Volumetric Homogenization for Knitwear Simulation

CHUN YUAN*, University of Utah, USA
HAOYANG SHI*, University of Utah, USA
LEI LAN, University of Utah, USA
YUXING QIU, LightSpeed Studios, USA
CEM YUKSEL, University of Utah, USA
HUAMIN WANG, Style3D Research, USA
CHENFANFU JIANG, UCLA, USA
KUI WU, LightSpeed Studios, USA
YIN YANG, University of Utah, USA

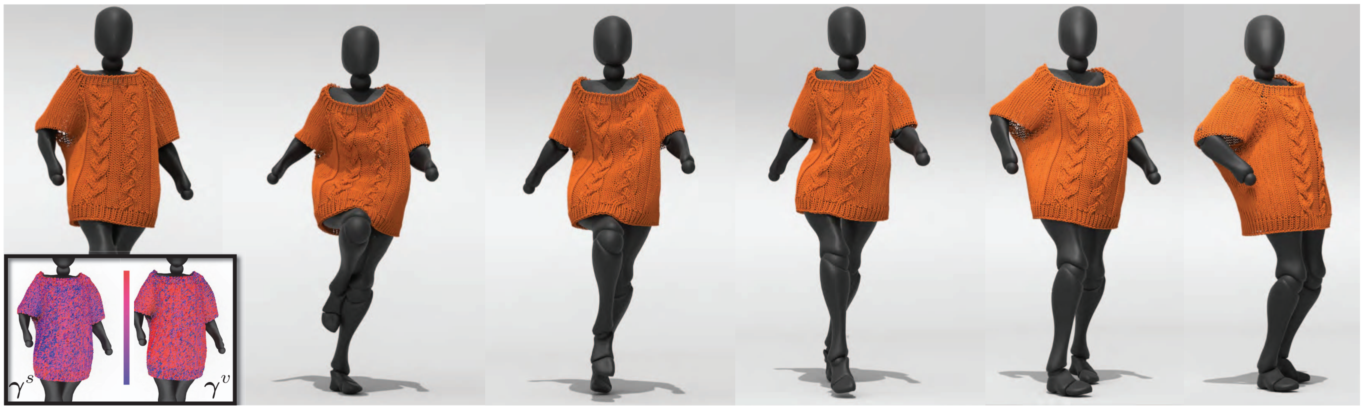


Fig. 1. ***Yangge dance.*** *We propose a volumetric homogenization algorithm for knitwear simulation. Our pipeline takes full-scale yarn-level simulation observations as input and learns a homogenization out of the input poses. We name this procedure volumetric-homogenization because we do not homogenize yarn-level dynamics to a codimensional sheet/shell domain but to a volumetric enclosure. Doing so allows us to use an intuitive hyperelastic material model with a volume-preserving constraint to capture nonlinear behaviors like bending and twisting. volumetric homogenization varies spatially. Each volumetric element has its own material parameter, which substantially enhances the expressivity of our model. This is a challenging problem, we utilize the adjoint Gauss-Newton method[Zehnder et al. 2021] to enhance the convergence of the adjoint method. Additionally, we propose harmonic initialization, sample-based optimization, and a novel domain-decomposed projective dynamics solver, which significantly accelerates the entire optimization process. In the teaser figure, we show snapshots of an animated virtual character performing a Yangge dance, where we simulate a homogenized tetrahedron mesh with about* 390K *elements that encapsulate a sweater. Cable patterns on the garment contribute to an irregular and sophisticated material distribution (as visualized in the sub-figure). In this example, the mesh-level simulation runs at* 604 *ms per frame on average under* $\Delta t = 1/150$ *sec. The yarn-level sweater model, on the other hand, consists of over* 3M *DOFs. Our volumetric homogenization makes the simulation* $\sim 100\times$ *faster than simulating the sweater at the level.*

