

Mil2: Efficient Cloth Simulation Using Non-distance Barriers and Subspace Reuse

LEI LAN, University of Utah, USA

ZIXUAN LU, University of Utah, USA and Institute of software, Chinese Academy of Sciences

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

HUAMIN WANG, Style3D Research

CHENFANFU JIANG, UCLA, USA and Style3D Research

YIN YANG, University of Utah, USA and Style3D Research



Fig. 1. **Fashion show.** We present a cloth simulation framework that can be conveniently parallelized on the GPU. We name this new pipeline *Mil2* as it is able to simulate high-resolution cloth meshes (e.g., around one **million** degrees of freedom) at an interactive rate (in **milliseconds**). *Mil2* offers new insights into cloth and deformable body simulation. With a non-distance-based barrier formulation, we save a lot of computations of traditional CCD with a 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. *Mil2* also features a residual forwarding trick to alleviate the damping issues generated by 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 1/200 sec, *Mil2* runs at 5.4 FPS.

Mil2 pushes the performance of high-resolution cloth simulation, making the simulation interactive (in milliseconds) for models with one million degrees of freedom (DOFs) while keeping every triangle untangled. The guarantee of being penetration-free is inspired by the interior-point method, which

Authors' addresses: Lei Lan, University of Utah, USA; Zixuan Lu, University of Utah, USA and Institute of software, Chinese Academy of Sciences; 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; Huamin Wang, Style3D Research; Chenfanfu Jiang, UCLA, USA and Style3D Research; Yin Yang, University of Utah, USA and Style3D Research.

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 ACM 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.

© 2024 Association for Computing Machinery.

0730-0301/2024/4-ART \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

converts the inequality constraints to barrier potentials. Nevertheless, we propose a major overhaul of this modality by defining a novel and simple barrier formulation which does not depend on the distance between mesh primitives. Such a non-distance barrier model allows a new way to integrate collision detection into the simulation pipeline. Another contributor to the performance boost comes from the so-called subspace reuse strategy. This is based on the observation that low-frequency strain vibrations are 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 nearly one order while keeping the entire simulation penetration-free and producing high-equality animations of high-resolution models.

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

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

ACM Reference Format:

Lei Lan, Zixuan Lu, Jingyi Long, Chun Yuan, Xuan Li, Xiaowei He, Huamin Wang, Chenfanfu Jiang, and Yin Yang. 2024. Mil2: Efficient Cloth Simulation Using Non-distance Barriers and Subspace Reuse. *ACM Trans. Graph.* 1, 1 (April 2024), 15 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Being one of the most researched problems in computer graphics, 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 unleash their talents and inspiration 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 needed, and the increased number of degrees of freedom (DOFs) further stresses the simulation performance. This paper presents a GPU-based cloth simulation framework that is one or even two orders faster than the state-of-the-art GPU simulation algorithms for high-resolution cloth models. We name our method Mil2 suggesting it enables an interactive frame rate (in milliseconds) for simulation scenes of a million DOFs.

Efficient processing collision and self-collision of cloth is the pivotal concern for a high-performance cloth simulator. We are inspired by the recent success of the barrier method a.k.a. incremental potential contact or IPC [Lan et al. 2022b; Li et al. 2020, 2021], and choose it as the primary modality for contact handling. IPC 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 in cloth animations. As a result, the most dominant computation in many cloth simulators becomes the CCD processing. To address this issue, Mil2 uses a novel barrier formulation, which does not depend on the actual distance between primitives. Instead, the barrier energy is defined based on the virtual life span of a collision event – the “longer” a constraint/contact persists during the nonlinear solve, the stiffer it should be. Because of that, 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 techniques, which project the system into a prescribed kinematic subspace. The real difficulty comes from the nonlinearity of the system matrix, which varies under different cloth poses, and constructing a subspace for each pose is not an option. To this end, Mil2 employs 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.

Mil2 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, Mil2 is a parallel simulator integrating several novel features to address the core challenges in cloth simulation. By deeply optimizing the pipeline, Mil2 further pushes the quality and efficiency of cloth simulation. Thanks to the barrier method, 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. Our experiments show that Mil2 delivers high-quality animation results while being one order faster than existing methods. Just as suggested by its naming, Mil2 enables an interactive frame rate (in milliseconds) for high-resolution cloth simulations (of a million DOFs).

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. The cloth is regarded as a codimensional object [Terzopoulos et al. 1987], whose thickness is much smaller than its other two dimensions. A common practice 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; 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. Straining 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 FEM. One can fine-tune 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 for cloth. Due to the high surface-to-volume ratio, collision handling has always been an indispensable part of cloth simulation. Based on CCD, Provot [1997] and Bridson et al.

[2002] suggest a post-simulation step grouping penetrating vertices into impact zones. Huh et al. [2001] decomposes the impact zone into smaller colliding clusters based on their positions on the original mesh. Harmon et al. [2008] relaxes impact zones with an inelastic projection, allowing relative tangential movement. CCD is known to be expensive – it solves a cubic function for the first time of impact (TOI). The analytic formulation of cubic function is available but error-prone [Wang et al. 2021]. Numerical root finding is therefore preferred [Wang 2014; Yuksel 2022], which is clearly more costly.

