	Rings*	0.001	5×10^6
CR (Full)	0.8 (0.6)	0.84 (0.80)	1.6 (1.40)	154 (103)	1,898 (2,232)	120	12,960	17,280	Rings*	0.001	5×10^6
ICA	174 (160)	4.95 (4.92)	179 (165)	7,206 (6,261)	876 (1,023)	120	152,400	88,560	Bunnies	0.001	5×10^6
CR (Full)	31.3 (17.8)	4.89 (4.84)	36.2 (22.7)	900 (534)	700 (793)	120	152,400	88,560	Bunnies	0.001	5×10^6
SP (Active Set)	395 (310)	6.8 (4.5)	404 (317)	6,839 (5,393)	4,335 (4,101)	1	385,739	20,978	Armadillo	0.0005	Non-linear
ICA	255 (95)	4.15 (3.81)	259 (99)	344 (286)	4,170 (3,705)	1	385,739	20,978	Armadillo	0.0005	Non-linear
CR (Full)	6.55 (5.44)	4.14 (3.6)	10.71 (9.65)	81 (71)	4,350 (3,633)	1	385,739	20,978	Armadillo	0.0005	Non-linear
CR (Full)	19.38 (18.88)	3.54 (3.32)	22.9 (22.4)	480 (463)	4,044 (4,405)	9	115,659	46,080	Bearing	0.001	Non-linear

The numbers represent mean (and median) values over a fixed stretch of timesteps. The number of active constraints equals three times the number of contacts.

more elements than surface faces (due to internal vertices), the ratio collision detection/collision response shifts such that collision detection uses a smaller portion of the total time. Experiments "Bunnies" and "Armadillo" in Table 1 depict such cases. The Armadillo test case shows large differences in time and iteration count between the methods (see also Figure 10). The differences are mainly caused by the large compression of the material, which results in large contact forces and more linearization steps of the constraints due to the larger deformations. SP has difficulties in finding an accurate global minimum, whereas ICA spends more time on setting up and solving the LCPs.

Figure 8 shows additional example results by our method. The first snapshot shows various elastic objects, of different sizes and resolutions, interacting with each other. In total, 201 objects were simulated, containing about 150K tetrahedral elements; the maximum number of constraints peaked at 2K. The average speedup compared to ICA was 4.35×, whereas the time spent on collision detection was negligible. Figure 8(d) shows an example of 2K elastic cubes that form a pile. The simulation contains about 88K tetrahedral elements, and the number of constraints peaked at 10K. Figure 8(e) through (g) shows examples in which external forces and motions are used along with the simulations. For these examples, a Young modulus of $E = 5 \times 10^5 Pa$ and a Poisson ratio of 0.4 were used. Additionally, Figure 8(e) depicts the Pyramid test case, in which a pyramid is compressed by a large and rotating glass plate, resulting in large contact forces between the objects.

6.2.6 Memory Usage. The experiment shown in Figure 8(h) required about 380MB of RAM for the whole simulation. Sparse matrix $\mathbf{A}$ had dimension $25{,}920 \times 25{,}920$ with an average density of 17 elements per row. The simulation shown in Figure 8(e) peaked to about 1.8GB of memory with $\mathbf{A}$ of dimension $109{,}200 \times 109{,}200$ and an average density of 17 elements per row. For the Bunnies test case, about 2.2GB of RAM was allocated, with $\mathbf{A}$ of dimension $147{,}600 \times 147{,}600$ and an average density of 27 elements per row. Most of the allocated memory is used for collision detection–related structures and the FEM machinery. We have also compared our CR-based method to a MINRES-based implementation. Both versions require similar amounts of memory; this is because both methods store the same matrix and allocate a similar amount of vectors. Therefore, no significant difference exists with respect to memory consumption.