This paper presents volumetric homogenization, a spatially varying homogenization scheme for knitwear simulation. We are motivated by the observation that macro-scale fabric dynamics is strongly correlated with its underlying knitting patterns. Therefore, homogenization towards a single material is less effective when the knitting is complex and non-repetitive. Our method tackles this challenge by homogenizing the yarn-level material locally at volumetric elements. Assigning a virtual volume of a knitting structure enables us to model bending and twisting effects via a simple volume-preserving penalty and thus effectively alleviates the material nonlinearity. We employ an adjoint Gauss-Newton formulation[Zehnder et al. 2021] to battle the dimensionality challenge of such per-element material optimization. This intuitive material model makes the forward simulation GPU-friendly. To this end, our pipeline also equips a novel domain-decomposed subspace

*joint first authors

Authors' addresses: Chun Yuan, yuanchunisme@gmail.com, University of Utah, USA; Haoyang Shi, u1431587@utah.edu, University of Utah, USA; Lei Lan, lanlei.virhum@gmail.com, University of Utah, USA; Yuxing Qiu, yuxqiu@gmail.com, LightSpeed Studios, Palo Alto, USA; Cem Yuksel, cem@cemyuksel.com, University of Utah, Salt Lake City, USA; Huamin Wang, wanghmin@gmail.com, Style3D Research, Galena, USA; Chenfanfu Jiang, chenfanfu.jiang@gmail.com, UCLA, Los Angeles, USA; Kui Wu, walker.kui.wu@gmail.com, LightSpeed Studios, Los Angeles, USA; Yin Yang, yangzzzy@gmail.com, University of Utah, Salt Lake City, USA.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 0730-0301/2024/12-ART207
https://doi.org/10.1145/3687911

solver crafted for GPU projective dynamics, which makes our simulator hundreds of times faster than the yarn-level simulator. Experiments validate the capability and effectiveness of volumetric homogenization. Our method produces realistic animations of knitwear matching the quality of full-scale yarn-level simulations. It is also orders of magnitude faster than existing homogenization techniques in both the training and simulation stages.

CCS Concepts: • **Computing methodologies** → *Simulation by animation*; *Volumetric models*.

Additional Key Words and Phrases: Yarn-level simulation, Homogenization, Physics-based simulation, Domain decomposition

ACM Reference Format:
Chun Yuan, Haoyang Shi, Lei Lan, Yuxing Qiu, Cem Yuksel, Huamin Wang, Chenfanfu Jiang, Kui Wu, and Yin Yang. 2024. Volumetric Homogenization for Knitwear Simulation. *ACM Trans. Graph.* 43, 6, Article 207 (December 2024), 19 pages. https://doi.org/10.1145/3687911

1 INTRODUCTION

Creating realistic garment animation is a core problem of computer graphics. A common practice is to leverage a mass-spring system [Liu et al. 2013; Provot 1995] or a triangle mesh [Baraff and Witkin 1998] to simulate the cloth motion as a thin membrane. The resultant cloth dynamics are controlled by a few macroscopic (triangle-level) material parameters like bending and stretching stiffness. On the other hand, knitwear like sweaters, scarves, and cardigans that are fabricated by interlocking thick yarn threads exhibit "non-rubbery" behaviors due to intricate yarn-level interactions. The intriguing mechanical responses of knitted fabrics are beyond the expressivity of an elastic sheet and require dedicated algorithms to simulate microscopic dynamics, such as yarn-level simulation (YLS) [Kaldor et al. 2008]. While YLS is able to produce high-fidelity dynamics, it is also known to be computationally expensive as one needs a large number of degrees of freedom (DOFs) to capture the motions and interactions of individual yarn threads.