In time-sensitive cases, discrete collision detection (DCD) may be used as a cheap alternative. For collisions between cloth and volumetric colliders (e.g., the human body), the penalty force based on projection [Guan et al. 2012] or signed distance field (SDF) [Chen et al. 2013] are popular choices. Buffet et al. [2019] extends the implicit field of volumetric objects to open surfaces to resolve the inter-penetration of multiple-layer cloth. For collisions between cloths, Baraff et al. [2003] and Wicke et al. [2006] employ an untangling method that applies repulsion forces to minimize the colliding region. Volino and Magnenat-Thalmann [2006] and Ye and Zhao [2012] minimize the length of the collision contour of the colliding region.

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 cloths/thin shells [Li et al. 2021], 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 within the feasible region. Lan et al. [2022b] dynamically approximates 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 and solve them in parallel. To reduce the cost of CCD, Wu et al. [2020] uses point-point distance constraints between triangle pairs to avoid intersection.

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 a.k.a. constraint-based methods. One can locally and inexactly solve those constraints via the so-called constraint projection [Goldenthal et al. 2007]. 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 becomes 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]. In cloth simulation, our human perception is sensitive to the richness of local details. Given limited computation time, we often opt for incorporating more DOFs over more complicated materials for improved visual plausibility.

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]. An et al. [2008] present a data-driven approach to allow the integration to be performed only at a few sample elements (cubature). 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. Reduced models need expensive pre-computations such as computing the generalized eigendecomposition or training a deep network. While some efforts have been made to speed up such pre-computations [Yang et al. 2015], any changes to the simulation setup e.g., the boundary condition, nullify the pre-computation. On the contrary, Mil2 employs a novel subspace reuse method – the subspace constructed at the rest shape remains effective, and one only needs incremental computations for the subspace integration.

Multigrid method [Trottenberg et al. 2001] is another common technique to boost simulation efficiency when a large number of DOFs are 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] utilizes a B-spline subspace, and Tamstorf et al. [2015] built the subspace by QR decomposition on near-kernel components. Though most works treat the multigrid method as a linear solver, Wang et al. [2018] integrates multigrid into a nonlinear optimization process, which updates the residue and system matrix periodically. Mil2 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 i.e. [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:

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

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

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

as a known vector based on the previous displacement x^* , velocity \dot{x}^* , and an external force f_{ext} . M is the mass matrix, and h is the time step size. The target 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:

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

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 is followed by local projections as a standard linear solve:

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

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, 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. 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:

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

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] showed that IPC can also be integrated in LG iterations by setting w_i of the collision constraint as $B(d_i)$.

4 NON-DISTANCE BARRIER

While IPC and its variations 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 that 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 (see [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 non-linear penalty, which becomes stiffer when the collision constraint is about to be violated. It geometrically approximates the indicator functions ($\delta_{\mathcal{A}}(x) = 0$, if $x \in \mathcal{A}$; $\delta_{\mathcal{A}}(x) = \infty$, if $x \notin \mathcal{A}$). However, 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 sufficiently close to zero (the complementary slackness is fully satisfied). In practice, what we want is an increasingly strong repulsion to correct the constraint violation. Given the fact that one always needs multiple LG iterations to solve the barrier-in-the-loop PD problem, we define an exponential barrier (EXP) based on the *how hard to resolve a penetrating primitive*, and use EXP as w_i of this constraint:

$$w_i = B_i^{\text{EXP}} = kA^a, \quad (7)$$

where $a \in \mathbb{Z}^+$ represents how many *consecutive LG iterations* has the constraint been active. k is an initial weight, and A is the base of the growth rate. Eq. (7) suggests B_i^{EXP} quickly approaches ∞ if a collision constraint C_i remains unresolved over few LG iterations. When this is the case, the local target position of the constraint shall be satisfied with the highest priority to resolve the corresponding collision. B_i in Eq. (7) 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.

4.1 Partial CCD

An immediate benefit of EXP is relieved CCD computation. It is easy to see from Eq. (7) that updating $B(x)$ only checks if a collision remains active after the previous LG iteration i.e., a true-or-false answer suffices, not the exact TOI. This feature allows a simplified approach for EXP CCD, which converts the cubic root-finding problem to multiple dot products, and we name this method *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:

$$p_1^0 + t^* (p_1^1 - p_1^0) = p_2^0 + t^* (p_2^1 - p_2^0), \quad (8)$$

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:

$$(p_2^1 - p_1^1) \cdot (p_2^0 - p_1^0) = \frac{t^* - 1}{t^*} (p_2^0 - p_1^0) \cdot (p_2^0 - p_1^0) \leq 0. \quad (9)$$

Eq. (9) suggests a non-positive inner product of $(p_2^1 - p_1^1) \cdot (p_2^0 - p_1^0)$ 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:

$$Q(\lambda) = (p_2^1(\lambda) - p_1^1(\lambda)) \cdot (p_2^0(\lambda) - p_1^0(\lambda)). \quad (10)$$

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 $0 \leq \lambda_1, \lambda_2 \leq 1$. Let Ω_λ be the domain of the query function Q . It is easy to see that Ω_λ forms a triangle $\lambda_{1,2} \geq 0, \lambda_1 + \lambda_2 \leq 1$ 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 $\mathcal{S} = \{\lambda_{(0)}, \lambda_{(1)}, \dots\} \subset \Omega_\lambda$ be the set of sample points. We define that local sample interval $\rho(\lambda)$ for any $\lambda \in \Omega_\lambda, \lambda \neq \lambda_{(i)}$ 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: $\rho_S = \max_{\lambda \in \Omega_\lambda} \rho(\lambda)$. Based on Lagrange Remainder Theorem, there exists an upper bound of $Q(\lambda)$ for any $\lambda \in \Omega_\lambda$:

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

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

$$Q(\lambda^*) + \left(\max_{\lambda \in \Omega_\lambda} \|\nabla_\lambda Q\| \right) \|\lambda - \lambda^*\| \leq 0 \Rightarrow \|\lambda - \lambda^*\| \leq \rho = -\frac{Q(\lambda^*)}{\max_{\lambda \in \Omega_\lambda} \|\nabla_\lambda Q\|}. \quad (12)$$

As the cloth is less stretchable, we assume that edge lengths of \mathcal{P}_1 and \mathcal{P}_2 are bounded by L . Under the finite velocity, it is reasonable to assume 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 $\nabla_\lambda Q$:

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

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

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

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

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

That said 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. Sampling-based partial CCD can be considered an *inexact approximate* of exact CCD algorithms [Brochu et al. 2012], given the premise that EXP is in use. Eq. (15) provides a sufficient condition eliminating false negative collision detections. Nevertheless, false positive detections are still possible. Fortunately, the main objective of partial CCD is to estimate if a collision constraint remains alive. False positives do not affect the simulation quality or performance. Mil2 does not exclusively rely on partial CCD, and regular CCDs are still performed during the simulation (but at a much lower frequency). Therefore unless partial CCD is largely inaccurate, simulation performance and quality are not sensitive to partial CCD accuracy (e.g., see Fig. 10).

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. 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.

5 SUBSPACE REUSE

Exp enables a more affordable CCD 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 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 both 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 :

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

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

As U is eigenvectors, $U^T H U = \Lambda$ 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 $U^T (b - HX)$, and diagonalized subspace solve is neglectable. 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 $\tilde{x} = X + Uq$ 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 \tilde{x} :

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

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

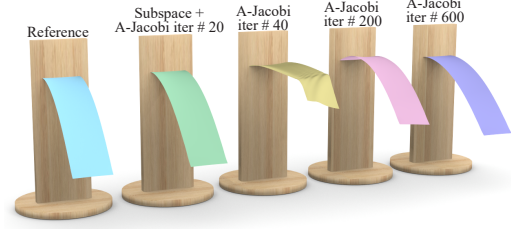


Fig. 2. **Bending beams w. and w/o subspace.** We simulate a collision-free scene where a cloth strip bends under gravity. The model consists of 30K DOFs. The resulting deformation is of low-frequency, which is challenging for Jacobi-like methods. A subspace solve effectively resolves this issue: only 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. Mil2 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.

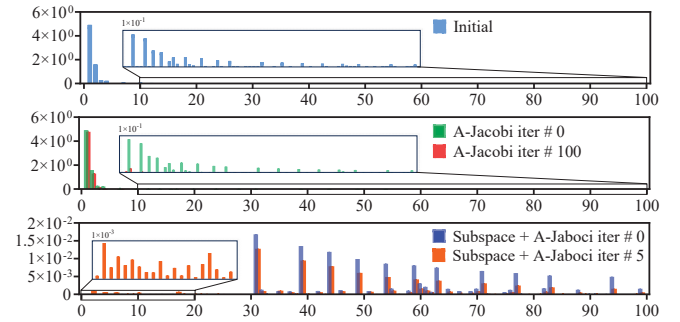


Fig. 3. **Spectral distribution of residual errors (w/o collision).** We plot the distribution of residual error over the first 100 modal bases of the beam test (Fig. 2). As shown at the top, the dominant deformation is low-frequency, which is conveniently solved within the subspace (bottom). On the other hand, A-Jacobi iterations are not effective in dealing with low-frequency residuals. As shown in the middle, 100 A-Jacobi iterations barely lower the low-frequency errors, while the high-frequency strains are well relaxed.

($r = 30$), it only takes 20 A-Jacobi iterations to converge the simulation to the ground truth. General A-Jacobi 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 SOR-like 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 constantly varies during the simulation. 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

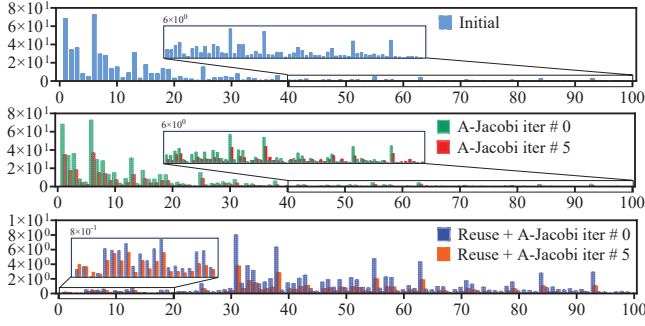


Fig. 4. **Spectral distribution of residual errors (w. collision).** We visualize the distribution of residual errors over the first 100 modal bases of the rest-pose global matrix H when the tablecloth covers the Armadillo (as shown in Fig. 5). Subspace reuse does generate some low-frequency errors,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

still be effective while the matrix is modified? This is because low-frequency modes built at the rest shape depict general and global cloth motions. Those motions are not strongly coupled with high-frequency deformations. For example, the distinct geometries of local wrinkles are influenced less by the overarching movements of the cloth and more by the specific configurations of the corresponding collisions. Interestingly, it is the low-frequency deformations that most significantly 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 $H + \Delta H$ i.e., ΔH is the modification of H induced by collisions or self-collisions. Despite the matrix change, we stick with rest-shape subspace and solve q out of the following reduced global-stage system:

$$U^T(H + \Delta H)Uq = (\Lambda + U^T\Delta HU)q = U^T(b - HX - \Delta HX). \quad (18)$$

Solving the above system is slightly more expensive than Eq. (16) since $(\Lambda + U^T\Delta HU)$ 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. 5 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.



