Fast But Accurate: A Real-Time Hyperelastic Simulator with Robust Frictional Contact

ZIQIU ZENG, University of Strasbourg, France and CAIR CAS, Hong Kong
SIYUAN LUO*, National University of Singapore, Singapore
FAN SHI, National University of Singapore, Singapore
ZHONGKAI ZHANG, CAIR CAS, Hong Kong

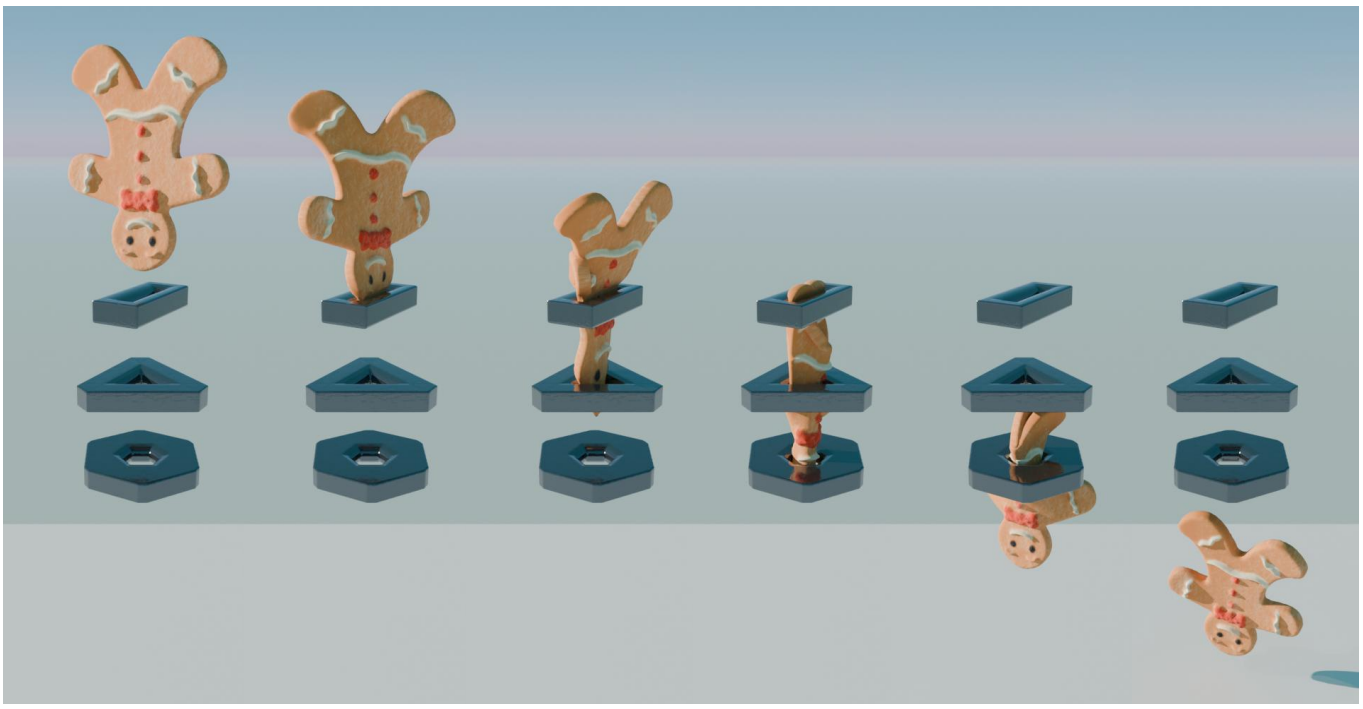


Fig. 1. Crossing Gingerbread Man: we propose a new framework for stable simulation of hyperelastic materials with nodes under large deformation and generic contact constraints in real-time. Pulling the gingerbread man (58.5k DoFs for a single object) through the thin and irregular obstacles is simulated at 11.95ms/iteration using 5 local-global iterations per frame when maximum contact pairs are involved (800 contact constraints).

We present a GPU-friendly framework for real-time implicit simulation of elastic material in the presence of frictional contacts. The integration of hyperelasticity, non-interpenetration contact, and friction in real-time simulations presents formidable nonlinear and non-smooth problems, which are highly challenging to solve. By incorporating nonlinear complementarity conditions within the local-global framework, we achieve rapid convergence in addressing these challenges. While the structure of local-global methods is not fully GPU-friendly, our proposal of a simple yet efficient solver with sparse presentation of the system inverse enables highly parallel computing while maintaining a fast convergence rate. Moreover, our novel splitting strategy for non-smooth indicators not only amplifies overall performance but also refines the complementarity preconditioner, enhancing the accuracy of frictional behavior modeling. Through extensive experimentation, the robustness of our framework in managing real-time contact scenarios, ranging from large-scale systems and extreme deformations to non-smooth contacts and precise friction interactions, has been validated. Compatible with a wide range of hyperelastic models, our approach maintains efficiency across both low and high stiffness materials. Despite its remarkable efficiency, robustness, and generality, our method is elegantly simple, with its core contributions grounded solely on standard matrix operations.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Real-time simulation**; *Parallel algorithms.*

Additional Key Words and Phrases: Physics-based animations, Real-time simulations, GPU parallelization

*Corresponding author.

Authors' Contact Information: Ziqiu Zeng, University of Strasbourg, France and and CAIR CAS, Hong Kong, zengziqiu1995@gmail.com; Siyuan Luo, National University of Singapore, Singapore, sy.luo@nus.edu.sg; Fan Shi, National University of Singapore, Singapore, fan.shi@nus.edu.sg; Zhongkai Zhang, CAIR CAS, Hong Kong, zhongkai.zhangzkz@gmail.com.

ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3730834

ACM Reference Format:
Ziqiu Zeng, Siyuan Luo, Fan Shi, and Zhongkai Zhang. 2025. Fast But Accurate: A Real-Time Hyperelastic Simulator with Robust Frictional Contact. *ACM Trans. Graph.* 44, 4 (August 2025), 19 pages. https://doi.org/10.1145/3730834

1 Introduction

Physics-based simulation of deformable materials plays a crucial role across a variety of disciplines, such as computer graphics, robotics, and medical imaging. Within these fields, it enhances visual experiences through accurate simulations of soft objects and their interactions with the environment, often utilizing finite element discretization for complex nonlinear materials. An ideal physical simulator should possess key attributes like the capacity for *generality* in simulating different materials and the *robustness* to minimize failure instances. In interactive applications, achieving *real-time performance* is essential, while a framework's *simplicity* is highly valued for simplifying maintenance and development processes, making it widely applicable across various downstream fields.

Position-based dynamics (PBD) [Müller et al. 2006] has gained widespread adoption in real-time simulation engines due to its simplicity, robustness, and efficiency. However, as PBD is not derived from continuum mechanics principles, its real-time performance comes at the cost of accuracy. To address this limitation, Projective Dynamics (PD) [Bouaziz et al. 2014] employs local-global optimization, combining localized constraint solving with global optimization for better accuracy and efficiency. In multi-body systems, addressing contact problems typically involves utilizing either penalty methods with soft constraints or Lagrange multiplier methods with hard constraints. While penalty methods are computationally efficient, they encounter difficulties in enforcing strict non-penetration and accurate friction. Lagrange multiplier methods are robust and accurate but face challenges with convergence and parallelization in relaxation techniques like projected Gauss-Seidel [Duriez 2013]. Non-smooth Newton methods [Macklin et al. 2019], combined with Krylov subspace solvers and complementarity preconditioner, offer rapid convergence, high accuracy, and efficient parallelization.

We propose a framework where the local-global iterations are constrained by nonlinear complementarity conditions. Our simulator, empowered by efficient matrix operations, exhibits many favorable features: First, our framework is *efficient* and enables large-scale simulations in real-time, in the presence of nonlinear and non-smooth problems. Within the real-time computational constraints, our solver is able to converge to a desirable *accuracy*, thereby enhancing the *stability*. Second, our method is not only *general* for a wide range of hyperelastic models but also preserves efficiency across materials with both low and high stiffness. Through extensive experiments, our framework has been proven to be *robust* when solving the complex contact problem with different challenges, including large deformation, non-smooth contact, and accurate friction. Beyond these advantages, our method shines in its simplicity, as the core contributions only rely on standard matrix operations. In summary, our contributions are listed as follows:

(1) Our method achieves a highly parallelizable structure while maintaining a high convergence rate by transforming the linear global system into sparse matrix multiplications.
(2) We reformulate Lagrange multiplier methods, particularly the non-smooth Newton method, to integrate seamlessly with local-global integrators.
(3) We introduce a strategy to separate non-smooth indicators, resulting in both reduced Schur-complement computations and enhanced complementarity preconditioner.
(4) Our algorithm establishes a unified, GPU-friendly system for real-time dynamics and contact resolution, with a modular design that ensures easy integration with other frameworks.

2 Related work

Physics-based simulation in computer graphics has been extensively studied. In this section, we focus on discussing the most recent work related to our method.

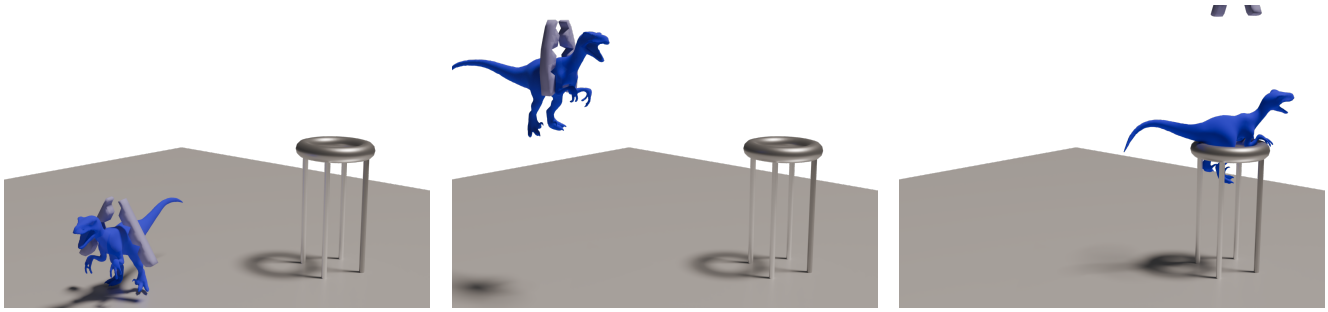
2.1 Implicit simulation for elastic dynamics

In computer graphics, implicit simulations [Bro-Nielsen and Cotin 1996] using backward Euler integration allow for larger time steps compared to explicit methods [Comas et al. 2008], significantly improving computational stability in stiff systems. Finite element (FE) models [Kim and Eberle 2022; Sifakis and Barbic 2012] are a valuable tool for understanding the underlying mechanisms in the real world, as they provide a direct explanation of soft tissue behavior through constitutive relations. With the rapid advancement of computational power and methods, FE models have become increasingly suitable for real-time and interactive simulations.

In terms of integration, the traditional implicit method employs the Newton iteration technique to solve nonlinear problems. Initially restricted to linear elastic models [Bro-Nielsen and Cotin 1996], the method is later extended to the co-rotational formulation [Felippa 2000] and hyperelastic and viscoelastic materials [Marchesseau et al. 2010]. Although the Newton's method converges fast in solving nonlinear systems, it involves re-evaluate and invert the Hessian matrix in each Newton iteration, which implies large computing costs. In practice, Newton integrators typically perform only one iteration in real-time applications [Faure et al. 2012].

In computer graphics, PBD [Müller et al. 2006] is very popular in applications like cloth simulations owning to its high efficiency and stable behavior. As an extension, the extended PBD (XBPD) [Macklin et al. 2016] addresses the limitation of stiffness dependence, providing a better approximation to the implicit Euler method. Recently, the VBD method [Chen et al. 2024] is proposed, which constrains the positions with hyperelastic constraints. We categorize these methods as PBD-like methods [Bender et al. 2017], where the positions of the objects are iteratively adjusted to satisfy a set of local physical constraints. As a result, the PBD-like methods iterate in a Gauss-Seidel-like manner, and their convergence rates are limited by the inefficiency of propagation.

On the other hand, PD [Bouaziz et al. 2014] uses a different way of integration (initially proposed in [Liu et al. 2013]), solving the nonlinear implicit Euler problem through iterative local and global steps. While the PD initially only supports the as-rigid-as-possible (ARAP) model [Chao et al. 2010], advanced PD methods [Liu et al. 2017; Overby et al. 2017] extend the elastic model to generic hyperelastic models. Moreover, [Liu et al. 2017] regards the local-global



(a) Frame 0 (b) Frame 1100 (c) Frame 3200

Fig. 2. Grabbing Raptor: stable catching of an elastic raptor with a soft gripper actuated by cables. Lifting, rotating, and moving the raptor by the fingers are complex operations where friction constraints are necessary. Gripper details can be seen at Appendix.C.

iterations as a Quasi-Newton method and accelerates it using the L-BFGS method, thus creating what is known as LBFGS-PD. On the other hand, another point of view is proposed in [Brown et al. 2018; Narain et al. 2016; Overby et al. 2017], considering the local-global method as a special case of the ADMM method. This work is extended in the WRAPD [Brown and Narain 2021] and the mixed variational FEM [Trusty et al. 2022] to improve the convergence rate with rotation-aware global steps. We categorize these methods as PD-like local-global methods, also briefly local-global methods in the rest of this paper. The key difference between the PD-like methods and the PBD-like methods is how to normalize the local projection results to the global coupling. Unlike PBD-like methods that directly adjust the positions, PD-like ones normalize the results onto the right-hand-side of the global step, via a linear mapping. Accordingly, PD-like methods couple the local solutions through a constant system that can be pre-factorized for efficient computing. Compared with PBD-like methods, PD-like methods provide faster propagation of the local results, leading to much higher convergence rates.

Although local-global methods provide fast convergence and efficient global solution owning to pre-factorization, the global step is hard to be parallelized on the GPU due to the strong data dependence in forward and backward substitutions. To tackle this issue, [Wang 2015] proposes replacing the global linear system with a single Jacobi iteration and employing the Chebyshev approach to accelerate the convergence. The method is further extended to A-Jacobi in [Lan et al. 2022], which incorporates multiple Jacobian iterations to enhance the potential for parallelization. To address Jacobi methods' poor convergence, [Fratarcangeli et al. 2016] suggests using a Gauss-Seidel approach and employing graph coloring techniques [Saad 2003] to improve parallelization. Nevertheless, these methods for parallelizing the global steps only solve the global system more or less approximately, thereby largely decreasing the convergence rate.

2.2 Muti-body Contact

The collision modeling in computer graphics inherits from the numerical methods in constrained optimization theory [Nocedal and Wright 2006]. By simplifying the non-interpenetration contact as inequality constraints, one can address the contact problems via penalty methods [Hasegawa and Sato 2004; Kugelstadt et al. 2018] and augmented Lagrangian methods. These methods handle constraints by adding penalty terms to the objective function such that the overall variational optimization becomes unconstrained. Despite being simple and straightforward, such methods can lead to numerical instability under complex constraints. Another general issue of these methods is the difficulty of including friction in the penalty-based formulation due to the complexity in the non-smooth complementarity conditions. To address this, [Geilinger et al. 2020] uses a hybrid approach to enforce static friction with hard constraints. To tackle the interpenetration challenge, IPC [Li et al. 2020] employs a logarithmic barrier penalty to create a stronger impulse for separating objects as they approach each other closely. Furthermore, in every Newton iteration, it utilizes a CCD-aware line search to geometrically ensure intersection-free results. The friction modeling is also included in this pipeline through a delayed evaluation.

