Subspace-Preconditioned GPU Projective Dynamics with Contact for Cloth Simulation

Xuan Li
UCLA
xuan.shayne.li@gmail.com

Yu Fang
UCLA
University of Pennsylvania
squarefk@gmail.com

Lei Lan
University of Utah
lei.lan@utah.edu

Huamin Wang
Style 3D Research
wanghmin@gmail.com

Yin Yang
University of Utah
yin.yang@utah.edu

Minchen Li
UCLA
Carnegie Mellon University
minchernl@gmail.com

Chenfanfu Jiang
UCLA
Style 3D Research
chenfanfu.jiang@gmail.com

Figure 1: Kick (High-Res). Our method can efficiently simulate a high-resolution version (more than 120K nodes) of the Kick animation in CIPC [Li et al. 2021] at 23s per frame, which is $6.5 \times$ faster than a heavily optimized and GPU accelerated CIPC solver. All collisions are robustly handled with intricate wrinkles captured on the cloth, highlighting the efficacy of our approach in handling fine-detailed garment simulations in complex animation scenarios.

ABSTRACT

We propose an efficient cloth simulation method that combines the merits of two drastically different numerical procedures, namely the subspace integration and parallelizable iterative relaxation. We show those two methods can be organically coupled within the framework of projective dynamics (PD), where both low- and high-frequency cloth motions are effectively and efficiently computed. Our method works seamlessly with the state-of-the-art contact handling algorithm, the incremental potential contact (IPC), to offer the non-penetration guarantee of the resulting animation. Our core ingredient centers around the utilization of subspace for the expedited convergence of Jacobi-PD. This involves solving the reduced global system and smartly employing its precomputed factorization. In addition, we incorporate a time-splitting strategy to handle the frictional self-contacts.

Specifically, during the PD solve, we employ a quadratic proxy to approximate the contact barrier. The prefactorized subspace system matrix is exploited in a reduced-space LBFGS. The LBFGS method starts with the reduced system matrix of the rest shape as the initial Hessian approximation, incorporating contact information into the reduced system progressively, while the full-space Jacobi iteration captures high-frequency details. Furthermore, we address penetration issues through a penetration correction step. It minimizes an incremental potential without elasticity using Newton-PCG. Our method can be efficiently executed on modern GPUs. Experiments show significant performance improvements over existing GPU solvers for high-resolution cloth simulation.

CCS CONCEPTS

• Computing methodologies → Physical simulation.

KEYWORDS

subspace, projective dynamics, domain decomposition, quasi-Newton methods

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0315-7/23/12...
https://doi.org/10.1145/3618548.3618157

ACM Reference Format:
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 (SA
1 INTRODUCTION

In cloth simulation, a fine and high-resolution discretization is often needed for rich and vivid effects like detailed wrinkles, folds, and creases, which coarser models cannot produce. The complexity, however, increases disproportionately w.r.t to the degrees of freedom (DOFs) of the system, making high-resolution simulation prohibitive for time-critical applications, whereas animating virtual garments at an interactive rate is always desired.

It is well understood that the primary obstacle for high-resolution cloth animation lies in the computational cost associated with the system solve for numerical integration at each time step. The cloth dynamics is nonlinear, and commonly used solvers rely on incremental linearization of the equation of motion e.g., see [Baraff and Witkin 1998]. Collisions and self-collisions, which are ubiquitous in cloth simulation, impose extra difficulties. Traditional methods often leverage soft repulsion to handle collisions [Tang et al. 2018; Wu et al. 2020]. These approaches require careful parameter tuning to prevent undesirable artifacts like cloth interpenetration. The state-of-the-art solution to contact modeling is incremental potential contact (IPC) [Li et al. 2020], which utilizes log barriers and continuous collision detection (CCD) to strictly maintain objects’ separation. The joint optimization with cloth elasticity is numerically challenging as the collision component is considerably stiffer and of higher frequency. The increased resolution also vastly complicates the spectrum of the system. While a wide range of numerical algorithms are available, they are proven only effective in limited or specific situations. For instance, gradient-based strategies [Wang and Yang 2016] are quite parallelizable and efficient, but the performance declines quickly for stiffer instances. On the other hand, direct solvers are robust against numerical stiffness when paired with line search [Wolfe 1969, 1971], but they are highly sequential and expensive for large-scale problems.

