Efficient GPU Cloth Simulation with Non-distance Barriers and Subspace Reuse

LEI LAN, University of Utah, USA
ZIXUAN LU, University of Utah, USA
JINGYI LONG, University of Utah, USA
CHUN YUAN, University of Utah, USA
XUAN LI, UCLA, USA
XIAOWEI HE, Institute of Software, Chinese Academy of Sciences, China
HUAMIN WANG, Style3D Research, China
CHENFANFU JIANG, UCLA, USA
YIN YANG, University of Utah, USA

Fig. 1. **Fashion show.** We present a new GPU-based cloth simulation framework with projective dynamics. Our method is able to simulate high-resolution cloth meshes at an interactive rate. With a non-distance-based barrier formulation, we can replace a large portion of traditional CCDs with the partial CCD procedure, which is much less expensive. The subspace reuse strategy relaxes the low-frequency errors effectively at the cost of single-digit milliseconds. Our method also features a residual forwarding trick to alleviate the damping issues generated by early termination and small-step line search filtering. In the teaser, we show an animated scene of the virtual fashion show. The model, dressed in a soft and light midi skirt, walks to the front and then turns around. These series of movements cause complex fabric dynamics, vividly showcasing the design concept of the garment. The garment is of high resolution and has 340K vertices. The corresponding simulation involves over one million unknowns, and detailed local wrinkles can be well perceived. With a time step of $\Delta t = 1/200$ sec, the simulation runs at 4.8 FPS. Please refer to the supplementary video for the corresponding animations.

This paper pushes the performance of cloth simulation, making the simulation interactive even for high-resolution garment models while keeping every triangle untangled. The penetration-free guarantee is inspired by the interior point method, which converts the inequality constraints to barrier potentials. We propose a major overhaul of this modality within the projective dynamics framework by leveraging an adaptive weighting mechanism inspired by barrier formulation. This approach does not depend on the distance between mesh primitives, but on the virtual life span of a collision event and thus keeps all the vertices within feasible region. Such a non-distance barrier model allows a new way to integrate collision resolution into the simulation pipeline. Another contributor to the performance boost comes from the subspace reuse strategy. This is based on the observation that low-frequency strain propagation is near orthogonal to the deformation induced by collisions or self-collisions, often of high frequency. Subspace reuse then takes care of low-frequency residuals, while high-frequency residuals can also be effectively smoothed by GPU-based iterative solvers. We show that our method outperforms existing fast cloth simulators by at least one order while producing high-quality animations of high-resolution models.

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

Additional Key Words and Phrases: GPU simulation, cloth animation, collision detection, parallel computation

Authors' addresses: Lei Lan, University of Utah, USA, lanlei.virhum@gmail.com; Zixuan Lu, University of Utah, USA, birdpeople1984@gmail.com; Jingyi Long, University of Utah, USA, u6046121@umail.utah.edu; Chun Yuan, University of Utah, USA, yuanchunisme@gmail.com; Xuan Li, UCLA, USA, xuan.shayne.li@gmail.com; Xiaowei He, Institute of Software, Chinese Academy of Sciences, China, xiaowei@iscas.ac.cn; Huamin Wang, Style3D Research, China, wanghmin@gmail.com; Chenfanfu Jiang, UCLA, USA, chenfanfu.jiang@gmail.com; Yin Yang, University of Utah, USA, yangzzzy@gmail.com.

1 INTRODUCTION

Cloth animation brings the simulated world to life in a vivid way, endows virtual characters with an infinite array of new appearances, and allows artists to lay their talents and inspirations on the triangular mesh. The primary challenge for today's cloth simulation arises from the irreconcilability between the desired visual quality and the limited computing resources – it is often the case that the actual time budget allocated for the simulator is strictly capped e.g., in interactive design or games. Cloths and fabrics demonstrate intricate dynamics under collisions and contacts, yielding captivating fine deformations of wrinkles and folds. To faithfully capture those effects, a high-resolution mesh is preferred, and the increased number of degrees of freedom (DOFs) further stresses the simulation performance. This paper presents a GPU-based cloth simulation framework which is one or even two orders faster than the state-of-the-art GPU simulation algorithms for high-resolution cloth models.

Efficient processing of collision and self-collision of cloth is the pivotal concern for a high-performance cloth simulator. Our method is inspired by the recent success of incremental potential contact or IPC [Li et al. 2020, 2021a]. When combined with CCD line search filtering and projection-Newton solver, IPC offers a non-penetration guarantee throughout the simulation process. It introduces a nonlinear repulsion between a pair of colliding or near-colliding primitives, which becomes infinitely strong if they get closer to each other. While this method has been proven robust, it requires repetitive CCDs (continuous collision detection) to calculate the distance between primitives in proximity – this is costly for detailed meshes since all the triangles may be in contact. As a result, a dominant computation in many cloth simulators becomes the CCD processing. To address this issue, we design a novel barrier mechanism, which adaptively re-weights the collision constraints when the collision event exists over a continuous period of iterations, and does not depend on the actual distance between primitives. Because of this, most CCD procedures invoked in the simulation can be substantially simplified.

On the solve side, we observe that existing GPU-based iterative solvers are less effective for smoothing low-frequency errors. In theory, this drawback could be remedied with model reduction or multigrid techniques, which project the system into a prescribed kinematic subspace. The difficulty comes from the nonlinearity of the system matrix, which varies under changing cloth poses, and constructing a subspace for each pose is not an option. To this end, we design a subspace reuse scheme that leverages a low-frequency rest-shape subspace for different deformed poses. This strategy well synergizes with the projective dynamics (PD) framework [Bouaziz et al. 2014] because the geometric nonlinearity of cloth dynamics is taken care of in the local step, and the low-frequency subspace tends to be less sensitive to high-frequency deformations. We carefully exploit the structure of the global matrix in PD so that the expensive subspace projection of the full global matrix can be pre-computed.

Our experiments show that subspace reuse cut the follow-up aggregated Jacobi iteration by 70% on average.

Our algorithm is also equipped with a residual forwarding scheme. As time-critical applications allocate a limited time budget for the simulator, early termination may lead to visual artifacts like over-stiffening and damping due to the CCD-based line search filtering. Residual forward estimates a ghost external force as the residual force inherited from the previous step to relieve this issue.

In a nutshell, this paper proposes a simulator integrating several novel features to address the core challenges in cloth simulation. By deeply optimizing the pipeline, our method further pushes the quality and efficiency of the simulation. It is guaranteed that the resulting cloth poses are free of inter-penetration. Meanwhile, the combination of reused subspace and aggregated Jacobi iterations makes the solver effective for both low- and high-frequency deformations. More importantly, most calculations are friendly for GPU or any parallel computation platforms. The experiments show that our method delivers high-quality animation results while being one order faster than existing methods.

2 RELATED WORK

Cloth simulation has been extensively studied in past decades. A vast volume of excellent contributions exists. Due to the page limit, this section only briefly surveys a few representative prior works.

Cloth simulation. A common practice for cloth simulation is to discretize its geometry with a mass-spring network [Choi and Ko 2002; Liu et al. 2013] or a triangle mesh [Baraff and Witkin 1998; Terzopoulos et al. 1987; Volino et al. 2009]. Early techniques use explicit integration with small time steps [Provot et al. 1995]. The stability is improved by switching to the implicit integration [Baraff and Witkin 1998], at the cost of assembling and solving the resulting linearized systems. Cloth fabric is less extensible, showing strong resistance to stretching. The discrepancy between its stretching and bending behavior induces extra numerical difficulty. Strain limiting is a simple and effective approach to mitigate this challenge [Provot et al. 1995; Thomaszewski et al. 2009; Wang et al. 2010]. On the other hand, being unstretchable also inspires a simplified quadratic bending model [Bergou et al. 2006]. Cloth is more than an isotropic hyperelastic continuum. Kim [2020] reveals the underlying connection between the Baraff-Witkin model [Baraff and Witkin 1998] and anisotropic finite element method (FEM). One can finetune the strain-stress relation to obtain an accurate material model i.e., see [Volino et al. 2009]. Data-driven methods have also been used for this purpose [Feng et al. 2022; Sperl et al. 2022; Wang et al. 2011].

Collision processing. Collision handling has always been an integral part of cloth simulation. Discrete collision detection (DCD) is an efficient method to identify the list of colliding primitives. As the name suggests, DCD detects the inter-penetration at a specific time instance, and it fails to capture all the collision events for models with thin geometries (such as garments) or fast-moving objects. On the other hand, CCD offers a more robust way to detect inter-penetration. Assuming the trajectory is linear within a time interval, CCD calculates the first time of impact (TOI) between two primitives. For triangularized surfaces, CCD is often modeled as the

root finding of cubic equations. It is error-prone [Wang et al. 2021], and robust numerical procedures are preferred [Wang 2014; Yuksel 2022]. The brute-force collision detection among all the primitive pairs is prohibitive in general. A commonly adopted method is to use some bounding volume hierarchy (BVH) [Langetepe and Zachmann 2006] to avoid excessive triangle-triangle intersection tests, a.k.a collision culling. Various BVH types have been explored such as AABB [Bergen 1997], OBB [Gottschalk et al. 1996], sphere [Hubbard 1995; James and Pai 2004], Boxtree [Zachmann 2002], spherical shell [Krishnan et al. 1998] and so on.

To resolve the intersected triangles, the penalty method has been a popular choice for its simplicity [Guan et al. 2012]. Buffet et al. [2019] extend the implicit field of volumetric objects to open surfaces to resolve the inter-penetration of multiple-layer cloth. Baraff et al. [2003] and Wicke et al. [2006] employ an untangling method that applies repulsion forces to minimize the colliding region for cloth-cloth collisions. Volino and Magnenat-Thalmann [2006] and Ye and Zhao [2012] minimize the length of the collision contour of the colliding region. Instead of solving the collision and cloth dynamics in a two-way coupled manner, Provot [1997] and Bridson et al. [2002] suggest a post-simulation step grouping penetrating vertices into impact zones. Huh et al. [2001] decompose the impact zone into smaller colliding clusters based on their positions on the original mesh. Harmon et al. [2008] relax impact zones with an inelastic projection, allowing relative tangential movement.

Recently, an interior-point-based algorithm called incremental potential contact (IPC) [Li et al. 2020] has been proposed, which guarantees the simulation to be free of intersections. It has then been generalized to the simulation of cloth/thin shells [Li et al. 2021a], rigid/stiff bodies [Ferguson et al. 2021; Lan et al. 2022a], and curved meshes [Ferguson et al. 2023]. IPC is highly time-consuming as a CCD is needed at each nonlinear iteration to ensure the simulation results stay within the feasible region. Lan et al. [2022b] approximate the logarithm barrier with an increasingly stronger quadratic function so that the simulation fits the PD framework [Bouaziz et al. 2014]. Lan et al. [2023] decompose the global collision configuration into local stencils. To reduce the cost of CCD, Wu et al. [2020] use point-point distance constraints between triangle pairs to avoid intersections. Another relevant prior art is from Ly et al. [2020]. This method is also based on PD, and it incorporates Signorini-Coulomb law [Brogliato and Brogliato 1999; Daviet et al. 2011] using constraint projections in a semi-implicit way. With a matrix splitting scheme, it handles frictional contact robustly and efficiently. Our method is orthogonal to [Ly et al. 2020] in the sense that we aim to improve the performance for contact resolution using a non-distance barrier-like formulation to adaptively adjust the collision weight instead of resorting to complementarity programming [Moreau 1988]. Our subspace reuse technique may also be used in [Ly et al. 2020].