Fig. 5. **Subspace reuse.** A piece of tablecloth drops on a wooden Armadillo. There are 66K DOFs on the cloth, and over 30% of them are involved in collision constraints.

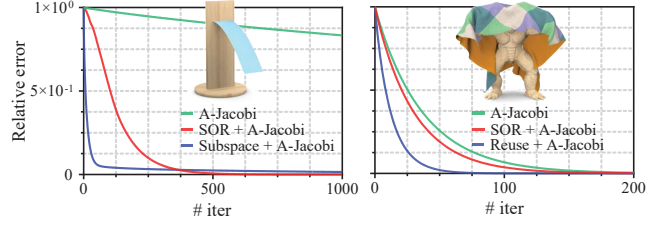


Fig. 6. **Convergence curves w. and w/o subspace (reuse).** We plot the convergence curves for experiments in Figs. 2 and 5. We can see subspace solve saves a large fraction of A-Jacobi iterations even under intensive collisions. The computation cost of one subspace solve, on the other hand, is similar to performing one or two A-Jacobi iterations on average.

5.1 Pre-computed subspace update

The most expensive computation however is the subspace projection of ΔH i.e., evaluating $U^T\Delta HU$. The complexity is $O(N^2r)$ 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 unique structures of $U^T\Delta HU$. First, we set $A_i = B_i = \text{Id}$ in Eq. (3) as an identity matrix (Id) for local projection of collision constraints. Intuitively, doing so is equivalent to deploying a nonlinear spring pulling a penetrating vertex to its target (collision-free) position. With EXP, the strength of the spring grows exponentially after each LG iteration if the vertex keeps penetrating. It is immediately noticed that ΔH becomes a diagonal matrix under this treatment.

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

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

That said 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 $U^T\Delta HU$ 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 till the stopping criterion is satisfied. After an LG iteration, partial CCD ensues, and w_i of the collision constraints are updated per Eq. (7). Nevertheless, our subspace reuse scheme can efficiently calculate the updated (reduced) global matrix and allows Mil2 to enjoy the advantages of both subspace solvers and iterative solvers with minimum computational costs. Such combined efficiency and convergence are unseen in previous GPU algorithms (e.g., see Fig. 6). 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: $\Delta x \leftarrow \alpha t_{\text{TOI}} \cdot \Delta x$. If the TOI is a smaller quantity, severe damping or locking artifacts could be produced as a significant 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^* of Eq. (5), which ideally should possess a vanished gradient $\nabla E(x^*) = 0$. By the end of a time step, if LG iterations fail to fully converge, the resulting x^* is different from x^* such that $x^* = x^* + \delta x$. The gradient of the variational energy $\nabla E(x^*) = f_r \neq 0$ represents unbalanced residual forces in the system. RF seeks a kind of virtual force δf at the next time step to mitigate the damping artifacts induced by δx . That said, Eq. (2) becomes:

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

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

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

Note $\dot{x}^* = \dot{x}^* + \frac{\delta \dot{x}}{h}$ 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^* as:

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

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

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

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

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

Taking a closer look, it is noted that computing $(\nabla^2 E(x^*))^{-1} f_r$ 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 barriers is k i.e., a 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 severe artifacts. Honestly, there is nothing much we can do if the application aggressively restrains the available time budget for simulation since Eq. (5) is highly nonlinear by itself. Conceptually, RF moves some computation e.g., solve for

$(\nabla^2 E(x^*))^{-1} f_r$ 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.

ALGORITHM 1: Mil2 simulation pipeline.

```

1:  $z \leftarrow x^* + h\dot{x}^* + h^2(M^{-1}f_{\text{ext}} + \delta f)$  //  $\delta f$  is from RF
2:  $x \leftarrow z$ 
3:  $\|\Delta x\| \leftarrow \infty$ 
4: while  $\|\Delta x\| < \epsilon_{\text{initial}}$  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:  $\langle B, t_{\text{TOI}} \rangle \leftarrow \text{FullCCD}(x^-, x^+)$  //  $B$  is the latest barrier
10:  $x \leftarrow x + t_{\text{TOI}} \cdot \Delta x$ 
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:    $\langle B, t_{\text{TOI}} \rangle \leftarrow \text{FullCCD}(x^-, x^+)$  // update  $B$  at each outer loop
19: end
20:  $x \leftarrow x + t_{\text{TOI}} \cdot \Delta x$  // exiting line search filtering
21: if  $t_{\text{TOI}} < \epsilon_{\text{TOI}}$  then
22:    $B_i = \kappa$  // quadratic approximation of  $B$ 
23:   while  $\|\Delta x\| < \epsilon_{\text{inner}}$  do
24:      $\delta x \leftarrow \text{ModalLGReuse}(I + \Psi + B)$ 
25:   end
26:    $\delta f \leftarrow \frac{2\delta x M}{h^2}$ 
27: end