To accurately model non-smooth conditions, one can formulate frictional contact problems as complementarity problems. In optimization theory, these problems can be solved using Quadratic Programming (QP) or Sequential QP for simulating linear elastic or nonlinear hyperelastic materials [Kane et al. 1999; Kaufman et al. 2008; Nocedal and Wright 2006]. Different to the penalty methods, these methods generally constrain the linearized system (e.g., at each Newton solve) with Lagrange multipliers [Baraff 1996], thereby known as Lagrange multiplier methods. Early works like [STEWART and TRINKLE 1996] linearize the frictional constraints along the friction cones, thereby converting the original problem to a linear complementarity problem (LCP). The LCP can be then solved with relaxation methods such as Projected Gauss-Seidel (PGS) [Daviet 2020; Duriez et al. 2006; Li et al. 2018], and direct methods like pivoting methods [Erleben 2013]. It has been observed in [Todorov 2010] that the friction cone linearization could be unnecessary since the non-smooth contact model can be simply treated as an additional set of non-linear equations in the system. [Larionov et al. 2021] proposes a smooth local implicit surface representation to accurately

handle frictional contact between smooth objects. The Newton-based approaches are successful in handling friction model [Bertails-Descoubes et al. 2011]. By transforming the Signorini-Coulomb conditions [Brogliato 2016] to equaly nonlinear complementarity problem (NCP) functions, one can convert the origin non-smooth problem to a root-finding one. Such approach can yield quadratic convergence, and is decoupled with the linear solver, which means that one can choose a fast linear solver like Krylov space solver (e.g., Conjugate Residual [Frâncu and Moldoveanu 2015]) for both fast convergence and high potential for parallelization. [Macklin et al. 2019] extended this method to handle the soft materials and proposed an efficient complementarity preconditioner for smoothing the NCP functions.

In local-global methods, collisions can be handled either through the penalty way (i.e., adding local penalty contact energies), or with the Lagrange multipliers (i.e., constraining the global steps). The penalty way follows the simple collision handing in PBD, dynamically adding contact energies while potential collisions are detected [Bouaziz et al. 2014; Wang et al. 2021]. The IPC energy can also be included in this way [Lan et al. 2022]. However, such methods are typically only adaptable to incomplete methods that are less sensitive to changes in the system [Fratarcangeli et al. 2016]. In contrast, [Overby et al. 2017] proposes constraining the global steps with Lagrange multipliers when interpenetration is detected. In their work, the non-smooth conditions are omitted, thereby simplifying the complementarity problems into linear systems. [Komaritzan and Botsch 2019] and [Ly et al. 2020] continue to use this strategy while verifying the Signorini-Coulomb conditions in each global step. To improve computational efficiency, [Ly et al. 2020] uses a semi-implicit approach that approximates the system inverse.

3 Background

3.1 Implicit Euler scheme in elastic dynamics

Following [Martin et al. 2011], the implicit Euler time integration solves the following optimization problem within the time interval $[t, t+h]$:

$$\mathbf{q}_{t+h} = \min_{\mathbf{q}} \left(\frac{1}{2h^2} ||\mathbf{M}^{\frac{1}{2}}(\mathbf{q} - \tilde{\mathbf{q}})||_F^2 + \sum_i \boldsymbol{\psi}_i(\mathbf{q}) \right) \tag{1}$$

where $\mathbf{M}$ represents the mass matrix; $\mathbf{q}$ and $\mathbf{v}$ denote positions and velocity; $\tilde{\mathbf{q}} = \mathbf{q}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$ is the predicted state with external force $\mathbf{f}_{\text{ext}}$ when implicit forces are not considered; $\boldsymbol{\psi}$ signifies the elastic energy for the finite elements.

3.2 Multi-body dynamics

In multi-body systems, the typical Lagrange multiplier method incorporates contact forces $\boldsymbol{\Lambda}_j$ into the dynamic system. Constraining the derivative of the total energy in Equation (1) with contact conditions yields:

$$\mathbf{M}(\mathbf{q} - \tilde{\mathbf{q}}) - h^2\mathbf{f}_{\text{int}}(\mathbf{q}) - h^2 \sum_{j \in \mathcal{L}} \mathbf{C}_j^{\mathrm{T}} \boldsymbol{\Lambda}_j = \mathbf{0} \tag{2a}$$

$$\forall j \in \mathcal{L}, \quad \mathbf{C}_j\mathbf{q} - \hat{\mathbf{p}}_j = \boldsymbol{\Delta}_j \tag{2b}$$

$$\forall j \in \mathcal{B}, \quad \boldsymbol{\Delta}_j = \mathbf{0} \tag{2c}$$

$$\forall j \in C, \quad (\boldsymbol{\Lambda}_j, \boldsymbol{\Delta}_j) \in \boldsymbol{\xi}_{\mu_j} \tag{2d}$$

where $\mathbf{f}_{\text{int}}$ denotes implicit internal forces, and $\mathcal{L}$ represents indices for Lagrangian constraints which are categorized into binding pairs $\mathcal{B}$ and non-interpenetration pairs C. The linear mapping $\mathbf{C}_j$ maps the mechanical state to the contact space. The item $\hat{\mathbf{p}}$ could be the projected state in binding constraints or a parameter of minimum separation in unilateral constraints (as in [Macklin et al. 2019]).

For each contact pair $j \in \mathcal{L}$, interpenetration $\boldsymbol{\Delta}_j$ and contact forces $\boldsymbol{\Lambda}_j$ must satisfy bilateral condition or Signorini-Coulomb condition $\boldsymbol{\xi}_{\mu_j}$ [Brogliato 2016] for accurate representation of bilateral and frictional contacts. Generally, to solve the problem with numerical methods, linearizing the constraints along specific directions is performed:

Binding contact. Bilateral constraints are prevalent in binding scenarios such as joint connections and special cases like needle constraints [Adagolodjo et al. 2019]. The directions of bilateral constraints $\vec{b} \in \mathcal{R}^{1\times3}$ are defined for binding connections or special collision events.

Non-interpenetration contact. Non-interpenetration contacts are typically linearized along normal and tangent directions to prevent interpenetration and model friction. Generally, the contact normal should be defined by the collision detection. We refer the readers to [Erleben 2018] for more details about the methodologies to generate the contact normal for stable numerical solution. Given a contact normal $\vec{n} \in \mathcal{R}^{1\times3}$, two tangent directions $\vec{f}_1, \vec{f}_2 \in \mathcal{R}^{1\times3}$ can be generated (e.g., via the Gram-Schmidt process) to represent the space perpendicular to the normal. In the subsequent sections, friction directions are unified as $\vec{f}$ for simplicity.

The formulation in Appendix A.1 converts the governing equation (2) into a linearized form, as follows:

$$\mathbf{M}(\mathbf{q} - \tilde{\mathbf{q}}) - h^2\mathbf{f}_{\text{int}}(\mathbf{q}) - h^2\mathbf{H}_b^{\mathrm{T}}\boldsymbol{\lambda}_b - h^2\mathbf{H}_n^{\mathrm{T}}\boldsymbol{\lambda}_n - h^2\mathbf{H}_f^{\mathrm{T}}\boldsymbol{\lambda}_f = \mathbf{0} \tag{3a}$$

$$\mathbf{H}_b\mathbf{q} - \mathbf{d}_b = \boldsymbol{\delta}_b \tag{3b}$$

$$\forall j \in \mathcal{B}, \quad \delta_j = 0 \tag{3c}$$

$$\mathbf{H}_n\mathbf{q} - \mathbf{d}_n = \boldsymbol{\delta}_n \tag{3d}$$

$$\mathbf{H}_f\mathbf{v} - \mathbf{d}_f = \dot{\boldsymbol{\delta}}_f \tag{3e}$$

$$\forall j \in C, \quad (\lambda_j, \delta_j, \dot{\delta}_j) \in \boldsymbol{\xi}_{\mu_j} \tag{3f}$$

In this formulation, $\mathcal{B} = \{b_1, ..., b_m\}$ represents bilateral constraints, and $C = \{c_1, ..., c_n\}$ represents unilateral constraints along with frictional constraints. The Contact Jacobian matrix $\mathbf{H}$ compiles the linearized contact mappings along specific constraint directions, exemplified by $\vec{b}$, $\vec{n}$, or $\vec{f}$. The elements $\mathbf{d}$, $\boldsymbol{\delta}$, and $\boldsymbol{\lambda}$ linearize the contact items $\hat{\mathbf{p}}$ (or $\hat{\mathbf{u}}$), $\boldsymbol{\Delta}$, and $\boldsymbol{\Lambda}$, respectively, along these constraint directions. To accurately capture the frictional behavior of contacts, the Signorini-Coulomb condition $\boldsymbol{\xi}_{\mu_j}$ should be satisfied as follows:

$$\forall j \in C, \quad 0 \le \delta_{n,j} \perp \lambda_{n,j} \ge 0 \tag{4a}$$

$$\forall j \in \mathcal{A}, \quad \dot{\delta}_{f,j} + \frac{|\dot{\delta}_{f,j}|}{|\lambda_{f,j}|}\lambda_{f,j} = 0 \tag{4b}$$

$$\forall j \in \mathcal{A}, \quad 0 \le |\dot{\delta}_{f,j}| \perp \mu_j\lambda_{n,j} - |\lambda_{f,j}| \ge 0 \tag{4c}$$

$$\forall j \in \mathcal{I}, \quad \lambda_{f,j} = 0 \tag{4d}$$

where $\mathcal{A} = \{j \in C \mid \lambda_{n,j} > 0\}$ is the set of all active contact indices, and $\mathcal{I} = \{j \in C \mid \lambda_{n,j} \leq 0\}$ is its complement in C. We refer readers to the Appendix A.2 for more details on the frictional contact formulation. The constrained system in Equation (3) actually refers to a nonlinear complementarity problem. Various numerical methods for solving such problems have been proposed in the literature [Andrews and Erleben 2021; Erleben 2013], such as direct methods, relaxation methods, and non-smooth Newton methods.

3.3 Local-global iterative methods

The PD-like local-global methods address the nonlinear optimization in Equation (1) through recursive local-global iterations [Bouaziz et al. 2014; Liu et al. 2013; Overby et al. 2017]. In the local step, the positions $\mathbf{q}$ are fixed, and suitable projected local states $\mathbf{p}_i$ are sought by solving local and independent sub-problems:

$$\mathbf{p}_i^k = \min_{\mathbf{p}_i} \frac{w_i}{2} ||\mathbf{p}_i - \mathbf{G}_i \mathbf{q}^k||_F^2 + \zeta_i(\mathbf{p}_i) \tag{5}$$

where k denotes the current local-global (L-G) iteration; w is a nonnegative weight for each constraint; $\mathbf{G}$ is the linear mapping from the mechanical state $\mathbf{q}$ to the projection state $\mathbf{p}$. For elastic energy, the nonlinear energy function ζ could be generic hyperelastic energy ψ or an indicator function as in the Projective Dynamics [Bouaziz et al. 2014] ($\zeta = 0$ if $\mathbf{p} \in SO(3)$ and $\zeta = +\infty$ otherwise). Moreover, the local step formulation provides large flexibility to model other generic constraints, such as bending energy in cloth simulations and positional constraints.

After solving the independent local problems, the global step couples all the projected results in a global system:

$$\overbrace{\left(\mathbf{M} + h^2 \sum_i w_i \mathbf{G}_i^{\mathrm{T}} \mathbf{G}_i\right)}^{\mathbf{A}} \mathbf{q}^{k+1} = \overbrace{\left(\mathbf{M}\tilde{\mathbf{q}} + h^2 \sum_i w_i \mathbf{G}_i^{\mathrm{T}} \mathbf{p}_i^k\right)}^{\mathbf{b}^k} \tag{6}$$

In the global step, the right-hand-side (RHS) $\mathbf{b}^k$ assembles the projected state $\mathbf{p}^k$ in the current L-G iteration. Since $\mathbf{G}$ depends only on predefined projection constraints, the global system matrix $\mathbf{A}$ remains invariant throughout the simulation. Repeating the local and the global step recursively (Algorithm 1), the integrator efficiently converges to the solution of Newton's method.

ALGORITHM 1: Local-global integrators

```
while simulation do
    q̃ = q_t + h v_t + h² M⁻¹ f_ext;
    for k ∈ {0, ..., n} do
        p_i^k = project(G_i q^k);                  // local step
        b^k = M q̃ + h² Σ_i w_i G_i^T p_i^k;        // assemble the RHS
        q^{k+1} = A⁻¹ b^k;                         // global step
    end
    q_{t+h} = q^k;                                 // integration
    v_{t+h} = (1/h)(q^k − q_t);
end
```

The success of local-global methods can be attributed to several factors: First, the flexibility in the local step allows for simulating different constraints in a unified framework. The local steps are parallelizable due to their independence. Second, the global step efficiently couples the constraints through the system matrix $\mathbf{A}$. Since local solutions are propagated immediately through global solving, the efficient constraint coupling leads to rapid convergence, especially in the initial iterations. In general, local-global methods converge in significantly fewer iterations than PBD-like methods, although not as few as Newton's method. Most importantly, the system matrix $\mathbf{A}$ remains constant throughout the simulation. Exploiting this property could significantly reduce the computational cost of the global step. A common approach is to pre-factorize the system $\mathbf{A}$ once at initialization, which reduces the global steps to solving only sparse triangular systems (STS).

Despite its efficiency, the forward and backward substitutions in STS are difficult to be computed in parallel due to strong data dependencies, limiting its applicability to large-scale problems. Moreover, it is also challenging to model accurate frictional contacts in local-global methods due to the requirement of satisfying complex non-smooth conditions described in Section 3.2. In this paper, we seek for a method that addresses all these challenges. For this purpose, we propose a simple yet highly efficient method for parallelizing the global step (Section 4), and develop a unified framework where the local-global iterations are constrained by nonlinear complementarity conditions to handle frictional contacts (Section 5).

4 Sparse Inverse Solution

4.1 Global step solution strategy

In Section 2.1, we review different numerical solvers used for solving the global step in the literature. These solvers can be generally classified into complete and incomplete solutions.

Complete Solution. The traditional strategy [Bouaziz et al. 2014; Brown and Narain 2021; Liu et al. 2013; Overby et al. 2017] involves fully resolving the linear system $\mathbf{q}^{k+1} = \mathbf{A}^{-1}\mathbf{b}^k$ during each L-G iteration with pre-factorized system. As discussed in Section 3.3, although immediate propagation via complete global solving achieves fast convergence, data dependencies in the STS make it difficult to implement efficient parallel computation.

Incomplete Solution. An alternative perspective [Wang 2015] suggests that computing an exact solution is both redundant and computationally inefficient, especially considering that the local step requires modifications to the linear system in subsequent local-global iterations. Therefore, incomplete methods [Fratarcangeli et al. 2016; Lan et al. 2022; Wang 2015] approximate the exact solution $\mathbf{q}^{k+1} = \mathbf{A}^{-1}\mathbf{b}^k$ with $\mathbf{q}^{k+1} = \tilde{\mathbf{P}}\mathbf{b}^k + \tilde{\mathbf{Q}}\mathbf{q}^k$, where $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{Q}}$ depend on specific iterative algorithms. These methods offer substantial potential for parallelization since the iterative solvers typically involve sparse matrix-vector multiplications (SpMV) that are easily parallelized on both CPU and GPU architectures. However, approximating the global solve significantly decreases the efficiency of propagating local solutions, leading to slower convergence. Consequently, a larger number of L-G iterations (usually one or more orders of magnitude higher) are required to reach the same level of accuracy as the complete solution. Moreover, each additional iteration leads to an extra local step, which possibly becomes the bottleneck for real-time

performance, especially in solving hyperelastic local problems.

Our goal is to develop a method that both efficiently couples local results with the complete solution for fast convergence and offers extensive parallelization capabilities. A naive propose is to explicitly pre-compute the inverse of $\mathbf{A}$, such that the global steps simply turn to a matrix-vector product, which is straightforward to be parallelized. However, inverting the matrix requires large precomputation cost and, more importantly, massive memory usage for large-scale simulations.

In practice, although the system matrix $\mathbf{A}$ is sparse, its inverse $\mathbf{A}^{-1}$ is typically dense. For instance, a deformable object with $10k$ vertices requires more than $3GB$ of memory for storing dense matrices (as evidenced by experimental data in Tables 1 and 2). Due to memory constraints and the overhead of computing and transferring large matrices to GPUs, this method becomes impractical for large-scale simulations.

An immediate question is: does there exist a sparse system that can accurately represent the inverse $\mathbf{A}^{-1}$? The answer is affirmative, and we give the details in the following subsection.