The pros and cons of existing algorithms endorse their strong complementarity, which forms our key rationale, illustrated in Fig. 2. Specifically, we seek algorithmic synergy between direct and iterative numerical procedures for efficient high-resolution cloth animations. We argue that parallel local relaxation schemes e.g., Jacobi or Gauss-Seidel are most effective if low-frequency residuals can be pre-eliminated. The latter happens to be the task in which subspace methods excel. The coordination of those two simulation modalities collectively delivers combined efficiency and convergence that existing methods can hardly match. Model reduction limits is extensively used to prevent over-constraint [Goldenthal et al. 2007; Provot et al. 1995; Thomaszewski et al. 2009; Wang et al. 2010].

Considerable research has focused on refining material modeling to accurately emulate cloth’s mechanical behaviors. For example, Volino et al. [2009] introduced a nonlinear and anisotropic tensile model based on continuum mechanics. Cloth bending is closely tied to the parameterization of the dihedral angle [Volino et al. 1995], with Breen et al. [1994] leveraging linear beam theory to link bending moment and curvature. Bridson et al. [2005] developed the bending model of a hinge-based element orthogonal to in-plane deformations. In the context of inextensible fabrics, discrete mean curvature approximates bending, yielding a quadratic energy with a constant Hessian [Bergou et al. 2006; Wardetzky et al. 2007]. The bending model introduced by Choi and Ko [2005a; Gingold et al. 2004; Grinspun et al. 2003]. Modern cloth animation workflows largely incorporate an implicit time integration scheme, a practice pioneered by Baraff and Witkin [1998]. Given the reduced stretchability of many fabrics, strain limiting is extensively used to prevent over-constraint [Goldenthal et al. 2007; Provot et al. 1995; Thomaszewski et al. 2009; Wang et al. 2010].

Figure 2: Subspace simulation, Newton method and Chebyshev-Jacobi excel at reducing residuals in different frequency ranges. Our method combines the advantages of Subspace BFGS and Chebyshev-Jacobi to achieve similar outcomes as Newton-PCG, but with improved performance.

2 RELATED WORK

2.1 Cloth Simulation

Cloth and thin shell simulation, ubiquitous in daily life, continue to be central in the realm of computer graphics and animation [Choi and Ko 2005a; Gingold et al. 2004; Grinspun et al. 2003]. Modern cloth animation workflows largely incorporate an implicit time integration scheme, a practice pioneered by Baraff and Witkin [1998]. Given the reduced stretchability of many fabrics, strain limiting is extensively used to prevent over-constraint [Goldenthal et al. 2007; Provot et al. 1995; Thomaszewski et al. 2009; Wang et al. 2010].

In this paper, we propose a novel subspace-preconditioned projective dynamics (PD) framework [Bouaziz et al. 2014] for cloth simulation. At each global step of PD, we solve for the displacement field within the subspace to capture low-frequency motion modes, with a pre-factorized subspace global matrix. The high-frequency details are dealt with using parallel Chebyshev-Jacobi relaxation [Wang 2015; Wathen 2008] on the original full-order system. We adopt a time-splitting scheme along with a quadratic contact proxy [Xie et al. 2023] to handle complex contacts. A novel modified Bryden-Fletcher-Goldfarb-Shanno (BFGS) method is incorporated to progressively integrate the quadratic contact proxy into the pre-factorized subspace global matrix, which circumvents the need for re-factorization. Our comparative evaluation demonstrates remarkable performance gain, enabling another 10x acceleration over the state-of-the-art simulation techniques - like IPC, the resulting animation is guaranteed to be free of any interpenetrations.
2.2 Collision Handling

Developing accurate contact models is crucial in mechanics, robotics, and computer graphics [Andrews et al. 2022; Johnson and Johnson 1987]. Traditional methods often handle contacts as constraint-based linear complementarity problems (LCP) [Baraff 1994; Kaufman et al. 2008], resolved using projected Gauss-Seidel (PGS) method. An alternate approach uses quadratic programming (QP) [Macklin James 2005; Sifakis and Barbic 2012]. These subspaces, usually built as manifolds for Reduced-Order Modeling (ROM), with neural networks being used to construct the dynamics of a fine model. For instance, Capell et al. [2002] include geometric shape coarsening, akin to skin rigging to preprocess fluids [Kim and Delaney 2013; Treuille et al. 2006], and computational design problems [Xu et al. 2015]. Alternative approaches include geometric shape coarsening, akin to skin rigging to prescribe the dynamics of a fine model. For instance, Capell et al. [2002] deformed an elastic body using an embedded skeleton, Gilles et al. [2011] employed 6-DOF rigid frames, Brandt et al. [2018]; Faure et al. [2011] employed scattered handles, and Martin et al. [2010] used sparsely-distributed integrators for rods, shells, and solids.