6.3 Error Measurements

The Fischer-Burmeister function,

$$f(x, y) := x + y - \sqrt{x^2 + y^2}, \tag{24}$$

can be used to measure the complementarity error made when solving complementarity problems (i.e., $\|(\mathbf{J}\mathbf{v} - \mathbf{c})^T \boldsymbol{\lambda}\|$). Figure 13 shows the maximum error measured per constraint for our method for the experiment shown in Figure 8(h). For each individual constraint, x is set to the corresponding multiplier and y is set to the corresponding distance value (e.g., $\mathbf{j}_k\mathbf{v} - c_k$). For kinetic friction constraints, x is set to the difference between $\mu\lambda_k$ and $\|\boldsymbol{\gamma}_k\|$, which should evaluate to zero if the constraint is satisfied. Furthermore, since x and y contain two different quantities, x is scaled by the corresponding value on the diagonal of matrix $\mathbf{S}_d$ (see Section 3.4) such that both x and y represent a distance error. As shown in Figure 13, both static friction and non-penetration constraints measure, in general, a complementarity error that is below the specified solver tolerance $\epsilon_1 = 5 \times 10^{-5}$. Furthermore, the plot shows the maximum error among all constraints for a given timestep. Kinetic friction constraints yield, in general, an error smaller than $\epsilon_2 = 5 \times 10^{-6}$.

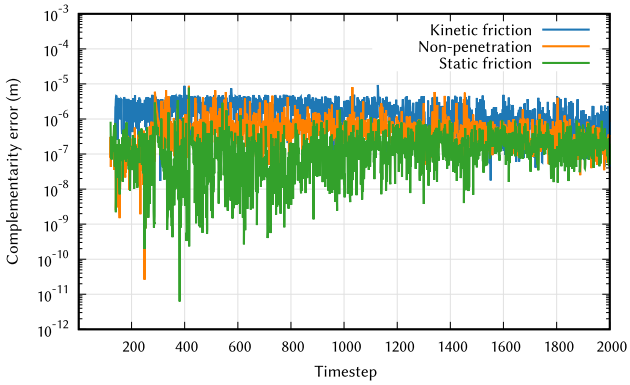


Fig. 13. Plot of the Fischer-Burmeister function for our method, corresponding to the experiment shown in Figure 8(h). The solver tolerance was set to $\epsilon_1 = 5 \times 10^{-5}$, and constraints were allowed to change if the corresponding state change was larger than $\epsilon_2 = 5 \times 10^{-6}$.

7 DISCUSSION

Solvers for indefinite problems. The CR method is very similar to the MINRES method (Paige and Saunders 1975), and in fact, for positive-definite matrices, both methods perform identically (Fong and Saunders 2011). Even for indefinite problems, both methods perform similar. The main difference between CR and MINRES is the computation of the updated search vector. CR uses a two-term recurrence, similar to the CG method, whereas MINRES uses a three-term recurrence. We have implemented a MINRES version of the method described in Algorithm 1 using many combinations of preconditioners (None, Diag, and Full) and friction treatment (cone and pyramid). For all tested cases (mentioned in Table 1), the MINRES version performed most timesteps very similarly to our CR version. However, sometimes it had serious convergence issues leading to failure. Especially when multiple bodies were colliding, convergence issues were observed, which were not observed within our CR-based method. Thus, when constraints are updated prior to convergence (as in our method), MINRES is not recommended. Conversely, we successfully applied MINRES as an Active Set QP solver in the SP method. The results were very similar to the CR-based method, in which the constraints were kept fixed. Additionally, both CR and MINRES can break down in similar situations, in which a least-square approximation is computed. To alleviate this, an extra stopping condition is typically used (see Section 4.3). Within CR, vector $\mathbf{BC}^{-1}\mathbf{r}$ is already available, whereas MINRES should explicitly compute it, which makes the method less efficient.

Other popular methods for solving indefinite problems are *generalized* versions of CR and MINRES, such as GCR (Eisenstat et al. 1983) and GMRES (Saad and Schultz 1986), which also apply to non-symmetric problems. Both methods store one additional vector per iteration, which is used in all subsequent iterations. This results in growing storage and computational requirements. Since these methods rely on a history of search vectors, one could ask how GCR or GMRES performs when the methods are restarted after every constraint update. We leave this for future work.