GPU-based simulation. In addition to collision, another computational bottleneck is the (nonlinear) system solve due to the use of implicit integration. A widely used strategy is to convert the force equilibrium (i.e., strong form) to the variational form (i.e., weak form) [Gast et al. 2015; Kharevych et al. 2006; Li et al. 2019]. Doing so offers new perspectives to the simulation in the light of optimization such as constraint-based methods. One can locally and inexactly

solve those constraints via the the constraint projection [Goldenthal et al. 2007], which highlights the potential for parallelization. For instance, position-based dynamics [Macklin et al. 2016; Müller et al. 2007] uses per-vertex constraint projection making the simulation matrix-free. Projective dynamics (PD) [Bouaziz et al. 2014] presents a global and local alternation scheme to solve the nonlinear dynamic system. PD quickly became a popular simulation modality because its local projections are trivially parallelizable. Instead of solving its global system exactly e.g., using Cholesky factorization, iterative linear solvers can be used, such as Jacobi [Lan et al. 2022b; Wang 2015], Gauss-Seidel [Fratarcangeli et al. 2016] and preconditional conjugate gradient (PCG) [Tang et al. 2013], which bring significant speedups on the GPU. For more general and nonlinear models, sophisticated GPU algorithms are needed to decouple unknown DOFs to obtain the global solution [Lan et al. 2023; Tang et al. 2018; Wang and Yang 2016; Wang et al. 2023].

Model reduction & multigrid method. Model reduction is an acceleration technique that reduces the simulation cost. It uses a set of reduced coordinates to pre-parameterize the simulation in the subspace and lowers the total number of unknown DOFs. Linear modal analysis offers the optimal subspace approximate around the rest shape [Choi and Ko 2005; O'Brien et al. 2003; Pentland and Williams 1989]. Large and rotational deformations are less intuitive. For StVK models, one can pre-compute the coefficients of the reduced Hessian and the internal force [Sifakis and Barbic 2012]. It is also possible to combine modal analysis with stiffness warping [Müller et al. 2002] at per-vertex local frames [Choi and Ko 2005]. However, doing so suffers from ghost forces when simulating free-floating objects. Another collection of contributions builds data-driven reduced models. For instance, Kim and James [2009] use recent simulation results to construct the subspace at the simulation runtime. Shen et al. [2021] and Fulton et al. [2019] use an autoencoder to encode the fullspace DOFs using the latent representation. Model reduction can also be coupled with PD. For instance, Brandt et al. [2018] design a reduced model for both local projection and global solve. It is highly efficient when the global matrix is constant (at the cost of missing some local deformations). When the global matrix varies due to collision and contact events or mesh's topology changes, efficiently estimating the updated subspace matrix becomes a challenge. A strategy is to use Woodbury formulation to leverage the pre-factorized global matrix for the rest shape and low-rank matrix update as in [Li et al. 2021b; Modi et al. 2021]. Apart from those existing methods, we employ a novel subspace reuse method – the subspace constructed at the rest shape is used for relax low-frequency errors. This strategy requires minimum computational cost, while the high-frequency is left for GPU-based iterative solvers.

Multigrid method [Trottenberg et al. 2001] is another common technique to boost simulation efficiency when a large number of DOFs is present. It was originally proposed to solve Poisson-like equations abounded in fluid simulation [McAdams et al. 2010; Molemaker et al. 2008]. For deformable/cloth simulation, one needs to represent the dynamics at different levels. The geometric multigrid (GMG) [Georgii and Westermann 2006] approaches this by generating spatial discretization (e.g., meshes or grids) of different resolutions. Xian et al. [2019] further simplifies this process

by sampling points from the finest grid to form coarser grids. The algebraic multigrid (AMG), on the other hand, approaches this by generating a subspace of the fine dynamics, which shares a similar nature of model reduction. For example, Li et al. [2023] utilize a B-spline subspace, and Tamstorf et al. [2015] built the subspace by QR decomposition on near-kernel components. Wang et al. [2018] integrate multigrid into a nonlinear optimization process, which updates the residue and system matrix periodically. Our method can be understood as a two-level multigrid for the global step solve. The reused subspace solve eliminates the low-frequency errors, leaving the high-frequency to the aggregated Jacobi iterations.

3 BACKGROUND

To make the exposition self-contained, we start with a brief review of projective dynamics and the distance-based interior point method for collision resolution. The reader can find more details from the relevant literature e.g., [Bouaziz et al. 2014; Lan et al. 2022b; Li et al. 2020].

Given an implicit time integration scheme such as backward Euler, many state-of-the-art cloth simulators rely on the variational formulation of:

$$\arg\min_x E = I(x, \dot{x}) + \Psi(x), \ I = \frac{1}{2h^2}\|M^{\frac{1}{2}}(x - z)\|^2. \quad (1)$$

Here, x is the unknown variable we would like to compute at the next time step i.e., the position of all the cloth vertices. We also have:

$$z = x^* + h\dot{x}^* + h^2 M^{-1} f_{ext}, \quad (2)$$

as a known vector based on the previous position x^*, velocity $\dot{x}^*$, and an external force f_{ext}. M is the mass matrix, and h is the time step size. The objective function E consists of the inertia momentum (I) offering a mass-weighted regularization over x, and the elasticity potential (Ψ) controlling the deformation of the cloth.

Under the framework of PD, Eq. (1) is split into two stages in the form of local-global (LG) iterations. In the local stage, the unknown DOFs are duplicated at individual constraints, which measure the strain/deformation under various metrics like the change of the edge length or the bending angle. The local step is formatted as:

$$\arg\min_{y_i} \frac{1}{2}\|A_i S_i x - B_i y_i\|^2, \ \text{s.t.} \ C_i(y_i) = 0. \quad (3)$$

In other words, the local step computes a *target position* y_i, which not only satisfies the constraint C_i exactly but is also closest to the current value of x_i i.e., a projection-like operator. Here, S_i is a selection matrix picking DOFs relevant to C_i from x such that $x_i = S_i x$. A_i and B_i map the positional information of x_i and y_i to the specific coordinate that the constraint C_i measures. For instance, they can be a differential operator computing the deformation gradient of a triangle. The local step is highly parallelizable as the computation at each constraint is independent.

The global stage follows as a standard linear solve:

$$\left(\frac{M}{h^2} + \sum_i w_i S_i^\top A_i^\top A_i S_i\right) x = \frac{M}{h^2} z + \sum_i w_i S_i^\top A_i^\top B_i y_i. \quad (4)$$

Intuitively, the goal of Eq. (4) is to blend duplicated DOFs y_i to produce a global solution of x since x_i could have multiple replicates if it is involved in several constraints. As a result, the weight, i.e., w_i

in Eq. (4) embodies the *priority* of a constraint – a bigger w_i (relative to other constraints) means the global solve favors x being closer to the corresponding y_i. In an extreme case when $w_i \rightarrow \infty$, $x_i \rightarrow y_i$ ignoring all the other constraints. One should not confuse w_i with the *stiffness* of the constraint: a high constraint stiffness produces a big internal force, which could overshoot and must be coupled with a line search for extra safeguards.

The presence of the collision and self-collision introduces a new energy into Eq. (1):

$$\arg\min_x E = I(x, \dot{x}) + \Psi(x) + B(x). \quad (5)$$

Codimensional geometries of clothes make the simulation sensitive to inter-penetrations – once collisions or self-collisions are generated, the fabric often gets more and more tangled in the following time steps. IPC [Li et al. 2020] offers a potential solution to the challenge, which formulates $B(x)$ as a log-barrier such that:

$$B_i = \begin{cases} -\kappa(d_i - \hat{d})^2 \ln\left(\frac{d}{\hat{d}}\right), & 0 < d_i < \hat{d} \\ 0, & d_i \geq \hat{d}. \end{cases} \quad (6)$$

Here, $\hat{d}$ is a user-provided tolerance of the collision resolution. $d_i(x)$ denotes the closest distance between the i-th pair of surface primitives, either a vertex-triangle pair or an edge-edge pair. $B(d_i)$ diverges if $d_i < \hat{d}$ and approaches ∞ when $d_i \rightarrow 0$. Consequently, as long as we keep d_i positive at the beginning of a time step (i.e., all primitives are separate), the existence of $B(d_i)$ prevents any future inter-penetration with an increasingly stronger repulsion. Lan et al. [2022b] further showed that IPC barrier function can also be integrated into LG iterations by setting the weighting function w_i of the collision constraint as $B(d_i)$.

4 NON-DISTANCE BARRIER

While IPC [Li et al. 2021a] and its PD variations [Lan et al. 2022b] offer robust treatment for collisions, they are all based on d_i. Here, we name this family of barrier functions as the distance-based barrier or DBB. Updating DBB is expensive – one needs to perform a broad phrase collision culling to generate the list of vertex-triangle or edge-edge pairs, compute TOI for each pair, identify the smallest TOI which is between zero and one of all pairs, compute d_i, and eventually obtain the latest value of $\sum B(d_i)$. Such a *full CCD procedure* is invoked frequently during LG iterations [Lan et al. 2022b] and becomes the dominant computation along the pipeline.

We argue that not all DBBs are indispensable, and most of them can be replaced with a more economic alternative model under the PD framework. The underlying philosophy of DBB is to offer a nonlinear penalty, which becomes stiffer when the collision constraint is about to be violated. DBB geometrically approximates the indicator functions ($\delta_{\mathcal{A}}(x) = 0$, if $x \in \mathcal{A}$; $\delta_{\mathcal{A}}(x) = \infty$, if $x \notin \mathcal{A}$). A DBB itself is *not* physically accurate i.e., the gradient of DBB differs from the actual collision force unless κ in Eq. (6) is close to zero (i.e., the complementary slackness is sufficiently satisfied). In practice, what we want is an increasingly strong repulsion to correct the constraint violation, and this goal can be enabled without d_i, the actual distance between a pair of primitives. To this end, we design

a non-distance barrier weight with exponential formulation:

$$w_i = B_i^{NDB} = kK^{a_i}, \tag{7}$$

where $a_i \in \mathbb{Z}^+$ represents how many *consecutive LG iterations* has the constraint C_i been active. A contact constraint is considered active if the primitive pair remains sufficiently close to each other. k is an initial weight, and K is the base of the growth rate. We use NDB to denote this simplified barrier model. Similar to DBB, NDB approaches to ∞ if a collision constraint C_i remains unresolved over several LG iterations. In this case, the local target position of the constraint shall be satisfied with the highest priority in the global solve. Unlike DBB on the other hand, NDB no longer depends on the distance between primitives or any physically/geometrically meaningful measures. It becomes a self-adjusting variable as the optimization proceeds with minimum computation costs. As we elaborate in the next subsection, this feature delivers substantial convenience and speedup for cloth simulation.

Discussion. DBB takes the primitive distance $d(x)$ as its parameter. In contrast, our non-distance barrier (NDB) strategy does not explicitly depend on the collision distance. Instead, it employs the collision's life span (i.e., number of iterations) to adaptively increase the weight of the constraint. NDB is not a barrier function in a strict sense since NDB is not explicitly related to x. Nevertheless, it introduces a penalty that asymptotically tends to infinity to prevent the interpenetration between primitives. This feature ensures the optimization variable x remains within the feasible region, and works in a similar way as the conventional interior-point method. One limitation, however, is the accuracy of such a weighting mechanism. Since the iteration index is a discrete count, it is sometimes challenging to produce sufficiently different repulsion forces to distinguish nearby vertices. More iterations are therefore needed.

4.1 Partial CCD

An immediate advantage of using NDB is the alleviation of computational effort required for CCD. When updating B_i^{NDB}, it is sufficient to determine whether a collision remains active after the previous LG iteration, requiring a true-or-false response rather than an exact time of impact. This characteristic enables a more streamlined approach for NDB CCD, transforming the cubic root-finding problem into a series of dot product calculations, a technique we refer to as partial CCD.