Recent work has started to investigate nonlinear low-dimensional manifolds for ROM, with neural networks being used to construct these spaces [Lee and Carlberg 2020]. This approach can require smaller latent space dimensions compared to linear methods [Fulton et al. 2019; Shen et al. 2021]. There has also been significant progress in data-driven latent space dynamics [Lusch et al. 2018], with neural networks being used to learn the evolution of the entire latent space [Wiewel et al. 2019]. To construct a subspace with sparse basis for general cloth dynamics, we stick with linear subspaces and use 2D B-spline functions as the building block.

3 BACKGROUND

In this section, we provide a brief overview of the Projective Dynamics (PD) and Incremental Potential Contact (IPC) techniques, with a specific focus on cloth simulation, to ensure self-containment.

3.1 Projective Dynamics for Cloth Simulation

In the absence of collision, the PD solver employs the following optimization time integration for time stepping:

\[
\min_{\hat{x}} \frac{1}{2\Delta t} \| \hat{x} - \dot{x} \|^2_M + \frac{E_{\text{mem}}}{2} \sum_t \| F_t - R(F_t) \|^2 + \frac{E_{\text{bend}}}{2} \sum_c \| \hat{x}_c \|^2_Q,
\]

Here, \( h \) is the time step size, \( \hat{x} = x^* + \hat{x}^* + gh^2 \) is the predictive position with \( x^* \) being the current position, \( F_t \) denotes the deformation gradient of triangle \( t \), \( R(F) \) represents the closest rotation matrix to \( F \), \( E_{\text{mem}} \) and \( E_{\text{bend}} \) correspond to the membrane stiffness and bending stiffness, respectively, and \( Q_e \) is the local quadratic bending stiffness matrix for inner edge \( e \), as outlined by Bergou et al. [2006].

Rather than directly optimizing the energy equation (1), PD decouples it by introducing auxiliary rotations \( R_k \) for each triangle, enabling optimization through a global-local alternating approach:

\[
\min_{x^k} \frac{1}{2\Delta t} \| x^k - \hat{x} \|^2_M + \frac{E_{\text{mem}}}{2} \sum_t \| F_k^t - R_k^t \|^2 + \frac{E_{\text{bend}}}{2} \sum_c \| x^k_c \|^2_Q,
\]

\[ R_k^{t+1} = R(F_k^t). \]

The global step involves solving a linear system with a fixed system matrix \( H = \frac{1}{\Delta t} M + K_{\text{mem}} + K_{\text{bend}}, \) where \( K_{\text{mem}}, K_{\text{bend}} \) are membrane energy Hessian and bending energy Hessian at the rest shape. The local step can be executed efficiently in parallel. This alternating procedure continues until convergence is achieved. A precomputed Cholesky factorization can be applied to solve the global step. However, as the resolution increases, the computational time grows significantly. Moreover, multiple updates are required for the rotation matrices to ensure accuracy. To mitigate this, it is common to solve the global step inexacty using a few or even just one Jacobi iteration, while updating the rotation matrices as frequently as possible. To accelerate convergence, Chebyshev acceleration techniques can be applied, as suggested by Wang [2015]. However, a substantial number of iterations are still required for high-resolution scenes.

3.2 Incremental Potential Contact

IPC [Li et al. 2020, 2021] is an approach that handles contact constraints using smooth log barriers on unsigned distances to ensure separations between objects. It provides a robust method for processing collisions, where the log barrier potential is included as an


Algorithm 1 Timestepping of subspace-preconditioned PD

if it is the first time step then
    Construct subspace basis sparse matrix \( \mathcal{P} \). \( \triangleright \) Sec. 4.2
    Factorize the reduced-order global matrix \( \mathcal{P}^T \mathcal{H} \mathcal{P} \). \( \triangleright \) Sec. 4.3
end if

Update predictive position \( \hat{x} \).
Run a reduced-order global step w.o. contact for an initial guess.
Construct quadratic barrier proxy at current state \( x^* \).
Initialize subspace BFGS history.
while not converged do
    Run 2 iterations of subspace BFGS and update the history.
    Run 5 fullspace Jacobi iterations.
    Run PD local projections in parallel.
end while
Run penetration correction step. \( \triangleright \) Sec. 4.4.2

4 METHOD

4.1 Algorithm Overview

Our subspace preconditioned PD pipeline is summarized in Algorithm 1. We elaborate further details in the following subsections.