Differences with ICA. The speed difference between our method and ICA is mainly due to the different solvers used: Gauss-Seidel in ICA versus preconditioned CR in our method. Using the full preconditioner from Section 3.4, the CR solver converges significantly faster compared, for example, to a simple diagonal preconditioner, and this is therefore crucial for efficiently computing the collision response. Within ICA, per iteration, one Gauss-Seidel approximation is computed for refining the velocity correction and one Jacobi approximation for estimating the right-hand side of the LCP. We have observed that if the unconstrained problem is solved relatively fast, the differences between our method and ICA are smaller. Conversely, if the unconstrained problem is more difficult to solve (especially for stiffer and/or more complex models), our method performs significantly better than ICA (see the Bunnies test case in Table 1). However, we expect smaller differences in cases where $\mathbf{A}$ has a low density and low condition number. Apart from the underlying numerical methods, other differences also exist. For example, at the beginning of each timestep, ICA computes an unconstrained velocity that is used as input for the collision detection phase. Then, the collision detection stage results in a large number of collisions, which are subsequently used for computing the velocity correction. The problem with performing collision detection using the unconstrained velocity is that all elastic energy is released at once. The collisions found afterward do not necessarily match the collision state at the end of the previous timestep, and hence the multipliers computed in the previous step will not result in a nearly resolved collision state. Due to this, ICA requires more steps to find the proper set of constraints and needs more iterations to converge. This becomes clearly visible in Figure 10. The difference in total computation time can become larger than 10 times. To make ICA more efficient in these situations, we suggest following our approach in which the constrained velocity is used for the collision test. The main advantage of this is that valid active constraints are reused, so there is no sudden release of the internal elastic energy, resulting in fewer new collisions. For the test case described in Section 6.2.1, the cylinders were made of rigid objects. As a result, the computed LCPs will have large and dense regions, which affects the construction time of the LCP matrix and running time of the PGS method. This additional overhead is visible as the time per timestep increases more than the number of iterations. This can be largely factored out using a modified PGS method (see Miguel and Otaduy (2011) for more details). However, the total convergence behavior remains similar. This overhead is not observed while running the same test case using our method.

Differences with SP. SP uses a loose coupling of the non-penetration and friction constraints by sequentially solving the non-penetration and friction sub-problems. The advantage of this approach is that kinetic friction is modeled through a BLCP for which the bounds are computed by the non-penetration sub-problem. The disadvantage of this coupling is that the sub-problems respond much later to state changes in the other sub-problem. This can require more iterations to obtain convergence (see Figures 10 and 11). If we compare our method to SP with Active Set solvers and a hybrid version using our modified solver, we can also see the effects of updating the constraint states while the method has not converged yet (see the results of Rings in Table 1). Within the Active Set version, the solvers converge quickly, although not necessarily to the optimal solution. By changing states, eventually the method

converges. The hybrid version allows changing of non-penetration or friction constraints, depending on the current sub-problem. This results in a faster convergence, but still the non-penetration and friction constraints have the same loose coupling, which still can lead to a slow converging sequence of approximations. Within our approach, this coupling is much tighter, as constraints are allowed to change at much smaller intervals. Hence, the method converges faster, but this mechanism introduces some additional overhead.

Discretization methods. We have used a (first-order) semi-implicit time integration scheme that solves an implicit system for the unknown vertex velocities. The new velocities are then used explicitly to compute new vertex positions. However, our method is not bound to this scheme, and instead it can use other/higher-order (implicit) integration schemes, such as the Newmark-Beta method. When higher-order methods are used, the constraint matrices $\mathbf{J}$ and $\mathbf{J}_t$ are extended and the evaluation of the constraints involves additional quantities, but the basic principle remains the same. Furthermore, we have used the FEM for the spatial discretization (represented by matrix $\mathbf{A}$), but this can be replaced by other discretization schemes.

