Preconditioned Nonlinear Conjugate Gradient Method for Real-time Interior-point Hyperelasticity

Xing Shen
Fuxi AI Lab, NetEase Inc
Hangzhou, Zhejiang, China
shenxing03@corp.netease.com

Runyuan Cai
Fuxi AI Lab, NetEase Inc
Hangzhou, Zhejiang, China
cairunyuan@corp.netease.com

Mengxiao Bi
Fuxi AI Lab, NetEase Inc
Hangzhou, Zhejiang, China
bimengxiao@corp.netease.com

Tangjie Lv
Fuxi AI Lab, NetEase Inc
Hangzhou, Zhejiang, China
hzlvtangjie@corp.netease.com

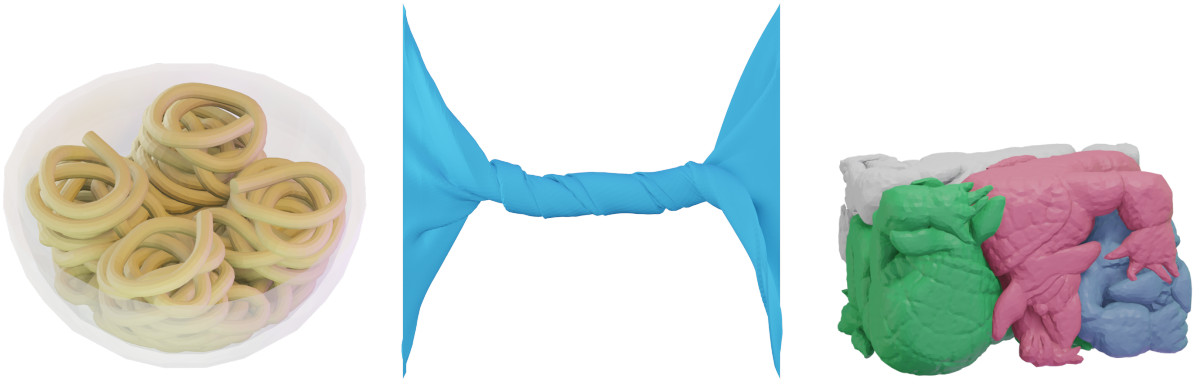


Figure 1: Example simulation results involving complex self-collision scenarios.

ABSTRACT

The linear conjugate gradient method is widely used in physical simulation, particularly for solving large-scale linear systems derived from Newton's method. The nonlinear conjugate gradient method generalizes the conjugate gradient method to nonlinear optimization, which is extensively utilized in solving practical large-scale unconstrained optimization problems. However, it is rarely discussed in physical simulation due to the requirement of multiple vector-vector dot products. Fortunately, with the advancement of GPU-parallel acceleration techniques, it is no longer a bottleneck. In this paper, we propose a Jacobi preconditioned nonlinear conjugate gradient method for elastic deformation using interior-point methods. Our method is straightforward, GPU-parallelizable, and exhibits fast convergence and robustness against large time steps. The employment of the barrier function in interior-point methods necessitates continuous collision detection per iteration to obtain a penetration-free step size, which is computationally expensive and challenging to parallelize on GPUs. To address this issue, we introduce a line search strategy that deduces an appropriate step size in a single pass, eliminating the need for additional collision detection. Furthermore, we simplify and accelerate the computations of Jacobi preconditioning and Hessian-vector product for hyperelasticity and barrier function. Our method can accurately simulate objects comprising over 100,000 tetrahedra in complex self-collision scenarios at real-time speeds.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**.

KEYWORDS

Physics-based simulation, Nonlinear conjugate gradient method, GPU

ACM Reference Format:
Xing Shen, Runyuan Cai, Mengxiao Bi, and Tangjie Lv. 2024. Preconditioned Nonlinear Conjugate Gradient Method for Real-time Interior-point Hyperelasticity. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657490

SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657490

1 INTRODUCTION

The contact simulation of elastically deformable objects is an important research topic in numerous applications, including surgical training, robotics, augmented reality/virtual reality (AR/VR), and digital fashion [Choi and Ko 2005; Meier et al. 2005; Popescu et al. 1999; Umedachi et al. 2013; Wang 2018]. However, accurately and robustly simulating the dynamic behavior of elastic objects with complex self-contact configurations under different external conditions is a major challenge due to the nonlinear and non-convex nature of elasticity. To address this challenge, a recent method called Incremental Potential Contact (IPC) [Li et al. 2020] has been proposed.

IPC incorporates the interior-point method to enable robust, accurate, and differentiable simulation of elastodynamics and contact. The method utilizes a log barrier function to approximate the inequality constraints arising from collisions and contacts, thereby converting the problem into an unconstrained optimization problem, Newton's method is then employed to solve this optimization problem. The robustness of IPC is attributed to its line search mechanism. Initially, continuous collision detection (CCD) is employed to determine the maximum step size, ensuring that objects remain penetration-free. Subsequently, a backtracking line search is implemented to achieve a decrease in the objective function. However, the practical application of IPC is hindered by its relatively slow simulation speed.

One main computational cost of IPC is solving the large-scale linear system derived from Newton's method at every iteration. The linear conjugate gradient method is a common choice to solve large-scale linear systems because it can be easily parallelized on GPUs. Since the entire optimization problem is highly nonlinear, instead of first using Newton's method for optimization and then solving the corresponding linear system with the conjugate gradient method, why not directly use the nonlinear conjugate gradient method to solve the entire optimization problem?

In paper [Wang and Yang 2016], nonlinear conjugate gradient is compared with the proposed Jacobi preconditioned gradient method with Chebyshev acceleration. Although the nonlinear conjugate gradient method demonstrates better convergence than the proposed method, it exhibits poorer overall performance due to the computation cost of the vector-vector dot product. The author reports a time consumption of 0.41ms for a vector-vector dot product over a 15K-vertices model, utilizing the NVIDIA GeForce GTX TITAN X GPU and CUDA thrust library. Fortunately, with advancements in GPU hardware and parallel techniques, the computational cost of the dot product has significantly decreased. By leveraging the Taichi language [Hu et al. 2019], MeshTaichi package [Yu et al. 2022] and NVIDIA RTX 4090 GPU, the vector-vector dot product calculation now requires only 0.012 ms for the same model, and performing 4 dot products within one for-loop requires just 0.016 ms. With the significantly reduced time consumption of vector-vector dot product, the nonlinear conjugate gradient method becomes a promising approach to discuss.

Nonlinear conjugate gradient(NCG) method is a well-established approach for unconstrained optimization. It demonstrates a convergence speed comparable to that of the Quasi-Newton method and is particularly suitable for GPU parallel acceleration. Numerous variations of the algorithm have been proposed to improve convergence for different optimization problems [Andrei et al. 2020]. However, in the field of physics simulation, researchers typically rely on the classic Fletcher-Reeves and Polak-Ribière algorithms [Fletcher and Reeves 1964; Polak and Ribiere 1969]. The essence of the nonlinear conjugate gradient method lies in its strategy for calculating the search direction, which primarily involves performing several vector-vector dot products. With our implementation, the computational cost of vector-vector dot product is relatively low, and the differences in computational time per iteration of different algorithms are negligible. Therefore, by finding a conjugate gradient algorithm that converges faster, we can directly accelerate the entire process of physical simulation.

The line search is another significant computational burden in IPC. Firstly, the computational cost of CCD is relatively high and it is challenging to accelerate through GPU parallelization. Secondly, despite CCD providing an upper bound for the step size, a backtracking line search is still necessary for convergence. All of these operations aim to determine an appropriate step size. Therefore, if we can directly deduce a suitable step size from the available information, it would eliminate the need for collision detection and significantly speed up the algorithm.

In this paper, we propose a preconditioned nonlinear conjugate gradient (PNCG) method for interior-point hyperelasticity and demonstrate that Dai-Kou [Dai and Kou 2013] conjugate gradient algorithm has the fastest convergence rate. Our PNCG method is straightforward, GPU-parallelizable, and exhibits fast convergence and robustness against large time steps. We introduce a line search strategy to deduce an appropriate step size in one pass and achieve penetration-free simulation in complex scenarios without any additional collision detection module. Computational costs associated with the Hessian matrix are typically significant, so we simplify and parallelize the computation of the Hessian matrix for hyperelasticity and barrier function separately. The entire algorithm is parallelized on the GPU using the Taichi programming language and the MeshTaichi package. Leveraging the benefits of fast convergence optimization algorithms, improved line search strategies, simplified Hessian computations, and efficient GPU parallelization, our method enables real-time simulations of objects with over 100,000 tetrahedra in complex self-collision scenarios throughout the entire simulation. The source code is available at https://github.com/Xingbaji/PNCG_IPC/.