4.2 Sparse inverted local-global method

Given the Cholesky factor $\mathbf{L}$ of the system, its inverse $\mathbf{L}^{-1}$ is exactly the sparse system we are seeking. Following [Benzi and Tuma 2000; Bridson and Tang 1999; Scott and Tůma 2023], we present the following theorem:

Theorem 1. *Let $\mathbf{A}$ be a symmetric positive definite (SPD) matrix with its Cholesky factor $\mathbf{L}$. The sparsity structure $\mathcal{S}\{\mathbf{L}^{-1}\}$ is the union of all entries (i, j) where i is an ancestor of j in the elimination tree $\mathcal{T}(\mathbf{A})$.*

The theorem implies that $\mathbf{L}^{-1}$ does not need to be fully dense. In practice, for SPD systems in local-global methods, techniques like fill-in reduction ordering (e.g., nested dissection [George 1973]) can efficiently reduce the number of ancestors of vertices in $\mathcal{T}(\mathbf{A})$, resulting in a highly sparse $\mathbf{L}^{-1}$. A simple example in Figure 3 illustrates this concept. The sparse scheme outperforms a full traversal of $\mathbf{L}$, yielding both efficient computation of $\mathbf{S} = \mathbf{L}^{-1}$ and high sparsity in $\mathbf{S}$. The same conclusion can be drawn from [Zeng et al. 2022] concerning the STS solution in their method. By explicitly computing $\mathbf{S}$ and storing it in sparse format, we achieve the desired sparse system which is capable of representing the system inverse. Specifically, the system inverse $\mathbf{A}^{-1}$ is given by the following product:

$$\mathbf{A}^{-1} = \mathbf{S}^{\mathrm{T}}\mathbf{S} \tag{7}$$

such that the overall global solution is transformed into two SpMV operations:

$$\mathbf{q}^{k+1} = \mathbf{S}^{\mathrm{T}}\mathbf{S}\mathbf{b}^{k} \tag{8}$$

Consequently, our sparse inverse method combines both fast convergence and high potential for parallelization, while maintaining simplicity through its use of standard matrix operations. As a trade-off, our method requires additional memory usage for $\mathbf{S}$ and extra time for its computation. We give comprehensive evaluation of memory usage, convergence rate, and computational efficiency in Sections 7.2, 7.3, and 7.4, respectively. On the other hand, our method heavily depends on the system invariability, which makes it challenging to incorporate penalty-based methods that dynamically modify the system during contact handling. To address this, in the next section, we propose using the Lagrange multiplier methods to constrain the global steps where the complementarity conditions are effectively verified in each L-G iteration.

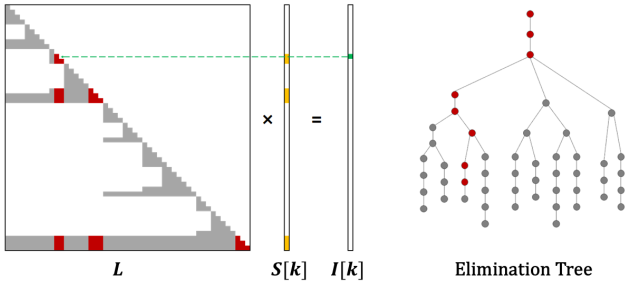


Fig. 3. Sparse solution: after reducing the matrix pattern fill-in through nested dissection, the Cholesky factor $\mathbf{L}$ is reordered and partitioned into sub-blocks. For a given column k in the identity matrix $\mathbf{I}$, the requisite structure to be processed in $\mathbf{L}$ consists of k and its ancestors (red nodes) in the elimination tree.

ALGORITHM 2: Sparse inverted local-global integrators

$\mathbf{L} = \mathit{Cholesky}(\mathbf{A})$;
$\mathbf{S} = \mathbf{L}^{-1}\mathbf{I}$;
while *simulation* **do**
  $\tilde{\mathbf{q}} = \mathbf{q}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$;
  for $k \in \{0, ..., n\}$ **do**
    $\mathbf{p}_i^k = \mathit{project}(\mathbf{G}_i\mathbf{q}^k)$; // local step
    $\mathbf{b}^k = \mathbf{M}\tilde{\mathbf{q}} + h^2 \sum_i w_i \mathbf{G}_i^{\mathrm{T}}\mathbf{p}_i^k$; // assemble the RHS
    $\mathbf{q}^{k+1} = \mathbf{S}^{\mathrm{T}}\mathbf{S}\mathbf{b}^k$; // global step
  end
  $\mathbf{q}_{t+h} = \mathbf{q}^k$; // integration
  $\mathbf{v}_{t+h} = \frac{1}{h}(\mathbf{q}^k - \mathbf{q}_t)$;
end

5 Multi-body dynamics with accurate frictional contact

In Section 4, we introduce the sparse inverse approach for local-global methods. This section now shifts its focus to addressing contact-related issues within the local-global framework. Although several efficient techniques have been developed to handle non-penetration contacts within local-global frameworks, none fully captures the accurate contact and friction behaviors. The non-smooth Newton method has proven effective for its accuracy and stability in multi-body dynamics. However, it is not inherently compatible with local-global methods, as existing approaches mainly combine Newton's method with impulse-based integration.

In this section, we derive a reformulation to integrate the non-smooth Newton method into the local-global framework. In Section

5.1, we establish the compatibility of the Lagrange multiplier methods within the local-global framework. By considering the local-global iterations as quasi-Newton iterations (referencing [Liu et al. 2017]), we demonstrate the equivalence between position-based local-global integration and impulse-based integration, effectively merging these two methodologies within a unified framework (Section 5.2). Subsequently, in Section 5.3, we propose a strategy that splits out the non-smooth indicators, allowing us to significantly reduce computational costs in the Schur complement calculation. This approach also enables us to develop an effective complementarity preconditioner (Section 5.4). Finally, in Section 6, we conclude our approach as a GPU-friendly framework.

5.1 Constrained global step

As illustrated in Algorithm 1, general local-global methods directly update positions $\mathbf{q}$ without explicitly computing internal forces. Therefore, it is not straightforward to integrate the Lagrange multiplier method shown in Equation (2) into the local-global algorithm. To bridge this gap, we first evaluate the impulse $\Delta\mathbf{v}$ in the global steps in Equation (6) within each L-G iteration:

$$\begin{aligned}\mathbf{M}\Delta\mathbf{v} &= \frac{1}{h}\mathbf{M}\Big(\mathbf{q}^{k+1} - (\mathbf{q}_t + h\mathbf{v}_t)\Big) \\ &= h\mathbf{f}_{\text{ext}} + h^2\Big(\sum_i w_i\mathbf{G}_i^{\mathrm{T}}\mathbf{p}_i^k - \sum_i w_i\mathbf{G}_i^{\mathrm{T}}\mathbf{G}_i\mathbf{q}^{k+1}\Big)\end{aligned} \tag{9}$$

In an isolated system (excluding contact forces), the impulse $\Delta\mathbf{v}$ is generated by both external and internal forces, allowing us to compute the internal forces for the current L-G iteration k:

$$\mathbf{f}_{\text{int}}(\mathbf{q}^{k+1}) = h\Big(\sum_i w_i\mathbf{G}_i^{\mathrm{T}}\mathbf{p}_i^k - \sum_i w_i\mathbf{G}_i^{\mathrm{T}}\mathbf{G}_i\mathbf{q}^{k+1}\Big) \tag{10}$$

By incorporating Equation (10) into the constrained implicit Euler solver in Equation (3a), we derive the constrained global step:

$$\begin{aligned}&\overbrace{\Big(\mathbf{M} + h^2\sum_i w_i\mathbf{G}_i^{\mathrm{T}}\mathbf{G}_i\Big)}^{\mathbf{A}}\mathbf{q}^{k+1} - \overbrace{\Big(\mathbf{M}\tilde{\mathbf{q}} + h^2\sum_i w_i\mathbf{G}_i^{\mathrm{T}}\mathbf{p}_i^k\Big)}^{\mathbf{b}^k} \\ &\quad -h^2\mathbf{H}_b^{\mathrm{T}}\boldsymbol{\lambda}_b^{k+1} - h^2\mathbf{H}_n^{\mathrm{T}}\boldsymbol{\lambda}_n^{k+1} - h^2\mathbf{H}_f^{\mathrm{T}}\boldsymbol{\lambda}_f^{k+1} = \mathbf{0} \\ &\quad \text{subject to Equations (3b) - (3f)}\end{aligned} \tag{11}$$

A similar formulation can be found in [Overby et al. 2017], which simplifies the NCP by disregarding the Signorini-Coulomb conditions and converting it into a linear problem. For a thorough solution to the contact problem, we propose using the non-smooth Newton method, which will be explained in detail in the following section.

5.2 Non-smooth constrained local-global integration

The non-smooth Newton method reformulates the complementarity conditions into equivalent NCP functions:

$$0 \le a \perp b \ge 0 \quad \leftrightarrows \quad \boldsymbol{\phi}(a, b) = 0 \tag{12}$$

where the NCP function $\boldsymbol{\phi}$ transforms the original problem into a root-finding one. In practice, $\boldsymbol{\phi}$ could be formulated as either the *minimum-map* formulation or the *Fischer-Burmeister* formulation:

$$\boldsymbol{\phi}_{\min}(a, b) = \min(a, b) \tag{13}$$

$$\boldsymbol{\phi}_{\text{FB}}(a, b) = a + b - \sqrt{a^2 + b^2} \tag{14}$$

As in [Macklin et al. 2019], we use complementarity preconditioners $\mathbf{r}$ to improve the convergence in nonlinear integration. We refer the readers to Appendix B for the detail formulation of different NCP functions and their derivatives. By considering interpenetration $\boldsymbol{\delta}$ and contact forces $\boldsymbol{\lambda}$ as arguments of NCP functions, the non-smooth Newton method reformulates the constrained global step (Equation (11)) as follows:

$$\mathbf{A}\mathbf{q}^{k+1} - \mathbf{b}^k - h^2\mathbf{J}_b^{\mathrm{T}}\boldsymbol{\lambda}_b^{k+1} - h^2\mathbf{J}_n^{\mathrm{T}}\boldsymbol{\lambda}_n^{k+1} - h^2\mathbf{J}_f^{\mathrm{T}}\boldsymbol{\lambda}_f^{k+1} = \mathbf{0} \tag{15a}$$

$$\mathbf{J}_b\mathbf{q}^{k+1} - \mathbf{d}_b + \mathbf{E}_b\boldsymbol{\lambda}_b^{k+1} = \mathbf{0} \tag{15b}$$

$$\boldsymbol{\phi}_n(\boldsymbol{\delta}_n^{k+1}, \boldsymbol{\lambda}_n^{k+1}) = \mathbf{0} \tag{15c}$$

$$\boldsymbol{\phi}_f(\dot{\boldsymbol{\delta}}_f^{k+1}, \boldsymbol{\lambda}_f^{k+1}) = \mathbf{0} \tag{15d}$$

where solving the non-smooth conditions in Equations (3d) - (3f) involves finding roots for NCP functions.

For bilateral constraints, $\mathbf{J}_b = \mathbf{H}_b$ represents the non-smooth Jacobian matrix, while $\mathbf{E}_b = diag(e_{b_0}, ..., e_{b_m})$ is a diagonal compliance matrix consisting of inverse stiffness coefficients e (following [Macklin et al. 2019]). By setting $e > 0$, we establish quadratic energy potentials with a stiffness weight of e^{-1}. Besides, setting $e = 0$ is equivalent to the bilateral condition in Equations (3b) - (3c), resulting in a hard constraint with infinite stiffness.

For unilateral and frictional constraints, the non-smooth Jacobian matrix $\mathbf{J}$ is derived from the partial derivatives of $\boldsymbol{\phi}$ with respect to position $\mathbf{q}$ or velocity $\mathbf{v}$. The non-smooth compliance matrix $\mathbf{E}$ is derived from the partial derivatives of $\boldsymbol{\phi}$ with respect to $\boldsymbol{\lambda}_n$ and $\boldsymbol{\lambda}_f$:

$$\begin{aligned}\mathbf{J}_n &= \frac{\partial\boldsymbol{\phi}_n}{\partial\mathbf{q}}, & \mathbf{E}_n &= \frac{\partial\boldsymbol{\phi}_n}{\partial\boldsymbol{\lambda}_n} \\ \mathbf{J}_f &= \frac{\partial\boldsymbol{\phi}_f}{\partial\mathbf{v}}, & \mathbf{E}_f &= \frac{\partial\boldsymbol{\phi}_f}{\partial\boldsymbol{\lambda}_f}\end{aligned} \tag{16}$$

With $\Delta\mathbf{q} = \mathbf{q}^{k+1} - \mathbf{q}^k$ and $\Delta\mathbf{v} = \mathbf{v}^{k+1} - \mathbf{v}^k = \frac{1}{h}(\mathbf{q}^{k+1} - \mathbf{q}^k)$, we write the conditions in Equations (15b) - (15d) into the position-based form using a first-order Taylor expansion:

$$\begin{aligned}\mathbf{0} &= \mathbf{J}_b\mathbf{q}^{k+1} - \mathbf{d}_b + \mathbf{E}_b\boldsymbol{\lambda}_b^{k+1} \\ &= \mathbf{J}_b\mathbf{q}^{k+1} + \mathbf{E}_b\Delta\boldsymbol{\lambda}_b - \mathbf{d}_b + \mathbf{E}_b\boldsymbol{\lambda}_b^k\end{aligned} \tag{17a}$$

$$\begin{aligned}\mathbf{0} &= \boldsymbol{\phi}_n(\boldsymbol{\delta}_n^{k+1}, \boldsymbol{\lambda}_n^{k+1}) \\ &= \boldsymbol{\phi}_n(\boldsymbol{\delta}_n^k, \boldsymbol{\lambda}_n^k) + \mathbf{J}_n\Delta\mathbf{q} + \mathbf{E}_n\Delta\boldsymbol{\lambda}_n \\ &= \mathbf{J}_n\mathbf{q}^{k+1} + \mathbf{E}_n\Delta\boldsymbol{\lambda}_n + \boldsymbol{\phi}_n(\boldsymbol{\delta}_n^k, \boldsymbol{\lambda}_n^k) - \mathbf{J}_n\mathbf{q}^k\end{aligned} \tag{17b}$$

$$\begin{aligned}\mathbf{0} &= \boldsymbol{\phi}_f(\dot{\boldsymbol{\delta}}_f^{k+1}, \boldsymbol{\lambda}_f^{k+1}) \\ &= \boldsymbol{\phi}_f(\dot{\boldsymbol{\delta}}_f^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_n^k) + \mathbf{J}_f\Delta\mathbf{v} + \mathbf{E}_f\Delta\boldsymbol{\lambda}_f \\ &= \frac{1}{h}\Big(\mathbf{J}_f\mathbf{q}^{k+1} + h\mathbf{E}_f\Delta\boldsymbol{\lambda}_f + h\boldsymbol{\phi}_f(\dot{\boldsymbol{\delta}}_f^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_n^k) - \mathbf{J}_f\mathbf{q}^k\Big)\end{aligned} \tag{17c}$$

Assembling the system for Equations (15a) and (17), we have:

$$\begin{bmatrix} \mathbf{A} & -\mathbf{J}_b^{\mathrm{T}} & -\mathbf{J}_n^{\mathrm{T}} & -\mathbf{J}_f^{\mathrm{T}} \\ \mathbf{J}_b & \frac{1}{h^2}\mathbf{E}_b & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_n & \mathbf{0} & \frac{1}{h^2}\mathbf{E}_n & \mathbf{0} \\ \mathbf{J}_f & \mathbf{0} & \mathbf{0} & \frac{1}{h}\mathbf{E}_f \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ h^2\Delta\boldsymbol{\lambda}_b \\ h^2\Delta\boldsymbol{\lambda}_n \\ h^2\Delta\boldsymbol{\lambda}_f \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{h}_b \\ \mathbf{h}_n \\ \mathbf{h}_f \end{bmatrix} \tag{18}$$

where

$$\mathbf{g} = \mathbf{b}^k + h^2\mathbf{J}_b^{\mathrm{T}}\boldsymbol{\lambda}_b^k + h^2\mathbf{J}_n^{\mathrm{T}}\boldsymbol{\lambda}_n^k + h^2\mathbf{J}_f^{\mathrm{T}}\boldsymbol{\lambda}_f^k \tag{19a}$$

$$\mathbf{h}_b = \mathbf{d}_b - \mathbf{E}_b\boldsymbol{\lambda}_b^k \tag{19b}$$

$$\mathbf{h}_n = -\boldsymbol{\phi}_n(\boldsymbol{\delta}_n^k, \boldsymbol{\lambda}_n^k) + \mathbf{J}_n\mathbf{q}^k \tag{19c}$$

$$\mathbf{h}_f = -h\boldsymbol{\phi}_f(\dot{\boldsymbol{\delta}}_f^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_n^k) + \mathbf{J}_f\mathbf{q}^k \tag{19d}$$

Grouping the contact items such that

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_b \\ \mathbf{J}_n \\ \mathbf{J}_f \end{bmatrix}, \qquad \boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_b \\ \boldsymbol{\lambda}_n \\ \boldsymbol{\lambda}_f \end{bmatrix}, \qquad \mathbf{h} = \begin{bmatrix} \mathbf{h}_b \\ \mathbf{h}_n \\ \mathbf{h}_f \end{bmatrix},$$
$$\text{and} \quad \mathbf{E} = \begin{bmatrix} \frac{1}{h^2}\mathbf{E}_b & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1}{h^2}\mathbf{E}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{1}{h}\mathbf{E}_f \end{bmatrix}, \tag{20}$$

we can rewrite Equation (18) as

$$\begin{bmatrix} \mathbf{A} & -\mathbf{J}^{\mathrm{T}} \\ \mathbf{J} & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ h^2\Delta\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{h} \end{bmatrix} \tag{21}$$

which eventually leads to a saddle point problem after a Schur-complement:

$$\left(\mathbf{J}\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}} + \mathbf{E}\right)\Delta\boldsymbol{\lambda} = \frac{1}{h^2}\left(\mathbf{J}\mathbf{A}^{-1}\mathbf{g} - \mathbf{h}\right) \tag{22}$$

A key advantage of the non-smooth Newton method is its independence from the linear solver used for the saddle point system in Equation (22). This flexibility allows for various solver choices: decomposition methods can be used for precise solutions in small-scale problems, while Krylov subspace methods offer both good convergence and parallelization potential for large-scale problems.