4.2 Construction of Subspace

Clothing items are typically composed of several flat fabric pieces, connected by stitches. Taking this into account, it is sufficient to utilize basis functions defined in \( \mathbb{R}^2 \).

Suppose the cloth domain \( \Omega \) is divided into multiple patches interconnected by stitches: \( \Omega = \bigcup_{k=1}^{K} \Omega_k \). For each cloth patch \( \Omega_k \), we employ the As-Rigid-As-Possible (ARAP) parameterization technique [Liu et al. 2008] to obtain a bijective parameterization \( \Omega_k \). In certain cases, additional patch decompositions might be required to ensure bijectivity. Next, we embed each \( \Omega_k \) into a regular 2D Cartesian grid and employ Material Point Method (MPM) quadratic spline shape functions on the grid points [Jiang et al. 2016] as the basis for one spatial dimension of \( \mathbb{R}^3 \). Specifically, each basis is a product of two one-dimensional quadratic B-splines and is discretized on mesh nodes (see the inset figure):

\[
B_{ij}(X) = N(u/Lx - i)N(v/Ly - j).
\]

Here, \((i, j)\) denotes a grid index, \( u \) and \( v \) represent the parameterization of \( X \). \( Lx \) corresponds to the spline’s kernel size and the spacing of the 2D grid, and \( N(x) \) is defined as:

\[
N(x) = \begin{cases} 
\frac{3}{4} - x^2, & |x| < \frac{1}{2}, \\
\frac{1}{4}(|x|^2 - |x|)^2, & \frac{1}{2} \leq |x| < \frac{3}{2}, \\
0, & \frac{3}{2} \leq |x|.
\end{cases}
\]

We decouple the three dimensions of the ambient space of \( \Omega \), meaning that the complete sparse basis matrix \( \mathcal{P} = \mathcal{B} \otimes I_3 \in \mathbb{R}^{M \times 3N} \) is represented by the Kronecker product between the spline basis matrix \( \mathcal{B} \in \mathbb{R}^{N \times M} \) for scalar functions and the 3D identity matrix, where \( M \) is the number of bases and \( N \) is the number of vertices. Given a reference state of mesh position \( x_0 \), we express the states in the subspace as \( \{x : x = x_0 + \mathcal{P}u, y \in \mathbb{R}^{3M}\} \). The displacement w.r.t. the reference state is constrained within the linear space expanded by the bases in \( \mathcal{P} \). This basis satisfies the partition of unity property and is \( C^1 \)-continuity w.r.t. the parameterization space.

Bending on Stitch. If decompositions are necessary to achieve non-overlapping parameterization, such as for a tube-shaped cloth, we propose an approach to incorporate bending energy on the artificially generated stitches. As depicted in Figure 3, assume an artificial stitch passing through the shared edge of two triangles, namely \( (v_1, v_2, v_3) \) and \( (v_4, v_5, v_6) \). We introduce two bending energies, each with half of the original bending stiffness, on the 4-stencils \( (v_1, v_2, v_4, v_5) \) and \( (v_2, v_3, v_5, v_6) \). By doing so, the original energy is now split into two parts. As the stitch penalty pulls the vertices \( v_2 \) and \( v_5 \), as well as \( v_3 \) and \( v_6 \), closer together, the combined energy of these two parts will recover the energy prior to the decomposition.

4.3 Subspace Projective Dynamics

In each global step, as described in Equation 2, we need to solve a linear system \( Hu = b \), where \( u \) is the displacement increment w.r.t. the previous global step \( x^k \). To optimize the quadratic energy
We follow the same strategy of Xie et al. [2023] and incorporate \( \text{we employ subspace BFGS iterations to minimize the quadratic} \) within the subspace, we can restrict the displacement to the form \( u = \mathcal{P}y \). This leads to solving a reduced-order global system

\[
\mathcal{P}^T \mathcal{H} \mathcal{P} y = \mathcal{P}^T b.
\]  

(8)

Here, the reduced-order global system matrix \( \mathcal{H} = \mathcal{P}^T \mathcal{H} \mathcal{P} \) has a dimension that corresponds to the number of bases in \( \mathcal{P} \). This dimension can be much smaller than the total number of degrees of freedom. The advantage of this smaller matrix is that it can be prefactorized using the Cholesky decomposition, enabling efficient reuse for backsolving of Equation 8.