Starting from a collision-free configuration, if two primitives $\mathcal{P}_1$, $\mathcal{P}_2$ collide with each other within a normalized time interval $(0, 1]$, there exist two points $p_1 \in \mathcal{P}_1$ and $p_2 \in \mathcal{P}_2$ which have intersecting trajectories such that:

$$p_1^0 + t^* \left(p_1^1 - p_1^0\right) = p_2^0 + t^* \left(p_2^1 - p_2^0\right), \tag{8}$$

where $p_{1,2}^0$ and $p_{1,2}^1$ denote the positions of those two points at the beginning and end of the time interval. $t^* \in (0, 1]$ is the intersection time. With some manipulations, Eq. (8) can be re-written as:

$$\left(p_2^1 - p_1^1\right) \cdot \left(p_2^0 - p_1^0\right) = \frac{t^* - 1}{t^*} \left(p_2^0 - p_1^0\right) \cdot \left(p_2^0 - p_1^0\right) \le 0. \tag{9}$$

Eq. (9) suggests a non-positive inner product of $\left(p_2^1 - p_1^1\right) \cdot \left(p_2^0 - p_1^0\right)$ at certain locations on $\mathcal{P}_1$ and $\mathcal{P}_2$ being a necessary condition of the collision between primitives.

We then build a query function based on l.h.s. of Eq. (9) by parameterizing p_1 and p_2 with $\lambda = [\lambda_1, \lambda_2]^\top$ in their corresponding primitives:

$$Q(\lambda) = \left(p_2^1(\lambda) - p_1^1(\lambda)\right) \cdot \left(p_2^0(\lambda) - p_1^0(\lambda)\right). \tag{10}$$

For a vertex-triangle pair specified by x, x_1, x_2, x_3 i.e., x is the position of the vertex and $x_{1,2,3}$ are the three vertices of the triangle as shown in the inset on right, we have $p_1 = x$ and $p_2(\lambda) = x_1 + \lambda_1(x_2 - x_1) + \lambda_2(x_3 - x_1)$, where λ_1, λ_2 and $1 - \lambda_1 - \lambda_2$ are barycentric coordinates of p_2 in the triangle. Similarly, for an edge-edge pair specified by $x_{1,2}$ and $x_{3,4}$, $p_{1,2}$ are: $p_1 = x_1 + \lambda_1(x_2 - x_1)$, $p_2 = x_3 + \lambda_2(x_4 - x_3)$, for $0 \le \lambda_1, \lambda_2 \le 1$. Let Ω_λ be the domain of the query function Q. It is easy to see that Ω_λ forms a triangle $\lambda_{1,2} \ge 0, \lambda_1 + \lambda_2 \le 1$ for a vertex-triangle pair, and a box $\lambda_{1,2} \in [0, 1]$ for an edge-edge pair.

Since $Q(\lambda)$ is continuous, $Q < 0$ prescribes some neighborhoods around λ^*, where $p_1(\lambda^*)$ and $p_2(\lambda^*)$ converges at $t = t^*$. Instead of solving λ^* and the corresponding t^* i.e., as in most CCD algorithms, we query the value of $Q(\lambda)$ at multiple sample points in Ω_λ. As long as our sampling is sufficiently dense to not miss those neighborhoods of $Q < 0$, and all the queried $Q(\lambda)$ values are positive, we conclude that no collision occurs between $\mathcal{P}_1$ and $\mathcal{P}_2$. Eq. (10) does not involve t^*, meaning we skip the calculation for the actual TOI and only compute inner products of 3-vectors. The question is: how dense should the sampling be?

To answer this question, we first set up the metric of sampling density. Let $\mathcal{S} = \left\{\lambda_{(0)}, \lambda_{(1)}, ...\right\} \subset \Omega_\lambda$ be the set of sample points. We define that local sample interval $\rho(\lambda)$ for any $\lambda \in \Omega_\lambda, \lambda \ne \lambda_{(i)}$ is the distance from λ to its nearest sample point in the parameter space. More formally, $\rho(\lambda)$ gives the largest radius of the disc $\Omega(\lambda)$ centered at λ such that $\Omega(\lambda) \cap \mathcal{S} = \emptyset$. If $\lambda = \lambda_{(i)}$ happens to be a sample point, its local sample interval is zero. The *sample interval* of the whole sample set is defined as: $\rho_\mathcal{S} = \max_{\lambda \in \Omega_\lambda} \rho(\lambda)$. Based on Lagrange Remainder Theorem, there exists an upper bound of $Q(\lambda)$ for any $\lambda \in \Omega_\lambda$:

$$Q(\lambda) \le Q(\lambda^*) + \left(\max_{\lambda \in \Omega_\lambda} \|\nabla_\lambda Q\|\right) \|\lambda - \lambda^*\|, \tag{11}$$

which leads to a sufficient condition of $Q(\lambda) \le 0$:

$$Q(\lambda^*) + \left(\max_{\lambda \in \Omega_\lambda} \|\nabla_\lambda Q\|\right) \|\lambda - \lambda^*\| \le 0 \Rightarrow$$

$$\|\lambda - \lambda^*\| \le \rho = -\frac{Q(\lambda^*)}{\max_{\lambda \in \Omega_\lambda} \|\nabla_\lambda Q\|}. \tag{12}$$

As the cloth is less stretchable, we assume that edge lengths of $\mathcal{P}_1$ and $\mathcal{P}_2$ are bounded by L. Given a finite velocity, the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ is bounded above by H^1 during $t \in (0, 1]$, and bounded below by H^0 at $t = 0$. We can then obtain an upper bound

of the norm of $\nabla_\lambda Q$:

$$\left|\frac{\partial Q}{\partial \lambda_{1,2}}\right| \le 2L(H^1 + 2L) \implies \|\nabla_\lambda Q\| \le 2\sqrt{2}L(H^1 + 2L). \quad (13)$$

According to Eq. (9), we also have:

$$Q(\lambda^*) = \frac{t^* - 1}{t^*}\left\|p_2^0 - p_1^0\right\|^2 \le \frac{t^* - 1}{t^*}\left(H^0\right)^2. \quad (14)$$

Together with Eq. (13), a thresholding sample interval over Ω_λ can then be obtained:

$$\rho^* = \left(\frac{1}{\alpha} - 1\right)\frac{\left(H^0\right)^2}{2\sqrt{2}L(H^1 + 2L)}. \quad (15)$$

When ρ_S is smaller than ρ^*, there is at least one sample point sitting in a neighborhood of $Q < 0$. Here, $\alpha < 1$ is the hyperparameter used in line search filtering, which is typically set as 0.8.

Discussion. Partial CCD is sample-based and can be considered as an *approximation* of exact CCD algorithms [Brochu et al. 2012]. We would like to mention that our method does not exclusively rely on partial CCD. In other words, partial CCD, together with exponential-based NDB collision constraint projection works as a warm start, and regular CCDs are still performed during the simulation (but at a much lower frequency). Therefore, we do not need to set the sampling density to ρ^* in practice (as this theoretical bound depends on various factors and varies at each iteration). The performance and the quality of the simulation are not sensitive to partial CCD accuracy (e.g., see Fig. 9).

Eq. (15) appears sensitive to H^0, the closest distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ for $t \in (0, 1]$. For instance, if a vertex is in the proximity of the triangle at $t = 0$ and hits the triangle at $t = t^*$, $\rho(\lambda^*)$ of the corresponding contact approaches zero i.e., the disc of $\Omega(\lambda^*)$ shrinks to a point. This issue can be easily fixed by setting the projection of the vertex on the triangle as a sample point if they are close to each other. In some extreme cases, where the cloth is substantially stretched or accelerated, we increase the sample density by scaling the sample interval by the change ratio of ρ^*.

A quick benchmark well demonstrates the potential of such a sample-based strategy: solving a cubic equation on the GPU as in the original IPC implementation [Li et al. 2020] is slower than evaluating Eq. (9) on a RTX 3090 GPU by over 6, 000 times. Partial CCD is about 2, 000 times faster than the state-of-the-art polynomial solver [Yuksel 2022]. For a high-resolution cloth model, we do not need a lot of sample points on a primitive, making partial CCD faster than regular CCD by orders of magnitude.

5 SUBSPACE REUSE

NDB enables a more affordable collision processing, but it does not alter the fact that the optimization of Eq. (5) is highly nonlinear. Solving the system within a limited time budget remains a computational challenge for time-sensitive applications. Under the framework of PD, the bottleneck is the linear solve at the global stage i.e., Eq. (4). Commonly used strategies resort to iterative GPU solvers such as Jacobi [Wang 2015] or Gauss-Seidel [Fratarcangeli et al. 2016] to solve Eq. (4) inexactly. Unfortunately, the presence of barriers makes those solvers less beneficial. Specifically, it is known that iterative solvers are effective in smoothing high-frequency errors but become
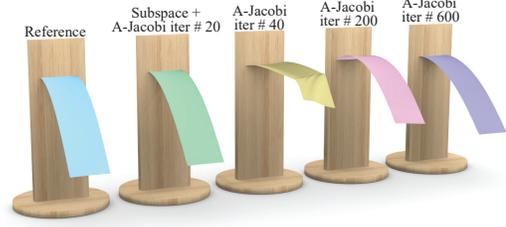
Fig. 2. **Bending strips w. and w/o subspace.** We simulate a collision-free scene where a cloth strip bends under gravity. The model consists of 30K DOFs. The resulting deformation is of low frequency, which is challenging for Jacobi-like methods. A subspace solve effectively resolves this issue: only 20 A-Jacobi iterations are needed to fully converge the simulation, which otherwise takes over one thousand iterations. Because the bending stiffness is quite strong in this example, we need to assign a big SOR-like weight ($\omega = 0.9$) to dampen each A-Jacobi iteration. Our method runs over 300 FPS for this example, while PD-IPC [Lan et al. 2022b] is less than 0.5 FPS due to the large number of A-Jacobi iterations.

cumbersome when dealing with low-frequency residuals. In contrast, low-frequency cloth deformation can be efficiently handled using subspace methods. A concrete example is shown in Figs. 2 and 3. To this end, we tackle Eq. (4) using the reduced direct solver and fullspace iterative solver, aiming to reap the goods from both sides.

Our subspace is for the global stage only. While building a reduced model for the local step is also possible [Brandt et al. 2018], we explicitly avoid doing so to retain local details like wrinkles, folds, and creases. The subspace matrix U is made of eigenvectors of l.h.s. of Eq. (4), corresponding to r smallest eigenvalues. We first solve Eq. (4) in the column space of U for the reduced displacement q:

$$U^\top H(X + Uq) = U^\top b \implies (U^\top H U) q = U^\top (b - HX)$$
$$\implies \Lambda q = U^\top (b - HX), \quad (16)$$

where $H = \left(\frac{M}{h^2} + \sum_i w_i S_i^\top A_i^\top A_i S_i\right)$ and $b = \frac{M}{h^2}z + \sum_i w_i S_i^\top A_i^\top B_i y_i$ per Eq. (4). X is rest-pose vertex positions.