After computing $\Delta\boldsymbol{\lambda}$, we solve the global step by integrating the contact forces $\boldsymbol{\lambda}^{k+1}$, with corrections to the right-hand side:

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \Delta\boldsymbol{\lambda} \tag{23a}$$

$$\mathbf{q}^{k+1} = \mathbf{A}^{-1}\left(\mathbf{b}^k + h^2\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}^{k+1}\right) \tag{23b}$$

The local-global iteration constrained with non-smooth functions is outlined in Algorithm 3. Implementing the non-smooth constraints into the local-global methods brings two major additional computing cost within each L-G iteration: the Schur-complement $\mathbf{J}\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}}$ and the linear system solution in Equation (22). As discussed earlier, the flexibility in choosing linear solvers enables us to use Krylov subspace methods, such as Conjugate Gradient (CG) and Conjugate Residual (CR), which are well-suited for parallel implementation on both CPUs and GPUs for large-scale problems.

In contrast, efficiently computing the Schur-complement is challenging as it involves solving a large linear system (scale of DOFs n) with multiple right-hand sides (scale of contact constraints c).

ALGORITHM 3: Non-smooth constrained local-global method

while *simulation* **do**
  Perform collision detection;
  $\tilde{\mathbf{q}}^k = \mathbf{q}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$;
  for $k \in \{0, ..., n\}$ **do**
    $\mathbf{p}_i^k = project(\mathbf{G}_i\mathbf{q}^k)$; // local step
    $\mathbf{b} = \mathbf{M}\tilde{\mathbf{q}} + h^2\sum_i w_i\mathbf{G}_i^{\mathrm{T}}\mathbf{p}_i^k$;
    Evaluate $\mathbf{J}$, $\mathbf{E}$, $\mathbf{g}$, $\mathbf{h}$;
    $\Delta\boldsymbol{\lambda} = \frac{1}{h^2}\left(\mathbf{J}\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}} + \mathbf{E}\right)^{-1}\left(\mathbf{h} - \mathbf{J}\mathbf{A}^{-1}\mathbf{g}\right)$; // solve constraint linear system
    $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \Delta\boldsymbol{\lambda}$;
    $\mathbf{q}^{k+1} = \mathbf{A}^{-1}\left(\mathbf{b} + h^2\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}^{k+1}\right)$; // apply constraint correction
  end
  $\mathbf{q}_{t+h} = \mathbf{q}^{k+1}$; // integration
  $\mathbf{v}_{t+h} = \frac{1}{h}\left(\mathbf{q}^{k+1} - \mathbf{q}_t\right)$;
end

Such computation can easily become the bottleneck in case of a detailed mesh discretization or large amount of contacts. The solution in [Macklin et al. 2019] simplifies the computing by replacing the Hessian by a diagonal approximation computed in each Newton iteration. In general, this should cause slower convergence since propagating the impact of constraints through the non-diagonal entries is eliminated, leading to weak coupling between the constraints. To address this problem, we present effective enhancements in subsequent sections.

5.3 Splitting out non-smooth indicators

We propose a splitting strategy to address the expensive computation of the Schur-complement $\mathbf{J}\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}}$ in each L-G iteration. Given the formulations of non-smooth Jacobian in Equations (47) (50) (53) (56) for different NCP functions, we unify these formulations in following way:

$$\mathbf{J}_{n_{\min}} = \omega_{n_{\min}}\mathbf{H}_n \quad \text{with} \quad \omega_{n_{\min}} = \begin{cases} \mathbf{1} & \text{if } \boldsymbol{\delta}_n \le \mathbf{r}\boldsymbol{\lambda}_n \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{24a}$$

$$\mathbf{J}_{f_{\min}} = \omega_{f_{\min}}\mathbf{H}_f \quad \text{with} \quad \omega_{f_{\min}} = \begin{cases} \mathbf{1} & \text{if } \boldsymbol{\lambda}_n > \mathbf{0} \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{24b}$$

$$\mathbf{J}_{n_{\text{FB}}} = \omega_{n_{\text{FB}}}\mathbf{H}_n \quad \text{with} \quad \omega_{n_{\text{FB}}} = \mathbf{1} - \frac{\boldsymbol{\delta}_n}{\sqrt{\boldsymbol{\delta}_n^2 + \mathbf{r}^2\boldsymbol{\lambda}_n^2}} \tag{24c}$$

$$\mathbf{J}_{f_{\text{FB}}} = \omega_{f_{\text{FB}}}\mathbf{H}_f \quad \text{with} \quad \omega_{f_{\text{FB}}} = \begin{cases} \mathbf{1} & \text{if } \boldsymbol{\lambda}_n > \mathbf{0} \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{24d}$$

where we define the non-smooth indicators ω as the weighting parameters for $\mathbf{H}$.

In matrix form, we can reformulate the non-smooth Jacobian as follows:

$$\mathbf{J} = \begin{bmatrix} \omega_0 & & \\ & \ddots & \\ & & \omega_c \end{bmatrix} \mathbf{H} = \Omega\mathbf{H} \tag{25}$$

where Ω is a diagonal matrix containing the non-smooth indicators.

A key finding is that the non-smooth indicators ω dominate the dynamic behavior of the non-smooth Jacobian $\mathbf{J}$, while $\mathbf{H}$ remains static throughout the current time step. This is because $\mathbf{H}$ only depends on the output of the collision detection and the constraint linearization which are performed only once at the beginning of the time step. With Equation (25), the Schur-complement in each Newton iteration becomes:

$$\mathbf{J}\mathbf{A}^{-1}\mathbf{J}^{\mathrm{T}} = \Omega\mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\Omega^{\mathrm{T}} = \Omega\mathbf{W}\Omega^{\mathrm{T}} \tag{26}$$

where $\mathbf{W} = \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}$ is called *delasus operator* in multi-body dynamics, and it only needs to be computed once per time step. Consequently, this converts the original m Schur-complement computations (where m is the number of Newton or local-global iterations) into a single Schur-complement computation at the beginning of the time step, plus $2 \times m$ diagonal matrix - matrix multiplications, which are typically much more efficient.

The splitting strategy can be applied for non-smooth Newton methods with generic linear solvers. Moreover, referring back to the sparse inverse resolution outlined in Section 4, we can efficiently compute the Schur-complement through matrix product operations as follows:

$$\mathbf{W} = \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}} = \mathbf{H}\mathbf{S}^{\mathrm{T}}\mathbf{S}\mathbf{H}^{\mathrm{T}} \tag{27}$$

In contrast to the diagonal approximation detailed in [Macklin et al. 2019], our exact solution can efficiently couple the constraints using sparse matrix - sparse matrix multiplication (SpGEMM) operations which are fundamental operations for sparse matrices.

To accelerate the computation of Equation (27), one can utilize the reuse strategy described in [Zeng et al. 2022] to exploit shared contact data between consecutive time steps.

5.4 Complementarity precondition

Another advantage of explicitly computing $\mathbf{W}$ is that it provides an effective complementarity preconditioner. As discussed in [Macklin et al. 2019], applying a preconditioner r to NCP functions dose not change the solution of the original problem. However, using the preconditioner significantly affects the convergence rate of the non-smooth Newton method and our local-global method constrained with NCP functions.

The strategy in [Macklin et al. 2019] for choosing the preconditioner is to ensure both sides of the complementarity equation have the same slop: For a unilateral constraint, $\mathbf{r}_j = h^2\left[\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\right]_{jj}$ is used to achieve appropriate scaling in the position-force relationship between $\boldsymbol{\delta}_j$ and $\boldsymbol{\lambda}_j$. For a frictional constraint, the scaling factor should be $\mathbf{r}_j = h\left[\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\right]_{jj}$ to follow the velocity-force relationship between $\dot{\boldsymbol{\delta}}_j$ and $\boldsymbol{\lambda}_j$.

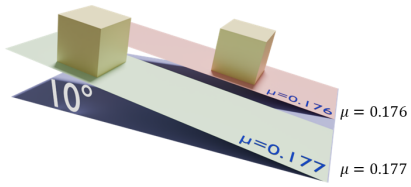
However, there are two problems to use such preconditioner. First, the preconditioner $\mathbf{r}$ is computed by $\mathbf{J}$ while $\mathbf{J}$ actually depends on $\mathbf{r}$, leading to a circular dependency issue. Second, using the mass inverse will decrease the efficiency of the preconditioner. This is particularly obvious in soft body simulations with high-stiffness materials, where the stiffness matrix, rather than the mass matrix, dominates the system's diagonal elements.

To address these problems, we propose computing the preconditioner using the diagonal elements in the *delasus operator*:

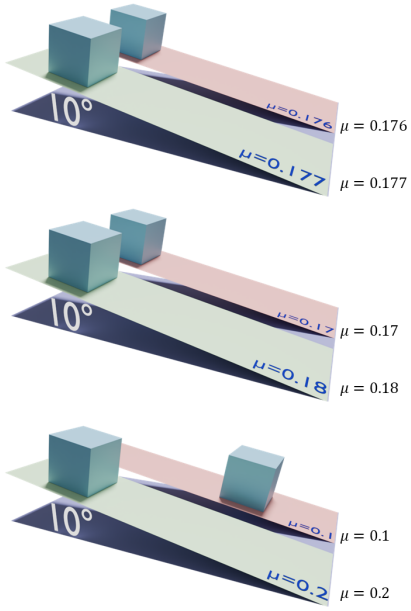
$$\text{Unilateral constraint:} \quad \mathbf{r}_j = h^2\mathbf{W}_{jj} \tag{28a}$$

$$\text{Frictional constraint:} \quad \mathbf{r}_j = h\mathbf{W}_{jj} \tag{28b}$$

We give several reasons for using this strategy: First, our preconditioner uses $\mathbf{A}^{-1}$, instead of $\mathbf{M}^{-1}$ to couple the constraints, efficiently taking the stiffness into consideration. Second, the *delasus operator* $\mathbf{W}$ is computed at the beginning of each time step, thereby avoiding the circular dependency issue. Finally, as shown in previous studies of complementarity problems [Erleben 2013], the *delasus operator*



(a) Our system inverse $\left[\mathbf{H}\mathbf{A}^{-1}\mathbf{H}^{\mathrm{T}}\right]_{jj}$ complementarity preconditioner.



(b) Mass inverse $\left[\mathbf{H}\mathbf{M}^{-1}\mathbf{H}^{\mathrm{T}}\right]_{jj}$ complementarity preconditioner.

Fig. 4. Stick-Sliding test: simulating a high-stiff block on a slope. With the analytical stick-sliding discontinuity $\mu^* = tan(10\pi/180) = 0.17632698$, the ideal behavior is that the box sticks to the plane when $\mu > \mu^*$, and slides when $\mu \leq \mu^*$. (a) Our complementarity preconditioner with the system inverse achieves the accuracy level at 0.001. (b) The complementarity preconditioner computed with the mass inverse can only capture the discontinuity for stick-sliding behavior with an accuracy of 0.1.

ALGORITHM 4: A unified and GPU-friendly pipeline for solving elastic dynamics involving fricitonal contacts

$\mathbf{L} = Cholesky(\mathbf{A})$;
$\mathbf{S} = \mathbf{L}^{-1}\mathbf{I}$;
while *simulation* **do**
  Perform collision detection;
  $\mathbf{W} = \mathbf{H}\mathbf{S}^{\mathrm{T}}\mathbf{S}\mathbf{H}^{\mathrm{T}}$; // computing delasus operator
  $\tilde{\mathbf{q}} = \mathbf{q}_t + h\mathbf{v}_t + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$;
  for $k \in \{0, ..., n\}$ **do**
    $\mathbf{p}_i^k = project(\mathbf{G}_i\mathbf{q}^k)$; // local step
    $\mathbf{b} = \mathbf{M}\tilde{\mathbf{q}} + h^2 \sum_i w_i \mathbf{G}_i^{\mathrm{T}} \mathbf{p}_i^k$;
    Evaluate Ω, $\mathbf{J}$, $\mathbf{E}$, $\mathbf{g}$, $\mathbf{h}$;
    $\Delta\boldsymbol{\lambda} = \frac{1}{h^2}\left(\Omega\mathbf{W}\Omega^{\mathrm{T}} + \mathbf{E}\right)^{-1}\left(\mathbf{h} - \mathbf{J}\mathbf{S}^{\mathrm{T}}\mathbf{S}\mathbf{g}\right)$; // solve constraint linear system
    $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \Delta\boldsymbol{\lambda}$;
    $\mathbf{q}^{k+1} = \mathbf{S}^{\mathrm{T}}\mathbf{S}\left(\mathbf{b} + h^2\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}^{k+1}\right)$; // apply constraint correction
  end
  $\mathbf{q}_{t+h} = \mathbf{q}^{k+1}$; // integration
  $\mathbf{v}_{t+h} = \frac{1}{h}\left(\mathbf{q}^{k+1} - \mathbf{q}_t\right)$;
end

$\mathbf{W}$ inherently acts as a scaling factor by establishing the connection between interpenetration and contact force (e.g., $\boldsymbol{\delta} = h^2\mathbf{W}\boldsymbol{\lambda}$ for LCP [Duriez et al. 2006; Zeng et al. 2022]).

Consequently, our complementarity preconditioning strategy provides effective scaling for the NCP parameters. In Section, 7.6 we present experimental results demonstrating our method's efficiency in accurately capturing the discontinuities between sticking and sliding behavior.

6 A GPU-friendly framework and implementation

Our final integrated algorithm is presented in Algorithm 4, where the modifications in orange highlight our core contributions with efficient forward matrix operations.

In conclusion, we elaborate on the synergistic efficiency of combining the sparse inverse local-global method with the non-smooth Newton method.