Homogenized yarn-level cloth (HYLC) [Sperl et al. 2020] aims to improve the YLS efficiency by regressing a nonlinear hyperelastic thin shell model based on quasi-static YLS responses so that the triangle-level simulation reflects the dominant effects of the knits. It is based on the classic theory of computational homogenization [Geers et al. 2010] that uniform macro-scale material properties can be extracted out of micro-scale structural variations. This applies to the knit structure with relatively simple and periodic patterns, as the inconsistency between macroscopic and microscopic motion is believed to be averaged out. However, knit sweaters often feature intricate patterns and complex yarn structures across their front panels, and the spatial disparity makes the traditional homogenization technique cumbersome. Moreover, determining the appropriate size for the representative volume element (RVE) [De Souza Neto et al. 2015; Liu and Reina 2016] is not straightforward in such cases. Another limitation (maybe more as a natural consequence) of HYLC is its reliance on a highly nonlinear material model to capture the non-smooth yarn movements at micro scales, leading to potential numerical instability, such as energy singularity and Hessians that are not positive definite. Therefore, HYLC simulation uses a (very) conservative time step size, resulting in more steps to compute, which undermines its original purpose of computational efficiency.

Simply increasing the material complexity in the homogenization, e.g., using spline-based strain-stress model [Sperl et al. 2020; Xu et al. 2015] or even neural-network-based materials [Feng et al. 2024], is less helpful to capture the dynamics of complicated interlocked yarn structure with large variations and often induce numerical instability (e.g. see Wang et al. [2023]).

We propose an entirely different perspective for efficient simulation of knits that can even handle non-repeating knitting patterns. In contrast to traditional material homogenization methods or HYLC, we introduce a novel approach: volumemetric homogenization. We focus on studying the *heterogeneity*, i.e., the spatial variation of material parameters while keeping the constitutional relation simple. In particular, we incorporate a virtual volume representing the yarn material and exploit spatially varying volume-preserving penalties to capture desired knit characteristics, such as stretching, shearing, and bending, without over-complicating the material model.

We discretize the fabric at the yarn level and embed it within a volumetric mesh. This mesh is of high resolution to capture the local structural characteristics of yarn loops. We learn a macro-scale material model over this volumetric space, aligning it with dynamic YLS results. In that respect, our method can also be understood as applying local homogenization at each mesh element. Our rationale resembles curve fitting, where using multiple low-degree polynomials (e.g., splines) is often preferred over employing a single high-degree polynomial. Likewise, a composite material model, despite its simplicity, offers a more versatile design space compared to a homogenized material, since each volumetric element possesses independent material freedoms. We employ the adjoint Gauss-Newton method[Zehnder et al. 2021] to regress the yarn material via a high-dimension space-time optimization problem. The simplicity of the material model makes the volumetric homogenization well-suited for efficient forward GPU solvers such as projective dynamics (PD) [Bouaziz et al. 2014]. To this end, we devise a domain-decomposed multi-level solver for the global stage of PD specially crafted for our runtime pipeline.

Although simulating over a volume mesh may appear expensive initially, the aforementioned benefits outweigh the overhead of mesh DOFs. As a result, our method is more stable, runs orders of magnitude faster than existing homogenized YLS, and produces realistic animations of knitwear with complex knit patterns.

2 RELATED WORK

There exists a large volume of excellent work on relevant topics of cloth and yarn simulation. Due to the page count limits, this section only briefly discusses some closely relevant prior work.

Sheet-level cloth model. Modern fabrics and garments are fabricated through sophisticated workmanship and exhibit intricate material response. Nevertheless, modeling cloth with a sheet-like shell remains prevalent [Choi and Ko 2002; Grinspun et al. 2003] due to its simplicity and intuitiveness. Mass-spring and/or triangle meshes offer a natural discretization of sheet-based cloth, and its equation of motion can be derived via elastic energies that measure the cloth deformation [Terzopoulos et al. 1987]. Implicit integration has become a standard component since the seminal work of Baraff and Witkin [1998]. Unlike a rubber membrane, cloth fabrics are less stretchable but tend to be bent more easily. This feature inspires various techniques to enforce the inextensibility [English and Bridson 2008; Goldenthal et al. 2007; Provot 1995; Wang et al. 2010] with numerical stability. Cloth bending, on the other hand, can be parameterized by the dihedral angle or bending modes [Bridson et al. 2003]. It is also possible to exploit isometric mesh deformations to use the so-called quadratic bending [Bergou et al. 2006; Garg et al. 2007], which possesses a constant Hessian. Kim [2020] reveals the connection between model of Baraff and Witkin [1998] and the finite element method (FEM) [Bathe 2006]. To capture detailed wrinkles and folds on the cloth, adaptive remeshing [Narain et al. 2013, 2012] or mixed discretization [Guo et al. 2018; Weidner et al. 2018] are proven to be effective. Another important aspect of sheet-level cloth animation is to handle collisions, especially self-collisions [Baraff et al. 2003; Bridson et al. 2002; Volino and Thalmann 2000]. Penalty-based methods [Bouaziz et al. 2014; Wu et al. 2020] are straightforward to implement but often with stability issues. Harmon et al. [2008]; Provot [1997] introduce approaches that group multiple collisions into "impact zones" and treat them as rigid bodies, allowing for some sliding motion. Constraint-based collision handling methods represent contact as constraints based on exact Coulomb friction [Li et al. 2018; Otaduy et al. 2009].