By exclusively solving the global step increment within the subspace, we effectively address the low-frequency modes that primarily govern the overall motion of the cloth. However, this approach tends to overlook the high-frequency details that showcase the benefits of high-resolution meshes. To reintroduce the high-frequency modes, we employ several Jacobi iterations on the original global system \( \mathbf{H}u = b \), starting from the displacement obtained through the subspace solution. This two-level scheme bears a resemblance to a two-level multigrid method, wherein \( \mathcal{P} \) and \( \mathcal{P}^T \) can be perceived as the restriction and prolongation matrices, respectively. Nevertheless, due to the necessity of a local projection step to update the membrane rotation reference, the right-hand side of the global linear system undergoes changes.

### 4.4 Subspace-Preconditioned Projective Dynamics with Contact

We follow the same strategy of Xie et al. [2023] and incorporate contact in a specifically designed time-splitting manner. At the beginning of each time step, we introduce a quadratic proxy of the contact potential into the PD solver, allowing for penetration. At the beginning of each time step, we introduce a quadratic proxy of the contact potential into the PD solver, allowing for penetration.

**4.4.1 PD with Quadratic Contact Proxy.** The quadratic proxy is defined as the second-order expansion of the barrier potential at the initial state \( x^* \) of the current time step:

\[
\tilde{B}(x; x^*) = B(x) + \nabla B(x^*)^T (x - x^*) + \frac{1}{2} \|x - x^*\|^2_{\nabla^2 B(x^*)}.
\]  

(9)

By combining this quadratic proxy, the global system of the PD solver is still linear:

\[
(H + \nabla^2 B(x^*))u = b - \nabla B(x^*) - \nabla^2 B(x^*)(u + x^k - x^*),
\]  

(10)

where \( Hu = b \) represents the global step system without contact.

However, the global system matrix is subject to change over time. To reuse the prefactorized matrix obtained without contact, we employ subspace BFGS iterations to minimize the quadratic energy of the global step within the subspace, i.e., we solve the quadratic problem with the increment constrained to the subspace. It is important to note that the subspace global matrix is now:

\[
\mathcal{P}^T (H + \nabla^2 B(x^*)) \mathcal{P} = \mathcal{P}^T \mathcal{H} \mathcal{P}^T + \mathcal{P}^T \nabla^2 B(x^*) \mathcal{P}^T.
\]  

(11)

The prefactorized matrix \( \mathcal{P}^T \mathcal{H} \mathcal{P} \) can serve as the initial approximation of the Hessian for the subspace BFGS iterations at the beginning of each time step. Importantly, the global system matrix remains unchanged within a single time step, allowing for the reuse of BFGS history across different global steps within the time step. To ensure efficiency, we limit the number of BFGS iterations to 2 in each global step, effectively providing only one additional 2-rank update to the initial reduced-order Hessian matrix based on previous updates. For quadratic problems, the optimal step size for line search can be computed analytically, requiring only one matrix-vector multiplication. It is worth mentioning that at the beginning of the local-alternations, we solve one reduced-order global step, where one backsolve using the prefactorized subspace global system can give us the exact solution. This initial guess significantly decreases the number of PD iterations.

Following the subspace BFGS iterations, we perform 5 block-diagonal Jacobi iterations on the original full-order linear system (Eq. 10) to enrich high-frequency details in the solution. This choice aligns with the common practice of multigrid, which incorporates 3–5 smoothing iterations per cycle. The block size is 3 because all matrices are assembled with 3x3 blocks, during which the diagonal blocks are tracked. However, with contact proxy stiffness matrix, the eigenvalue of the iteration matrix can easily exceed 1. Here we use a modified Jacobi with automatically tuned weight. We observe that each naive Jacobi iteration is essentially a block-diagonal preconditioned gradient descent for the corresponding quadratic problem. The steepest descent step size can be analytically computed as discussed above. We use that step size as the weight for each Jacobi iterations. This can make sure that the energy for the global step in Equation 2 is always decreasing.