The sparse inverse local-global method relies on system invariability, precluding penalty-based contact methods that would modify the system matrix. As a Lagrange multiplier approach, the non-smooth Newton method operates without such alterations, effectively ensuring the fulfillment of complementarity conditions during global steps.

In contrast, complete-solution-based local-global methods facilitate achieving the desired precision with a limited number of iterations. This characteristic aligns well with the preferences of the non-smooth Newton method, as it helps reduce significant computational cost. Notably, the sparse inverse method provides efficient computation of the Schur-complement by transforming the linear problem into matrix multiplication operations.

This unified approach enables parallelization across all major computational stages, making it well-suited for GPU acceleration. The implementation details are as follows.

Matrix operations. As discussed in Section 4.2, the matrix operations are highly parallelizable on both the CPU and the GPU. We use the NVIDIA's cuSPARSE library to implement SpMV operations in global steps. The Schur-complement (Equation (27)) can be implemented with SpGEMM operations. To accelerate this process, we implement the reusing strategy in [Zeng et al. 2022] to reduce computational costs.

Constraint linear solver and NCP functions. We choose the Conjugate Residual (CR) as our linear solver in the constraint space, and the Fischer-Burmeister function as our NCP function, since their efficiency has been proven in [Macklin et al. 2019]. We implement a GPU version of the CR solver using NVIDIA cuBLAS library.

Local step. Following [Overby et al. 2017], we implement the singular value decomposition (SVD) and a small-scale L-BFGS quasi-Newton method to solve the local nonlinear problems in Equation (5). Such process is naturally parallelizable owning to the independency of the local problems to each other. Since the local step is not the core of our methods, we implement a CPU-based parallelization for it. The migration of this process to create a fully GPU-based simulator remains future work. Our current implementation utilizes a hybrid CPU-GPU framework.

Collision detection. Our contact method, based on the non-smooth Newton method, is decoupled with the collision detection. It is compatible with any detection method that provides accurate collision pair information. Like in [Ly et al. 2020] and [Macklin et al. 2019], we implement collision detection using simple proximity queries with basic parallelization strategies. For simplicity, we do not activate the self-collision handling in our scenarios. However, our method supports self-collisions in principle, as the contact Jacobian $\mathbf{H}$ can effectively couple the DOFs within the same object.

Chebyshev acceleration. As indicated in [Wang 2015], local-global methods can be accelerated with the Chebyshev method. However, since the complete solution converges fast in the first iterations where the Chebyshev method shows minimal impact, we do not use the Chebyshev acceleration for our examples which generally achieve real-time performance with just a few local-global iterations.

7 Results

7.1 Experimental setup

In this section, we evaluate the performance and accuracy of our proposed framework. The simulation tests are performed on a computer equipped with an Intel@ i9-13900KF CPU 24-Core at 5.5GHz with 128GB RAM, and a GPU NVIDIA GeForce RTX 4090 with 24GB RAM.

Our tests cover various challenging scenarios, including large-scale systems, large deformation, non-smooth and massive contacts,

and accurate frictions. Each simulation typically runs with 5 local-global iterations, and 10 Conjugate Residual iterations for the constraint solver if contacts are involved. The detailed statistics for the tests are summarized in Table 4.

7.2 Memory usage

We evaluate the memory usage for storing the inverted Cholesky factor $\mathbf{S} = \mathbf{L}^{-1}\mathbf{I}$ for objects of varying shapes, discretizations, and element types. Tables 1 and 2 present memory usage comparisons for surface and volume meshes in our test scenarios. As discussed in Section 4, the system inverse $\mathbf{A}^{-1}$ shows a high density, and storing such matrix in the memory becomes soon inhibitive as the problem size increases. In contrast, our method requires reasonable memory usage for the sparse inverse of the Cholesky factor. For large objects with $20k$ nodes ($60k$ DOFs), the memory cost for storing $\mathbf{S}$ generally remains below 1GB, with the volumetric bar being the only exception.

The topology connection strongly affects matrix sparsity, as demonstrated in Table 2. Generally, objects with more complex shapes have fewer connections (fewer elements), resulting in sparser matrices and lower memory requirements for $\mathbf{S}$. This conclusion is also evident in the triangle meshes shown in Table 1 where the bending constraints introduce virtual topology connections and increase the matrix fill-in.

7.3 Performance and accuracy evaluation

In Figure 5, we compare the convergence rate and the performance between our method and incomplete solutions. In the test, a bar is twisted with rotational boundary conditions applied to its ends (Figures 5a). We take the Jacobi's method [Wang 2015] as a baseline and compute the relative error against the ground truth $\mathbf{q}^*$. Since both complete and incomplete solutions can be accelerated via the Chebyshev method [Wang 2015], we exclude the Chebyshev method from our comparison.

In Figure 5b, we plot the convergence rate and observe that the complete solution quickly reduces the errors in the first iterations, for both low and high stiffness systems. In contrast, the incomplete solution, approximating the global step with a single Jacobi iteration, converges slowly due to the inefficiency in propagating the local results. Furthermore, the incomplete method exhibits severely limited convergence for highly stiff systems. For example, when applied to an elastic bar with Young's modulus $E \geq 10^7$, the method fails to achieve an accuracy of 10^{-3} even after 1000 iterations.

In Figure 5c, we compare the relative error with respect to computational time. Combining the effect of computational efficiency (both local and global costs) and convergence rates, our method substantially outperforms the incomplete approach across all stiffness levels. This gap will be more significant when solving nonlinear local problems, such as simulating Neo-Hookean materials, which requires more computational costs in local steps. This highlights the inherent simplicity of our method: even with a basic implementation of the local step, our method achieves a high accuracy within real-time computational constraints. Conversely, while the incomplete method may achieve favorable results with optimized local steps, its performance exhibits high sensitivity to implementation details.

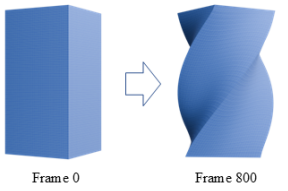
The magnified plot (Figure 5c, right) illustrates the rapid convergence of our method within real-time computational constraints, achieving high accuracy (error < 10^{-3}) for both low and high stiffness elastic materials. In contrast, for low-stiffness systems, the incomplete method achieves a limited accuracy (error=0.1) within real-time constraints. The performance deteriorates further in high-stiffness scenarios, where solutions deviate significantly from the ground truth.

Table 1. Memory usage (MB) for storing matrices inverse (in double floating) with triangular surface meshes.

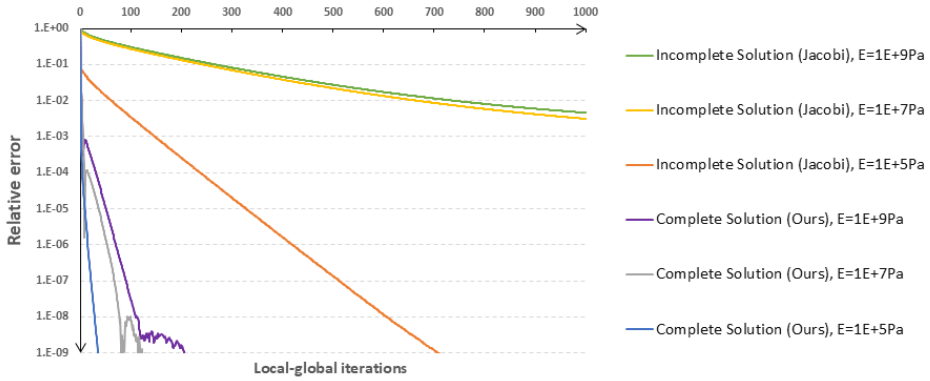
Shape	Square cloth (no bending)				Square cloth (isometric bending)				Square cloth (Laplace-Beltrami bending)			
Vertex	5.1k	10.2k	15.3k	20.2k	5.1k	10.2k	15.3k	20.2k	5.1k	10.2k	15.3k	20.2k
Tri.	10.0k	20.2k	30.3k	40.0k	10.0k	20.2k	30.3k	40.0k	10.0k	20.2k	30.3k	40.0k
$\mathbf{A}$	1.67	3.37	5.05	6.67	2.58	5.20	7.81	10.31	5.22	10.58	15.93	21.08
$\mathbf{L}$	3.34	7.58	11.87	16.42	9.81	22.83	37.29	51.51	18.37	43.36	71.58	99.85
$\mathbf{A}^{-1}$	893.33	3589.46	8050.51	14010.30	893.33	3589.46	8050.51	14010.30	893.33	3589.46	8050.51	14010.30
$\mathbf{S}$	34.80	102.51	183.79	280.41	62.37	183.82	342.68	523.36	87.08	258.85	487.14	749.10

Table 2. Memory usage (MB) for matrices inverse (in double floating) with tetrahedral volume meshes

Shape	Raptor			Ball		Bar			Wooper		Gingerbreadman	
Vertex	10.0k	15.0k	20.1k	7.1k	14.7k	5.3k	11.3k	20.8k	5.3k	11.1k	11.8k	19.5k
Tetra.	33.9k	54.4k	77.0k	37.5k	81.7k	26.4k	59.0k	111.1k	20.4k	52.5k	48.2k	93.6k
$\mathbf{A}$	5.49	8.40	11.48	4.68	9.95	3.36	7.35	13.67	3.00	7.14	6.66	12.25
$\mathbf{L}$	19.21	37.98	63.44	51.53	152.86	28.26	84.23	203.90	13.46	51.31	36.01	90.37
$\mathbf{A}^{-1}$	3661.43	8071.55	14308.70	1744.86	7452.21	961.48	4414.98	14870.70	973.51	4784.48	4255.27	13045.50
$\mathbf{S}$	120.00	260.85	467.58	266.87	946.97	125.02	463.03	1310.19	67.14	295.09	227.30	619.41

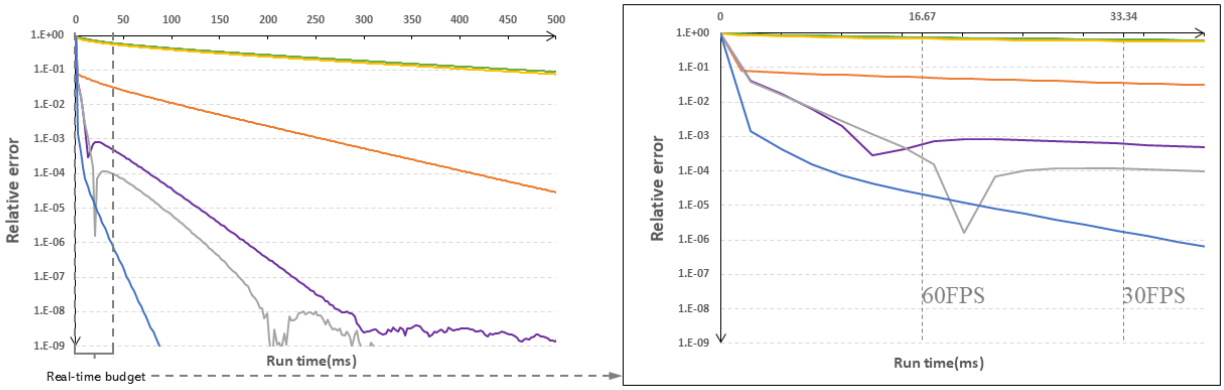


Vertices	*11.3k*
Tetrahedra	*59.0k*
ρ	$1000kg/m^3$
υ	0.45



(a) Test scenario: twisting bar.

(b) Comparison of relative error with respect to L-G iterations.



(c) Comparison of relative error with respect to computational time (ms): convergence rate (left) and performance at real-time frame rates of 30 FPS and 60 FPS (right).

Fig. 5. Performance and accuracy evaluation in scenario under large rotational deformation: The relative error is defined as $(\epsilon(\mathbf{q}^k) - \epsilon(\mathbf{q}^*))/(\epsilon(\mathbf{q}^0) - \epsilon(\mathbf{q}^*))$ with ϵ the system energy and $\mathbf{q}^*$ the ground truth (the converged solution in Newton's method). (a) At Frame 800 where the bar is largely deformed, we compare the convergence rate in different stiffness settings. (b) We plot the same comparison of convergence behavior with particular emphasis on the accuracy achieved within real-time computational constraints.

7.4 Comparison with forward/backward substitutions

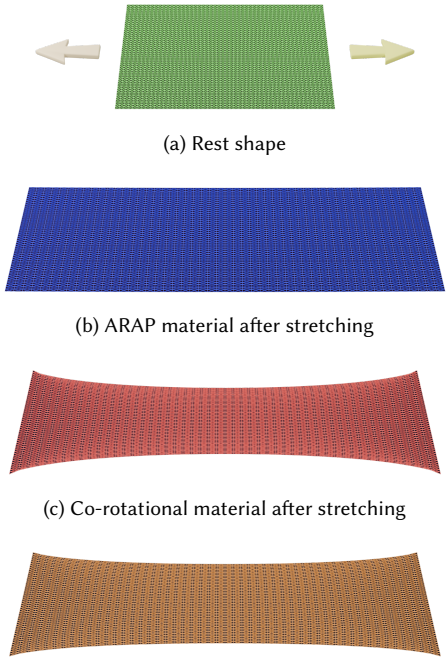
In order to comprehensively evaluate the performance of our sparse inverse method proposed in Section 4.2, we conducted experiments using the "Twisting Bar" benchmark in Section 7.3 to compare the matrix multiplications and forward/backward substitutions on the GPU. NVIDIA cuSPARSE provides the SpSV function for STS, enabling direct comparisons within a consistent computational framework. As illustrated in Table 3, the SpMV operation consistently outperforms SpSV by approximately 5×– 10× across various vertex counts, indicating a significant efficiency advantage. This performance gap is expected to widen further in scenarios involving Schur-complement operations in Equation (27) due to the presence of multiple RHS.

Table 3. Performance comparison between SpSV and SpMV in the "Twisting Bar" experiment in different scales of problem.

	Operation	5.3k	11.3k	20.8k
Forward	SpSV ($\mathbf{Lx} = \mathbf{y}$)	1.33 ms	2.92 ms	5.85 ms
	SpMV ($\mathbf{x} = \mathbf{Sy}$)	0.11 ms	0.39 ms	1.09 ms
Backward	SpSV ($\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{y}$)	1.34 ms	2.94 ms	5.94 ms
	SpMV ($\mathbf{x} = \mathbf{S}^{\mathrm{T}}\mathbf{y}$)	0.12 ms	0.41 ms	1.09 ms

7.5 Hyperelastic material

To show the compatibility with different hyperelastic materials, we make extension tests similar to those in [Trusty et al. 2022] and [Macklin et al. 2019]. Figure 6 compares different elastic materials (Neo-Hookean, co-rotational, and ARAP) applied to an identical square cloth. We observe different volume preservation capability: The linear co-rotational model exibits high volume loss compared to the Neo-Hookean model, while the ARAP material only deforms orthogonally due to the omission of the preservation item in its local energy formulation.



(a) Rest shape

(b) ARAP material after stretching

(c) Co-rotational material after stretching

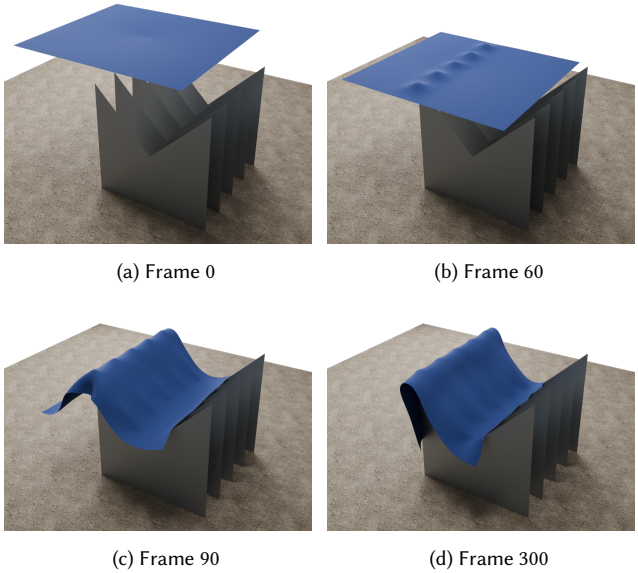
(d) Neo-Hookean material after stretching

Fig. 6. Cloth Extension: capability of volume preservation with different elastic materials.