```

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 eigendecomposition of the rest-shape global matrix H , which is agnostic on the collision or barrier constraints. The eigendecomposition computes \bar{r} eigenvectors corresponding to the smallest eigenvalues, and we typically set \bar{r} as a few hundred i.e., $\bar{r} > r$. Because the rest-shape modal global matrix $\Lambda = U^T H U$ is diagonal, and the subspace projection is efficient on GPU, increasing the dimensionality of U from r to \bar{r} does not induce extra computational costs but makes the system solve (w.o. collision) more effective. The first $r < \bar{r}$ eigenvectors form the set of basis vectors 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

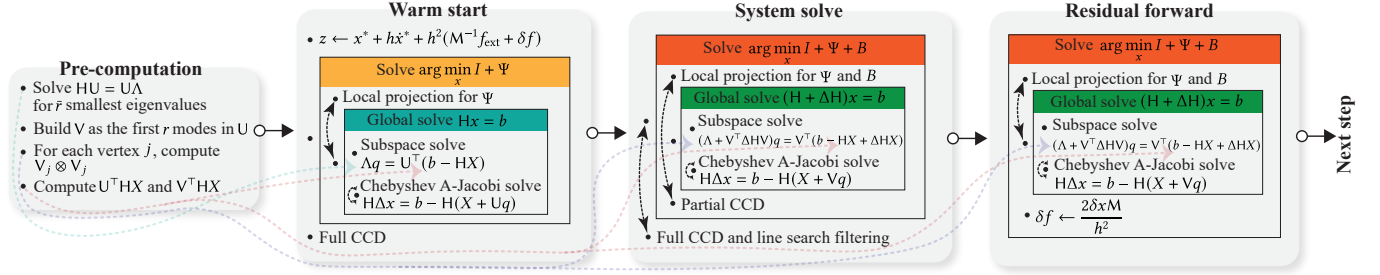


Fig. 7. **Algorithm overview.** Mil2 is an efficient cloth simulation pipeline. It leverages a subspace reuse technique to substantially improve the convergence of GPU-based iterative solvers. The costly CCD is also replaced with a non-distance-based barrier formulation i.e., EXP. The pipeline also features 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 EXP-in-the-loop optimization. Each time step consists of several blocks of LG iterations, which are visualized with curved double arrows.

computation of the updated subspace matrix. Lastly, $U^T HX$ and $V^T HX$ 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 $\Lambda + V^T \Delta H V$ 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 of the spectrum. As a result, expanding V to the high-frequency spectrum is not beneficial. Since we will need to save $V_i \otimes V_i$ for each vertex, a more compact V_i is also more memory-friendly.

Each time step begins with a warm-start computation. Typically, z (as in Eq. (2)) is used as the initial guess of x at the current time step and for setting up the initial barrier constraints B . 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 $\text{span}(U)$ 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 solver works on 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 change of $\|x\|$ between two consecutive outer loops. That said if $\|x\|$ nearly stays the same for two outer loops, we consider the time step is completed. Each inner loop begins with a standard LG iteration. The local projections are performed for both elasticity constraints (Ψ) and collision constraints (B). At the global solve, we re-use the subspace V constructed from H to smooth high-frequency errors and pass the residual to A-Jacobi iterations. The convergence of the inner loop is also based on the variation of $\|x\|$ from the previous inner loop. After the LG is done, the partial CCD routine is 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} 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 Mil2 pipeline 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 Mil2 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 of Mil2 on the GPU.

8.1 Implementation details

Most parts of the Mil2 framework are matrix-free, except for the subspace solve step of $V^T (H + \Delta H) V$. For solving a linear system of $Ax = b$, the most common practice is to pre-factorize A and compute x via forward and backward substitutions. In our implementation however, we directly solve $X \in \mathbb{R}^{r \times r}$ via:

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

to obtain an approximate of the matrix inverse $X \approx A^{-1}/\beta$. 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 $A = V^T (H + \Delta H) V$ 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 . 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

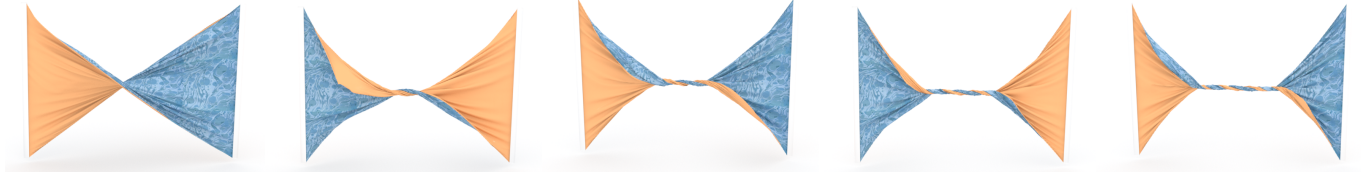


Fig. 8. **Cloth twisting with EXP.** We rotate the table cloth at both ends for 2,880 degrees. The simulation includes over 120K DOFs. Mil2 easily completes this test at 12 FPS. Compared with distance-based barrier i.e., [Lan et al. 2022b], EXP saves about 30% LG iterations (as plotted in Fig. 10).

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 of Mil2. In our implementation, we use rank-2 A-Jacobi as our default GPU solver. A rank-2 A-Jacobi iteration computes 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).

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]. When partial CCD suffices, 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 100K, very few sample points (e.g., three) work well for partial CCD in most simulation scenarios.