As U is eigenvectors, $U^\top H U = \Lambda$ becomes a diagonal matrix of eigenvalues. If the constraint set stays unchanged during the simulation, solving Eq. (16) is highly efficient on the GPU – we only need one subspace projection for evaluating $U^\top (b - HX)$, and diagonalized subspace solve is negligible. The resulting subspace displacement of q is then converted to its fullspace counterpart as $u = Uq$. At this point, most low-frequency errors have been eliminated by the subspace solve, and $\widetilde{x} = X + Uq$ represents an ideal guess of Eq (4) for iterative solvers, which only has high-frequency errors. We then use aggregated-Jacobi (A-Jacobi) as in [Lan et al. 2022b] to solve for Δx to relax the (high-frequency) residual of $\widetilde{x}$:

$$H(\widetilde{x} + \Delta x) = b \implies H\Delta x = b - H\widetilde{x} \implies H\Delta x = b - H(X + Uq). \quad (17)$$

A representative experiment is reported in Fig. 2, where a cloth strip is attached to the wall and gets bent under gravity. When the low-frequency deformation is computed within the subspace ($r = 30$), it only takes 20 A-Jacobi iterations to converge the simulation to the ground truth i.e., the exact global solve. General A-Jacobi
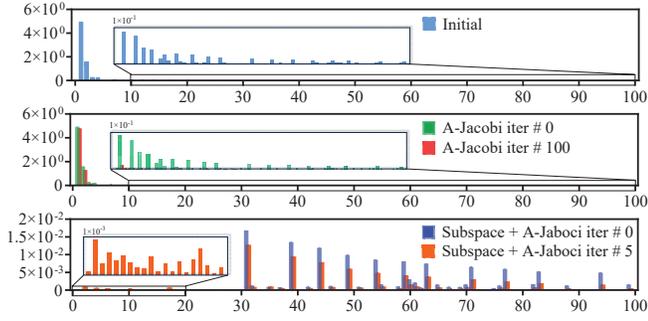
Fig. 3. **Spectral distribution of residual errors (w/o collision).** We plot the distribution of residual error over the first 100 modal bases of the strip test (Fig. 2). As shown at the top, the dominant deformation is low-frequency, which is efficiently solved within the subspace (bottom). On the other hand, A-Jacobi iterations are not effective in dealing with low-frequency residuals. 100 A-Jacobi iterations barely lower the low-frequency errors, while the high-frequency strains are well relaxed (middle).

iterations are not effective for such low-frequency deformations – we observe noticeable visual difference even after 600 iterations (the right-most beam). In this case, the bending stiffness of the cloth is relatively high, and the vanilla Jacobi or A-Jacobi do not even converge. We use a large successive over-relaxation-like (SOR) weight ($\omega = 0.9$) to dampen each A-Jacobi update. The error distribution of this experiment is plotted in Fig. 3, which is consistent with our previous analysis.

Collisions are ubiquitous in cloth simulation, and H varies during the time integration. Building a new subspace at the simulation runtime does not sound practical for interactive simulations. Our key observation here is: *while the global-stage matrix is altered by different collision constraints, the essential structure of its low-frequency subspace remains unchanged.* It seems counterintuitive at first sight: how can old eigenvectors still be effective while the matrix is modified? This is because low-frequency modes built at the rest shape depict general and global strain distribution over the garment. This is not strongly coupled with

Fig. 4. **Subspace reuse.** A piece of tablecloth (66K DOFs) drops on a wooden Armadillo. Over 30% of vertices are involved in collision constraints.

high-frequency deformations. For example, the appearance of local wrinkles is less influenced by the overarching movements of the cloth but more by the specific configurations of the corresponding collisions. Interestingly, it is the low-frequency deformations that most severely hinder the convergence of iterative solvers. By eliminating or even just reducing these low-frequency errors, we could significantly enhance the convergence of the subsequent Jacobi method.

Following this rationale, our subspace reuse strategy pre-computes U as eigenvectors of the smallest r eigenvalues of H at the rest pose. During the simulation, the new global matrix becomes H + ΔH i.e., ΔH is the modification of H induced by collisions or self-collisions.
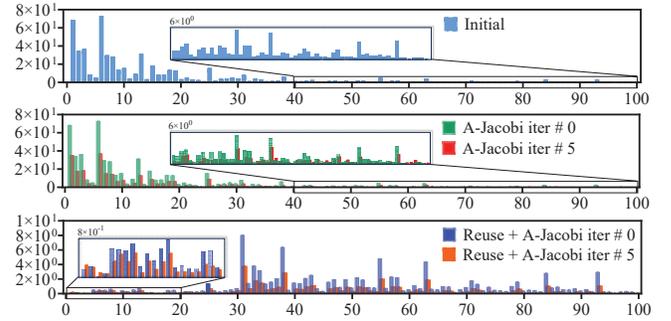

Fig. 5. **Spectral distribution of residual errors (w. collision).** We visualize the distribution of residual errors over the first 100 modal bases of the rest-pose global matrix H when the tablecloth covers the Armadillo (as shown in Fig. 4). Subspace reuse does generate some low-frequency errors, but it still helps the convergence of the A-Jacobi significantly.
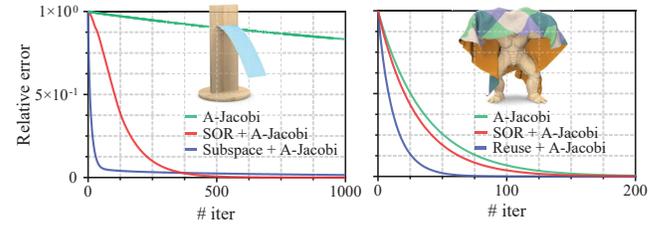

Fig. 6. **Convergence curves w. and w/o subspace (reuse).** We plot the convergence curves for experiments in Figs. 2 and 4. We can see subspace solve saves a large faction of A-Jacobi iterations even under intensive collisions. The computation cost of one subspace solve, on the other hand, is similar to performing one or two A-Jacobi iterations on average.

Despite the matrix change, we stick with rest-shape subspace and solve q out of the following reduced global-stage system:

$$U^\top (H + \Delta H) U q = \left(\Lambda + U^\top \Delta H U\right) q = U^\top (b - HX - \Delta H X). \quad (18)$$

Solving the above system is slightly more expensive than Eq. (16) since $\left(\Lambda + U^\top \Delta H U\right)$ is no longer diagonal. For a subspace of low dimension e.g., $r = 30$, it is still quite efficient using less than 0.1 *ms*. Fig. 4 shows an experiment where a piece of tablecloth covers a wooden Armadillo. The cloth has 66K DOFs, and over 25K DOFs are associated with collision constraints. Similar to Fig. 3, we report the error distribution for this simulation, where the subspace bases are constructed with the rest-pose matrix H. Clearly, reused subspace solve is less perfect compared with Fig. 3. It still effectively handles low-frequency errors, which will need several hundred A-Jacobi iterations otherwise. The convergence curves using subspace and subspace reuse are reported in Fig. 6.

5.1 Pre-computed subspace update

The most expensive computation is the subspace projection of ΔH i.e., evaluating $U^\top \Delta H U$. The complexity is $O(N^2 r)$ on the surface. This is prohibitive even on the GPU if we want the simulation to be interactive and high-resolution at the same time.

We obviate this difficulty exploiting the unique structure of $U^\top \Delta H U$. First, we set $A_i = B_i = \mathrm{Id}$ in Eq. (3) as an identity matrix (Id) for local projection of collision constraints whose target (collision-free) positions are computed as in [Lan et al. 2022b]. With NDB, the target positions can be achieved effectively within just a few LG iterations. We note that ΔH becomes a diagonal matrix under this treatment.

Suppose that there are k colliding vertices indexed as $c_1, c_2, \cdots, c_k$, and the weights of the corresponding collision constraints are $w_1, w_2, \cdots, w_k$ respectively. Let us denote each r-dimension row vector of U as $U_i^\top$ such that $U = [U_1, U_2, ...]^\top$. It can be verified that:

$$U^\top \Delta H U = \sum_{j=1}^{k} w_j U_{c_j} \otimes U_{c_j} \in \mathbb{R}^{r \times r}. \qquad (19)$$

Therefore the change of the subspace matrix caused by those k colliding vertices is the weighted summation over k rank-one matrices. All of these $U_{c_j} \otimes U_{c_j}$ can be pre-computed, and every $U_{c_j} \otimes U_{c_j}$ is a $r \times r$ symmetric matrix. In practice, we only save its upper triangle at each vertex, and the corresponding memory footprint is lightweight. The computation of the summation, on the other hand, is unfolded at each of $r(r-1)/2$ matrix elements using CUDA. The pre-computed subspace update using Eq. (19) is 50 to 150 times faster than directly evaluating $U^\top \Delta H U$ using cuBlas.

After q is computed, we use A-Jacobi to relax the remaining high-frequency residual as in Eq. (17) and move to the next LG iteration until the stopping criterion is satisfied. After an LG iteration, partial CCD ensues, and w_i of the collision constraints are updated per Eq. (7). Our subspace reuse scheme can efficiently calculate the updated (reduced) global matrix and allows the simulation to enjoy the advantages of both subspace solvers and iterative solvers with small computational costs. Such combined efficiency and convergence are unseen in previous GPU algorithms (e.g., see Fig. 6). In practice, we use two subspaces to handle simulation without and with collisions (as detailed in Sec. 8.1). It is noteworthy that this method is also highly effective for volumetric deformable models, where low-frequency motions are particularly challenging for iterative solvers.

6 RESIDUAL FORWARDING

In interactive applications, simulation modules normally have prescribed time budgets regardless if the solver reaches the convergence. This hard constraint forces the simulator to enter the final CCD-based linear search filtering and truncate the position update Δx by the global TOI: $\Delta x \leftarrow \alpha t_{TOI} \cdot \Delta x$. If the TOI is a smaller quantity, severe damping or locking artifacts could be produced as a considerable portion of the system energy dissipates by the early termination.

To *partially* alleviate this issue, we propose a post-step treatment namely residual forwarding, or RF in short. The idea of RF is to estimate the remaining residual generated by small-step line search filtering and/or non-convergent LG iterations. Recall that each time step, the simulation seeks a minimizer $x^\star$ of Eq. (5), which ideally should possess a vanished gradient $\nabla E(x^\star) = 0$. By the end of a time step, if LG iterations fail to fully converge, the resulting x^* is different from $x^\star$ such that $x^\star = x^* + \delta x$. The gradient of the variational energy $\nabla E(x^*) = -f_r \neq 0$ represents unbalanced

residual forces in the system. RF seeks the virtual force δf at the next time step to mitigate the damping artifacts induced by δx. Therefore, Eq. (2) becomes:

$$z = x^* + h\dot{x}^* + h^2 M^{-1}(f_{ext} + \delta f). \qquad (20)$$

On the other hand, if the actual minimizer $x^\star = x^* + \delta x$ were used, the ground truth z should be:

$$z = x^\star + h\dot{x}^\star + h^2 M^{-1} f_{ext} = x^* + h\dot{x}^* + 2\delta x + h^2 M^{-1} f_{ext}. \qquad (21)$$

Note $\dot{x}^\star = \dot{x}^* + \frac{\delta x}{h}$ also depends on the previous position under implicit Euler. By adding δf, RF offsets the derivation of $2\delta x$ with the compensation of $h^2 M^{-1} \delta f$.

To compute the optimal δf, we Taylor expand ∇E around $x^\star$ as:

$$\nabla E(x^\star) = \nabla E(x^*) + \nabla^2 E(x^*) \cdot \delta x + \epsilon(\|\delta x\|^2), \qquad (22)$$

where ϵ is a quadratic error term. Because $\nabla E(x^\star) = 0$, we obtain:

$$\delta x = \left(\nabla^2 E(x^*)\right)^{-1} \left(f_r - \epsilon(\|\delta x\|^2)\right). \qquad (23)$$

Assuming $\epsilon(\|\delta x\|^2)$ is sufficiently small, the most effective RF should minimize:

$$\arg\min_{\delta f} \left\| \frac{h^2}{2} M^{-1} \delta f - \left(\nabla^2 E(x^*)\right)^{-1} f_r \right\|. \qquad (24)$$

Taking a closer look, it is noted that computing $\left(\nabla^2 E(x^*)\right)^{-1} f_r$ is equivalent to taking one more Newton solve by the end of the previous time step, which can be efficiently approximated with subspace reuse. In RF, *we do not need to perform CCD-based line search filtering*, and the weights for all the collision constraints is k i.e., a_i for i-th collision constraint is set zero in Eq. (7).

Discussion. RF is a *heuristic treatment* when the current time step must end for other time-critical tasks. If $\epsilon(\|\delta x\|^2)$ is big, RF becomes erroneous and generates artifacts. Honestly, there is nothing much we can do if the available time budget is aggressively restrained. Conceptually, RF moves some computation e.g., solve for $\left(\nabla^2 E(x^*)\right)^{-1} f_r$ to the next time step. The advantage of RF is skipping the line search filtering since z could embody an overlapping and penetrating configuration. The collision constraint set is then updated in the follow-up LG iteration in the future time step. RF, on the other hand, allows non-colliding vertices to move under cloth momentum and elasticity even with a small t_{TOI} (i.e., from the previous time step). As a result, damping/locking artifacts are ameliorated.