2 RELATED WORK

The simulation of elastically deformable objects has been an important graphics research topic since the 1980s [Terzopoulos and Fleischer 1988; Terzopoulos et al. 1987, 1988]. The main objective of deformable simulation is to accurately reproduce the behavior of real-world materials in a digital environment. Modeling, theoretical analysis, and numerical simulation of elastodynamics and contact have been extensively discussed in [Bergou et al. 2008; Kane et al. 1999; Kaufman et al. 2008; Li et al. 2020; Liu et al. 2017; Teran et al. 2005; Verschoor and Jalba 2019]. Due to the highly nonlinear and non-convex nature of elasticity, as well as the typically non-smooth behavior of contact, achieving robust and accurate simulation of these effects remains a significant challenge.

Recently, Li et al. [Li et al. 2020] proposed the Incremental Potential Contact (IPC) method for robust, accurate, and differentiable elastodynamics and contact simulations. This method has the capability to consistently produce high-quality results for codimensional solids across a wide range of scenarios, ensuring interpenetration-free simulations. This method has been successfully applied in various areas, including co-dimensional simulation [Li et al. 2021], rigid body simulation [Ferguson et al. 2021; Lan et al. 2022a], embedded FEM [Zhao et al. 2022], FEM-MPM coupling [Li et al. 2022], and geometric modeling [Fang et al. 2021].

Although IPC has numerous advantages, its primary limitation lies in its computational speed. The IPC method utilizes logarithmic barrier functions to approximate inequality constraints induced by collisions and contacts. It converts the overall variational optimization into an unconstrained optimization problem and utilizes the Newton's method to solve it. A CCD-based line search is followed to ensure all the primitives are intersection-free before any displacement update is committed. The Newton solver and the CCD-based line search are both expensive and hard to implement GPU acceleration. Recent efforts have focused on accelerating IPC through the use of reduced-order models [Lan et al. 2021], projective dynamics [Lan et al. 2022b; Li et al. 2023], block coordinate descent [Lan et al. 2023], and time splitting [Wang et al. 2023; Xie et al. 2023]. However, enabling IPC method to perform comprehensive Finite Element Analysis (FEA) at real-time speeds remains a significant challenge.

The conjugate gradient (CG) method has been a classic algorithm for solving unconstrained optimization problems since the 1950s [Nocedal and Wright 1999].In the field of computer graphics, linear conjugate gradient algorithms, especially Fletcher-Reeves [Fletcher and Reeves 1964], are usually employed to solve the large linear system, due to the matrix-free property and suitability for parallel execution on GPUs. Recently, several new variants of CG method [Andrei 2013; Andrei et al. 2020; Dai and Kou 2013; Hager and Zhang 2005] are proposed and exhibit better convergence rate than Fletcher-Reeves algorithm. However, it is rarely discussed in the field of physical simulation. The conjugate gradient algorithms are usually sensitive to the step size, and the line search strategy is the crucial point in conjugate gradient algorithms. Numerous line search strategies have been proposed for various conjugate gradient algorithms, such as [Andrei 2009, 2013; Dai and Kou 2013; Hager and Zhang 2005; Liu and Liu 2018]. However, due to the limited number of experimental results on CG algorithms with over 100K variables, it remains a significant challenge to determine an appropriate step size.

3 METHOD

We first provide a brief overview of the formulation of elastodynamics using interior-point method [Mehrotra 1992]. By employing the implicit Euler method, the variational optimization of the elastic simulation is formulated as:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \frac{1}{2}(\mathbf{x}-\tilde{\mathbf{x}})^\top \mathbf{M}(\mathbf{x}-\tilde{\mathbf{x}}) + h^2\Psi(\mathbf{x}) \quad s.t.\ h_i(\mathbf{x}) \geq 0. \tag{1}$$

where $\mathbf{x}^{t+1}$ denotes the positions of all vertices within a three-dimensional tetrahedral meshed model at time step $t+1$. The term $\tilde{\mathbf{x}} = \mathbf{x}^t + h\mathbf{v}^t + h^2\mathbf{M}^{-1}\mathbf{f}_{ext}$, where h denotes time step size, $\mathbf{v}$ represents the velocity of vertices, $\mathbf{f}_{ext}$ is the external force, and $\mathbf{M}$ refers to the mass matrix. The first term $\frac{1}{2}(\mathbf{x}-\tilde{\mathbf{x}})^\top\mathbf{M}(\mathbf{x}-\tilde{\mathbf{x}})$ denotes the inertia potential. The hyperelastic energy $\Psi(\mathbf{x})$ is utilized to quantify the magnitude of deformation. The inequality constraints set C, given by $h_i(\mathbf{x}) \geq 0$, ensures that the simulation is free from both inter- and intra-model intersections.

Incremental potential contact [Li et al. 2020] is an implementation of the interior-point method that approximates constraints $h_i(\mathbf{x}) \geq 0$ with a barrier function:

$$\kappa \sum_{k\in C} b\left(d_k(\mathbf{x})\right).$$

The barrier function of IPC is defined as

$$b\left(d_k(\mathbf{x})\right) = \begin{cases} -\left(d_k-\hat{d}\right)^2 \log\left(\frac{d_k}{\hat{d}}\right), & 0 < d_k < \hat{d}, \\ 0, & d_k \geq \hat{d}, \end{cases} \tag{2}$$

where the hyperparameter $\hat{d}$ controls the threshold of collision repulsion, and κ determines the magnitude of collision repulsion.

Consequently, IPC converts the original problem into the following unconstrained optimization problem:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} E(\mathbf{x}), \tag{3}$$

$$E(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\tilde{\mathbf{x}})^\top\mathbf{M}(\mathbf{x}-\tilde{\mathbf{x}}) + h^2\Psi(\mathbf{x}) + \kappa\sum_{k\in C} b\left(d_k(\mathbf{x})\right). \tag{4}$$

The Hessian matrix of $E(\mathbf{x})$ is given by:

$$\mathbf{H} = \mathbf{M} + h^2\frac{\partial^2\Psi}{\partial\mathbf{x}^2} + \kappa\sum_{k\in C}\frac{\partial^2 b}{\partial\mathbf{x}^2}. \tag{5}$$

In this paper, we need to compute the diagonal matrix of $\mathbf{H}$ as the preconditioner and $\mathbf{p}^\top\mathbf{H}\mathbf{p}$ in the line search, where $\mathbf{p}$ represents the search direction in NCG algorithms. Note that the computational cost of these two terms will significantly impact the speed of algorithm. Therefore, we aim to optimize the computation of the Hessian matrix for Ψ and b separately.

3.1 Hyperelasticity

In this paper, we employ following invariants described in Chapter 5 of [Kim and Eberle 2022] to formulate the energy density function for a hyperelastic material.

$$I_1 = \mathrm{tr}(\mathbf{S}), \quad I_2 = \mathrm{tr}\left(\mathbf{F}^T\mathbf{F}\right), \quad I_3 = \det(\mathbf{F}).$$

where $\mathbf{F}$ denotes the deformation gradient and $\mathbf{S}$ is derived from Polar decomposition $\mathbf{F} = \mathbf{RS}$.

The elastic behavior of a deformable body can be characterized in terms of a hyperelastic energy density $\Psi(I_1, I_2, I_3)$, such as the Neo-Hookean elasticity [Bonet and Wood 2008]:

$$\Psi_{\mathrm{NH}} = \frac{\mu}{2}(I_2-3) - \mu\log(I_3) + \frac{\lambda}{2}(\log(I_3))^2.$$

The gradients and Hessians of each invariant with respect to $\mathbf{F}$ can be obtained as follows:

$$\mathbf{g}_1 = \mathrm{vec}(\mathbf{R}), \qquad \mathbf{H}_1 = \sum_{i=0}^{2} \lambda_i \mathbf{q}_i \mathbf{q}_i^\top,$$

$$\mathbf{g}_2 = 2\,\mathrm{vec}(\mathbf{F}), \qquad \mathbf{H}_2 = 2\mathbf{I}_{9\times 9},$$

$$\mathbf{g}_3 = \mathrm{vec}\left([\mathbf{f}_1 \times \mathbf{f}_2, \mathbf{f}_2 \times \mathbf{f}_0, \mathbf{f}_0 \times \mathbf{f}_1]\right), \quad \mathbf{H}_3 = \begin{bmatrix} \mathbf{0} & -\hat{\mathbf{f}}_2 & \hat{\mathbf{f}}_1 \\ \hat{\mathbf{f}}_2 & \mathbf{0} & -\hat{\mathbf{f}}_0 \\ -\hat{\mathbf{f}}_1 & \hat{\mathbf{f}}_0 & \mathbf{0} \end{bmatrix},$$

where $\mathbf{f}_0$, $\mathbf{f}_1$ and $\mathbf{f}_2$ represent the columns of $\mathbf{F}$. $\mathbf{I}$ represents the identity matrix, and $\mathbf{0}$ represents the zero matrix. Vectorization $vec(\cdot)$ denotes column-wise flattening of a matrix into a vector, the symbol $\hat{\mathbf{f}}_i$ indicates the cross-product matrix out of the vector $\mathbf{f}_i$. The detailed definitions of vector $\mathbf{q}_i$, and matrices $\mathbf{H}_1$ and $\mathbf{H}_3$ are provided in the supplemental document.