In summary, each global step of our solver consists of 2 L-BFGS iterations and 5 Jacobi iterations. The PD phase is terminated if the \( L \)-infinity distance between states from two consecutive global steps falls below a given tolerance \((5 \times 10^{-3})h \) in our experiments with time step size \( h \), or if the maximum number of iterations is reached \((200 \) in our experiments). Additionally, we apply Chebyshev acceleration to the global steps, following Liu et al. [2017]; Wang [2015], to accelerate convergence. Empirically, we found that Chebyshev weight 0.99 worked well across all our examples.

**4.4.2 Penetration Correction.** The above PD solver may provide a trial solution \( x^T \) with penetrations. To resolve these penetrations, we solve the following energy minimization using projected Newton method combined with a non-penetration line search, following the approach used in the IPC framework:

\[
\frac{1}{2k^2}\|x - x^T\|^2_M + B(x).
\]  

(12)

This objective function aims to find a balance between the trial state from PD and the collision constraints. Unlike the original IPC, which solves the linear system in the Newton method with Cholesky factorization, we employ a block-diagonal preconditioned conjugate gradient (PCG) method to solve the system. On GPUs,
iterative solvers are generally much faster than direct solvers. Furthermore, we use a smaller contact stiffness than the quadratic proxy. The stiffness for the proxy is slightly lower than the elasticity stiffness, while in penetration correction, the value is adjusted to 0.01\(x\). This adjustment can reduce the condition number of the nonlinear optimization problem, facilitating faster convergence.

We also propose an early-stop strategy to reduce the number of Newton iterations required in this step. The objective of this phase is to resolve collisions while preserving momentum as much as possible. By observing this, we can increase the tolerance for DOFs that are involved in contacts, while focusing on preserving momentum primarily for DOFs that are not in contact. By prioritizing momentum preservation for non-contact DOFs and allowing for a slightly larger tolerance for contact DOFs, we can reduce the number of Newton iterations required in the penetration resolution step while still achieving satisfactory results. In our experiments, the tolerances on non-contact DOFs and contact DOFs are \(10^{-2}\) and \(10^{-1}\) respectively.

4.4.3 Friction. We also introduce an approach designed specifically for our time-splitting contact model incorporating frictional effects. This model utilizes the Hessian of the frictional energy as a damping matrix. We note that the friction coefficient \(\mu\) no longer holds a physical meaning but still controls the magnitude of the frictional forces. In this friction proxy, we expand the frictional contact potential at \(x_k = x_k^* + 10\mu e_v s_v/k_s\), and remove the contributions of tangential velocities smaller than \(e_v\). That is, we only adopt dynamic friction Hessians for damping along the tangential directions. Our fuzzy friction proxy can be seamlessly integrated into the contact proxy and incorporated into the PD solve loops using the BFGS algorithm.

5 GPU IMPLEMENTATION

Our algorithm has been implemented to run efficiently on a single GPU using CUDA 12.1. To avoid write-write conflicts during the assembly of global gradients and Hessian matrices, we did not utilize coloring algorithms like the one proposed in Fratarcangeli et al. [2016]. Instead, we found that using atomic add operations on our GPU was already efficient enough and simpler to implement. For matrix-vector products in the Jacobi solver, CG method, and subspace restriction/prolongation, we employed cuSPARSE, a library for efficiently performing operations on compressed sparse row (CSR) matrices in CUDA. In addition, the dense Cholesky factorization of \(P^T H P\) was implemented using cuSOLVER, which enabled efficient factorization of the reduced-order global matrix. At each Newton iteration during the penetration correction, CCD is required on each search direction to guarantee non-penetration. We use the patch-based GPU collision culling from Lan et al. [2022b] to efficiently reduce the number of candidates.

6 EXPERIMENT

We implemented our algorithm on a desktop workstation with an NVIDIA RTX 3090 GPU and an Intel Core i9-10920X 3.5-GHz CPU with 12 cores. We also follow [Macklin et al. 2019] using simulation substeps for optimized performance.

6.1 Compare with Chebyshev-Accelerated Jacobi-PD

We compare our method with [Wang 2015], a classic GPU-accelerated PD algorithm using Jacobi method to solve the global system exactly. In this test, we simulate a piece of table cloth (250K vertices, 500K triangles) with two upper corners fixed. Here, we exclude collision and self-collision processing in both methods to only showcase the performance-wise difference. For the method described in Wang [2015], we performed three separate experiments with 1,000, 2,000, and 3,000 Jacobi iterations, respectively. We also simulate the scene using Newton’s method as the reference. The results of the comparison are illustrated in Figure 6. Even with 3,000 iterations, Wang [2015] exhibits more discrepancies with Newton’s results compared to our method. Furthermore, our method demonstrates faster computation times compared to Wang [2015] when utilizing only 1000 iterations, despite the presence of artifacts in their results.