7.6 Complementarity precondition and friction accuracy

To evaluate the accuracy of the complementarity preconditioner proposed in Section 5.4, we make a stick-sliding test similar to that in [Larionov et al. 2021]. In Figure 4, we simulate an elastic square box with high stiffness ($\rho = 1000kg/m^3$, $E = 10^8Pa$) on a slope inclined at 10 degrees to the horizontal. The analytical stick-sliding discontinuity occurs at $\mu^* = tan(10\pi/180) = 0.17632698$. In this test, the problems are solved with 10 L-G iterations and 24 CR iterations for constraint resolution.

While the mass-inverse approach [Macklin et al. 2019] achieves a sliding accuracy of only 0.1, our system-inverse method based on Equation (28) captures such discontinuity with a precision of 0.001. This aligns with our analysis in Section 5.4: For systems with high stiffness, it is generally the stiffness item that dominates the system diagonal. In this case, the mass inverse elements are generally significantly larger than those of the actual system inverse. As a result, the non-smooth indicators frequently fall within the range $|\dot{\boldsymbol{\delta}}_f| \leq \mathbf{r}(\mu\boldsymbol{\lambda}_n - |\boldsymbol{\lambda}_f|)$. In the phase of solving linear system, this forces the numerical solver to reduce $\mathbf{v}$ to zeros, resulting in undesired sticking behavior (Figure 4b). Consequently, this gap makes it hard to capture the discontinuities within a few iterations. In contrast, our method uses exactly the system inverse, resulting in efficient capture of the discontinuities and high accuracy of friction behavior.



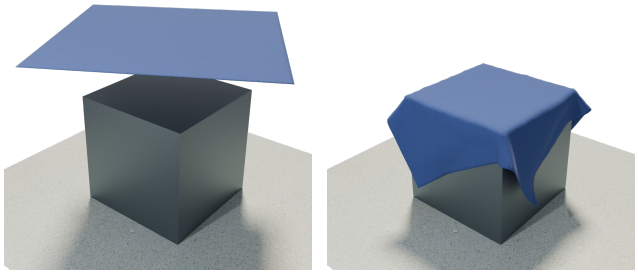
(a) Frame 0 (b) Frame 60

(c) Frame 90 (d) Frame 300

Fig. 7. Cloth on Knives: stable cloth simulation with non-smooth and codimensional contacts.

7.7 Non-smooth and codimensional contacts

We apply our methods in a scenario similar to that in [Li et al. 2020]. In Figure 7, a square cloth falls onto codimensional triangle-shaped obstacles. Unlike the IPC test, we use a triangle surface mesh instead of a thin volumetric mat. Collision handling for triangle meshes is generally more challenging than for volumetric meshes with finite thickness, as infinitely thin structures can generate large motions that typically lead to instabilities. Owning to the fast convergence of the CR solver, our method can efficiently reach a desired accuracy in a few iterations, thereby maintaining a good stability.

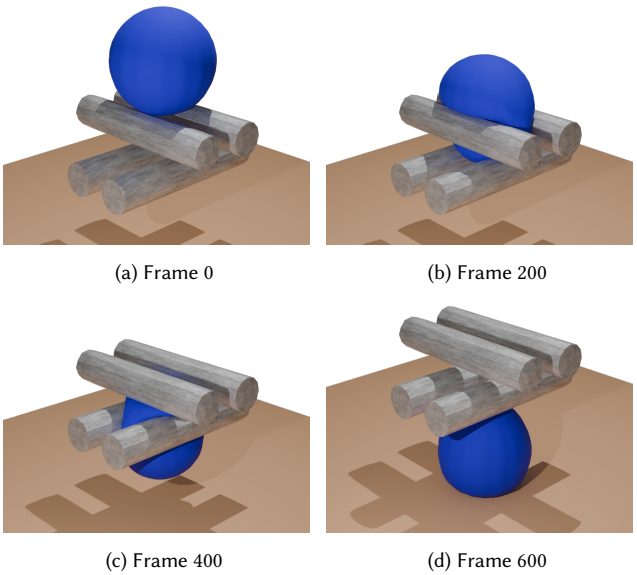


(a) Frame 0 (b) Frame 120

Fig. 8. Sharp Corner: stable simulation of cloth mesh on sharp corner obstacle, with rich and non-smooth contacts.

Another scenario with non-smooth contacts is demonstrated in Figure 8, where a cloth falls on to a static cube obstacle. Rich contact is the key difference between this example and the previous one.

For a cloth containing $10.2k$ vertices, such collision event generates $8.62k$ contact constraints (see Table 4), creating a challenging large-scale contact problem. The CR solver efficiently reduces errors within a few iterations, resulting in stable behavior. Computing the Schur complement is computationally intensive even with acceleration techniques. Owing to our splitting method for non-smooth indicators in Equation (24), such computation should only be performed once per frame (unlike per Newton iteration in [Macklin et al. 2019]), significantly reducing computational costs.



(a) Frame 0 (b) Frame 200

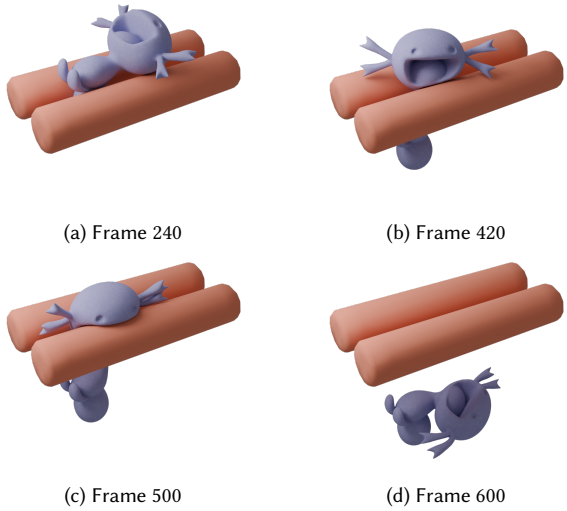
(c) Frame 400 (d) Frame 600

Fig. 9. Squeezing Ball: the rolling cylinders strongly squeeze the soft ball, thereby generating large frictional forces (with $\mu = 0.5$) that drive the ball through the gaps.

7.8 Large deformation with contacts

We reproduce another example from [Li et al. 2020]. In Figure 9, a soft ball drops onto a set of rollers. While being strongly squeezed by the rolling cylinders, the ball is under large compressions and therefore generates strong resistant contact forces with the Neo-Hookean material. This further generates large frictional forces that pulls the ball through the rollers. After passing through the obstacles, the soft ball is able to recover its shape, exhibiting a strong stability. In Table 4, our method efficiently simulates such example at near real-time rates ($46.15ms$ per frame for $7k$ vertices), significantly outperforming IPC technique ($60s$ per frame with same number of vertices).

We further test contact handling for highly stiff objects undergoing large deformations. In Figure 10, a complex-shaped soft wooper ($E = 10^7 Pa$) is pulled with a moving positional constraint. Another similar example is in Figure 1 where a gingerbread man ($E = 10^6 Pa$) is being pulled though several obstacles. In both cases, the elastic materials show an interesting behavior to automatically deform and fit with the obstacle shape. Our method is proven to be stable to simulate these high-stiffness objects, being a key difference with the



(a) Frame 240 (b) Frame 420

(c) Frame 500 (d) Frame 600

Fig. 10. Pulling Wooper: a moving positional constraint is applied on the tail of the wooper, pulling it through the thin gap between two cylinders.

incomplete solutions and PBD-like methods (e.g., VBD [Chen et al. 2024]). This is owning to the high efficiency on propagating the local results through the complete solution, as discussed in Section 4.1.

7.9 Comparison with [Ly et al. 2020]

Under the same global solution strategy, the method proposed in [Ly et al. 2020] is expected to achieve better performance in solving contact problems. This is because their approach introduces only a minor overhead per L-G iteration to check non-smooth conditions for each contact, whereas our method requires assembling and solving a linear system in the constraint space. However, in their method, contact constraints are not coupled within each L-G iteration, potentially resulting in more iterations to accurately resolve contact interactions. Moreover, as noted in their paper, their approach only supports nodal contacts, making it highly challenging to simulate non-smooth contacts, such as those presented in Figures 8 and 7.

8 Limitations and future work

Generic local-global methods. Our sparse inverse method efficiently accelerates the global steps in general local-global integrators. Although our implementations are mainly based on PD and ADMM-PD, the sparse inverse strategy applicable to various advanced techniques such as LBFGS-PD [Liu et al. 2017], WRAPD [Brown and Narain 2021], and Mixed-FEM [Trusty et al. 2022]. While these methods generally improve the L-G convergence through L-BFGS with line search, they all rely on a constant and pre-factorized system as defined in Equation (6) for efficient computing, thereby can effectively benefit from our sparse inverse method. We leave the parallelization of these methods as a future work.

Intersection-free collision. The collision response in our pipeline is penetration-based. Although the Signorini-Coulomb condition in

Table 4. Simulation statics: we report the detail of examples in Section 7. All simulations employ **5 L-G iterations**, **10 CR iterations** (if contacts are involved), and a fixed time interval $h = 0.01s$. For each example, we choose several discretizations (from $5k$ to $20k$ vertices) for the soft objects to compare the performance in different scales of problem. We detail the memory usage for storing the sparse inverse S, and maximum number of contact constraints during simulation. Owning to our splitting non-smooth indicator strategy, the Schur-complement (Schur.) is performed once per frame and the other steps are performed once per LG iteration. The SpMV operations in the global step (Global), the assembly phase (Assem.), the constraint linear solution (Cst. Solve), the constraint correction (Corr.), and the total constrained global step (Cst. Global) are presented respectively. The total cost of each frame without the collision detection (Frame max*) is measured when maximum contact pairs are presented. *The local step (Local*) is parallelized on the CPU, while the other steps are on the GPU, therefore the total time step (Time Step max*) being the measure of a hybrid implementation.

Example	Mesh		Elasticity		Memory	Contact	Time cost (ms)							
	Vert.	Elem.	Constitutive	$\rho(kg/m^3)$, $E(Pa)$, ν	S (MB)	max.	Schur.	Local*	Global	Assem.	Cst. Solve	Corr.	Cst. Global	Time Step max*
Twisting Bar	5.3k	26.4k	ARAP	1000, 1E+9, 0.45	125.02	N/A	N/A	0.87	0.27	N/A	N/A	N/A	N/A	6.23
	11.3k	59.0k	ARAP	1000, 1E+9, 0.45	463.03	N/A	N/A	1.67	0.86	N/A	N/A	N/A	N/A	13.55
	20.8k	111.1k	ARAP	1000, 1E+9, 0.45	1310.19	N/A	N/A	2.99	2.41	N/A	N/A	N/A	N/A	28.55
Cloth Extension	10.2k	20.2k	ARAP	1000, 1E+5, 0.45	102.51	N/A	N/A	0.39	0.33	N/A	N/A	N/A	N/A	4.44
	10.2k	20.2k	Co-rotation	1000, 1E+5, 0.45	102.51	N/A	N/A	0.87	0.23	N/A	N/A	N/A	N/A	6.43
	10.2k	20.2k	Neo-Hookean	1000, 1E+5, 0.45	102.51	N/A	N/A	0.93	0.23	N/A	N/A	N/A	N/A	6.65
	20.2k	40.0k	ARAP	1000, 1E+5, 0.45	280.41	N/A	N/A	0.59	0.58	N/A	N/A	N/A	N/A	7.53
	20.2k	40.0k	Co-rotation	1000, 1E+5, 0.45	280.41	N/A	N/A	1.59	0.51	N/A	N/A	N/A	N/A	12.11
	20.2k	40.0k	Neo-Hookean	1000, 1E+5, 0.45	280.41	N/A	N/A	1.73	0.52	N/A	N/A	N/A	N/A	12.88
Cloth on Knives	5.1k	10.0k	Neo-Hookean	1000, 1E+5, 0.4	62.37	0.76k	3.45	0.47	0.13	0.25	0.54	0.18	0.98	14.73
	10.2k	20.2k	Neo-Hookean	1000, 1E+5, 0.4	183.82	1.19k	5.37	0.79	0.33	0.46	0.52	0.36	1.35	21.42
	15.3k	30.3k	Neo-Hookean	1000, 1E+5, 0.4	342.68	1.78k	11.07	1.09	0.59	0.80	0.60	0.62	2.02	33.02
Sharp Corner	5.1k	10.0k	Neo-Hookean	1000, 1E+5, 0.4	62.37	4.57k	41.45	0.34	0.14	0.87	2.79	0.15	3.82	70.77
	10.2k	20.2k	Neo-Hookean	1000, 1E+5, 0.4	183.82	8.62k	200.08	0.49	0.37	2.66	9.13	0.37	12.15	277.36
Squeezing Ball	7.1k	37.5k	Neo-Hookean	1000, 1E+4, 0.4	266.87	3.18k	11.60	2.75	0.50	0.99	1.39	0.51	2.89	46.15
	14.7k	81.7k	Neo-Hookean	1000, 1E+4, 0.4	946.97	4.62k	27.91	6.09	1.61	2.46	2.80	1.67	6.92	102.01
Pulling Wooper	5.3k	20.4k	Neo-Hookean	1000, 1E+7, 0.3	67.14	0.42k	4.76	1.25	0.14	0.26	0.54	0.19	0.98	19.95
	11.8k	52.5k	Neo-Hookean	1000, 1E+7, 0.3	295.09	0.73k	16.30	3.06	0.54	0.77	0.51	0.59	1.87	46.20
Gingerbread Man	11.1k	48.2k	Neo-Hookean	1000, 1E+6, 0.3	227.30	0.74k	8.37	2.77	0.41	0.59	0.48	0.45	1.52	34.75
	19.5k	93.6k	Neo-Hookean	1000, 1E+6, 0.3	619.41	0.80k	14.16	5.03	1.05	1.30	0.50	1.15	2.95	59.76
Grabbing Raptor	10.3k	34.7k	Neo-Hookean	10, 2E+4, 0.1	120.00	0.14k	2.42	1.72	0.23	0.37	0.62	0.28	1.27	21.49
	15.3k	55.2k	Neo-Hookean	10, 2E+4, 0.1	260.85	0.13k	2.43	2.63	0.45	0.62	0.71	0.51	1.84	29.68
	20.4k	77.8k	Neo-Hookean	10, 2E+4, 0.1	467.58	0.14k	2.62	3.51	0.80	1.08	0.63	0.89	2.61	38.62

Equation (45) theoretically ensures intersection-free contacts, such condition is generally never exactly met due to numerical reasons. Therefore, unlike IPC [Li et al. 2020], our method cannot guarantee intersection-free. Exploring such intersection-free collision could be an interesting topic in future work.

Topology changes. Since the efficiency of our sparse inverse method strongly relies on the pre-computation of S, it is challenging to to handle topology-altering events such as cutting and tearing. A possibly valuable exploration is to efficiently update S with the progressively updated Cholesky factor proposed in [Herholz and Sorkine-Hornung 2020; Zhang et al. 2022].

Hyper-rich contacts. As shown in Table 4, the Sharp Corner example exhibits significant computational costs in both the Schur-complement and the constraint linear solver when contact constraints reach a massive scale ($8.62k$). Although our splitting strategy reduces the Schur-complement computation to once per time step, such computing process, plus the large-scale linear solution process, still prevent a real-time performance. We suggest simplifying contact constraints by reducing the contact space dimension, such as grouping the contacts in specific areas [Allard et al. 2010].

9 Conclusion

In this manuscript, we introduce a unified, GPU-friendly framework for simulating elastic objects in the presence of general contacts. Through our proposed reformulation, we adeptly address nonlinear and non-smooth challenges by integrating nonlinear complementarity conditions into the local-global iterations. To optimize performance, we introduce two crucial strategies: a sparse inverse method for parallelizing the local-global integrators while maintaining a rapid convergence rate, and a splitting strategy for non-smooth indicators, which largely reduces Schur-complement computation while refining the complementarity preconditioner. With the experimental results in various examples, we effectively demonstrate the generality, efficiency, accuracy, and robustness of our method in handling hyperelasticity and frictional contacts. Notably, our approach emphasizes simplicity, as the core operations rely on standard sparse matrix operations. We believe this simplicity makes our framework applicable across various downstream fields.