Toward larger timesteps. Figure 9 shows results of our method for the same simulation, executed with different values of Δt. In general, the method converges faster for smaller Δt, but more timesteps are required to simulate a certain time span. Therefore, the total number of solver iterations remains approximately similar while simulating the same time span with different Δt. For larger timesteps, more Newton steps are required, resulting in more re-linearizations of the constraints, and the set of potential candidates increases; both affect the runtime. When the timetep becomes larger than 0.004s, the constraint tolerance ϵ_2 should also be scaled with Δt. We have tested the method with timesteps of 0.02s. In this case, the problem to solve per timestep is more difficult but agrees with the findings in Figure 9. To apply this method in real-time applications, such as haptics (Lobo et al. 2017), either the resolution of the meshes should be decreased and/or parallelization should be used. With lower-resolution meshes, the number of potential collisions decreases as well.

Limitations. We are bound to use a customized collision detection system, whereas ICA and SP, for example, can use any of the highly optimized systems available. Since our method is very accurate, it is less forgiving when conflicting constraints are created due to wrongly detected collisions. This puts additional requirements on the collision detection method such that no incorrect or conflicting constraints are created. Furthermore, the friction treatment in our approach may not allow for an easy replacement of the proposed CR-based solver, for example, by a PATH solver.

8 FUTURE WORK

In this article, we have mainly focused on collision response handling for deformable models. However, we plan to further investigate the performance of our approach for coupled deformable and rigid-body simulations. As shown in Section 6.2.1, the LCP matrices lose their sparsity in such cases, resulting in larger computation times for the LCPs. Such a problem is not observed in our method. In Section 6.2.1, non-linear materials were used by linearizing the underlying energy density functions of the materials at the beginning of each timestep. One step further would be to extend our method such that the complete non-linear problem is also solved by using our approach in Newton-based solvers, which are usually built on top of linear solvers. Finally, all computations of our method can be easily parallelized, and we are in the process of porting the entire method on the GPU. Based on the results reported in Verschoor and Jalba (2012) and our initial experiments, we expect a speedup of at least one order of magnitude.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and Miguel Otaduy for their useful comments, and Juan José Casafranca for his help with the non-linear elasticity models. The work was funded in part by the European Research Council (ERC Consolidator Grant no. 772738 TouchDesign).

REFERENCES

Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G. Kry. 2010. Volume contact constraints at arbitrary resolution. *ACM Trans. Graph.* 29, 4 (2010), 82:1–82:10.

Mihai Anitescu and Gary D. Hart. 2004. A fixed-point iteration approach for multi-body dynamics with contact and small friction. *Math. Program.* 101, 1 (2004), 3–32.

David Baraff. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proc. of the 16th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'89)*. ACM, New York, NY, 223–232.

David Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. of the 21st Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'94)*. ACM, New York, NY, 23–34.

David Baraff. 1996. Linear-time dynamics using Lagrange multipliers. In *Proc. of the 23rd Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'96)*. ACM, New York, NY, 137–146.

David Baraff and Andrew Witkin. 1992. Dynamic simulation of non-penetrating flexible bodies. In *Proc. of the 19th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'92)*. ACM, New York, NY, 303–308.

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *Proc. of the 25th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'98)*. ACM, New York, NY, 43–54.

Jernej Barbič and Doug James. 2007. Time-critical distributed contact for 6-DoF haptic rendering of adaptively sampled reduced deformable models. In *Proc. of the 2007 ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA'07)*. 171–180.

Jan Bender, Kenny Erleben, and Jeff Trinkle. 2014. Interactive simulation of rigid body dynamics in computer graphics. *Comput. Graph. Forum* 33, 1 (2014), 246–270.

Florence Bertails-Descoubes, Florent Cadoux, Gilles Daviet, and Vincent Acary. 2011. A nonsmooth Newton solver for capturing exact Coulomb friction in fiber assemblies. *ACM Trans. Graph.* 30, 1 (2011), 6:1–6:14.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.* 21, 3 (2002), 594–603.