6.2 Benchmarks

We further compare our method with two known IPC-based cloth simulation methods, namely CIPC [Li et al. 2021] and PD-IPC [Lan et al. 2023], in multiple high-resolution cloth simulation setups. The original CIPC implementation was on the CPU, which is quite expensive. To avoid misleading benchmarks from different platforms, we re-implemented CIPC on the GPU, and we refer to our own implementation as GPU CIPC. GPU CIPC port most costly computations to CUDA including collision detection, culling, CCD, and Newton solve, and it already exhibits notable performance gains compared to its CPU counterpart. Nevertheless, our method (proposed in this paper) offers further speedups. PD-IPC is our closest competitor as it is also based on the PD. However, they utilize simplified formulations for the membrane and bending energies.

Figure 4: Ribbons. Twenty-one long ribbons are dropped into a round bowl, leading to numerous collisions and self-collisions among the ribbons.
compared to our method. We made best efforts to visually match their results with ours under the same numerical settings.

In the comparative analysis, we used a frame duration of 0.04s. We try our best to tune the time step size to ensure a fair comparison of performance among different methods. In the case of CPU IPC, the most efficient time step size is typically the frame duration. This is because the direct solver used in CPU IPC is not sensitive to the condition numbers of the Hessian matrix. However, for most iterative methods, including ours, substepping is beneficial as it is more dependent on the condition numbers of the problem. The per-frame computational costs of different approaches are listed in Table 1.

### Table 1: Average computational cost per frame (s) in the comparisons.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>#V</th>
<th>#Basis</th>
<th>#Step</th>
<th>Ours</th>
<th>PD Step</th>
<th>GPU IPC</th>
<th>CIPC</th>
<th>PD-IPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ribbons</td>
<td>285K</td>
<td>3e-4</td>
<td>7119</td>
<td>16</td>
<td>46</td>
<td>43.3%</td>
<td>1603</td>
<td>1579</td>
</tr>
<tr>
<td>Cloth on sphere</td>
<td>252K</td>
<td>1e-4</td>
<td>8427</td>
<td>8</td>
<td>27</td>
<td>37.2%</td>
<td>241</td>
<td>2040</td>
</tr>
<tr>
<td>Funnel</td>
<td>232K</td>
<td>5e-4</td>
<td>9432</td>
<td>10</td>
<td>23</td>
<td>63.3%</td>
<td>362</td>
<td>2740</td>
</tr>
<tr>
<td>Reef knot</td>
<td>104K</td>
<td>3e-4</td>
<td>7665</td>
<td>8</td>
<td>10</td>
<td>81.9%</td>
<td>90</td>
<td>144</td>
</tr>
</tbody>
</table>

Figure 5: Funnel. Three pieces of cloth are dropped onto a shallow funnel. A sphere pushes the layers of cloth completely through the hole of the funnel.

Figure 6: The comparison between our method and Wang [2015] on a cloth hanging experiment. The cloth used in the experiment consists of 250K vertices and 500K triangles. Our method demonstrates closer agreement with the results obtained using Newton’s method, but is much faster.

**Ribbons (Figure 4).** We simulate the behavior of 21 long ribbons dropped into a bowl. The ribbons interact with each other, resulting in multiple collisions and self-collisions. In this example, we achieve 22× acceleration compared to GPU CIPC, 34× acceleration compared to CPU CIPC, and 2.9× acceleration compared to PD-IPC. We note that PD-IPC utilizes simplified strain and bending models. In this example, it is not able to capture fine-detailed wrinkles, resulting in fewer contact interactions compared to our method.

**Cloth on sphere (Figure 7).** A square cloth is dropped onto a sphere, with a ground surface located beneath the sphere. There are persistent contact between the center of cloth with the top of sphere, and persistent contact between the cloth and the ground. The contact results in numerous intricate wrinkles. We achieve 8.9× acceleration compared to GPU CIPC, 75× acceleration compared to CPU CIPC and 1.8× acceleration with PD-IPC.

**Reef knot (Figure 9).** Two curved ribbons are initially intertwined and then pulled in opposite directions to form a knot. It is worth mentioning that, to apply our method, each ribbon is decomposed into three pieces combined with stitch bendings, as depicted in Figure 3. Our method manages to generate a tiny and tight knot. And we achieve 9× acceleration compared to GPU-IPC, 14× acceleration compared to CPU IPC, and 1.8× acceleration compared to PD-IPC.