Yarn-level cloth model. The rapid development of computing hardware makes animated yarn-level fabric feasible. YLS models each yarn thread as an elastic rod [Bergou et al. 2010, 2008; Pai 2002; Spillmann and Teschner 2007] and exploits yarn-yarn contact to trigger fabric deformation. This effort is pioneered by Kaldor et al. [2008], where yarn threads are modeled as cubic B-splines. With YLS, different knit structures like garter, rib, and stockinette showcase unique stretching behaviors, which are beyond the expressivity of average sheet-level models. This approach is subsequently accelerated by approximating yarn-yarn collisions using a co-rotated force model [Kaldor et al. 2010]. Persistent contact [Cirio et al. 2014, 2015] is an efficient method to handle the interaction among heavily interlaced yarns. This method is further combined with a triangle-based model to create a hybrid system that enhances yarn-level details only in areas of interest [Casafranca et al. 2020]. Additionally, Sánchez-Banderas et al. [2020] extend the concept of persistent contact to handle stacked fabrics, addressing both intra-fabric and inter-fabric contacts implicitly. Computational design and machine fabrication of yarn patterns are also of great interest to various communities. Leaf et al. [2018] propose an efficient GPU yarn-level simulator to enable the interactive designing of periodic yarn patches. Yuksel et al. [2012] introduce an efficient 3D design and modeling interface at the stitch level, which is later extended for hand knitting [Wu et al. 2019], machine-knitting [Narayanan et al. 2019], and enforcing wearability [Wu et al. 2021] via cloth simulation.

Data- and learning-based cloth model. Data-driven or learning-based cloth simulation is considered an effective approach to enhance the realism of continuum shell models. Wang et al. [2011] estimate planar and bending stiffness through separate tests, while automatic devices have been developed for acquiring fabric stiffness [Miguel et al. 2012] and friction parameters [Miguel et al. 2013]. Similar tensile tests have also been utilized by Clyde et al. [2017] to estimate the planar stiffness of woven fabrics. Feng et al. [2022] use a simulation-in-the-loop framework to estimate the fabric bending stiffness. ClothCap aims to reconstruct multiple garments from the 3D scans [Pons-Moll et al. 2017]. Differentiable simulation techniques [Li et al. 2022; Liang et al. 2019] become requisite for such inverse problems. Prior arts are often designed for inverse problems of a handful of parameters. They either become prohibitive or do not converge when the inverse problem has a large number of unknown variables to be optimized. Deep neural networks offer a powerful modality for extracting knowledge from observations. They have also been applied for garment and cloth animation. For instance, Santesteban et al. [2019] show that deep nets can be exploited to predict garment deformation based on physics-based simulation results. PBNS [Bertiche et al. 2021] and SNUG [Santesteban et al. 2022] use unsupervised learning for the synthesis of garment deformation. Bertiche et al. [2022] further generalizes this idea to learn dynamic garment movements. However, existing methods mostly focus on sheet-based models. A learning-based yarn-level model remains under-explored. This is likely due to the lack of high-quality training data and efficient simulation algorithms.