7 SIMULATION PIPELINE

We now have all the pieces to assemble our simulator. Fig. 7 visualizes major steps along our pipeline, and the pseudocode is also outlined in Alg. 1. The pre-computation stage performs the eigen-decomposition of the rest-shape global matrix H, which ignores the collision constraints. We use two subspaces to handle global solves without and with collisions. Specifically, a subspace of higher dimension is first constructed out of $\bar{r}$ eigenvectors. Because the rest-shape modal global matrix $\Lambda = U^\top H U$ is diagonal, and the subspace projection of r.h.s. vector is efficient on GPU, increasing the dimensionality of the subspace for collision-free global solve is worthy and effective. The first $r < \bar{r}$ eigenvectors of U form a
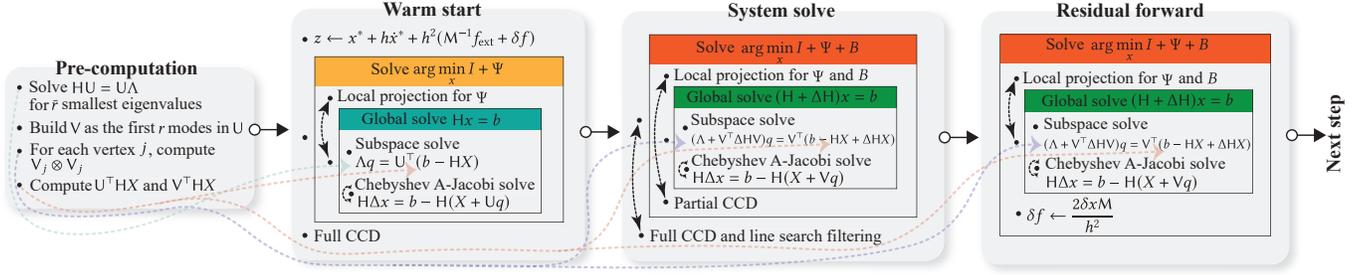
Fig. 7. Algorithm overview. Our method leverages a subspace reuse technique to improve the convergence of GPU-based iterative solvers. A large portion of costly CCDs is replaced with a non-distance non-distance barrier formulation i.e., NDB. The pipeline also features a residual forwarding mechanism, which generates plausible animation even with small exiting TOI. We use a bigger subspace of $\bar{r}$ dimension to obtain a good warm start at the beginning of each time step. A more compact r-dimension subspace is used for handling NDB-in-the-loop optimization. Each time step consists of several blocks of LG iterations, which are visualized with curved double arrows.

ALGORITHM 1: Our simulation pipeline.

1: $z \leftarrow x^* + h\dot{x}^* + h^2(M^{-1}f_{ext} + \delta f)$ // δf is from RF

2: $x \leftarrow z$

3: $\|\Delta x\| \leftarrow \infty$

4: **while** $\|\Delta x\| < \epsilon_{initial}$ **do**

5:     $\Delta x \leftarrow \text{ModalLG}(I + \Psi)$ // in span(U)

6: **end**

7: $x^- \leftarrow x^*$

8: $x^+ \leftarrow x + \Delta x$

9: $\langle B, t_{TOI} \rangle \leftarrow \text{FullCCD}(x^-, x^+)$// B is the latest barrier

10: $x \leftarrow x + t_{TOI} \cdot \Delta x$

11: $x^- \leftarrow x$

12: **while** *outer loop convergence check fails* **do**

13:     **while** *inner loop convergence check fails* **do**

14:         $\Delta x \leftarrow \text{ModalLGReuse}(I + \Psi + B)$ // in span(V)

15:         $x^+ \leftarrow x + \Delta x$

16:         $B_i \leftarrow \text{PartialCCD}(x^-, x^+)$

17:     **end**

18:     $\langle B, t_{TOI} \rangle \leftarrow \text{FullCCD}(x^-, x^+)$ // update B at each outer loop

19: **end**

20: $x \leftarrow x + t_{TOI} \cdot \Delta x$// exiting line search filtering

21: **if** $t_{TOI} < \epsilon_{TOI}$ **then**

22:     $B_i = \kappa$ // quadratic approximation of B

23:     **while** $\|\Delta x\| < \epsilon_{inner}$ **do**

24:         $\delta x \leftarrow \text{ModalLGReuse}(I + \Psi + B)$

25:     **end**

26:     $\delta f \leftarrow \dfrac{2\delta x M}{h^2}$

27: **end**

---

smaller set of bases V of a more compact subspace, and we set $r = 30$ in our implementation. V is for subspace reuse when collision constraints are taken into account. For each vertex j, we pre-compute the corresponding $V_j \otimes V_j$ as in Eq. (19) for fast computation of the updated subspace matrix. Lastly, $U^\top HX$ and $V^\top HX$ are also pre-computed for faster assembly of the r.h.s. of the global solve (i.e., Eqs. (16) and (18)).

Keeping V condensed is helpful for both efficiency and efficacy of the simulation. From the efficiency point of view, we know $\Lambda +$

$V^\top \Delta HV$ is not diagonal. Solving a dense linear system is only feasible when the system is of low dimension. From the efficacy point of view, the subspace reuse is effective in the low-frequency realm. Barrier constraints induced by collisions and contacts can drastically change the higher-frequency landscape. As a result, expanding V to the high-frequency spectrum is not beneficial. Since we will need to save $V_j \otimes V_j$ for each vertex, a more compact V_j is also more memory-friendly.

Each time step begins with a warm-start computation. Typically, z (i.e., in Eq. (20)) is used as the initial guess of x at the current time step and for setting up the initial constraints B^{NDB}. As the collision is ignored for the warm start, $\bar{r}$-dimension subspace provides a better initialization by solving $\arg\min_x I + \Psi$ (line 5 in Alg. 1). The decoupled computations in the modal space of span(U) make this procedure highly efficient. For instance, the warm start only takes three to five LG iterations and less than five milliseconds for a 300K-DOF simulation. The resulting x is forwarded to the full CCD procedure. The constraint list is then built, and $B(x)$ is initialized (line 9).

Afterwards, the system solves for the optimization of $\arg\min_x I + \Psi + B$. Similar to existing barrier-in-the-loop PD algorithms [Lan et al. 2022b], we employ a two-level iteration scheme. The outer loop consists of multiple LG iterations, and the outer convergence check is based on the norm of the change of x between two consecutive outer loops. If we uniformly scale the cloth meshes to a unit size, $\|\Delta x\| \leq 1E - 3$ is a good choice for convergence check. For simulations involving fast-moving objects, setting $\|\Delta x\| \leq 5E - 4$ may be needed. Each inner loop begins with a standard LG iteration. The local projections are performed for both elasticity constraints (Ψ) and collision/barrier constraints (B). At the global solve, we re-use the subspace V to smooth high-frequency errors and pass the residual to A-Jacobi iterations. The convergence of the inner loop is also based on $\|\Delta x\|$ from the previous inner loop ($\|\Delta x\| \leq 5E - 2$). The partial CCD routine is then invoked (line 16) to adjust the weights of collision/contact constraints. As discussed in Sec. 4, this computation boils down to computing the inner products of 3-vectors.

By the end of each time step, a full CCD and an exiting line search filtering are performed. They offer the algorithmic guarantee that the simulation is free of inter-penetration. If the exiting t_{TOI}

Fig. 8. **Cloth twisting NDB.** We rotate the table cloth at both ends for 2, 880 degrees. The simulation includes over 120K DOFs. Our method completes this test at 12 FPS. Compared with the distance-based barrier weighted PD method i.e., [Lan et al. 2022b]. NDB saves about 30% LG iterations (as plotted in Fig. 9).

is too small suggesting possible locking and overdamping due to insufficient outer loops, we trigger the residual forward, which estimates an optimal correction force δf for the next step.

8 EXPERIMENTAL RESULTS

We implemented the proposed simulation framework on a desktop computer with an `intel i7-12700` CPU and an `nVidia 3090 RTX` GPU. We used `Spectra` library for computing the eigendecomposition of the global stage matrix H. It should be noted that our method is also friendly with other parallel computing platforms – one can easily parallelize local projections using multi-threading, and multicore CPUs are well suited for the Gauss-Seidel method. Nevertheless, we only report the performance on the GPU. The reader can find animated results in the accompanying video demo.

8.1 Implementation details

Most parts of our framework are matrix-free, except for the subspace solve step of $V^\top (H + \Delta H)V$. For solving a linear system of $Ax = b$, the common practice is to pre-factorize A and compute x via forward and backward substitutions. In our implementation however, we directly obtain $X \in \mathbb{R}^{r \times r}$ via:

$$AX = \frac{1}{\beta}\text{Id}, \qquad (25)$$

to get an approximation of $X \approx A^{-1}/\beta$. The reader should not confuse A in Eq. (25) with A_i used in local projection (e.g., Eq. (3)). Here, A refers to the reduced global-stage matrix $V^\top (H + \Delta H)V$ with subspace reuse. β is a scaling factor estimated as $\beta = \sum |b_i|/r$ i.e., the average of absolute values of elements in b. Doing so could mitigate numerical drift induced by small or large values in b. This is because calculating X via $AX = \text{Id}$ implicitly assumes the r.h.s. of the system is around one.

With X computed, the system solve becomes a matrix-vector multiplication of Xb, which can be parallelized on the GPU. The standard forward/backward substitutions are sequential (even on the GPU). When many LG iterations are needed, computing Xb is more efficient than using the factorized matrix. In our implementation, we simply send A back to the CPU, compute X, and return it to the GPU. Since the reused subspace has a very low dimensionality, it ensures that the associated computations and CPU-GPU communications are fast and require minimal resources. The whole procedure takes less than 0.1 ms, which is $30\% - 35\%$ faster than factorizing A on the GPU. Nevertheless, the system solve is *not* the bottleneck. In our implementation, we use rank-2 A-Jacobi method, which computes
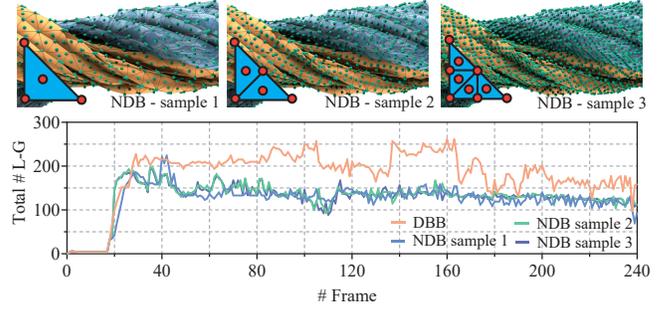

Fig. 9. **NDB vs DBB.** We plot the total number of LG iterations at each time step when twisting the cloth (as in Fig. 8) using DBB and exponential NDB. It can be seen that our NDB formulation uses 25% to 30% fewer iterations on average than DBB when allows the constraints to be adaptively re-weighted in time. Partial CCD is not sensitive to the sample density. As shown in the figure, increasing or decreasing partial CCD samples do not vastly alter the convergence behavior of the simulation. Different sample patterns are also visualized in the figure.

two regular Jacobi iterations with one step but using the same computation time [Lan et al. 2022b]. Thanks to subspace reuse, weighted SOR is never needed even for stiff simulations (e.g., Fig. 2). The base of exponential NDB (i.e., Eq. (7)) is set as $K = 2$ in our experiments. Setting K to 3 seems to produce a similar result. However, an over-aggressive K could negatively impact the convergence. In this case, this exponential NDB behaviors like a geometric projection.