Now we can derive the Hessian matrix of hyperelasticity:

$$\begin{aligned}\frac{\partial^2 \Psi}{\partial \mathbf{x}^2} &= \mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^\top \mathrm{vec}\left(\frac{\partial^2 \Psi}{\partial \mathbf{F}^2}\right) \mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right) \\ &= \mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^\top \left(\sum_{i=1}^{3} \left(\frac{\partial^2 \Psi}{\partial I_i^2} \mathbf{g}_i \mathbf{g}_i^\top + \frac{\partial \Psi}{\partial I_i} \mathbf{H}_i\right)\right) \mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right),\end{aligned}$$

where $\mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right) \in \mathcal{R}^{9\times 12}$ and $\mathrm{vec}\left(\frac{\partial^2 \Psi}{\partial \mathbf{F}^2}\right) \in \mathcal{R}^{9\times 9}$.

Hence we can split the Hessian matrix into six terms,

$$\frac{\partial^2 \Psi}{\partial \mathbf{x}^2} = \sum_{i=1}^{3} \left(\frac{\partial^2 \Psi}{\partial I_i^2} \mathbf{h}_i + \frac{\partial \Psi}{\partial I_i} \mathbf{h}_{i+3}\right),$$

where

$$\mathbf{h}_i = \mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^\top \mathbf{g}_i \mathbf{g}_i^\top \mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right), \text{ for } i = 1, 2, 3,$$

$$\mathbf{h}_i = \mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^\top \mathbf{H}_i \,\mathrm{vec}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right), \text{ for } i = 4, 5, 6.$$

So we have

$$\overline{diag}\left(\frac{\partial^2 \Psi}{\partial \mathbf{x}^2}\right) = \sum_{i=1}^{3} \left(\frac{\partial^2 \Psi}{\partial I_i^2} \overline{diag}(\mathbf{h}_i) + \frac{\partial \Psi}{\partial I_i} \overline{diag}(\mathbf{h}_{i+3})\right),$$

$$\mathbf{p}^\top \frac{\partial^2 \Psi}{\partial \mathbf{x}^2} \mathbf{p} = \sum_{i=1}^{3} \left(\frac{\partial^2 \Psi}{\partial I_i^2} \mathbf{p}^\top \mathbf{h}_i \mathbf{p} + \frac{\partial \Psi}{\partial I_i} \mathbf{p}^\top \mathbf{h}_{i+3} \mathbf{p}\right),$$

where the notation $\overline{diag}(A)$ to represent the vector composed of the diagonal elements of matrix A, and vector $\mathbf{p} \in \mathcal{R}^{12\times 1}$ is the search direction of NCG algorithm.

In this approach, the computation of $\frac{\partial^2 \Psi}{\partial \mathbf{x}^2}$ is divided into six parts, each of which can be efficiently calculated with minimal floating-point operations (FLOPs). In the supplemental document, we provide detailed instructions on computing $\overline{diag}(\mathbf{h_i})$ and $\mathbf{p}^\top \mathbf{h}_i \mathbf{p}$. For the Neo-Hookean elasticity model, the calculation of $\overline{diag}\left(\frac{\partial^2 \Psi}{\partial \mathbf{x}^2}\right)$ and $\mathbf{p}^\top \frac{\partial^2 \Psi}{\partial \mathbf{x}^2} \mathbf{p}$ requires only 143 and 314 FLOPs, respectively. However, if we first compute $\frac{\partial^2 \Psi}{\partial \mathbf{F}^2}$ and then compute $\frac{\partial^2 \Psi}{\partial \mathbf{x}^2}$ using two matrix-matrix products, it necessitates 624 FLOPs and 3590 FLOPs, respectively. Since these two terms will be computed in parallel for every tetrahedral element, the reduced computational cost with our method would be significant. Additionally, the computation of gradient and $\overline{diag}(\mathbf{H})$ can be included in the same for-loop, further enhancing the computational speed.

Compared to the implementation in [Lan et al. 2023], which computes Hessian-vector product with accelerated complex-step finite difference (CSFD) [Luo et al. 2019], our method is more concise in form and faster in speed. Moreover, our method can handle any hyperelastic material and does not rely on simplifications of the material models such as projective dynamics [Bouaziz et al. 2014; Lan et al. 2022b] and position-based dynamics [Macklin et al. 2016].

3.2 Incremental Potential Contact

Inspired by Chapter 14 of [Kim and Eberle 2022], we make slight modifications to the definition of the distance in IPC between primitives in order to improve parallel acceleration.

The distance between a point $\mathbf{x}_0$ and a triangle $T = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is defined as follows:

$$\begin{aligned} d^{PT} =& \|\mathbf{x}_0 - \mathbf{x}_b\| = \min_{\alpha,\beta,\gamma} \|\mathbf{x}_0 - (\alpha\mathbf{x}_1 + \beta\mathbf{x}_2 + \gamma\mathbf{x}_3)\|, \\ & s.t.\ \alpha, \beta, \gamma \geq 0,\ \alpha + \beta + \gamma = 1, \end{aligned} \tag{6}$$

where $\mathbf{x}_b = \alpha\mathbf{x}_1 + \beta\mathbf{x}_2 + \gamma\mathbf{x}_3$ represents the point within the triangle that is closest to the point $\mathbf{x}_0$. Similarly, the distance between edges $\mathbf{x}_0$-$\mathbf{x}_1$ and $\mathbf{x}_2$-$\mathbf{x}_3$ can be defined as follows:

$$\begin{aligned} d^{EE} =& \|\mathbf{x}_a - \mathbf{x}_b\| = \min_{a_0,a_1,b_0,b_1} \|(a_0\mathbf{x}_0 + a_1\mathbf{x}_1) - (b_0\mathbf{x}_2 + b_1\mathbf{x}_3)\|, \\ & s.t.\ a_0, a_1, b_0, b_1 \geq 0,\ a_0 + a_1 = 1,\ b_0 + b_1 = 1, \end{aligned} \tag{7}$$

where $\mathbf{x}_a = a_0\mathbf{x}_0 + a_1\mathbf{x}_1$ and $\mathbf{x}_b = b_0\mathbf{x}_2 + b_1\mathbf{x}_3$ are the closest points between the edges. By combining Equation (6) and Equation(7), we can reformulate the distance computation as follows:

$$\begin{aligned} d =& \|\mathbf{t}\| = \min_{c_0,c_1,c_2,c_3} \|c_0\mathbf{x}_0 + c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + c_3\mathbf{x}_3\|, \\ & s.t.\ c_0 = 1,\ c_1, c_2, c_3 \leq 0,\ c_1 + c_2 + c_3 = -1 \text{ for PT primitives}, \\ & c_0, c_1 \geq 0,\ c_2, c_3 \leq 0,\ c_0 + c_1 = 1,\ c_2 + c_3 = -1 \text{ for EE primitives}. \end{aligned} \tag{8}$$

Distance d is represented as the norm of $\mathbf{t} \in \mathcal{R}^{3\times 1}$, which is the linear combination of $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. If either c_1, c_2 or c_3 is equal to 0 in PT primitives, the point-edge(PE) distance is derived. If exactly two of c_1, c_2 or c_3 are equal to 0, the point-point(PP) distance is derived. Similarly, the PP and PE distances can be derived from EE primitives.

Hence, the gradient and Hessian of t are

$$\frac{\partial \mathbf{t}}{\partial \mathbf{x}} = \begin{bmatrix} c_0\mathbf{I}_{3\times 3} & c_1\mathbf{I}_{3\times 3} & c_2\mathbf{I}_{3\times 3} & c_3\mathbf{I}_{3\times 3} \end{bmatrix}^\top \in \mathcal{R}^{12\times 3}, \frac{\partial^2 \mathbf{t}}{\partial \mathbf{x}^2} = \mathbf{0},$$

It is worth noting that the definition of distance remains the same as the original formulation presented in [Li et al. 2020], the only difference is that we introduce $\mathbf{t}$ to simplify the computation.