8.2 EXP vs DBB

The first test we would like to show is a comparison between the distance-based barrier (DBB) and non-distance barrier i.e., EXP. The major difference lies in the fact that EXP allows the stiffness 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 EXP and DBB for this twisting test, and the plots are shown in Fig. 10. It can be seen that EXP helps reduce iterations, as each collision constraint is more likely to find an appropriate weight during iterations. EXP is not sensitive to the sample density (ρ). To this end, Fig. 10 also provides iteration counts for EXP with different sampling densities. We can see that EXP still works well even using one sample point at the center of Ω_λ .

8.3 Residual forwarding

Mil2 exploits residual forwarding to alleviate the damping and locking artifacts induced by line search filtering under a small TOI. Fig. 9 shows a side-by-side comparison with and without using RF. In this experiment, a piece of tablecloth falls on a teapot. The cloth has 16,641 vertices and 32,768 triangles. We use a small-size line search filtering at the end of each time step by setting $\alpha = 0.1$. The resulting animation suffers from severe damping artifacts since such a close-to-zero TOI freezes the deformation of all the triangles. This observation is highlighted in the figure. Residual forwarding recovers most missing dynamics and produces plausible results. Please refer to the supplementary video for an animated comparison.

8.4 Mil2 vs PD-IPC

Lan et al. [2022b] also combine PD with incremental potential contact (PD-IPC) for deformable and cloth simulation. Compared with Mil2, PD-IPC is based on DBB, and a full CCD must be used every time the constraint set is to be updated. PD-IPC solely relies on A-Jacobi for the global solve. For stiff simulations, one must use weighted A-Jacobi to ensure the iteration does not diverge. This is not a problem for Mil2 since the subspace solve removes most low-frequency errors (e.g., as shown in Figs. 2 and 5). As a result, Mil2 outperforms PD-IPC by a significant margin in general. Fig. 11 shows 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 weight of the A-Jacobi must be set conservatively ($\omega \geq 0.85$), which impairs the convergence. In this example, Mil2 is 12× faster than PD-IPC.

8.5 Mil2 vs BFGS-PD

Another relevant competitor is from a recent contribution by Li et al. [2023] or BFGS-PD. BFGS-PD combines Jacobi iteration with BFGS for global matrix update in a spline-based subspace. BFGS-PD is also based on DBB and uses reduced IPC Hessian for the global-stage solve. BFGS requires a low-rank matrix update at LG iterations, and the B-spline subspace is of high dimension i.e., $r \approx 10,000$. As a result, BFGS-PD is much slower than the pre-computed subspace update used in Mil2. 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. Mil2 runs this example at an interactive rate of 11.8 FPS, while BFGS-PD needs several seconds for each frame.

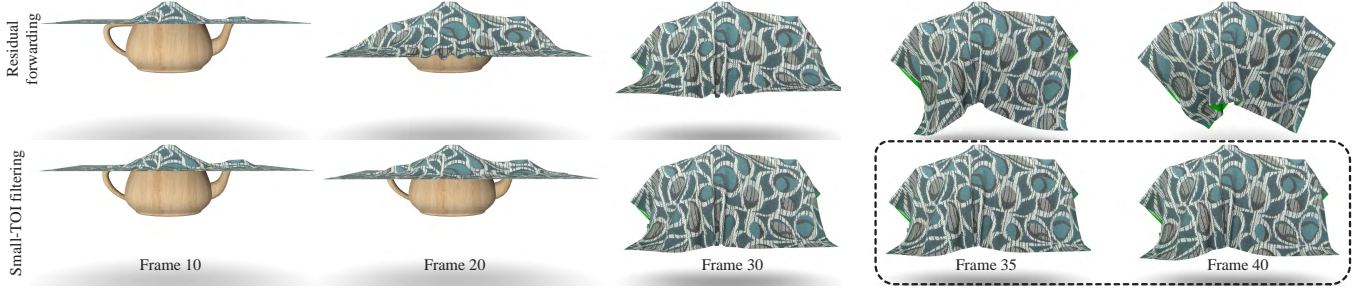


Fig. 9. **Residual forwarding.** We show an experiment where a piece of tablecloth falls on the teapot. If the TOI computed at the end of a time step is small, the corresponding line search filtering truncates the displacement update for all the vertices, leading to locking/damping artifacts (see highlighted snapshots). Residual forwarding estimates a virtual force to restore the dynamics at non-colliding vertices in the next step. Therefore, the animation looks more natural and realistic.

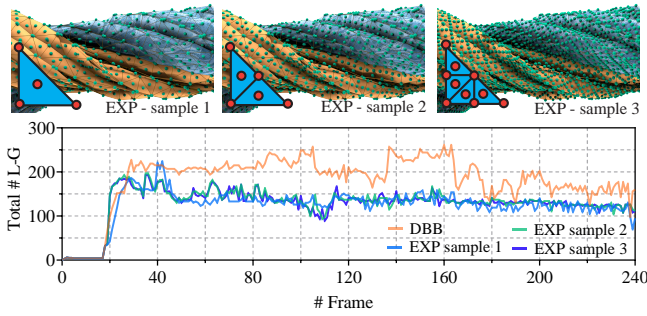


Fig. 10. **DBB vs EXP.** We plot the total number of LG iterations at each time step when twisting the cloth (as in Fig. 8) using DBB and EXP. It can be seen that EXP uses 25% to 30% fewer iterations on average than DBB. Partial CCD is not sensitive to the sample density. As shown in the figure, increasing or decreasing partial CCD samples do not vastly alter the convergence behavior of the simulation. Different sample patterns are also visualized in the figure.