To fully exploit the capacity of modern GPUs, the broad-phase collision culling leverages a patch-based BVH. Specifically, we build an incomplete BVH whose leaf houses a small patch of the cloth mesh. A patch consists of several inter-connected triangles, normally five to eight. After the initial AABB-based culling at each BVH level, from top to bottom, we exhaustively test triangle pairs between two nearby patches as well as pairs within a patch. At the narrow phase stage, if a full CCD is needed we solve t_{TOI} for each primitive pair using the polynomial solver proposed in [Yuksel 2022]. For partial CCD, we compute inner products of Eq. (10) at pre-selected sample points plus the projection points at $t = 0$. We periodically check if a denser sampling is needed given the current system velocity (since the time step size is assumed fixed). Partial CCD is more efficient for cloth models of higher resolutions. We noted that, as long as the total number of DOFs exceeds 50K, very few sample points (e.g., three) work well for partial CCD in most simulation scenarios.
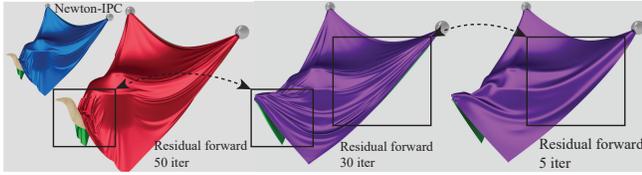
Fig. 10. **Residual forwarding.** We show an experiment where a piece of tablecloth falls on the teapot. Residual forwarding estimates a virtual force to restore the dynamics at non-colliding vertices in the next step. It produces reasonably good results when the residual errors are moderate. RF fails to handle all the errors with a highly constrained iteration cap. As we lower the per-step iteration count, we observe more artifacts even using RF.

8.2 NDB & DBB

The first test we would like to show is a comparison between the distance-based barrier (DBB) and exponential non-distance barrier i.e.,NDB. The major difference lies in the fact that NDB allows the weight of the collision constraint to be timely adjusted during LG iterations based on inexpensive partial CCD. The snapshots of the resulting simulation are reported in Fig. 8. We also compare the total number of LG iterations using NDB and DBB for this twisting test, and the plots are shown in Fig. 9. It can be seen that this adaptive weighting strategy helps reduce iterations, as each collision constraint is more likely to find an appropriate weight during iterations. NDB strategyis not sensitive to the sample density (ρ). To this end, Fig. 9 also plots iteration counts for NDB with different sampling densities. We can see that NDB works well even using one sample point at the center of Ω_λ.

8.3 Residual forwarding

Our method employs RF to mitigate damping and locking artifacts caused by limited time budgets. As discussed in Sec. 6, RF can enhance animation quality to a certain extent by "inheriting" unresolved residual errors from one time step to the next. The effectiveness of RF is primarily up to the magnitude of these errors. We observe that small and localized errors, resulting from early termination or line search filtering under a small TOI, are generally well-managed by RF. Another factor is the capability of minimizing residuals at the current time step. If the solver does not fully converge at the current step, the carried-over residuals may exacerbate simulation inaccuracies. RF is particularly designed for applications facing strict time constraints, such as gaming, where high-speed collisions might cause spiky FPS drops.

To better illustrate the effectiveness of RF, we simulate a scene where the cloth drops on a teapot (Fig. 10). There are 120K DOFs on the cloth. After the cloth comes in contact with the spout of the teapot, the inertia effect further moves the cloth towards the right, and the cloth eventually settles at the recess between the spout and the body of the teapot. When we have the time budget to complete 50 iterations per time step (which *do not* fully converge the solver), RF produces a high-quality result nearly identical to the ground truth – the one generated using a fully converged Newton IPC solver. Our method runs at 20 FPS, which is 500× faster than [Li et al. 2021a]. If we half the iteration number and exit the current time step with the line search filtering, the simulator is unable to deal

Fig. 11. **"Animal crossing".** The subspace reuse technique allows our solver to handle stiff materials and deformable bodies with ease. This example includes 33 animal toys, 10 elastic ribbons, 678K DOFs, and one million elements. Both our method and PD-IPC [Lan et al. 2022b] produce penetration-free animations, while our method is 12× faster than PD-IPC on RTX 3090.

with all the residuals, and we can see dampened cloth movement after it touches the teapot. Because of the early termination, the cloth "sticks" to the tip of the spout as the rest part of the cloth is locked (see highlighted area in the figure). Lastly, we further lower the iteration cap to only five. We observe more severe damping artifacts. Because of the subspace resue, our method still handles low-frequency residuals well, but most high-frequency information is lost due to an insufficient number of A-Jacobi iterations. The entire cloth exhibits rubber-like dynamics, and the collision is highly inelastic. This comparison is available in the supplementary video.

8.4 Comparison with existing methods

PD is a popular framework, based on which many excellent cloth and deformable simulation algorithms have been developed. In this subsection, we compare our method with several representative PD-based cloth simulation methods, including PD-IPC [Lan et al. 2022b], PD-BFGS [Li et al. 2023], and PD-Coulomb [Ly et al. 2020].

Our method vs. PD-IPC. PD-IPC [Lan et al. 2022b] is a full GPU simulator. It also integrates PD with IPC for deformable and cloth simulation. Unlike our method, PD-IPC uses DBB, and a full CCD must be invoked every time the constraint set is to be updated. PD-IPC solves the global step system only using A-Jacobi iterations on the GPU. For stiff simulations, one must adopt a close-to-one SOR weight to ensure A-Jacobi iterations do not diverge. This is not an issue for us since the subspace solve removes dominant low-frequency errors. As a result, our method outperforms PD-IPC by a significant margin in general. Fig. 11 reports a deformable body simulation result. In this example, we have 33 deformable animal toys, 10 elastic ribbons, and over one million elements falling into a glass tank. Some animal toys are five times stiffer than others. Such heterogeneous materials/bodies are particularly challenging for PD-IPC as the SOR weight of the A-Jacobi must be set conservatively ($\omega \geq 0.85$), which impairs the convergence. In this example, our method is 12× faster. As NDB does not depend on the distance, updating NDB can be processed using partial CCD. This strategy significantly speeds up the evaluation of barrier weighting and reduces more than 60% of computation time.

Our method vs PD-BFGS. Another relevant competitor is from a recent contribution by Li et al. [2023] or PD-BFGS. PD-BFGS designs a two-step global solve combining Jacobi iteration with BFGS

Table 1. **Experiment statistics.** We report detailed time statistics for experiments mentioned in the paper. # **B** gives the total number of objects in the scene. # **DOF** is the total number of simulation DOFs. # **Ele.** records the total number of elements (i.e., triangles and tetrahedrons). # **Con.** and # **Col.** are average numbers of elasticity constraints and collision constraints during the simulation. In the column of # **Col.**, the first quantity reports the number of collision constraints involved in CCD and the second number is the total number of primitive pairs after the broad phase collision detection. $\bar{r}|r$ represent the subspace size and reused subspace size. Δt is the time step size. **Pre.** is the pre-computation time (measured in seconds). The column **B|N|P|F** gives the timing information (in milliseconds) used for collision detection. Specifically, **B** is the broad phase time, and **N** is the narrow phase time. The narrow phase also includes partial CCD (**P**) and full CCD (**F**) procedures. ρ_m gives the mass density of the cloth (and the deformable objects e.g., in Figs. 11 and 17). The unit is $\frac{kg}{m^2}$ for the cloth (and $\frac{kg}{m^3}$ for deformable objects). κ (in MPa) gives the stretching and bending stiffness of the cloth (and Young's modulus of deformable objects). $\|\Delta x\|$ is the convergence condition. **LG (#)** gives the total time (first row) used and total number of LG iterations (second row) on average for each step. **Misc.** corresponds to other additional computations. Except for **Pre.**, other timings are measured in milliseconds. **FPS** is the overall FPS of the simulation.

Scene	# B	# DOF	# Ele.	# Con.	# Col.	$\bar{r}\|r$	Δt	Pre.	B|N|P|F	ρ_m	κ	$\|\Delta x\|$	LG (#)	Misc.	FPS
Twisting cloth (Fig. 8)	1	121K	80K	161K	110K 8M	120|30	$\frac{1}{150}$	6.3	8|15|2|13	0.3	160 $3 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	47.4 (157)	10.2	12.4
"Animal cross." (Fig. 11)	44	678K	1.1M	1.1M	67K 9.2M	120|30	$\frac{1}{150}$	46.8	20|10|0.4|10	0.9	$2 \cdot 10^3$	$1 \cdot 10^{-3}$	103.2 (47)	9.2	7.0
Make a knot (Fig. 12)	1	310K	203K	410K	13K 4M	120|30	$\frac{1}{150}$	13.6	8|7|0.2|7	0.3	160 $3 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	63.6 (74)	6.1	11.8
Drape one a sphere (Fig. 13)	2	120K	80K	160K	7K 3M	120|30	$\frac{1}{200}$	6.2	4|4|0.2|4	0.3	100 $2 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	30.8 (38)	1.2	25.0
Stack on teapot (Fig. 14)	11	500K	338K	677K	28K 7M	120|30	$\frac{1}{150}$	24.1	8.4|9.1|0.3|8.8	0.3	160 $3 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	35.9 (31)	6.6	16.6
Just folding (Fig. 15)	5	485K	320K	498K	23K 7M	120|30	$\frac{1}{150}$	7.6	5|7.5|0.5|7	0.3	160 $5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	64.4 (46)	3.6	12.4
Cloth blender (Fig. 16)	8	820K	540K	992K	160K 26M	120|30	$\frac{1}{150}$	25.4	10|11.6|2.6|9	0.3	160 $3 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	160.7 (73)	10.2	5.2
Cover the ship (Fig. 17)	5	930K	941K	1.2M	63K 9.2M	120|30	$\frac{1}{150}$	31.8	8|8.3|1.2|7.1	0.8|0.9	$160\|8 \cdot 10^3$ $2 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	117.8 (51)	5.3	7.2
Funnel (Fig. 18)	6	692K	458K	923K	276K 32M	120|30	$\frac{1}{150}$	32.8	17|13|2|9	0.3	160 $3 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	112.4 (52)	8.5	6.6
Kicking (Fig. 19)	2	450K	294K	525K	47K 4M	120|30	$\frac{1}{200}$	11.6	14|11|0.6|10	0.3	200 $3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	113.5 (94)	8.5	6.8
Fashion show (Fig. 1)	2	1.1M	656K	996K	84K 10M	120|30	$\frac{1}{200}$	28.4	14|14.7|1.6|13.1	0.3	160 $3 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	167.3 (67)	12.4	4.8

Fig. 12. **Make a knot.** Two cloth strips are pulled from opposite directions to form a tight knot. Both our method and PD-BFGS [Li et al. 2023] successfully handle this challenging simulation. Nevertheless, our method is 130× faster due to sample-based partial CCD with exponential NDB re-weighting strategy, and a more efficient subspace reuse strategy on global solve. In this experiment, there are 104K vertices and 203K triangles on the mesh.