Therefore, the gradient and Hessian of barrier function are

$$\frac{\partial b}{\partial \mathbf{x}} = \frac{\partial b}{\partial d}\frac{\partial d}{\partial \mathbf{t}}\frac{\partial \mathbf{t}}{\partial \mathbf{x}} = \frac{\partial b}{\partial d}\frac{1}{d}\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\mathbf{t}, \tag{9}$$

$$\frac{\partial^2 b}{\partial \mathbf{x}^2} = \left(\frac{\partial^2 b}{\partial d^2}\frac{1}{d^2} - \frac{\partial b}{\partial d}\frac{1}{d^3}\right)\left(\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\mathbf{t}\right)\left(\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\mathbf{t}\right)^\top + \frac{\partial b}{\partial d}\frac{1}{d}\left(\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\right)\left(\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\right)^\top. \tag{10}$$

Because $(\frac{\partial^2 b}{\partial d^2}\frac{1}{d^2} - \frac{\partial b}{\partial d}\frac{1}{d^3})$ and $\frac{\partial b}{\partial d}\frac{1}{d}$ are scalar, we focus on the matrix terms. Therefore, we obtain

$$\overline{diag}((\frac{\partial \mathbf{t}}{\partial \mathbf{x}})(\frac{\partial \mathbf{t}}{\partial \mathbf{x}})^\top) = [c_0, c_0, c_0, c_1, c_1, c_1, c_2, c_2, c_2, c_3, c_3, c_3]^{\circ 2}, \tag{11}$$

$$\overline{diag}((\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\mathbf{t})(\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\mathbf{t})^\top) = [c_0\mathbf{t}^\top, c_1\mathbf{t}^\top, c_2\mathbf{t}^\top, c_3\mathbf{t}^\top]^{\circ 2}, \tag{12}$$

where the Hadamard power $\circ 2$ represents element-wise square of the vector. Thus, we can efficiently compute the diagonal matrix without the need to calculate the entire Hessian matrix.

Given a vector $\mathbf{p} = [\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]^\top \in \mathcal{R}^{12\times 1}$, where $\mathbf{p}_i \in \mathcal{R}^{1\times 3}$, we have

$$\mathbf{p}^\top(\frac{\partial \mathbf{t}}{\partial \mathbf{x}}t)(\frac{\partial \mathbf{t}}{\partial \mathbf{x}}t)^\top\mathbf{p} = (\mathbf{p}^\top\frac{\partial \mathbf{t}}{\partial \mathbf{x}}t)^2, \tag{13}$$

$$\mathbf{p}^\top(\frac{\partial \mathbf{t}}{\partial \mathbf{x}})(\frac{\partial t}{\partial \mathbf{x}})^\top\mathbf{p} = \|\mathbf{p}^\top\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\|_2^2, \tag{14}$$

where the computation of $\mathbf{p}^\top\frac{\partial t}{\partial \mathbf{x}} = \sum_{i=0}^{3} c_i\mathbf{p}_i$ is straightforward. This approach significantly reduces the computational cost compared to initially calculating a 12×12 matrix and subsequently performing two matrix multiplications.

With equation (10),(11) and (12), we can simplify the computation of $\overline{diag}(\frac{\partial^2 b}{\partial \mathbf{x}^2})$. Similarly, with equation (10), (13) and (14), we can simplify the computation of $\mathbf{p}^\top\frac{\partial^2 b}{\partial \mathbf{x}^2}\mathbf{p}$. Additionally, the procedure for identifying the constraints derived from PT and EE primitives closely follows that of [Li et al. 2020], with the only difference being the computation and storage of $\mathbf{t}$ and the corresponding coefficients c_0, c_1, c_2, c_3. The impact of these differences on computational time and memory usage is negligible. Furthermore, by merging the definition and corresponding computation of PT distance and EE distance, we can handle all the constraints within a single for-loop. Thus the entire process can be parallelized for acceleration.

Through the above implementation, the computation of $\overline{diag}(\mathbf{H})$ and $\mathbf{p}^\top\mathbf{H}\mathbf{p}$ only requires two for-loops, one for all the tetrahedral elements and another for all the contact constraints. Both of these loops are totally parallelizable and very fast.

4 NONLINEAR CONJUGATE GRADIENT ALGORITHM

As illustrated in Algorithm 1, the pipeline of the proposed preconditioned nonlinear conjugate gradient method is illustrated. In this section, we describe the techniques employed in our method.

4.1 Conjugate Gradient Algorithm

Through a series of experiments, we find that the Dai-Kou (DK) algorithm [Dai and Kou 2013] exhibits superior performance in terms of robustness and convergence speed. The formula of DK conjugate gradient algorithm is

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k^{DK}\mathbf{p}_k, \tag{15}$$

$$\beta_k^{DK} = \frac{\mathbf{g}_{k+1}^T\mathbf{y}_k}{\mathbf{y}_k^T\mathbf{p}_k} - \frac{\mathbf{y}_k^T\mathbf{y}_k}{\mathbf{y}_k^T\mathbf{p}_k}\frac{\mathbf{p}_k^T\mathbf{g}_{k+1}}{\mathbf{y}_k^T\mathbf{p}_k}, \tag{16}$$

where $\mathbf{p}_k$ is the conjugate gradient direction at iteration k, $\mathbf{g}_k$ is the gradient of objective function, and $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.

The formula for β_{DK} is derived by determining the conjugate gradient direction that is closest to the direction of the scaled memoryless BFGS method (SSML-BFGS), and the numerical experiments in [Dai and Kou 2013] showed that DK algorithm performs more efficiently and robustly than the SSML-BFGS method. The proof of convergence and theoretical analysis in [Dai and Kou 2013] also illustrate its efficiency.

4.2 Preconditioning

Preconditioning is a widely employed technique to accelerate the conjugate gradient algorithms by reducing the condition number of the problem. As discussed in Section 3, the diagonal elements of Hessian of IPC and hyperelastic can be easily computed and accelerated with GPU parallel technique. Therefore, we adopt Jacobi preconditioning matrix $\mathbf{P} = diag(\mathbf{H})^{-1}$, which is the inverse of the diagonal matrix of the Hessian matrix. So we can derive the preconditioned DK algorithm:

$$\mathbf{p}_{k+1} = -\mathbf{P}_{k+1}\mathbf{g}_{k+1} + \beta_k^{DK}\mathbf{p}_k \tag{17}$$

$$\beta_k^{DK} = \frac{\mathbf{g}_{k+1}^T\mathbf{P}_{k+1}\mathbf{y}_k}{\mathbf{y}_k^T\mathbf{p}_k} - \frac{\mathbf{y}_k^T\mathbf{P}_{k+1}\mathbf{y}_k}{\mathbf{y}_k^T\mathbf{p}_k}\frac{\mathbf{p}_k^T\mathbf{g}_{k+1}}{\mathbf{y}_k^T\mathbf{p}_k}, \tag{18}$$

where the computation of β^{DK} requires four vector-vector dot products.

For vertices with collision repulsion, as the distance between pairs of primitives approaches 0, the value of b exhibits a significant increase due to the logarithmic term of Equation (2). This increase leads to larger values of $diag(\mathbf{H})$, resulting in a smaller scale of search direction p. Conversely, vertices without collision repulsion exhibit a larger scale of search direction, leading to a similar convergence rate among all vertices. As a result, the application of preconditioning can effectively alleviate the "locking" issue mentioned in [Lan et al. 2023].

In Section 3, we have optimized the computational complexity of the diagonal Hessian matrix to the same level as computing the gradient. Additionally, the gradient and preconditioner can be evaluated in the same for-loop. Consequently, despite evaluating $\mathbf{P}$ at each iteration, the computational cost is still acceptable, as opposed to evaluating the Hessian matrix once every M ($M >> 1$) iterations in [Wang and Yang 2016], which may lead to artifacts or penetration in complex contact scenarios and exhibit slightly poorer convergence.