### 6.3 Ablation Study

#### 6.3.1 Convergence Under Refinement

To evaluate the accuracy of our cloth solver, we perform a convergence under refinement test w.r.t. the time step size \( h \). We begin by capturing a contact-rich snapshot of the ‘cloth on sphere’ experiment and using it as the initial state. We then simulate with CIPC using \( h=1e^{-4} \) for a duration of 0.1 seconds. The final state obtained from the CIPC simulation serves as the reference. We then use our cloth solver to simulate with...
6.3.2 Number of Spline Bases.

The rate of convergence in the PD phase depends on the number of spline bases. In our analysis, we take a contact-rich snapshot and focus on a single substep. The outcome from the CIPC solver serves as the reference state, against which we evaluate the convergence of our BFGS-based PD solver across varying numbers of spline bases. In our framework, the number of spline bases is controlled by the spacing between spline centers. As depicted in the inset figure, a smaller spline spacing $\Delta x$ typically leads to less iterations for convergence. However, extremely small $\Delta x$ values can result in too much computational time of the backsolve step on GPU. We estimated that this component’s timing is approximately $O(\frac{1}{\Delta x^3})$. Despite efforts, we haven’t identified an optimal empirical compromise between timing and PD convergence speed. Nevertheless, maintaining the number of bases slightly below 10K for a 100K-vertex cloth tends to yield favorable overall speedups.

6.5 Comparisons to Hyper-Reduced Projective Dynamics

Our work differs from hyper-reduced PD (HRPD) [Brandt et al. 2018] in several key aspects. The subspace design in our method is tailored for cloth and shell structures, whereas HRPD targets volumetric solids. Additionally, we solve the full-order system, while HRPD directly simulates within the reduced subspace. Furthermore, the B-spline bases in our approach inherently satisfy partition of unity on regular grids. This is advantageous for simulating cloth, as HRPD bases lack this property. To demonstrate, we compare HRPD to our method by directly simulating the subspace dynamics without Jacobi relaxation (ours used 1200 bases, while HRPD used 1497 bases). As shown in Fig. 10, the hyper-reduced subspace exhibits severe locking under large deformations, while our method does not.

6.6 Controllable Friction

In Figure 11, we present an experiment where we vary the friction coefficients between the cloth and the slope. The cloth used in the experiment is a long rectangle with 120K vertices, and it falls onto the slope under gravity. By increasing the friction coefficient, we observe that the cloth slides down the slope at a slower speed until it gets stuck and then undergoes turning motions.

6.7 Garment Animation

Fine-detailed garment simulations play a crucial role in the animation industry. In the animation pipeline, artist-designed character animation sequences serve as moving boundary conditions for the cloth simulations. However, simulating garments on animated characters presents significant challenges due to the dramatic motions involved, such as running, jumping, or dancing. Here we use a challenging test case from Li et al. [2021], where a character turns
Subspace-Preconditioned GPU Projective Dynamics with Contact for Cloth Simulation

7 CONCLUSION

In this paper, we propose an efficient cloth simulation method based on the projective dynamics (PD) framework. Our method combines subspace integration and parallelizable iterative relaxation techniques to effectively reduce both high-frequency and low-frequency residuals, leading to significantly improved convergence. We seamlessly integrate our method with the state-of-the-art contact handling framework, IPC, to ensure interpenetration-free results in a time-splitting manner. We have shown that our method has significant performance improvements over existing GPU solvers for high-resolution cloth simulation.

Indeed, when dealing with objects exhibiting high speeds, the time splitting error can become significant, leading to amplified damping effects. To address this issue, it would be valuable to explore adaptive substepping techniques that can enhance the accuracy of the time splitting process and alleviate the undesired damping artifacts.

Furthermore, the use of Newton’s method in the penetration correction step may give rise to overshooting problems, resulting in excessive optimization iterations. Notably, the penetration correction step typically consumes a substantial portion of the computation time. To further improve the overall performance of our algorithm, it is crucial to investigate dedicated solvers specifically tailored for the penetration correction step.

ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their valuable feedback. We thank Pingying Chen for narrating the video. This work is partially supported by NSF 2153851, 2153863, 2023780, 2301040, 2008915, 2244651, and 2008564.

REFERENCES