Fig. 13. **Drape on a sphere.** A tablecloth of 40K vertices drapes on the sphere. We compare our method with PD-Coulomb [Ly et al. 2020]. PD-Coulomb models Coulomb friction, while our method uses a lagged friction Hessian at each LG iteration as in [Li et al. 2023].

method in a spline-based subspace. PD-BFGS is also based on DBB, and thus slower than our method for IPC-based collision processing. At the global stage, PD-BFGS employs a reduced quasi-Newton procedure in a B-spline subspace. This subspace is of high dimension i.e., $r \approx 10,000$. As a result, PD-BFGS is much slower than the pre-computed subspace update used in our simulation. A concrete experiment is reported in Fig. 12, where we pull two intertwining cloth strips to make a tight knot. There are 104K vertices on the strips. Our method handles this example at an interactive rate of 11.8 FPS, while PD-BFGS needs seconds for each frame.
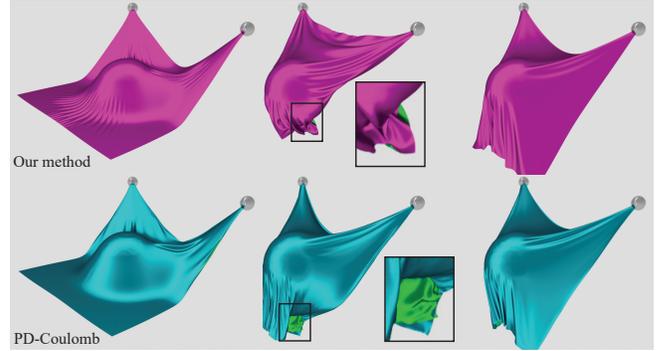
Our method vs PD-Coulomb. Our method is compatible with various friction models. In our implementation, we follow the strategy used in [Li et al. 2023], which leverages a quadratic barrier proxy to estimate the Hessian of the friction energy to simulate frictional surfaces with different friction coefficients – in a way similar to the original IPC. Alternatively, Ly et al. [2020] (PD-Coulomb) show that it is possible to incorporate full Coulomb friction by a novel splitting strategy so that the global matrix is kept constant. To this
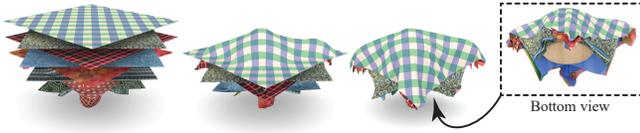
Fig. 14. **Stack on the teapot.** We stack ten pieces of tablecloth on the teapot simultaneously. The free fall of stacking clothes generates a large number of intertwining cloth-cloth collisions. EXP and partial CCD robustly handle this challenging scene and produce high-quality animation.

Fig. 15. **Just folding.** In this example, four clothes fall vertically on the desk one by one with different orientations. The contact between the cloth and the desktop folds the cloth with complex self-collisions. This is a good stress test because the velocity of falling cloth is high. The resulting multi-layer self-collisions are particularly challenging. There are nearly 500K DOFs involved, and the simulation runs at 12 FPS.

end, we show a side-by-side comparison between our method and PD-Coulomb [Ly et al. 2020].

In this experiment, we drape a piece of square cloth of 40K vertices on a sphere. The size of the cloth is $1m \times 1m$, and the time step is $\Delta t = 1/200$ sec. It can be seen from Fig. 13 (and also see the supplementary video) that both methods capture the frictional contact between the cloth and the sphere and produce similar animation results. In fact, the splitting method proposed in [Ly et al. 2020] can also be integrated with our pipeline. PD-Coulomb adopts DCD, and we follow the default setting as in the published code of [Ly et al. 2020]. The collision tolerance is set as $1mm$. This means as long as the distance between two triangles is smaller than $1mm$, DCD will generate a contact. While this is a conservative configuration, considering the movement of the cloth is moderate, one can still spot minor inter-penetration among the cloth triangles. On the other hand, our method uses CCD and a line search filtering at the end of each step, which guarantees all the triangles are separate.

It should be mentioned that PD-Coulomb is CPU-based, using OpenMP to speed up pre-factorized global solve, while our method is fully GPU-based. As a result, our method runs faster (at 25 FPS) than PD-Coulomb by 30 times. Such performance difference largely comes from the hardware platform. Nevertheless, our method is orthogonal to PD-Coulomb as we do not focus on friction modeling but more on solver optimization and collision detection.

8.5 More experiments

We have tested our method in a wide range of simulation scenes. The detailed time statistics of all the experiments shown in the paper

are reported in Tab. 1. Figs. 14, 15, and 16 show three tests with massive collisions. In Fig. 14, we test the robustness of our method under extensive stacking. In this test, ten falling tablecloths hang on the teapot. This simulation generates a lot of overlapping collisions when clothes stack under gravity. Fig. 15 reports another test where we fold four clothes by vertically dropping them on the desk. As the cloth makes contact with the desktop, it folds in a zig-zag pattern, resulting in a large number of self-collisions, particularly edge-edge collisions. This experiment is a good stress test showing the robustness of a cloth simulator. Fig. 16 mimics a washer with six clothes in a bowl-shaped container. The scene also involves a large number of dynamic collisions. Our method produces interesting animations in all of those experiments, and the results are free of any inter-penetration.

Our method is also capable of simulating deformable objects. In addition to the example reported in Fig. 11, we give another experiment with two-way coupling between cloth and deformable body under complex collisions. As shown in Fig. 17, a heavy cloth falls onto a barbarian ship. The ship base is fixed, and it has soft masts and ladders. As the cloth falls down, we can see detailed wrinkles at the contacting area between the cloth and the ship. There are 540K DOFs on the cloth and 390K DOFs on the ship. Our method generates interesting animation results at 7.2 FPS.

Fig. 18 shows two more simulations under different frictional setups. Three pieces of cloth cover a funnel, and we place a rigid/heavy ball on the funnel. We use affine body dynamics [Lan et al. 2022a] to simulate the ball's motion. There are nearly 700K DOFs in this scene, and our method produces plausible dynamics under different friction settings. The simulation runs at 6.6 FPS. As discussed, we can switch to a more accurate frictional model either as in [Ly et al. 2020], which is based on complementarity programming or as in [Li et al. 2020], which is based on the interior-point method.

Cloth animations play a pivotal role in the realm of digital fashion and design. The proposed method enhances this aspect by enabling virtual characters to interact seamlessly with a variety of garments, yielding high-quality simulations. As a demonstration of its capabilities, we report two additional examples. The first features a character executing a kicking action, illustrating the dynamic interaction between the motion and the garment (Fig. 19). This is a numerically challenging test as the body undergoes swift movements. Most DCD-based methods are unable to capture such high-velocity garment dynamics. The second example showcases a virtual model on a fashion runway as shown in Fig. 1. This model, attired in a knee-length skirt, walks to the front of the stage before turning around to return. Throughout this sequence, our method meticulously captures the intricacies of the motion. Please refer to the supplementary for the animated results, where we report two sequences of animations

9 CONCLUSION & LIMITATION

This paper presents a parallelizable cloth simulation framework, that delivers high-quality animation results, keeps computations lightweight, and separates all the triangles on the cloth models. To the best of our knowledge, such combined efficiency, quality, and performance are not possible with existing cloth animation algorithms. To achieve this goal, we employ a non-distance barrier,

Fig. 16. **Cloth blender.** We drop six clothes into a bowl-like container. The blender rotates and stirs all the clothes. This example generates fast rotational collisions between clothes and the collider, as well as a large number of cloth-cloth collisions. The simulation has 820K DOFs and runs at about 5 FPS.

Fig. 17. **Cover the ship.** We cover a deformable barbarian ship with a piece of heavy cloth ($\rho_m = 0.8kg/m^2$). The soft masts and ladders on the ship bend under the weight of the cloth cover. The ship-cloth contacts generate fine wrinkles that delineate the shape features of the ship. There are 540K DOFs on the cloth and 390K DOFs on the barbarian ship. With $\Delta t = 1/150$, our method resolves all the cloth-cloth, cloth-ship, and ship-ship collisions and runs at 7.2 FPS.
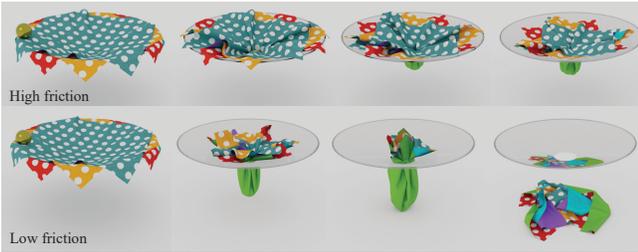

Fig. 18. **Funnel.** We place a rigid and heavy ball (with affine body dynamics [Lan et al. 2022a]) on a funnel covered by three layers of cloth. The cloths hold the sphere under high frictional contacts (top). When the friction is not strong enough, all the cloths eventually fall on the ground (bottom). The simulation involves nearly 700K DOFs, and our method runs at 6.6 FPS.

Table 2. **More statistics on parameters.** We report total LG iterations and time cost (in ms) using different simulation parameters. Detailed results (including artifacts and simulation failure cases) are presented in the supplementary videos.

Cloth folding			
$\Delta t(s)$	$\|\Delta x\| = 1 \cdot 10^{-4}$	$\|\Delta x\| = 5 \cdot 10^{-4}$	$\|\Delta x\| = 1 \cdot 10^{-3}$	$\|\Delta x\| = 1 \cdot 10^{-2}$
1/150	39(16.3ms)	18(8.3ms)	15(6.9ms) **Jittery**	4(2.8ms) **Jittery**
1/100	55(24.7ms)	26(13.5ms)	20(11.1ms) **Jittery**	**Fail**
1/50	95(56.4ms)	45(26.6ms)	40(20.9ms) **Jittery**	**Fail**
Cloth twisting			
$\Delta t(s)$	$\|\Delta x\| = 1 \cdot 10^{-4}$	$\|\Delta x\| = 5 \cdot 10^{-4}$	$\|\Delta x\| = 1 \cdot 10^{-3}$	$\|\Delta x\| = 1 \cdot 10^{-2}$
1/150	214(64.8ms)	171(52.2ms)	157(47.4ms)	72(21.6ms)
1/100	229(69.7ms)	182(56.1ms)	163(49.4ms)	**Fail**
1/50	255(77.3ms)	202(61.1ms)	171(51.9ms)	**Fail**

which is both simple and effective. This new barrier model allows the simulation to skip the computation of the distance between primitive pairs, and the weight of each collision constraint becomes self-adjusting during LG iterations given the current active set. The subspace reuse scheme significantly pushes the performance of the solver with minimum costs. By observing the fact that the low-frequency subspace is less sensitive to high-frequency collisional deformations, we reuse the rest-shape modal global matrix to solve global cloth deformation. The subspace matrix update is also efficient and can be pre-computed. The residual forwarding helps mitigate the dumpling artifacts due to small-TOI line searches. We show that this approach yields visually plausible animations with the penetration-free guarantee and makes the simulation efficient for high-resolution scenes.

Our method has some limitations. First, our method is based on projective dynamics. Therefore, the elasticity model (Ψ) should be shaped in a quadratic form. This prevents us from incorporating more complex fabric models in the simulation e.g., homogenized [Sperl et al. 2020] or data-driven models [Feng et al. 2022]. However, we believe it is possible to combine our subspace reuse strategy with block descent methods [Lan et al. 2023] for general materials. Residual forwarding trick becomes less effective if the residual errors are too large and may cause simulation failure: in some situations, RF can introduce unnatural bumpy artifacts, which may be even more visually disturbing (see Tab. 2, two columns on the right, Fig. 20 and supplementary videos). While our method adapts to various simulation parameters, it is more sensitive to timestep size and termination criteria compared to the projection-Newton.

Fig. 19. **Kicking.** In this example, the virtual character quickly performs a kicking action, which leads to nonlinear animation effects on the multi-layer skirt. Our method produces high-quality results. The frame rate reaches 6.8 FPS, which is two-order faster than the state-of-the-art GPU simulation [Li et al. 2023].
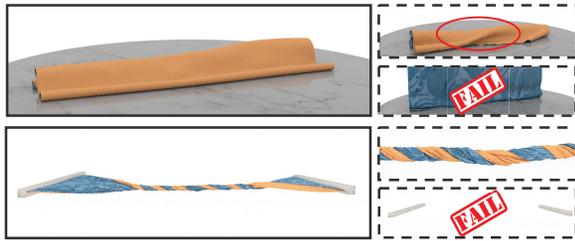

Fig. 20. **Folding and twisting.** As a non-Newton method, the performance of our method is correlated with time step size and termination criteria. RF may introduce unnatural jittery artifacts and may cause simulation failure when the residual error is too large. The animated results are available on supplementary video.

ACKNOWLEDGMENTS

REFERENCES

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98).* Association for Computing Machinery, New York, NY, USA, 43–54. https://doi.org/10.1145/280814.280821

David Baraff, Andrew Witkin, and Michael Kass. 2003. Untangling Cloth. 22, 3 (jul 2003), 862–870. https://doi.org/10.1145/882262.882357

Gino van den Bergen. 1997. Efficient collision detection of complex deformable models using AABB trees. *Journal of graphics tools* 2, 4 (1997), 1–13.

Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. 2006. A Quadratic Bending Model for Inextensible Surfaces. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) *(SGP '06).* Eurographics Association, Goslar, DEU, 227–230.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Transactions on Graphics* 33, 4 (2014), Article–No.

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. 21, 3 (jul 2002), 594–603. https://doi.org/10.1145/566654.566623

Tyson Brochu, Essex Edwards, and Robert Bridson. 2012. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–7.

Bernard Brogliato and B Brogliato. 1999. *Nonsmooth mechanics.* Vol. 3. Springer.

Thomas Buffet, Damien Rohmer, Loïc Barthe, Laurence Boissieux, and Marie-Paule Cani. 2019. Implicit Untangling: A Robust Solution for Modeling Layered Clothing. 38, 4, Article 120 (jul 2019), 12 pages. https://doi.org/10.1145/3306346.3323010

Kwang-Jin Choi and Hyeong-Seok Ko. 2002. Stable but Responsive Cloth. *ACM Trans. Graph.* 21, 3 (jul 2002), 604–611. https://doi.org/10.1145/566654.566624

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 (jan 2005), 91–101. https://doi.org/10.1109/TVCG.2005.13

Gilles Daviet, Florence Bertails-Descoubes, and Laurence Boissieux. 2011. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In *Proceedings of the 2011 SIGGRAPH Asia Conference.* 1–12.

Xudong Feng, Wenchao Huang, Weiwei Xu, and Huamin Wang. 2022. Learning-based bending stiffness parameter estimation by a drape tester. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–16.

Zachary Ferguson, Pranav Jain, Denis Zorin, Teseo Schneider, and Daniele Panozzo. 2023. High-Order Incremental Potential Contact for Elastodynamic Simulation on Curved Meshes. In *ACM SIGGRAPH 2023 Conference Proceedings.* 1–11.

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. 40, 4, Article 183 (jul 2021), 16 pages. https://doi.org/10.1145/3450626.3459802

Marco Fratarcangeli, Valentina Tibaldo, Fabio Pellacini, et al. 2016. Vivace: a practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph.* 35, 6 (2016), 214–1.

Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. 2019. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 379–391.

Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M. Teran. 2015. Optimization Integrator for Large Time Steps. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (2015), 1103–1115. https://doi.org/10.1109/TVCG.2015.2459687

Joachim Georgii and Rüdiger Westermann. 2006. A Multigrid Framework for Real-Time Simulation of Deformable Bodies. *Comput. Graph.* 30, 3 (jun 2006), 408–415. https://doi.org/10.1016/j.cag.2006.02.016

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient simulation of inextensible cloth. In *ACM SIGGRAPH 2007 papers.* 49–es.

Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. 1996. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques.* 171–180.

Peng Guan, Loretta Reiss, David A. Hirshberg, Alexander Weiss, and Michael J. Black. 2012. DRAPE: DRessing Any PErson. *ACM Trans. Graph.* 31, 4, Article 35 (jul 2012), 10 pages. https://doi.org/10.1145/2185520.2185531

David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. 2008. Robust Treatment of Simultaneous Collisions. 27, 3 (aug 2008), 1–4. https://doi.org/10.1145/1360612.1360622

Philip Martyn Hubbard. 1995. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics* 1, 3 (1995), 218–230.

S. Huh, D.N. Metaxas, and N.I. Badler. 2001. Collision resolutions in cloth simulation. In *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No.01TH8596).* 122–127. https://doi.org/10.1109/CA.2001.982385

Doug L James and Dinesh K Pai. 2004. BD-Tree: Output-sensitive collision detection for reduced deformable models. In *ACM SIGGRAPH 2004 Papers.* 393–398.

L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. 2006. Geometric, Variational Integrators for Computer Animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vienna, Austria) *(SCA '06).* Eurographics Association, Goslar, DEU, 43–51.

Theodore Kim. 2020. A Finite Element Formulation of Baraff-Witkin Cloth. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 171–179.

Theodore Kim and Doug L James. 2009. Skipping steps in deformable simulation with online model reduction. In *ACM SIGGRAPH Asia 2009 papers.* 1–9.

Shankar Krishnan, M Gopi, M Lin, Dinesh Manocha, and A Pattekar. 1998. Rapid and accurate contact determination between spline models using ShellTrees. In *Computer Graphics Forum*, Vol. 17. Wiley Online Library, 315–326.

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. *ACM Trans. Graph.* 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 Transactions on Graphics* 42, 4 (2023).

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.

Elmar Langetepe and Gabriel Zachmann. 2006. *Geometric data structures for computer graphics.* AK Peters/CRC Press.

Jing Li, Tiantian Liu, Ladislav Kavan, and Baoquan Chen. 2021b. Interactive cutting and tearing in projective dynamics with progressive cholesky updates. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–12.

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, Ming Gao, Timothy Langlois, Chenfanfu Jiang, and Danny M. Kaufman. 2019. Decomposed Optimization Time Integrator for Large-Step Elastodynamics. *ACM Trans. Graph.* 38, 4, Article 70 (jul 2019), 10 pages. https://doi.org/10.1145/3306346.3322951

Minchen Li, Danny M Kaufman, and Chenfanfu Jiang. 2021a. Codimensional incremental potential contact. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–24.

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.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-Spring Systems. 32, 6, Article 214 (nov 2013), 7 pages. https://doi.org/10.1145/2508363.2508406

Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective dynamics with dry frictional contact. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 57–1.

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.

A. McAdams, E. Sifakis, and J. Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. Eurographics Association, Goslar, DEU, 65–74.

Vismay Modi, Lawson Fulton, Alec Jacobson, Shinjiro Sueda, and David IW Levin. 2021. Emu: Efficient muscle simulation in deformation space. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 234–248.

Jeroen Molemaker, Jonathan M. Cohen, Sanjit Patel, and Jonyong Noh. 2008. Low Viscosity Flow Simulations for Animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 9–18.

Jean J Moreau. 1988. Unilateral contact and dry friction in finite freedom dynamics. In *Nonsmooth mechanics and Applications*. Springer, 1–82.

Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. 2002. Stable Real-Time Deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 49–54. https://doi.org/10.1145/545261.545269

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

James O'Brien, Kris Hauser, and Chen Shen. 2003. Interactive Deformation Using Modal Analysis with Constraints. *Graphics Interface* 3 (05 2003).

A. Pentland and J. Williams. 1989. Good Vibrations: Modal Dynamics for Graphics and Animation. *SIGGRAPH Comput. Graph.* 23, 3 (jul 1989), 207–214. https://doi.org/10.1145/74334.74355

Xavier Provot. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation '97*, Daniel Thalmann and Michiel van de Panne (Eds.). Springer Vienna, Vienna, 177–189.

Xavier Provot et al. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface*. Canadian Information Processing Society, 147–147.

Siyuan Shen, Yin Yang, Tianjia Shao, He Wang, Chenfanfu Jiang, Lei Lan, and Kun Zhou. 2021. High-order differentiable autoencoder for nonlinear model reduction. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–15.

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* (Los Angeles, California) *(SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA, Article 20, 50 pages. https://doi.org/10.1145/2343483.2343501

Georg Sperl, Rahul Narain, and Chris Wojtan. 2020. Homogenized yarn-level cloth. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 48–1.

Georg Sperl, Rosa M Sánchez-Banderas, Manwen Li, Chris Wojtan, and Miguel A Otaduy. 2022. Estimation of yarn-level simulation models for production fabrics. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–15.

Rasmus Tamstorf, Toby Jones, and Stephen F. McCormick. 2015. Smoothed Aggregation Multigrid for Cloth Simulation. *ACM Trans. Graph.* 34, 6, Article 245 (nov 2015), 13 pages. https://doi.org/10.1145/2816795.2818081

Min Tang, Ruofeng Tong, Rahul Narain, Chang Meng, and Dinesh Manocha. 2013. A GPU-based streaming algorithm for high-resolution cloth simulation. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 21–30.

Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. 2018. I-Cloth: Incremental Collision Handling for GPU-Based Interactive Cloth Simulation. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (November 2018), 204:1–10.

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 (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 205–214. https://doi.org/10.1145/37401.37427

Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. 2009. Continuum-based strain limiting. *Computer Graphics Forum* 28, 2 (3 2009), 569–576. https://doi.org/10.1111/j.1467-8659.2009.01397.x

Ulrich Trottenberg, Cornelius W. Oosterlee, Anton Schuller, and Achi Brandt. 2001. *Multigrid.* Academic Press.

Pascal Volino and Nadia Magnenat-Thalmann. 2006. Resolving Surface Collisions through Intersection Contour Minimization. 25, 3 (jul 2006), 1154–1159. https://doi.org/10.1145/1141911.1142007

Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. 2009. A Simple Approach to Nonlinear Tensile Stiffness for Accurate Cloth Simulation. 28, 4, Article 105 (sep 2009), 16 pages. https://doi.org/10.1145/1559755.1559762

Bolun Wang, Zachary Ferguson, Teseo Schneider, Xin Jiang, Marco Attene, and Daniele Panozzo. 2021. A Large-Scale Benchmark and an Inclusion-Based Algorithm for Continuous Collision Detection. *ACM Trans. Graph.* 40, 5, Article 188 (sep 2021), 16 pages. https://doi.org/10.1145/3460775

Huamin Wang. 2014. Defending continuous collision detection against errors. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Huamin Wang. 2015. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–9.

Huamin Wang, James O'Brien, and Ravi Ramamoorthi. 2010. Multi-Resolution Isotropic Strain Limiting. *ACM Trans. Graph.* 29, 6, Article 156 (dec 2010), 10 pages. https://doi.org/10.1145/1882261.1866182

Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. 2011. Data-driven elastic models for cloth: modeling and measurement. *ACM transactions on graphics (TOG)* 30, 4 (2011), 1–12.

Huamin Wang and Yin Yang. 2016. Descent Methods for Elastic Body Simulation on the GPU. *ACM Trans. Graph.* 35, 6, Article 212 (dec 2016), 10 pages. https://doi.org/10.1145/2980179.2980236

Tianyu Wang, Jiong Chen, Dongping Li, Xiaowei Liu, Huamin Wang, and Kun Zhou. 2023. Fast GPU-Based Two-Way Continuous Collision Handling. *ACM Trans. Graph.* 42, 5, Article 167 (jul 2023), 15 pages. https://doi.org/10.1145/3604551

Zhendong Wang, Longhua Wu, Marco Fratarcangeli, Min Tang, and Huamin Wang. 2018. Parallel multigrid for nonlinear cloth simulation. *Computer Graphics Forum* 37, 7 (10 2018), 131–141. https://doi.org/10.1111/cgf.13554

Martin Wicke, Hermes Lanker, and Markus Gross. 2006. Untangling Cloth With Boundaries. (01 2006).

Longhua Wu, Botao Wu, Yin Yang, and Huamin Wang. 2020. A Safe and Fast Repulsion Method for GPU-Based Cloth Self Collisions. 40, 1, Article 5 (dec 2020), 18 pages. https://doi.org/10.1145/3430025

Zangyueyang Xian, Xin Tong, and Tiantian Liu. 2019. A Scalable Galerkin Multigrid Method for Real-Time Simulation of Deformable Objects. *ACM Trans. Graph.* 38, 6, Article 162 (nov 2019), 13 pages. https://doi.org/10.1145/3355089.3356486

Juntao Ye and Jing Zhao. 2012. The Intersection Contour Minimization Method for Untangling Oriented Deformable Surfaces. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Goslar, DEU, 311–316.

Cem Yuksel. 2022. A Fast and Robust Solution for Cubic and Higher-order Polynomials. In *ACM SIGGRAPH 2022 Talks (SIGGRAPH 2022)*. ACM, New York, NY, USA. https://doi.org/10.1145/3532836.3536266

Gabriel Zachmann. 2002. Minimal hierarchical collision detection. In *Proceedings of the ACM symposium on Virtual reality software and technology*. 121–128.