4.3 Line Search

Conjugate gradient methods are usually sensitive to the step size. A step size that is too small would decrease the convergence speed, while a step size that is too large may result in penetration or artifacts. Therefore, it is crucial to compute an appropriate step size. Nevertheless, elasticity is highly nonlinear and non-convex, and the Hessian matrix of IPC may change drastically between two iterations due to the logarithmic barrier function. As a result, neither the interpolation method nor the Barzilai-Borwein method [Barzilai and Borwein 1988] can determine an appropriate step size. Therefore, we adopt Newton's method for line search, which secures an optimal step size by leveraging a quadratic approximation of the

objective function. By utilizing the Taylor series, we obtain

$$E(\mathbf{x}+\alpha\mathbf{p}) \approx E(\mathbf{x}) + \alpha\mathbf{g}^\top\mathbf{p} + \frac{\alpha^2}{2}\mathbf{p}^\top\mathbf{H}\mathbf{p} \tag{19}$$

Hence,

$$\frac{\partial E(\mathbf{x}+\alpha\mathbf{p})}{\partial\alpha} \approx \mathbf{g}^\top\mathbf{p} + \alpha\mathbf{p}^\top\mathbf{H}\mathbf{p} \tag{20}$$

The function $E(\mathbf{x}+\alpha\mathbf{p})$ is approximately minimized by setting the gradient with respect to α to be zero, so we have

$$\overline{\alpha} = -\frac{\mathbf{g}_{k+1}^\top\mathbf{p}_{k+1}}{\mathbf{p}_{k+1}^\top\mathbf{H}_{k+1}\mathbf{p}_{k+1}}. \tag{21}$$

In Section 3, we have significantly accelerated the computation of $\mathbf{p}_{k+1}^\top\mathbf{H}_{k+1}\mathbf{p}_{k+1}$, and the computation of the vector-vector dot product $\mathbf{g}_{k+1}^\top\mathbf{p}_{k+1}$ is inexpensive. Hence the overall computational cost is affordable. Moreover, Newton's method exhibits quadratic convergence, allowing for the direct determination of an appropriate step size.

Recall that the barrier function treats the distances between primitives larger than $\hat{d}$ as 0. In complex self-collision scenarios, when the displacement of vertices exceeds $\hat{d}$ during iterations, primitives may penetrate before contact constraints are detected. Inspired by the "CFL-Inspired Culling of CCD" technique proposed in [Li et al. 2020], we impose a restriction on the step size to ensure that the maximum displacement of vertices is less than $\frac{\hat{d}}{2}$. The upper bound of step size is

$$\alpha_{\text{upper}} = \frac{\hat{d}}{2\|\mathbf{p}_{k+1}\|_\infty}, \tag{22}$$

so the final step size is given as:

$$\alpha = \min(\alpha_{\text{upper}}, \overline{\alpha}) \tag{23}$$

Let us denote the optimal step size as α^*. When $\alpha^* < \alpha_{\text{upper}}$, the original IPC method initiates with α_{upper} and performs backtracking line search to determine a proper step size. In contrast, our method directly obtains an appropriate step size $\overline{\alpha}$. When $\alpha^* > \alpha_{\text{upper}}$, IPC employs CCD to establish a higher upper bound. Since CCD computation is computationally expensive, and each iteration of the PNCG method is computationally efficient, we prefer to utilize the conservative step size α_{upper} and perform additional iterations. Moreover, as the PNCG method operates as a first-order optimization technique, its search direction is comparatively less precise than that of Newton's method, typically resulting in a smaller optimal step size α^* for each iteration. Consequently, the scenario where $\overline{\alpha} > \alpha_{\text{upper}}$ is infrequent, primarily occurring during the initial few iterations. Limiting the step size in the early stages helps prevent optimization divergence.

4.4 Termination Condition

From Equation (19), we can derive

$$\Delta E = E(\mathbf{x}_k) - E(\mathbf{x}_{k+1}) \approx -\alpha\mathbf{g}_{k+1}^\top\mathbf{p}_{k+1} - \frac{\alpha^2}{2}\mathbf{p}_{k+1}^\top\mathbf{H}_{k+1}\mathbf{p}_{k+1}.$$

When ΔE is small, the algorithm can hardly decrease the objective function, indicating the termination of iterations. Since the initial energy scale can vary significantly across different materials and situations, we employ the condition $\Delta E < \epsilon\Delta E_0$ as the termination condition. This termination condition does not require additional computation, and experiments show that it is robust for PNCG algorithm.

ALGORITHM 1: Preconditioned Nonlinear Conjugate Gradient

$\mathbf{x}_0 \leftarrow x^t$
$\tilde{x} = x^t + hv^t + h^2M^{-1}f_{ext}$
for $k = 0$ *to IterMax* **do**
  $C \leftarrow \text{ComputeConstraintSet}(x, \hat{d})$
  $\mathbf{g}_{k+1}, \mathbf{P}_{k+1} \leftarrow$ ComputeGradientAndPreconditioning$(x, \tilde{x}, C)$
  if $k = 0$ **then**
    $\beta_k \leftarrow 0$
  else
    $\beta_k \leftarrow \text{ComputeBeta}(\mathbf{g}_{k+1}, \mathbf{g}_k, \mathbf{p}_{k+1}, \mathbf{p}_k)$
  end
  $\mathbf{p}_{k+1} \leftarrow -\mathbf{P}_{k+1}\mathbf{g}_{k+1} + \beta_k\mathbf{p}_k$
  $\alpha \leftarrow \min(\frac{\hat{d}}{2\|\mathbf{p}_{k+1}\|_\infty}, -\frac{\mathbf{g}_{k+1}^\top\mathbf{p}_{k+1}}{\mathbf{p}_{k+1}^\top\mathbf{H}_{k+1}\mathbf{p}_{k+1}})$.
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha\mathbf{p}_{k+1}$
  $\Delta E \leftarrow -\alpha\mathbf{g}_{k+1}^\top\mathbf{p}_{k+1} - \frac{\alpha^2}{2}\mathbf{p}_{k+1}^\top\mathbf{H}_{k+1}\mathbf{p}_{k+1}$
  if $i = 0$ **then**
    $\Delta E_0 \leftarrow \Delta E$
  end
  if $\Delta E < \epsilon\Delta E_0$ **then**
    break
  end
end

5 IMPLEMENTATION DETAILS

5.1 Larger $\hat{d}$ Strategy

In [Lan et al. 2023], the authors adopt a warm start strategy by using larger $\hat{d}$, which leads to a significant reduction in the total number of iterations. Our experiments also demonstrate that a larger $\hat{d}$ enhances the convergence of the PNCG algorithm. It enables the algorithm to detect collisions earlier and utilize a smaller step size, effectively preventing penetration. Moreover, a larger $\hat{d}$ relaxes the restriction on the maximum step size in Equation (23). Instead of using different customized $\hat{d}$ values for each collision pair, a constant $\hat{d}$ would be sufficient for our method. However, a larger $\hat{d}$ can cause additional self-collision repulsion at the rest pose. Therefore we propose a preprocessing stage. By setting the threshold as $1.5\hat{d}$ in the rest pose, we can first filter out the self-collision constraints, and store them with a spatial hashing table. By referring to the hashing table, we can exclude these self-collision constraints during the simulation.

Although a larger value of $\hat{d}$ would increase the number of contact constraints, the resulting increase in computational cost would be justifiable when considering the improvement in convergence speed, owing to the simplified and parallelized relevant calculations. Experimental results indicate that setting $\hat{d}$ within the range of 30% to 70% of the average boundary edge length is appropriate. Since collision detection operations are not included in the line search,

Table 1: Performance results. The "8 E Falling" scenario utilizes a time step size of $\Delta t = 0.01$, all other examples utilize $\Delta t = 0.04$. "# tets" refers to the number of tetrahedra. The abbreviations ARAP, FCR, NH, and SNH represent the As-Rigid-As-Possible, Fixed Corotated, Neo-Hookean, Stable Neo-Hookean[Smith et al. 2018] elasticity models, respectively.

Scene	Material	# nodes, # tets # faces	# contact avg(max)	IterMax	Tolerance ϵ	avg Iters	FPS avg(min)
Drag armadillo (Fig. 3c)	SNH	13K, 55K, 22K	0K (0K)	150	5×10^{-5}	108.1	40.1 (25.0)
8 "E" falling (Fig. 8)	NH	8K, 27K, 13K	2K (8K)	50	3×10^{-4}	32.8	46.6 (24.3)
48 "E" falling (Fig. 9)	ARAP	50K, 164K, 78K	63K (99K)	25	1×10^{-3}	25.0	32.2 (27.9)
Four long noodles (Fig. 4)	ARAP	39K, 101K, 73K	13K (29K)	25	1×10^{-3}	24.9	27.6 (25.8)
Squeeze four armadillo (Fig. 6)	ARAP	53K, 168K, 91K	71K (322K)	25	1×10^{-3}	24.1	28.9 (25.6)
Twist mat (Fig. 7)	FCR	45K, 133K, 90K	87K (222K)	150	1×10^{-3}	138	7.4 (5.6)
Twist four rods (Fig. 5)	FCR	53K, 202K, 80K	39K (92K)	150	1×10^{-3}	107.5	10.2 (6.9)

it is essential to choose appropriate hyperparameters for the barrier function. Increasing the value of $\hat{d}$ can enhance algorithm's robustness against variations in the value of κ.