Acknowledgments

We thank the graphics engineer in MiHoYo, Lei Sun, for discussions with the framework and solver. We also thank MiHoYo's artist Xuehan Zou for providing the beautiful assets. This work was supported in part by the NUS Presidential Young Professorship grant (A-0009982-00-00).

References

Yinoussa Adagolodjo, Laurent Goffin, Michel De Mathelin, and Hadrien Courtecuisse. 2019. Robotic Insertion of Flexible Needle in Deformable Structures Using Inverse Finite-Element Simulation. *IEEE Transactions on Robotics* 35, 3 (2019), 697–708. https://doi.org/10.1109/TRO.2019.2897858

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, Article 82 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778819

Sheldon Andrews and Kenny Erleben. 2021. Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2021 Courses*. ACM, Virtual Event USA, 1–124. https://doi.org/10.1145/3450508.3464571

David Baraff. 1996. Linear-time dynamics using Lagrange multipliers. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 137–146. https://doi.org/10.1145/237170.237226

Jan Bender, Matthias Müller, and Miles Macklin. 2017. A survey on position based dynamics, 2017. In *Proceedings of the European Association for Computer Graphics: Tutorials (EG '17)*. Eurographics Association, Goslar, DEU. https://doi.org/10.2312/egt.20171034 event-place: Lyon, France.

Michele Benzi and Miroslav Tuma. 2000. Orderings for Factorized Sparse Approximate Inverse Preconditioners. *SIAM Journal on Scientific Computing* 21, 5 (2000), 1851–1868. https://doi.org/10.1137/S1064827598339372 _eprint: https://doi.org/10.1137/S1064827598339372.

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, Article 6 (Feb. 2011), 14 pages. https://doi.org/10.1145/1899404.1899410

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33, 4 (July 2014). https://doi.org/10.1145/2601097.2601116 Place: New York, NY, USA Publisher: Association for Computing Machinery.

Robert Bridson and Wei-Pai Tang. 1999. Ordering, Anisotropy, and Factored Sparse Approximate Inverses. *SIAM Journal on Scientific Computing* 21, 3 (1999), 867–882. https://doi.org/10.1137/S1064827598335842 _eprint: https://doi.org/10.1137/S1064827598335842.

Morten Bro-Nielsen and Stephane Cotin. 1996. Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation. *Computer Graphics Forum* 15, 3 (1996), 57–66. https://doi.org/10.1111/1467-8659.1530057 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.1530057

Bernard Brogliato. 2016. *Nonsmooth Mechanics: Models, Dynamics and Control*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-28664-8

George E. Brown and Rahul Narain. 2021. WRAPD: weighted rotation-aware ADMM for parameterization and deformation. *ACM Trans. Graph.* 40, 4 (Aug. 2021), 1–14. https://doi.org/10.1145/3450626.3459942

George E. Brown, Matthew Overby, Zahra Forootaninia, and Rahul Narain. 2018. Accurate dissipative forces in optimization integrators. *ACM Trans. Graph.* 37, 6 (2018), 282:1–282:14. https://doi.org/10.1145/3272127.3275011

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph.* 29, 4 (July 2010), 1–6. https://doi.org/10.1145/1778765.1778775

Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. 2024. Vertex Block Descent. http://arxiv.org/abs/2403.06321 arXiv:2403.06321 [cs].

E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, and C. Duriez. 2017. Software toolkit for modeling, simulation, and control of soft robots. *Advanced Robotics* 31, 22 (Nov. 2017), 1208–1224. https://doi.org/10.1080/01691864.2017.1395362 Publisher: Taylor & Francis.

Olivier Comas, Zeike A. Taylor, Jérémie Allard, Sébastien Ourselin, Stéphane Cotin, and Josh Passenger. 2008. Efficient Nonlinear FEM for Soft Tissue Modelling and Its GPU Implementation within the Open Source Framework SOFA. In *Biomedical Simulation*, Fernando Bello and P. J. Eddie Edwards (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 28–39.

Gilles Daviet. 2020. Simple and scalable frictional contacts for thin nodal objects. *ACM Trans. Graph.* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392439

Christian Duriez. 2013. Control of elastic soft robots based on real-time finite element method. *Proceedings - IEEE International Conference on Robotics and Automation* (2013), 3982–3987. https://doi.org/10.1109/ICRA.2013.6631138

C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. 2006. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics* 12, 1 (2006), 36–47. https://doi.org/10.1109/TVCG.2006.13

Kenny Erleben. 2013. Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH 2013 Courses (SIGGRAPH '13)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2504435.2504443 event-place: Anaheim, California.

Kenny Erleben. 2018. Methodology for Assessing Mesh-Based Contact Point Methods. *ACM Trans. Graph.* 37, 3 (June 2018), 1–30. https://doi.org/10.1145/3096239

François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stéphane Cotin. 2012. SOFA: A Multi-Model Framework for Interactive Physical Simulation. In *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, Yohan Payan (Ed.). Studies in Mechanobiology, Tissue Engineering and Biomaterials, Vol. 11. Springer, 283–321. https://doi.org/10.1007/8415_2012_125

Ca Felippa. 2000. *A systematic approach to the element-independent corotational dynamics of finite elements*. Technical Report January. College Of Engineeringuniversity Of Colorado. http://www.colorado.edu/engineering/cas/Felippa.d/FelippaHome.d/Publications.d/Report.CU-CAS-00-03.pdf

Mihai Frâncu and Florica Moldoveanu. 2015. VIRTUAL TRY ON SYSTEMS FOR CLOTHES : ISSUES AND SOLUTIONS. https://api.semanticscholar.org/CorpusID:53507603

Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: a practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph.* 35, 6 (Dec. 2016). https://doi.org/10.1145/2980179.2982437 Place: New York, NY, USA Publisher: Association for Computing Machinery.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Trans. Graph.* 39, 6 (2020), 190:1–190:15. https://doi.org/10.1145/3414685.3417766

Alan George. 1973. Nested Dissection of a Regular Finite Element Mesh. *SIAM J. Numer. Anal.* 10, 2 (apr 1973), 345–363. https://doi.org/10.1137/0710032

S. Hasegawa and M. Sato. 2004. Real-time Rigid Body Simulation for Haptic Interactions Based on Contact Volume of Polygonal Objects. *Computer Graphics Forum* 23, 3 (2004), 529–538. https://doi.org/10.1111/j.1467-8659.2004.00784.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2004.00784.x

Philipp Herholz and Olga Sorkine-Hornung. 2020. Sparse cholesky updates for interactive mesh parameterization. *ACM Trans. Graph.* 39, 6 (Dec. 2020), 1–14. https://doi.org/10.1145/3414685.3417828

C. Kane, E. A. Repetto, M. Ortiz, and J. E. Marsden. 1999. Finite element analysis of nonsmooth contact. *Computer Methods in Applied Mechanics and Engineering* 180, 1 (Nov. 1999), 1–26. https://doi.org/10.1016/S0045-7825(99)00034-1

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. https://doi.org/10.1145/1409060.1409117

Theodore Kim and David Eberle. 2022. Dynamic deformables: implementation and production practicalities (now with code!). In *ACM SIGGRAPH 2022 Courses* (Vancouver, British Columbia, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 7, 259 pages. https://doi.org/10.1145/3532720.3535628

Martin Komaritzan and Mario Botsch. 2019. Fast Projective Skinning. In *Motion, Interaction and Games*. ACM, Newcastle upon Tyne United Kingdom, 1–10. https://doi.org/10.1145/3359566.3360073

T. Kugelstadt, D. Koschier, and J. Bender. 2018. Fast Corotated FEM using Operator Splitting. *Computer Graphics Forum* 37, 8 (2018), 149–160. https://doi.org/10.1111/cgf.13520 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13520

Lei Lan, Guanqun Ma, Yin Yang, Changxi Zheng, Minchen Li, and Chenfanfu Jiang. 2022. Penetration-free projective dynamics on the GPU. *ACM Trans. Graph.* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530069 Place: New York, NY, USA Publisher: Association for Computing Machinery.

Egor Larionov, Ye Fan, and Dinesh K. Pai. 2021. Frictional Contact on Smooth Elastic Solids. *ACM Trans. Graph.* 40, 2 (April 2021), 1–17. https://doi.org/10.1145/3446663

Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E. Brown, and Laurence Boissieux. 2018. An implicit frictional contact solver for adaptive cloth simulation. *ACM Trans. Graph.* 37, 4 (Aug. 2018), 1–15. https://doi.org/10.1145/3197517.3201308

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392425

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph.* 32, 6 (Nov. 2013). https://doi.org/10.1145/2508363.2508406 Place: New York, NY, USA Publisher: Association for Computing Machinery.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Trans. Graph.* 36, 3 (June 2017), 1–16. https://doi.org/10.1145/2990496

Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective dynamics with dry frictional contact. *ACM Trans. Graph.* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392396

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Viktor Makoviychuk. 2019. Non-Smooth Newton Methods for Deformable Multi-Body Dynamics. *ACM Trans. Graph.* 38, 5 (Oct. 2019), 1–20. https://doi.org/10.1145/3338695 arXiv:1907.04587 [cs].

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*. ACM, Burlingame California, 49–54. https://doi.org/10.1145/2994258.2994272

Stéphanie Marchesseau, Tobias Heimann, Simon Chatelin, Rémy Willinger, and Hervé Delingette. 2010. Multiplicative Jacobian Energy Decomposition Method for Fast Porous Visco-Hyperelastic Soft Tissue Model. In *Lecture notes in computer science*. Vol. 6361. Springer, 235–242. https://doi.org/10.1007/978-3-642-15705-9_29

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 72, 8 pages. https://doi.org/10.1145/1964921.1964967

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2006. Position Based Dynamics. *Vriphys: 3rd Workshop in Virtual Realitiy* Interactions (2006), 10 pages. https://doi.org/10.2312/PE/VRIPHYS/VRIPHYS06/071-080 Artwork Size: 10 pages ISBN: 9783905673616 Publisher: [object Object].

Rahul Narain, Matthew Overby, and George E Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. (2016).

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical optimization* (2nd ed ed.). Springer, New York. OCLC: ocm68629100.

Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Trans. Visual. Comput. Graphics* 23, 10 (Oct. 2017), 2222–2234. https://doi.org/10.1109/TVCG.2017.2730875

Yousef Saad. 2003. *Iterative Methods for Sparse Linear Systems* (second ed.). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718003 _eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718003.

Jennifer Scott and Miroslav Tůma. 2023. *Algorithms for Sparse Linear Systems*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-031-25820-6

Eftychios Sifakis and Jernej Barbic. 2012. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses (SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2343483.2343501 event-place: Los Angeles, California.

David E. Stewart. 2000. Rigid-Body Dynamics with Friction and Impact. *SIAM Rev.* 42, 1 (Jan. 2000), 3–39. https://doi.org/10.1137/S0036144599360110

D. E. STEWART and J. C. TRINKLE. 1996. AN IMPLICIT TIME-STEPPING SCHEME FOR RIGID BODY DYNAMICS WITH INELASTIC COLLISIONS AND COULOMB FRICTION. *Internat. J. Numer. Methods Engrg.* 39, 15 (1996), 2673–2691. https://doi.org/10.1002/(SICI)1097-0207(19960815)39:15<2673::AID-NME972>3.0.CO;2-I

Emanuel Todorov. 2010. Implicit nonlinear complementarity: A new approach to contact dynamics. In *2010 IEEE International Conference on Robotics and Automation*. 2322–2329. https://doi.org/10.1109/ROBOT.2010.5509739

Ty Trusty, Danny M. Kaufman, and David I. W. Levin. 2022. Mixed Variational Finite Elements for Implicit, General-Purpose Simulation of Deformables. http://arxiv.org/abs/2202.00183 arXiv:2202.00183 [cs].

Huamin Wang. 2015. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.* 34, 6 (Nov. 2015), 1–9. https://doi.org/10.1145/2816795.2818063

Qisi Wang, Yutian Tao, Eric Brandt, Court Cutting, and Eftychios Sifakis. 2021. Optimized Processing of Localized Collisions in Projective Dynamics. *Computer Graphics Forum* 40, 6 (Sept. 2021), 382–393. https://doi.org/10.1111/cgf.14385

Z. Zeng, S. Cotin, and H. Courtecuisse. 2022. Real-Time FE Simulation for Large-Scale Problems Using Precondition-Based Contact Resolution and Isolated DOFs Constraints. *Computer Graphics Forum* 41, 6 (Sept. 2022), 418–434. https://doi.org/10.1111/cgf.14563

Jiayi Eris Zhang, Jérémie Dumas, Yun (Raymond) Fei, Alec Jacobson, Doug L. James, and Danny M. Kaufman. 2022. Progressive Simulation for Cloth Quasistatics. *ACM Trans. Graph.* 41, 6 (Dec. 2022), 1–16. https://doi.org/10.1145/3550454.3555510

A Frictional contact constraints

A.1 Constraint linearization

For each contact pair j, we unify the constraint directions ($\vec{b}$, $\vec{n}$, and $\vec{f}$) as a general form $\vec{c}$:

$$\begin{aligned}\delta_{c,j} &= \vec{c}_j \cdot \mathbf{\Delta}_j \\ \lambda_{c,j} &= \vec{c}_j \cdot \mathbf{\Lambda}_j\end{aligned} \tag{29}$$

Interpenetration linearization. For the interpenetration measure along $\vec{b}$ and $\vec{n}$, we have:

$$\delta_{c,j} = \vec{c}_j \cdot \mathbf{\Delta}_j = \vec{c}_j \cdot (\mathbf{C}_j\mathbf{q} - \hat{\mathbf{p}}_j) = \vec{c}_j \cdot \mathbf{C}_j\mathbf{q} - \vec{c}_j \cdot \hat{\mathbf{p}}_j \tag{30}$$

while the relative velocity in the tangent space (which is needed in the Coulomb's law for friction formulation, see A.2) is given as follows:

$$\dot{\delta}_{f,j} = \vec{f}_j \cdot (\frac{\partial \mathbf{\Delta}_j}{\partial t} - \hat{\mathbf{u}}_j) = \vec{f}_j \cdot (\frac{\partial (\mathbf{C}_j\mathbf{q} - \hat{\mathbf{p}}_i)}{\partial t} - \hat{\mathbf{u}}_j) = \vec{f}_j \cdot \mathbf{C}_j\mathbf{v} - \vec{f}_j \cdot \hat{\mathbf{u}}_j \tag{31}$$

where $\hat{\mathbf{u}}$ is the velocity of the moving obstacle if there exist one within the contact pair (see details in [Ly et al. 2020]).