Numerical methods. An efficient forward simulation is often the key ingredient of the inverse problem. As the bottleneck in cloth simulation is often at solving the energy Hessian, a natural thought is to avoid a full linear solve in classic Newton's method. Following this idea, Hecht et al. [2012] propose a lagged factorization scheme that reuses existing Cholesky factorization to save the computation. Multi-resolution [Capell et al. 2002; Lee et al. 2010] and multigrid [Tamstorf et al. 2015; Wang et al. 2018] solvers project fine-grid residual errors onto a coarser grid, on which linear or nonlinear iterations are more effective [Bolz et al. 2003; Tamstorf et al. 2015; Xian et al. 2019; Zhu et al. 2010]. Liu et al. [2013] treat the implicit Euler integration as an energy minimization problem, in which spring constraints can be solved in parallel in the local step, while the global linear system remains constant on run-time. Projective Dynamics (PD) [Bouaziz et al. 2014] extend this concept to support a wider range of hyperelastic material models in quadratic form. Additionally, the computation of integration has been accelerated using methods such as the Chebyshev [Wang 2015; Wang and Yang 2016], Gauss-Seidel [Fratarcangeli et al. 2016], and various parallelization techniques on GPU [Fratarcangeli et al. 2016; Wang and Yang 2016; Wu et al. 2020]. Fully leveraging the obtained gradient to find a better search direction to improve convergence is also essential for achieving better performance. A common strategy involves estimating the second-order information of the system, e.g. L-BFGS[Du et al. 2021; Li et al. 2022] or Anderson acceleration[Peng et al. 2018]. Wang [2018] and Zimmermann et al. [2019] propose using the Gauss-Newton solver, which exhibits promising convergence. However, assembling and factorizing the dense Gauss-Newton solver becomes a new bottleneck. Similar to how the adjoint method circumvents dense matrices, Zehnder et al. [2021] introduces two additional adjoint variables that transform the Gauss-Newton solver to be a sparse linear system, greatly accelerating the performance.

Homogenization & coarsening. Our work is also closely related to computational homogenization [Allaire and Brizzi 2005; Andreassen

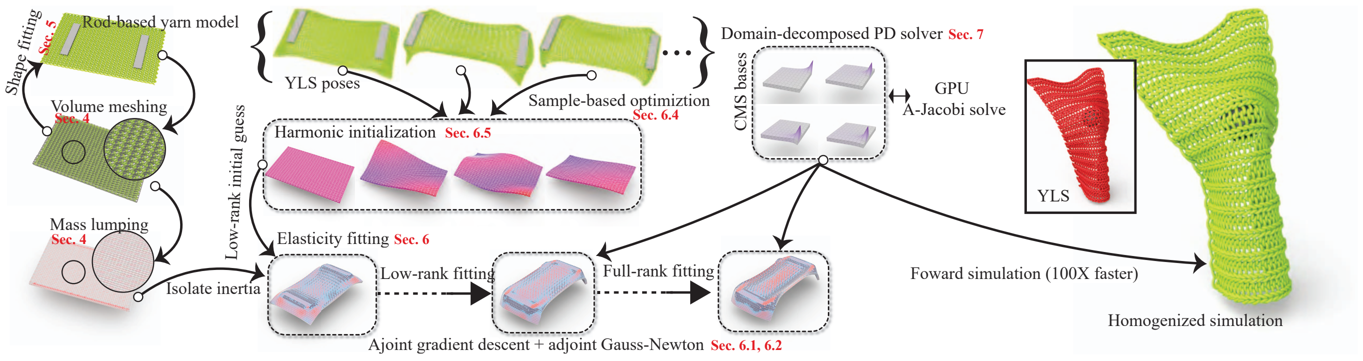


Fig. 2. ***Volumetric homogenization pipeline.*** *Given an input rod-based yarn model, our pipeline generates a volume mesh encapsulating the entire yarn structure. We lump the yarn-level mass to the mesh's nodal points so that the inertia effect(Eq. 14) can be isolated. The yarn-to-mesh and mesh-to-yarn shape fitting facilitates us to define the loss function for homogenization. Per-element material parameters are obtained via an adjoint Gauss-Newton procedure, in a sample-by-sample manner. With a domain-decomposed PD solver, our method not only produces high-quality animation of knitwear garments that is visually similar to the full-scale YLS result but also achieves these results two orders of magnitude faster.*