5.2 Splitting Computation

In scenarios involving multiple objects, such as Figure 9, certain objects may undergo collisions and compression, while others remain collision-free. It is not appropriate for all objects to be assigned the same step size in such cases. Hence, it is necessary to utilize separate values of step size α and search direction p for each object. This strategy can be easily achieved through parallel computation, minimizing the impact on computational cost. Similarly, in the case of the long noodles model discussed in Figure 4, collisions may occur between specific points of the same object, while leaving other parts unaffected. To address this, the objects can be divided into collision and non-collision parts at the start of iterations. Then, separate values of α and p can be assigned to each part, hence achieving higher convergence speed. More implementation details are elaborated in the supplementary material.

6 EXPERIMENTS

Our algorithm is implemented on a desktop workstation with an NVIDIA RTX 4090 GPU and an Intel Core i9-13900X CPU, using the Taichi language for GPU acceleration with single-precision float. Tetrahedral mesh computation is accelerated with MeshTaichi [Yu et al. 2022], and a grid-based spatial hashing technique [Müller et al. 2023] is employed for broad-phase collision culling.

6.1 Ablation study

In Figure 2a, we present the convergence analysis of various methods in Example 9. Jacobi preconditioning is utilized for all these methods. The ground truth is obtained by Newton-PCG with sufficient iterations. It is evident that the PNCG algorithms: CD [Fletcher 2000], HZ [Hager and Zhang 2005], FR [Fletcher and Reeves 1964], PRP [Polyak 1969], and DK [Dai and Kou 2013], outperform the Chebyshev-accelerated method [Wang and Yang 2016], with the DK algorithm exhibiting the highest performance among them. The time consumption of these PNCG algorithms is close, requiring only 1.2ms per iteration. Meanwhile, the Chebyshev-accelerated method takes slightly longer, at 1.4ms per iteration. Furthermore, the DK method demonstrates superior performance compared to the SSML-BFGS method with Jacobi preconditioning.

Additionally, although Newton's method exhibits the best convergence, it requires more than 400ms for each iteration, significantly larger than the PNCG method. Given that the height of "E" is 1.1, simulated results are considered sufficiently realistic when the error is less than 0.01. The line search strategy for Newton's method adheres to [Li et al. 2020], while the other methods employ our line search strategy for consistency. The detailed formulas for these methods are provided in the supplemental document.

Figure 2b compares the convergence rates of our method with and without contact scenarios. The results demonstrate that our method exhibits rapid convergence without contact. In scenarios involving contact, the convergence rate becomes slower. Because of the splitting computation strategy, the convergence rates for simulations involving two, four, and eight "E" are nearly identical.

Figure 3c tests our method's ability to handle extreme deformations by dragging the armadillo with large external forces in a collision-free scenario. Figure 3b compares corresponding convergence rates for different deformation magnitudes. Despite substantial external forces and extreme deformations, our method exhibits satisfactory convergence, and requirs only 0.25 ms per iteration.

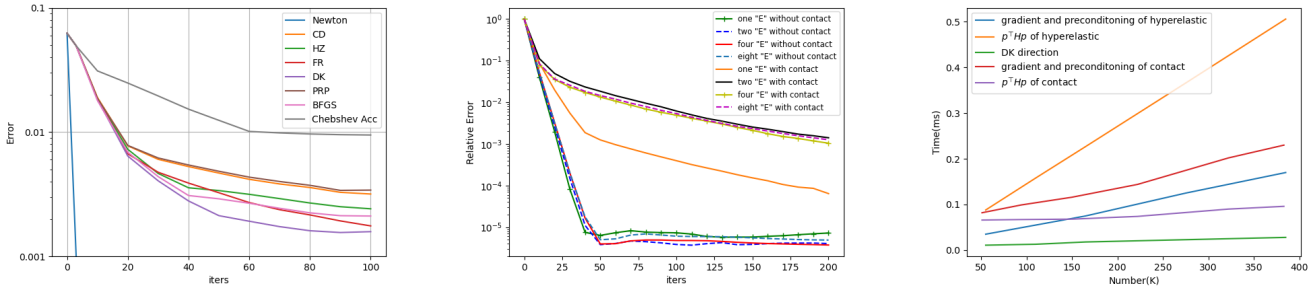
Figure 3a demonstrates the free fall of one "E" object under gravity $9.8m/s^2$ using Neo-Hookean hyperelasticity and $\Delta t = 0.01$, investigating the influence of stiffness on convergence. As Young's modulus increases from 1e4 to 1e6, the convergence rate decreases, indicating our method's unsuitability for simulating stiff objects.

6.2 Computational Cost

Figure 2c shows the computational costs of hyperelasticity and barrier contact, varying with the number of tetrahedral elements and contact constraints. The simplified computations outlined in Section 3 significantly reduce these costs. Notably, the DK search direction cost is extremely low at approximately 0.01 ms. For a 220K tetrahedra model on an NVIDIA GeForce RTX 2060 GPU, the DK search direction cost is still low at 0.021 ms.

6.3 Examples

As illustrated in Table 1, we evaluate the performance of our method in complex self-collision scenarios. To achieve real-time performance, we employ early stopping of the simulation before it fully converges in the "Squeeze four armadillo," "Four long noodles," and "48 E falling" scenarios. In Figure 6, we gradually compress four armadillo models from all six directions and subsequently release



(a) Convergence of different methods. DK algorithm demonstrates superior performance compared to other PNCG algorithms (CD,HZ,FR,PRP), and significantly outperforms the Chebyshev accelerated preconditioned descent method. Error is evaluated using the infinity norm with respect to the ground truth.

(b) Convergence with and without contact. In scenarios devoid of contact, an increase in the number of objects "E" has a minimal impact on convergence. In the presence of contact, the convergence speed decreases; yet, the adoption of a splitting computation strategy yields a convergence speed that is nearly identical for two, four, and eight "E".

(c) Computational cost of hyperelastic and contact with different number of elements and constraints, measured in milliseconds. All the operation of our method shows significantly low computational costs even with a large number of elements and constraints.

Figure 2: Ablation study.

them, the models exhibit an "explosion" effect upon release, demonstrating correct elastic behavior and the absence of penetrations. In Figure 4, four long strands of noodles simultaneously fall into the bowl with gravitational acceleration 9.8 m/s^2. In Figure 8, eight instances of the letter "E" undergo free fall toward the ground under a gravitational acceleration of 9.8 m/s^2, with a time step of 0.01 s. Figure 9 depicts 48 instances of "E" in free fall, subject to a gravitational acceleration of 0.5 m/s^2 and a time step of 0.04 s. As depicted in the corresponding figures, despite the early termination, our method still achieves visually plausible results. Furthermore, we perform discrete collision detection for each frame of all the examples to ensure the absence of penetrations. Futhermore, some unittests [Erleben 2018] are provides in the supplemental video.

In Figures 5 and 7, each side of the rods and mat is twisted with a rotating speed of $72^\circ/s$, and notable buckling appears after several rounds of twisting. The mat can still be successfully expanded and recovered by twisting in the opposite direction, demonstrating the absence of penetrations. In the supplementary material, we provide a result of twisting the mat for eight rounds. As reported in [Li et al. 2020], the vanilla CPU IPC requires 33 seconds per time step for the twisting mat demonstrations, whereas our method achieves significantly faster computational speed in comparison. The state-of-the-art method [Lan et al. 2023] achieves about 1 FPS speed in similar scenarios. The projective dynamics method [Lan et al. 2022b] displays similar speed, but it does not support general hyperelasticity as our method does.

7 CONCLUSION AND LIMITATION

In this paper, We investigate the application of the nonlinear conjugate gradient method for elastic deformation with barrier contact. Leveraging a fast convergence optimization algorithm, improved line search strategies, simplified Hessian computations, and efficient GPU parallelization, our proposed PNCG method enables real-time simulation of objects comprising over 100K tetrahedra in complex self-collision scenarios. It has the potential for integration with other collision repulsion methods and friction contact.

Since we employ fixed constant values of $\hat{d}$ and κ for the IPC potential, and the collision detection module is removed, choosing appropriate values is crucial. While all demonstrations in this paper achieve penetration-free results, in extreme scenarios such as twisting the mat for more rounds, where the mesh undergoes significant deformation, resulting in the dominance of the elastic term, frequently adjusting κ is necessary to ensure the updating direction points away from intersections. However, it is challenging to find suitable values that consistently maintain penetration-free results. Similarly, when the object's speed is high and the inertial potential dominates over the elastic and IPC potentials, the step size obtained by Equation (21) may cause penetrations. This is the reason why we employ a gravity value of 0.5 in the stacked "48 E falling" example. A robust adaptive barrier stiffness strategy is currently under investigation.