Erin Catto. 2005. Iterative dynamics with temporal coherence. In *Proc. of the 2005 Game Developer Conf.*

Richard Cottle, Jong-Shi Pang, and Richard E. Stone. 1992. *The Linear Complementarity Problem*. Academic Press.

Gilles Daviet, Florence Bertails-Descoubes, and Laurence Boissieux. 2011. A hybrid iterative solver for robustly capturing Coulomb friction in hair dynamics. *ACM Trans. Graph.* 30, 6 (2011), 139:1–139:12.

Christian Duriez, Frederic Dubois, Abderrahmane Kheddar, and Claude Andriot. 2006. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Trans. Vis. Comput. Graph.* 12, 1 (2006), 36–47.

Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. 1983. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.* 20, 2 (1983), 345–357.

Kenny Erleben. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.* 26, 2 (2007), Article 12.

Kenny Erleben. 2013. Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH 2013 Courses (SIGGRAPH'13)*. ACM, New York, NY, 8:1–8:42.

David C. Fong and Michael Saunders. 2011. *CG Versus MINRES: An Empirical Comparison.* Technical Report. Stanford University, Stanford, CA.

Nico Galoppo, Miguel A. Otaduy, Paul Mecklenburg, Markus Gross, and Ming C. Lin. 2006. Fast simulation of deformable models in contact using dynamic deformation textures. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA'06).* 73–82.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations* (3rd ed.). Johns Hopkins University Press, Baltimore, MD.

Eran Guendelman, Robert Bridson, and Ronald Fedkiw. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.* 22, 3 (2003), 871–878.

Ken Hayami. 2002. On the behaviour of the conjugate residual method for singular systems. In *Proc. of the 5th China-Japan Seminar on Numerical Mathematics.* 117–126.

T. Heyn, M. Anitescu, A. Tasora, and D. Negrut. 2012. Using Krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation. *Int. J. Numer. Meth. Eng.* 95, 7 (2012), 541–561.

Geoffrey Irving, Craig Schroeder, and Ronald Fedkiw. 2007. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph.* 26, 3 (2007), Article 13.

G. Irving, J. Teran, and R. Fedkiw. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA'04).* 131–140.

J. Judice and F. Pires. 1994. A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Comput. Oper. Res.* 21, 5 (1994), 587–596.

Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. 2008. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.* 27, 5 (2008), 164:1–164:11.

H. W. Kuhn and A. W. Tucker. 1950. Nonlinear programming. In *Proc. of the 2nd Berkeley Symp. on Mathematical Statistics and Probability.* 481–492.

Siwang Li, Zherong Pan, Jin Huang, Hujun Bao, and Xiaogang Jin. 2015. Deformable objects collision handling with fast convergence. *Comput. Graph. Forum* 34, 7 (2015), 269–278.

D. Lobo, M. Saraç, M. Verschoor, M. Solazzi, A. Frisoli, and M. A. Otaduy. 2017. Proxy-based haptic rendering for underactuated haptic devices. In *Proc. of the 2017 IEEE World Haptics Conf. (WHC'17).* 48–53.

David G. Luenberger. 1970. The conjugate residual method for constrained minimization problems. *SIAM J. Numer. Anal.* 7, 3 (1970), 390–398.

David G. Luenberger. 1973. An approach to nonlinear programming. *J. Optimiz. Theory App.* 11, 3 (1973), 219–227.

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), 32:1–32:14.

Eder Miguel and Miguel A. Otaduy. 2011. Efficient simulation of contact between rigid and deformable objects. In *ECCOMAS—Multibody Dynamics 2011.*

Matthew Moore and Jane Wilhelms. 1988. Collision detection and response for computer animation. In *Proc. of the 15th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'88).* ACM, New York, NY, 289–298.

Matthias Müller and Markus Gross. 2004. Interactive virtual materials. In *Proc. of Graphics Interface 2004 (GI'04).* 239–246.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *J. Vis. Comun. Image Represent.* 18, 2 (2007), 109–118.

Miguel A. Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. 2009. Implicit contact handling for deformable objects. *Comput. Graph. Forum* 28, 2 (2009), 559–568.