Contact forces linearization. We define the force $\vec{\boldsymbol{\lambda}}_{c,j} = \lambda_{c,j} \cdot \vec{c}_j^{\mathrm{T}} = \vec{c}_j \cdot \boldsymbol{\lambda}_j \cdot \vec{c}_j^{\mathrm{T}}$ that is the component of $\boldsymbol{\lambda}_j$ along $\vec{c}_j$. For a bilateral contact pair $j \in \mathcal{B}$, we have:

$$\mathbf{C}_j^{\mathrm{T}} \mathbf{\Lambda}_j = (\vec{b} \cdot \mathbf{C}_j)^{\mathrm{T}} \lambda_{b,j} \tag{32}$$

For a non-interpenetration contact pair $j \in C$, since the the basis $\vec{n}$ and $\vec{f}$ are orthonormal basis in euclidean space, we have:

$$\mathbf{\Lambda}_j = \vec{n}_j \cdot \mathbf{\Lambda}_j \cdot \vec{n}_j^{\mathrm{T}} + \vec{f}_j \cdot \mathbf{\Lambda}_j \cdot \vec{f}_j^{\mathrm{T}} = \lambda_{n,j} \cdot \vec{n}_j^{\mathrm{T}} + \lambda_{f,j} \cdot \vec{f}_j^{\mathrm{T}} \tag{33}$$

Consequently, the contact items in the constrained implicit Euler (2) can be separated into contact and friction items:

$$\mathbf{C}_j^{\mathrm{T}} \mathbf{\Lambda}_j = \mathbf{C}_j^{\mathrm{T}} (\vec{n}^{\mathrm{T}} \lambda_{n,j} + \vec{f}^{\mathrm{T}} \lambda_{f,j}) = (\vec{n} \cdot \mathbf{C}_j)^{\mathrm{T}} \lambda_{n,j} + (\vec{f} \cdot \mathbf{C}_j)^{\mathrm{T}} \lambda_{f,j} \tag{34}$$

Grouping linearized items. By grouping the contact terms in (30) (31) (32) (34), we defining the following contact items for different constraints $\vec{b}$, $\vec{n}$, and $\vec{f}$:

$$\begin{aligned}\mathbf{H}_b &= \begin{bmatrix} \vec{b}_{b_1} \cdot \mathbf{C}_{b_1} \\ \dots \\ \vec{b}_{b_m} \cdot \mathbf{C}_{b_m} \end{bmatrix}, & \boldsymbol{\lambda}_b &= \begin{bmatrix} \lambda_{b_1} \\ \dots \\ \lambda_{b_m} \end{bmatrix}, \\ \mathbf{d}_b &= \begin{bmatrix} \vec{b}_{b_1} \cdot \hat{\mathbf{p}}_{b_1} \\ \dots \\ \vec{b}_{b_m} \cdot \hat{\mathbf{p}}_{b_m} \end{bmatrix}, & \boldsymbol{\delta}_b &= \begin{bmatrix} \delta_{b_1} \\ \dots \\ \delta_{b_m} \end{bmatrix}\end{aligned} \tag{35}$$

$$\begin{aligned}\mathbf{H}_n &= \begin{bmatrix} \vec{n}_{c_1} \cdot \mathbf{C}_{c_1} \\ \dots \\ \vec{n}_{c_n} \cdot \mathbf{C}_{c_n} \end{bmatrix}, & \boldsymbol{\lambda}_n &= \begin{bmatrix} \lambda_{n,c_1} \\ \dots \\ \lambda_{n,c_n} \end{bmatrix}, \\ \mathbf{d}_n &= \begin{bmatrix} \vec{n}_{c_1} \cdot \hat{\mathbf{p}}_{c_1} \\ \dots \\ \vec{n}_{c_n} \cdot \hat{\mathbf{p}}_{c_n} \end{bmatrix}, & \boldsymbol{\delta}_n &= \begin{bmatrix} \delta_{n,c_1} \\ \dots \\ \delta_{n,c_n} \end{bmatrix}\end{aligned} \tag{36}$$

$$\begin{aligned}\mathbf{H}_f &= \begin{bmatrix} \vec{f}_{c_1} \cdot \mathbf{C}_{c_1} \\ \dots \\ \vec{f}_{c_n} \cdot \mathbf{C}_{c_n} \end{bmatrix}, & \boldsymbol{\lambda}_f &= \begin{bmatrix} \lambda_{f,c_1} \\ \dots \\ \lambda_{f,c_n} \end{bmatrix}, \\ \mathbf{d}_f &= \begin{bmatrix} \vec{f}_{c_1} \cdot \hat{\mathbf{u}}_{c_1} \\ \dots \\ \vec{f}_{c_n} \cdot \hat{\mathbf{u}}_{c_n} \end{bmatrix}, & \dot{\boldsymbol{\delta}}_f &= \begin{bmatrix} \dot{\delta}_{f,c_1} \\ \dots \\ \dot{\delta}_{f,c_n} \end{bmatrix}\end{aligned} \tag{37}$$

which allows us to convert the governing Equation 2 to linearized form:

$$\begin{aligned}\boldsymbol{\delta}_b &= \mathbf{H}_b\mathbf{q} - \mathbf{d}_b \\ \boldsymbol{\delta}_n &= \mathbf{H}_n\mathbf{q} - \mathbf{d}_n \\ \dot{\boldsymbol{\delta}}_f &= \mathbf{H}_f\mathbf{v} - \mathbf{d}_f\end{aligned} \tag{38}$$

$$\begin{aligned}\sum_{j\in\mathcal{L}} \mathbf{C}_j^{\mathrm{T}} \mathbf{\Lambda}_j &= \sum_{j\in\mathcal{B}} \mathbf{H}_{b,j}^{\mathrm{T}} \lambda_{b,j} + \sum_{j\in C} \mathbf{H}_{n,j}^{\mathrm{T}} \lambda_{n,j} + \sum_{j\in C} \mathbf{H}_{f,j}^{\mathrm{T}} \lambda_{f,j} \\ &= \mathbf{H}_b^{\mathrm{T}} \boldsymbol{\lambda}_b + \mathbf{H}_n^{\mathrm{T}} \boldsymbol{\lambda}_n + \mathbf{H}_f^{\mathrm{T}} \boldsymbol{\lambda}_f\end{aligned} \tag{39}$$

Computing interpenetration. Note that we do not need to explicitly compute the velocity in each L-G iteration. In practice, we can efficiently compute the interpenetration as follows:

$$\begin{bmatrix} \boldsymbol{\delta}_b^k \\ \boldsymbol{\delta}_n^k \\ h\dot{\boldsymbol{\delta}}_f^k \end{bmatrix} = \begin{bmatrix} \mathbf{H}_b\mathbf{q}^k - \mathbf{d}_b \\ \mathbf{H}_n\mathbf{q}^k - \mathbf{d}_n \\ \mathbf{H}_f(\mathbf{q}^k - \mathbf{q}_t) - h\mathbf{d}_f \end{bmatrix} = \begin{bmatrix} \mathbf{H}_b \\ \mathbf{H}_n \\ \mathbf{H}_f \end{bmatrix} \mathbf{q}^k - \begin{bmatrix} \mathbf{d}_b \\ \mathbf{d}_n \\ \mathbf{H}_f\mathbf{q}_t + h\mathbf{d}_f \end{bmatrix} \tag{40}$$

where the initial interpenetration $\begin{bmatrix} \mathbf{d}_b \\ \mathbf{d}_n \\ \mathbf{H}_f\mathbf{q}_t + h\mathbf{d}_f \end{bmatrix}$ can be computed at the beginning of the time step.

A.2 Frictional contact formulation

While the bilateral contact has a simple formula that eliminates the interpenetration $\delta_{b,j} = 0$, the non-interpenetration contact is a more complicated case that is usually formulated with the Signorini's law for the contact normal constraints:

$$0 \le \delta_{n,j} \perp \lambda_{n,j} \ge 0 \tag{41}$$

Following the principle of maximal dissipation [Stewart 2000], the frictional forces $\vec{\boldsymbol{\lambda}}_{f,j}$ remove the maximum amount of energy from the system while having their magnitude bounded by the normal forces:

$$\begin{aligned}\min_{\vec{\boldsymbol{\lambda}}_{f,j}} \quad & \dot{\vec{\boldsymbol{\delta}}}_{f,j}^{\mathrm{T}} \vec{\boldsymbol{\lambda}}_{f,j} \\ \text{s.t.} \quad & \lambda_{f,j} \le \mu_j \lambda_{n,j}\end{aligned} \tag{42}$$

where $\dot{\vec{\boldsymbol{\delta}}}_{f,j}$ measures the relative velocity in the contact space. When the contact is active ($\lambda_n > 0$), the first-order KKT conditions for Equation (42) is given by:

$$\dot{\delta}_{f,j} + \frac{|\dot{\delta}_{f,j}|}{|\lambda_{f,j}|} \lambda_{f,j} = 0 \tag{43}$$

$$0 \le |\dot{\delta}_{f,j}| \perp \mu_j \lambda_{n,j} - |\lambda_{f,j}| \ge 0 \tag{44}$$

The first condition (43) defines the direction of the frictional force $\vec{f}$ as the opposite of velocity direction. The second one (44) gives the complementarity conditions for the cases of sliding ($|\dot{\delta}_f| > 0$)

and stick ($|\dot{\delta}_f| = 0$). On the other hand, when the contact is inactive ($\lambda_n = 0$), the frictional forces should be set as 0.

Combining (41), (43), (44), (36), (37), we assemble the bilateral condition and Signorini-Coulomb condition:

$$\begin{aligned} \forall j \in C, \quad & 0 \le \delta_{n,j} \perp \lambda_{n,j} \ge 0 \\ \forall j \in \mathcal{A}, \quad & \dot{\delta}_{f,j} + \frac{|\dot{\delta}_{f,j}|}{|\lambda_{f,j}|}\lambda_{f,j} = 0 \\ \forall j \in \mathcal{A}, \quad & 0 \le |\dot{\delta}_{f,j}| \perp \mu_j \lambda_{n,j} - |\lambda_{f,j}| \ge 0 \\ \forall j \in \mathcal{I}, \quad & \lambda_{f,j} = 0 \end{aligned} \tag{45}$$

where $\mathcal{A} = \{j \in C \mid \lambda_{n,j} > 0\}$ is the set of all active contact indices, and $\mathcal{I} = \{j \in C \mid \lambda_{n,j} \le 0\}$ is its complement in C.

B Non-smooth functions

B.1 Minimum-Map formulation

For unilateral constraints:

$$\boldsymbol{\phi}_n = \begin{cases} \boldsymbol{\delta}_n & \text{if } \boldsymbol{\delta}_n \le \mathbf{r}\boldsymbol{\lambda}_n \\ \mathbf{r}\boldsymbol{\lambda}_n & \text{if } \boldsymbol{\delta}_n > \mathbf{r}\boldsymbol{\lambda}_n \end{cases} \tag{46}$$

$$\mathbf{J}_n = \frac{\partial \boldsymbol{\phi}_n}{\partial \mathbf{q}} = \begin{cases} \mathbf{H}_n & \text{if } \boldsymbol{\delta}_n \le \mathbf{r}\boldsymbol{\lambda}_n \\ \mathbf{0} & \text{if } \boldsymbol{\delta}_n > \mathbf{r}\boldsymbol{\lambda}_n \end{cases} \tag{47}$$

$$\mathbf{E}_n = \frac{\partial \boldsymbol{\phi}_n}{\partial \boldsymbol{\lambda}_n} = \begin{cases} \mathbf{0} & \text{if } \boldsymbol{\delta}_n \le \mathbf{r}\boldsymbol{\lambda}_n \\ \mathbf{r} & \text{if } \boldsymbol{\delta}_n > \mathbf{r}\boldsymbol{\lambda}_n \end{cases} \tag{48}$$

For friction constraints:

$$\boldsymbol{\phi}_f = \mathbf{J}_f \mathbf{v} + \mathbf{E}_f \boldsymbol{\lambda}_f \tag{49}$$

$$\mathbf{J}_f = \frac{\partial \boldsymbol{\phi}_f}{\partial \mathbf{v}} = \begin{cases} \mathbf{H}_f & \text{if } \boldsymbol{\lambda}_n > \mathbf{0} \\ \mathbf{0} & \text{if } \boldsymbol{\lambda}_n \le \mathbf{0} \end{cases} \tag{50}$$

$$\mathbf{E}_f = \frac{\partial \boldsymbol{\phi}_f}{\partial \boldsymbol{\lambda}_f} = \begin{cases} \mathbf{I} & \text{if } \boldsymbol{\lambda}_n \le \mathbf{0} \\ \mathbf{0} & \text{if } \boldsymbol{\lambda}_n > \mathbf{0} \;\&\; |\dot{\boldsymbol{\delta}}_f| \le \mathbf{r}(\mu\boldsymbol{\lambda}_n - |\boldsymbol{\lambda}_f|) \\ \frac{|\dot{\boldsymbol{\delta}}_f| - \mathbf{r}(\mu\boldsymbol{\lambda}_n - |\boldsymbol{\lambda}_f|)}{\mu\boldsymbol{\lambda}_n} & \text{if } \boldsymbol{\lambda}_n > \mathbf{0} \;\&\; |\dot{\boldsymbol{\delta}}_f| > \mathbf{r}(\mu\boldsymbol{\lambda}_n - |\boldsymbol{\lambda}_f|) \end{cases} \tag{51}$$

B.2 Fischer-Burmeister formulation

For unilateral constraints:

$$\boldsymbol{\phi}_n = \boldsymbol{\delta}_n + \mathbf{r}\boldsymbol{\lambda}_n - \sqrt{\boldsymbol{\delta}_n^2 + \mathbf{r}^2\boldsymbol{\lambda}_n^2} \tag{52}$$

$$\mathbf{J}_n = \frac{\partial \boldsymbol{\phi}_n}{\partial \mathbf{q}} = \left(\mathbf{1} - \frac{\boldsymbol{\delta}_n}{\sqrt{\boldsymbol{\delta}_n^2 + \mathbf{r}^2\boldsymbol{\lambda}_n^2}}\right)\mathbf{H}_n \tag{53}$$

$$\mathbf{E}_n = \frac{\partial \boldsymbol{\phi}_n}{\partial \boldsymbol{\lambda}_n} = \left(\mathbf{1} - \frac{\mathbf{r}\boldsymbol{\lambda}_n}{\sqrt{\boldsymbol{\delta}_n^2 + \mathbf{r}^2\boldsymbol{\lambda}_n^2}}\right)\mathbf{r} \tag{54}$$

For friction constraints:

$$\boldsymbol{\phi}_f = \mathbf{J}_f \mathbf{v} + \mathbf{E}_f \boldsymbol{\lambda}_f \tag{55}$$

$$\mathbf{J}_f = \frac{\partial \boldsymbol{\phi}_f}{\partial \mathbf{v}} = \begin{cases} \mathbf{H}_f & \text{if } \boldsymbol{\lambda}_n > \mathbf{0} \\ \mathbf{0} & \text{if } \boldsymbol{\lambda}_n \le \mathbf{0} \end{cases} \tag{56}$$

$$\mathbf{E}_f = \frac{\partial \boldsymbol{\phi}_f}{\partial \boldsymbol{\lambda}_f} = \begin{cases} \mathbf{I} & \text{if } \boldsymbol{\lambda}_n \le \mathbf{0} \\ \frac{\sqrt{\dot{\boldsymbol{\delta}}_f^2 + \mathbf{r}^2(\mu\boldsymbol{\lambda}_n - |\boldsymbol{\lambda}_f|)^2} - \mathbf{r}(\mu\boldsymbol{\lambda}_n - |\boldsymbol{\lambda}_f|)}{|\dot{\boldsymbol{\delta}}_f| + \mu\mathbf{r}\boldsymbol{\lambda}_n - \sqrt{\dot{\boldsymbol{\delta}}_f^2 + \mathbf{r}^2(\mu\boldsymbol{\lambda}_n - |\boldsymbol{\lambda}_f|)^2}}\mathbf{r} & \text{if } \boldsymbol{\lambda}_n > \mathbf{0} \end{cases} \tag{57}$$

C Soft manipulator

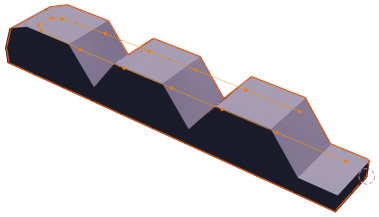


Fig. 11. Soft gripper driven by cables (simulated with bilateral constraints, highlighted with orange lines).

We apply our methods on a complex gripping task. In Figure 2, a soft gripper [Coevoet et al. 2017] performs a pick-and-place operation with a soft raptor. The gripper consists of two soft fingers (detailed in Figure 11), where bilateral constraints are used to simulate the cable constraints. By pulling and pushing the cable, the finger performs bending and straightening behavior.

While gripping the raptor, both the fingers and raptor deform to fit their contact surfaces, generating numbers of contact constraints. Lifting, rotating, and moving the raptor by the fingers are complex operations which require precise handling of frictional constraints. In this scenario, all objects and obstacles are coupled within a unified system through Equation (18) during each L-G iteration. Our pipeline efficiently simulates this multi-object system at real-time rates (see Table 4). Unlike the other scenarios, the contact constraints remain at the same level when we change the discretization of the soft raptor. This is because the gripper mesh remains unchanged, and the collision detection iterates on the finger mesh elements.