REFERENCES

Neculai Andrei. 2009. Acceleration of conjugate gradient algorithms for unconstrained optimization. *Appl. Math. Comput.* 213, 2 (2009), 361–369.

Neculai Andrei. 2013. Another conjugate gradient algorithm with guaranteed descent and conjugacy conditions for large-scale unconstrained optimization. *Journal of Optimization Theory and Applications* 159 (2013), 159–182.

Neculai Andrei et al. 2020. *Nonlinear conjugate gradient methods for unconstrained optimization.* Springer.

Jonathan Barzilai and Jonathan M Borwein. 1988. Two-point step size gradient methods. *IMA journal of numerical analysis* 8, 1 (1988), 141–148.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*. 1–12.

Javier Bonet and Richard D. Wood. 2008. *Nonlinear Continuum Mechanics for Finite Element Analysis* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511755446

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics. *ACM Transactions on Graphics* 33 (07 2014), 1–11. https://doi.org/10.1145/2601097.2601116

Min Gyu Choi and Hyeong-Seok Ko. 2005. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics* 11, 1 (2005), 91–101.

Yu-Hong Dai and Cai-Xia Kou. 2013. A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search. *SIAM Journal on Optimization* 23, 1 (2013), 296–320.

Kenny Erleben. 2018. Methodology for Assessing Mesh-Based Contact Point Methods. *ACM Trans. Graph.* 37, 3, Article 39 (jul 2018), 30 pages. https://doi.org/10.1145/3096239

Yu Fang, Minchen Li, Chenfanfu Jiang, and Danny M Kaufman. 2021. Guaranteed globally injective 3D deformation processing. *ACM Transactions on Graphics* 40, 4 (2021).

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M Kaufman, and Daniele Panozzo. 2021. Intersection-free rigid body dynamics. *ACM Transactions on Graphics* 40, 4 (2021).

Roger Fletcher. 2000. *Practical methods of optimization*. John Wiley & Sons.

Reeves Fletcher and Colin M Reeves. 1964. Function minimization by conjugate gradients. *The computer journal* 7, 2 (1964), 149–154.

William W Hager and Hongchao Zhang. 2005. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on optimization* 16, 1 (2005), 170–192.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

Couro Kane, Eduardo A Repetto, Michael Ortiz, and Jerrold E Marsden. 1999. Finite element analysis of nonsmooth contact. *Computer methods in applied mechanics and engineering* 180, 1-2 (1999), 1–26.

Danny M Kaufman, Shinjiro Sueda, Doug L James, and Dinesh K Pai. 2008. Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers*. 1–11.

Theodore Kim and David Eberle. 2022. Dynamic deformables: implementation and production practicalities (now with code!). In *ACM SIGGRAPH 2022 Courses*. 1–259.

Lei Lan, Danny M. Kaufman, Minchen Li, Chenfanfu Jiang, and Yin Yang. 2022a. Affine body dynamics: fast, stable and intersection-free simulation of stiff materials. 41, 4, Article 67 (jul 2022), 14 pages. https://doi.org/10.1145/3528223.3530064

Lei Lan, Minchen Li, Chenfanfu Jiang, Huamin Wang, and Yin Yang. 2023. Second-order Stencil Descent for Interior-point Hyperelasticity. *ACM Trans. Graph.* 42, 4, Article 108 (jul 2023), 16 pages. https://doi.org/10.1145/3592104

Lei Lan, Guanqun Ma, Yin Yang, Changxi Zheng, Minchen Li, and Chenfanfu Jiang. 2022b. Penetration-free projective dynamics on the GPU. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

Lei Lan, Yin Yang, Danny Kaufman, Junfeng Yao, Minchen Li, and Chenfanfu Jiang. 2021. Medial IPC: accelerated incremental potential contact with medial elastics. *ACM Transactions on Graphics* 40, 4 (2021).

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy R 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 (2020), 49.

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Trans. Graph. (SIGGRAPH)* 40, 4, Article 170 (2021).

Xuan Li, Yu Fang, Lei Lan, Huamin Wang, Yin Yang, Minchen Li, and Chenfanfu Jiang. 2023. Subspace-Preconditioned GPU Projective Dynamics with Contact for Cloth Simulation. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

Xuan Li, Yu Fang, Minchen Li, and Chenfanfu Jiang. 2022. BFEMP: Interpenetration-free MPM–FEM coupling with barrier contact. *Computer Methods in Applied Mechanics and Engineering* 390 (2022), 114350.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-newton methods for real-time simulation of hyperelastic materials. *Acm Transactions on Graphics (TOG)* 36, 3 (2017), 1–16.

Zexian Liu and Hongwei Liu. 2018. Several efficient gradient methods with approximate optimal stepsizes for large scale unconstrained optimization. *J. Comput. Appl. Math.* 328 (2018), 400–413.

Ran Luo, Weiwei Xu, Tianjia Shao, Hongyi Xu, and Yin Yang. 2019. Accelerated complex-step finite difference for expedient deformable simulation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

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*. 49–54.

Sanjay Mehrotra. 1992. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization* 2, 4 (1992), 575–601.

Ullrich Meier, Oscar López, Carlos Monserrat, Mari C Juan, and M Alcaniz. 2005. Real-time deformable models for surgery simulation: a survey. *Computer methods and programs in biomedicine* 77, 3 (2005), 183–197.

Matthias Müller, matthimf, Johnathon Selstad, Thomas Oberbichler, and Elliot Waite. 2023. *matthias-research/pages*. https://github.com/matthias-research/pages

Jorge Nocedal and Stephen J Wright. 1999. *Numerical optimization*. Springer.

Elijah Polak and Gerard Ribiere. 1969. Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle. Série rouge* 3, 16 (1969), 35–43.

Boris Teodorovich Polyak. 1969. The conjugate gradient method in extremal problems. *U. S. S. R. Comput. Math. and Math. Phys.* 9, 4 (1969), 94–112.

V Popescu, Grigore Burdea, and Mourad Bouzit. 1999. Virtual reality simulation modeling for a haptic glove. In *Proceedings Computer Animation 1999*. IEEE, 195–200.

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable neo-hookean flesh simulation. *ACM Transactions on Graphics (TOG)* 37, 2 (2018), 1–15.

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 181–190.

Demetri Terzopoulos and Kurt Fleischer. 1988. Deformable models. *The visual computer* 4, 6 (1988), 306–331.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 205–214.

Demetri Terzopoulos, Andrew Witkin, and Michael Kass. 1988. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial intelligence* 36, 1 (1988), 91–123.

Takuya Umedachi, Vishesh Vikas, and Barry A Trimmer. 2013. Highly deformable 3-D printed soft robot generating inching and crawling locomotions with variable friction legs. In *2013 IEEE/RSJ international conference on Intelligent Robots and Systems*. IEEE, 4590–4595.

Mickeal Verschoor and Andrei C Jalba. 2019. Efficient and accurate collision response for elastically deformable models. *ACM Transactions on Graphics (TOG)* 38, 2 (2019), 1–20.

Huamin Wang. 2018. Rule-free sewing pattern adjustment with precision and efficiency. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

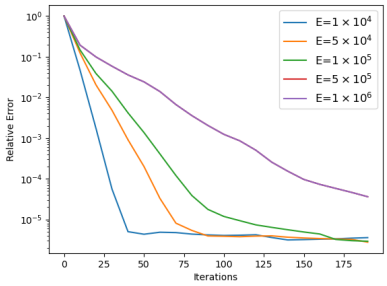
Huamin Wang and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–10.

Tianyu Wang, Jiong Chen, Dongping Li, Xiaowei Liu, Huamin Wang, and Kun Zhou. 2023. Fast GPU-Based Two-Way Continuous Collision Handling. *ACM Transactions on Graphics* 42, 5 (2023), 1–15.

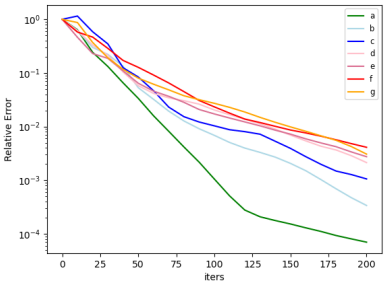
Tianyi Xie, Minchen Li, Yin Yang, and Chenfanfu Jiang. 2023. A Contact Proxy Splitting Method for Lagrangian Solid-Fluid Coupling. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–14.