Fig. 11. **“Animal crossing”.** The subspace reuse technique allows Mil2 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 Mil2 and PD-IPC [Lan et al. 2022b] produce penetration-free animations, while Mil2 is 12× faster than PD-IPC.

Mil2 is able to capture frictional contacts/collisions per user specifications. We follow the strategy used in [Li et al. 2023] to leverage barrier Hessian as the damping matrix to simulate frictional surfaces with different friction coefficients. It is also convenient to model friction as a tangential velocity damping. Fig. 13 reports a simulation

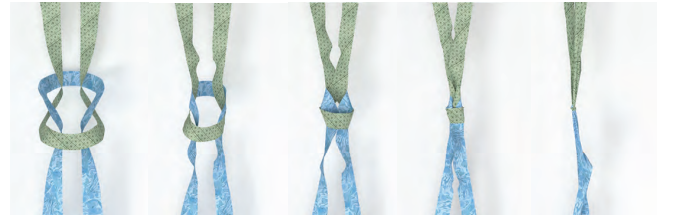


Fig. 12. **Make a knot.** Two cloth strips are pulled from opposite directions to form a tight knot. Both Mil2 and BFGS-PD [Li et al. 2023] successfully handle this challenging simulation. Mil2 is 130× faster due to EXP-based partial CCD, and more efficient subspace reuse. In this experiment, there are 104K vertices and 203K triangles on the mesh.

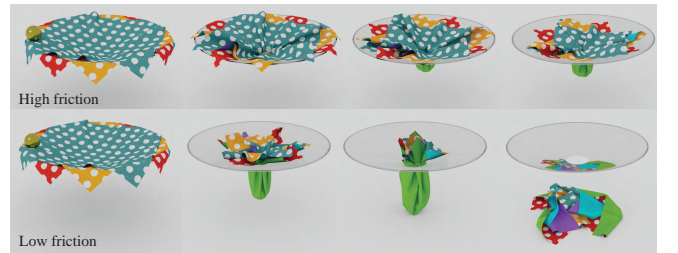


Fig. 13. **Funnel.** We place a rigid and heavy ball (simulated using affine body dynamics [Lan et al. 2022a]) on a funnel covered by three layers of cloth. The cloths hold the sphere under high frictional contacts (top). When the friction is not strong enough, all the cloths fall on the ground eventually (bottom). The simulation involves nearly 700K DOFs, and Mil2 runs at 6.6 FPS.

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 Mil2 produces physically correct dynamics under different friction settings. The simulation runs at 6.6 FPS.



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 inter-tangled cloth-cloth collisions. Mil2 robustly handles this challenging scene and produces high-quality animation.

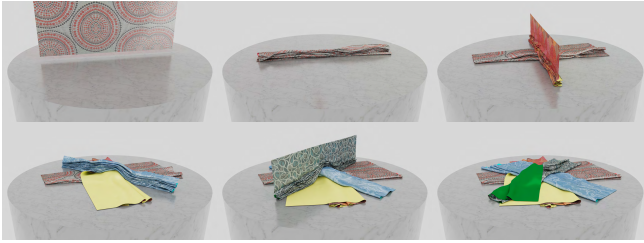


Fig. 15. **Just folding.** In this example, four clothes fall vertically on the desk one by one under different orientations. The contact between the cloth and the desktop folds the cloth with complex self-collisions. There are 200K DOFs involved, and the simulation runs at 15 FPS.

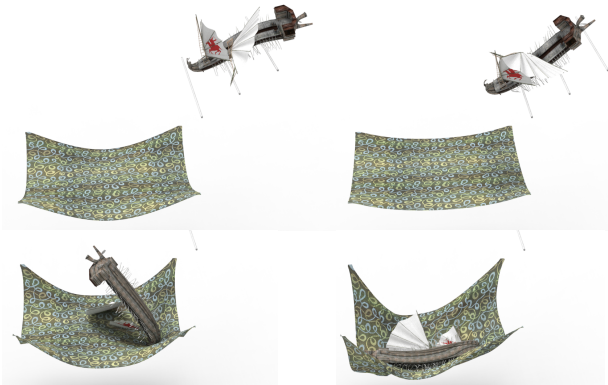


Fig. 16. **Barbarian ship in cloth.** A barbarian ship falls on three thin rods and slides down to a cloth. The ship model has complex geometries with a lot of local details. They not only generate intensive self-collisions but also lead to high-frequency cloth deformation when the ship reaches the cloth. Mil2 robustly handles this case, and the resulting simulation is 12 FPS.

8.6 More experiments

We have tested Mil2 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 17 show three tests with massive collisions. In Fig. 14, we test the robustness of Mil2 under extensive stacking. In this test, ten falling tablecloths hang on the teapot. This simulation generates a lot of interleaving 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 desk surface, it folds in a zig-zag pattern, resulting in numerous self-collisions, particularly edge-edge collisions.

Fig. 17 mimics a blender with 14 clothes in a bowl-shaped container. The scene also involves a large number of dynamic collisions. Mil2 produces interesting animations in all of those experiments, and the results are free of any inter-penetration.