C. C. Paige and M. A. Saunders. 1975. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* 12, 4 (1975), 617–629.

Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. 2004. Quasi-rigid objects in contact. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA'04).* 109–119.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1992. *Numerical Recipes: The Art of Scientific Computing* (2nd ed.). Cambridge University Press.

Laks Raghupathi and François Faure. 2006. QP-collide: A new approach to collision treatment. In *journées du groupe de travail Animation et Simulation (GTAS) (Annual French Working Group on Animation and Simulation).* Institut de Recherche en Informatique de Toulouse, 91–101.

Alison Ramage and Andrew J. Wathen. 1994. Iterative solution techniques for the stokes and Navier-Stokes equations. *Int. J. Numer. Methods Fluids* 19 (1994), 67–83.

Stéphane Redon, Abderrahmane Kheddar, and Sabine Coquillart. 2002. Gauss' least constraints principle and rigid body simulations. In *Proc. of the IEEE Int. Conf. on Robotics and Automation.* 11–15.

Mathieu Renouf and Pierre Alart. 2005. Conjugate gradient type algorithms for frictional multi-contact problems: Applications to granular materials. *Comput. Methods Appl. Mech. Eng.* 194, 18–20 (2005), 2019–2041.

Youcef Saad and Martin H. Schultz. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* 7, 3 (1986), 856–869.

Tamar Shinar, Craig Schroeder, and Ronald Fedkiw. 2008. Two-way coupling of rigid and deformable bodies. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA'08).* 95–103.

Morten Silcowitz-Hansen, Sarah Maria Niebe Abel, and Kenny Erleben. 2010a. Projected Gauss-Seidel subspace minimization method for interactive rigid body dynamics: Improving animation quality using a projected Gauss–Seidel subspace minimization method. In *Proc. of the Int. Conf. on Computer Graphics Theory and Applications (GRAPP'10)*, Vol. 1. 38–45.

Morten Silcowitz-Hansen, Sarah Niebe, and Kenny Erleben. 2010b. A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *Vis. Comput.* 26, 6–8 (2010), 893–901.

Funshing Sin, Yufeng Zhu, Yongqiang Li, and Daniel Schroeder. 2011. Invertible isotropic hyperelasticity using SVD gradients. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation (Posters).*

J. Spillmann, M. Becker, and M. Teschner. 2007. Non-iterative computation of contact forces for deformable objects. *J. WSCG* 14, (2007), 1–3.

D. Stewart and J. C. Trinkle. 1996. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *Int. J. Numer. Meth. Eng.* 39 (1996), 2673–2691.

Alexey Stomakhin, Russell Howes, Craig Schroeder, and Joseph M. Teran. 2012. Energetically consistent invertible elasticity. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA'12).* 25–32.

Min Tang, Dinesh Manocha, Miguel A. Otaduy, and Ruofeng Tong. 2012. Continuous penalty forces. *ACM Trans. Graph.* 31, 4 (2012), 107:1–107:9.

M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, et al. 2004. Collision detection for deformable objects. *Comput. Graph. Forum* 24, 1 (2004), 61–81.

Richard Tonge, Feodor Benevolenski, and Andrey Voroshilov. 2012. Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph.* 31, 4 (2012), 105:1–105:8.

Mickeal Verschoor and Andrei C. Jalba. 2012. Analysis and performance estimation of the conjugate gradient method on multiple GPUs. *Parallel Comput.* 38, 10–11 (2012), 552–575.

Pascal Volino and Nadia Magnenat Thalmann. 1994. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Comput. Graph. Forum* 13, 3 (1994), 155–166.

P. Wriggers. 2002. *Computational Contact Mechanics.* Wiley.

H. Xu, Y. Zhao, and J. Barbič. 2014. Implicit multibody penalty-based distributed contact. *IEEE Trans. Vis. Comput. Graph.* 20, 9 (2014), 1266–1279.

Received November 2017; revised October 2018; accepted January 2019