Chang Yu, Yi Xu, Ye Kuang, Yuanming Hu, and Tiantian Liu. 2022. MeshTaichi: A Compiler for Efficient Mesh-Based Operations. *ACM Trans. Graph.* 41, 6, Article 252 (nov 2022), 17 pages. https://doi.org/10.1145/3550454.3555430

Yidong Zhao, Jinhyun Choo, Yupeng Jiang, Minchen Li, Chenfanfu Jiang, and Kenichi Soga. 2022. A barrier method for frictional contact on embedded interfaces. *Computer Methods in Applied Mechanics and Engineering* 393 (2022), 114820.



(a) The Impact of Young's Modulus on Convergence.



(b) The convergence rates for different magnitudes of deformation.

(c) Drag armadillo. The markings and colors of the armadillo correspond one-to-one with Figure (b).

Figure 3: Additional ablation study.

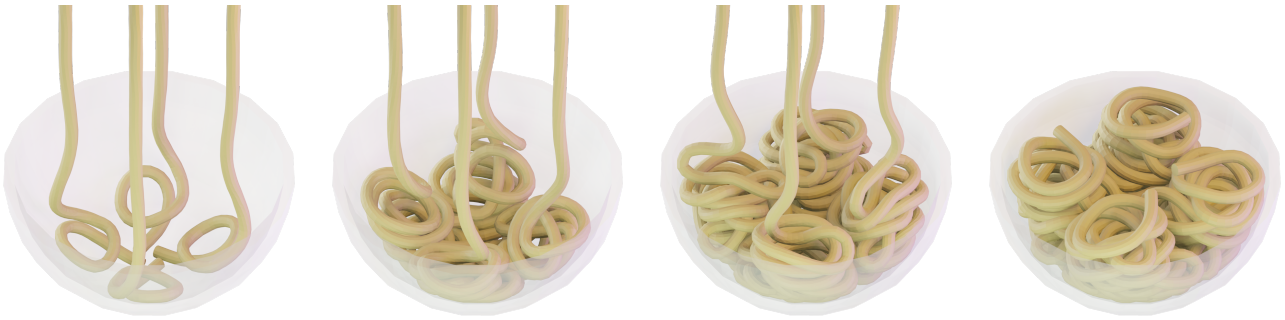


Figure 4: Long noodles in a bowl.

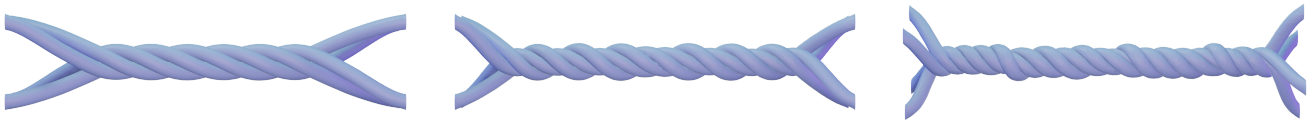


Figure 5: Twist rods for 4 rounds.

Figure 6: Squeeze and release of four armadillo models. We simultaneously compress the four small models from all six sides and then release them.

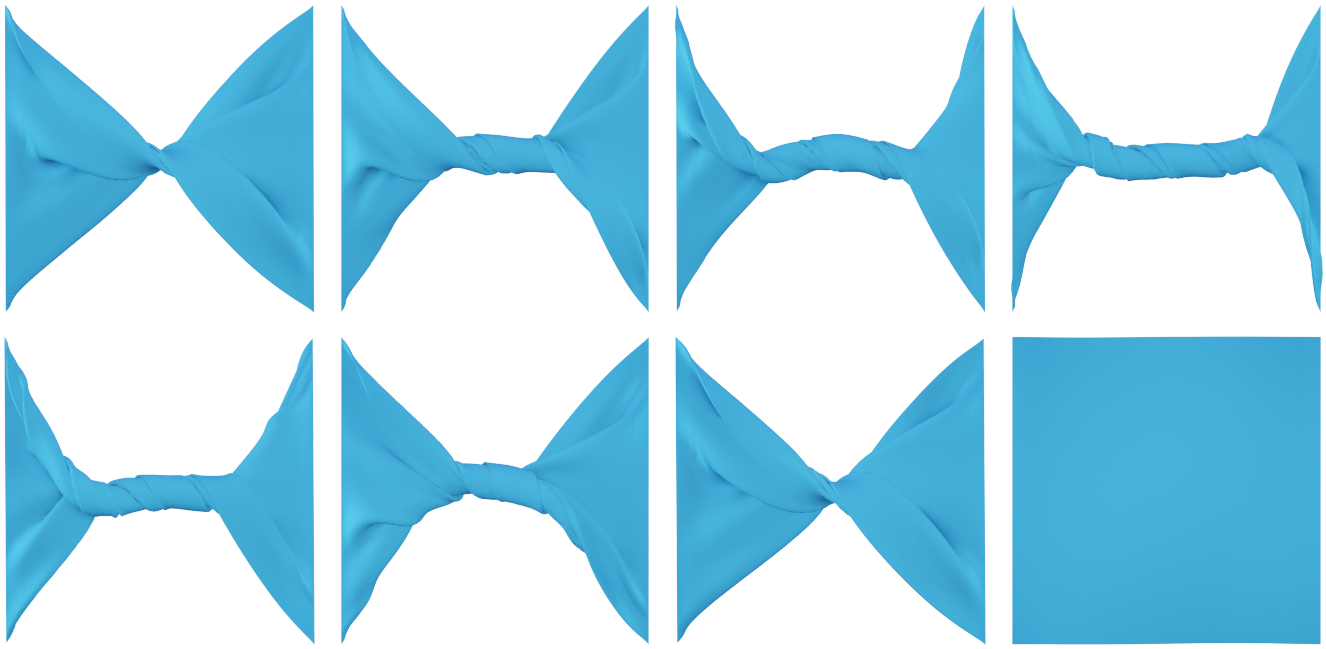


Figure 7: Twist mat. In the upper row, the mat is twisted for two rounds, while in the lower row, it is twisted back in the opposite direction.

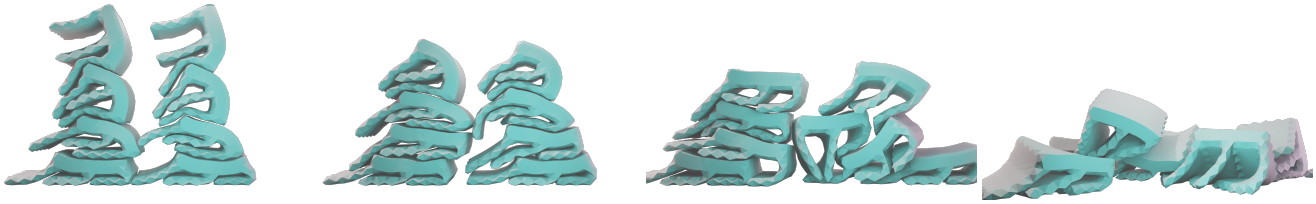


Figure 8: Free fall of 8 "E" models.

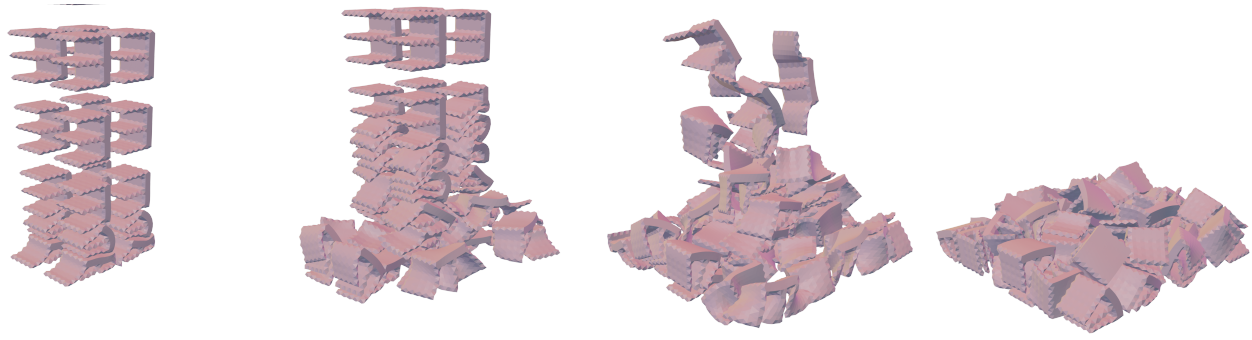


Figure 9: Free fall of stacked 48 "E" models.