While Mil2 is designed for cloth simulation, it 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. 16, a barbarian ship slides down over three thin rods and is held by a cloth on the right. The complex geometry of the ship, e.g., the masts, canvases, and paddles at the sides, induces fine-grain deformations, which are two-way coupled with the cloth. Our method robustly captures those effects, and the simulation is interactive at 12 FPS.

Cloth animations play a pivotal role in the realm of digital fashion and design. Mil2 enhances this aspect by enabling virtual characters to interact seamlessly with a variety of garments, yielding high-quality simulations. In demonstration of its capabilities, we present two additional examples. The first features a character executing a kicking action, illustrating the dynamic interaction between the motion and the garment. The second example showcases a virtual model on a fashion runway. This model, attired in a knee-length skirt, walks to the front of the stage before turning around to return. Throughout this sequence, Mil2 meticulously captures the intricacies of the motion.

9 CONCLUSION & LIMITATION

Mil2 is 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, Mil2 employs a non-distance barrier formulation, 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 stiffness 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 Mil2 yields visually plausible animations with the penetration-free guarantee and makes the simulation interactive for high-resolution scenes.

Mil2 is not perfect. It also has some limitations, leaving us interesting future research opportunities. First, Mil2 is based on projective dynamics. That said, 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. The residual forwarding trick used in Mil2 becomes less effective if the residual



Fig. 17. **Cloth blender.** We drop 14 clothes into a bowl-like container. The blender then rotates and stirs all the clothes. This example generates fast collisions between clothes and the collider, as well as a large number of cloth-cloth collisions. The simulation has 547K DOFs and runs at about 9 FPS.



Fig. 18. **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].

Table 1. **Time performance.** We report detailed time statistics for experiments mentioned in the paper. # **Bdy** 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 the numbers of elasticity constraints (Ψ) and collision constraints (B) on average 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). **Broad** gives the time used for the broad phase collision detection using patch-based BVH. **Narrow** is the total time used for the narrow phase CCD, which includes partial CCD (**P-CCD**) and full CCD (**F-CCD**). **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	# Bdy	# DOF	# Ele	# Con.	# Col.	$\bar{r} r$	Δt	Pre.	Broad	Narrow	P-CCD	F-CCD	LG (#)	Misc.	FPS
Twisting cloth (Fig. 8)	1	121K	80K	161K	110K 8M	120 30	1/150	6.3 s	8.4	14.9	1.6	13.3	47.4 (157)	10.2	12.4
“Animal cross.” (Fig. 11)	44	678K	1.1M	1.1M	67K 9.2M	120 30	1/150	46.8 s	19.6	10.1	0.4	9.7	103.2 (47)	9.2	7.0
Make a knot (Fig. 12)	1	310K	203K	410K	13K 4M	120 30	1/150	13.6 s	7.8	7.3	0.2	7.1	63.6 (74)	6.1	11.8
Funnel (Fig. 13)	6	692K	458K	923K	276K 32M	120 30	1/150	32.8 s	17.4	13.1	2.4	8.7	112.4 (52)	8.5	6.6
Stack on teapot (Fig. 14)	11	500K	338K	677K	28K 7M	120 30	1/150	24.1 s	8.4	9.1	0.3	8.8	35.9 (31)	6.6	16.6
Just folding (Fig. 15)	5	200K	131K	276K	20K 4M	120 30	1/150	8.3 s	6.4	7.9	0.3	7.6	46.1 (56)	4.3	15.5
Barbarian ship (Fig. 16)	5	546K	570K	648K	55K 6.5M	120 30	1/150	26.2 s	10.7	6.8	0.5	6.3	56.0 (50)	7.8	12.3
Cloth blender (Fig. 17)	16	547K	380K	725K	123K 14M	120 30	1/150	23.3 s	13.3	9.3	0.9	8.5	79.3 (64)	9.7	8.9
Kicking (Fig. 18)	2	450K	294K	525K	47K 4M	120 30	1/200	11.6 s	13.5	10.9	0.6	10.3	113.5 (94)	8.5	6.8
Fashion show (Fig. 1)	2	1.1M	656K	996K	87K 10M	120 30	1/200	28.4 s	14.2	9.8	0.9	8.9	149.7 (60)	10.1	5.4

errors are too large. In this situation, RF can introduce unnatural bumpy artifacts, which may be even more visually disturbing. Nevertheless, we do believe Mil2 echoes a general philosophy for developing post-Moore simulators as an integration of mutually complementary numerical procedures crafted for a specific problem.

This enables deeper optimization of the simulation and stronger synergy with emerging computing hardware.

REFERENCES

Steven S An, Theodore Kim, and Doug L James. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM transactions on graphics (TOG)* 27, 5 (2008), 1–10.

- 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>
- 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.
- 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>
- Zhili Chen, Renguo Feng, and Huamin Wang. 2013. Modeling Friction and Air Effects between Cloth and Deformable Bodies. *ACM Trans. Graph.* 32, 4, Article 88 (jul 2013), 8 pages. <https://doi.org/10.1145/2461912.2461941>
- Kwang-Jin Choi and Hyeon-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 Hyeon-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>
- 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 Duvinaud, 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.
- 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>
- 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>
- 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.
- 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.
- 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. 2021. 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>
- 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.
- 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.
- Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Culler. 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>
- Yin Yang, Dingzeyu Li, Weiwei Xu, Yuan Tian, and Changxi Zheng. 2015. Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph* 34, 6 (2015).
- 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>