and Andreasen 2014; Grinspun et al. 2002] and numerical coarsening [Chen et al. 2017, 2018]. This category of computing techniques aims to extract a low-rank representation of complex systems via global (coarse mesh) to local (fine mesh) optimization so that simulation at runtime efficiently captures the correct dynamics even on coarse grids [Kharevych et al. 2009; Nesme et al. 2009]. It has been successfully applied for fabrication [Chen et al. 2017; Panetta et al. 2015] or for accelerated simulation [Torres et al. 2016]. Chen et al. [2018] fit a material-aware shape function so that the coarse mesh replicates the behavior of a fine model. Constitutive model homogenization focuses more on macroscopic material responses [Blanco et al. 2016; De Souza Neto et al. 2015], where the underlying strain-stress relation is synthesized at RVE (representative volume element). This method has also been applied to the inverse problem of micro-structure optimization or topology optimization [Eschenauer and Olhoff 2001]. For instance, Schumacher et al. [2015] build deformable objects with target stiffness using cells of prescribed bulking behavior. Chen et al. [2015] propose a data-driven extension of this potential energy fitting idea to non-linear materials where now a coarse constitutive model is found through linear regression based on a set of deformation samples obtained from random forcing.

Our method shares many similarities in algorithmic rationale and method design with those prior arts and specifically targeting realistic and efficient knitwear simulation. Our method is directly relevant and strongly inspired by recent contributions on HYLC [Sperl et al. 2020, 2022]. HYLC shows a sheet-level homogenization paradigm that regresses a spline-based constitutive model. The homogenization substantially reduces the DOF, and the sheet-level simulation is much faster than YLS. The major difference between our method and HYLC is that our homogenization escalates the dimensionality of coarse mesh i.e., from a sheet- or rod-based discretization to a volumetric one. On the surface, such a strategy does not align with the original motivation of homogenization or coarsening, as it uses more DOF. Yet, we show that the increased dimensionality simplifies the material design when incorporated with a volume preservation constraint to homogenize bending and twisting effects of codimensional models. We are not the first to leverage volume preserving to model nonlinear bending deformation. Chen et al. [2023] use a volumetric prism element to simulate thick garments and have demonstrated the feasibility of this approach. Another major difference is that we learn a set of independent material parameters at each volume element, resulting in a high-dimensional space-time optimization problem. We name our method *volumetric homogenization*, hinting at these novel features of our formulation. The complexity of volumetric homogenization is unlike most prior methods. To address this, we propose to leverage a domain-decomposed PD solver and employ adjoint Gauss-Newton[Zehnder et al. 2021] to make the homogenization manageable. Volumetric homogenization is more expressive than HYLC–we can replicate the dynamics of complex and non-repetitious knits. It is also more efficient–the use of simpler material models with a fast solver offsets the increase in homogenized DOF.

3 METHOD OVERVIEW

Our framework allows stable and efficient knitwear simulation that closely mimics the behavior of a full-scale YLS. To achieve this objective, our pipeline estimates a spatially varying homogenization scheme for the yarn fabric. Unlike prior methods that homogenize a triangular mesh approximating the fabric mid-surface [Feng et al. 2024; Sperl et al. 2020] and assign uniform material parameters across the mesh, our pipeline, as shown in Fig. 2, begins by constructing a volumetric mesh to encapsulate the given yarn-level model and computes the mass for each nodal point (Sec. 4). With the sequence of yarn-level simulated results, we proceed to compute the best-fitting material parameter for each volume element. This is a difficult task, because we do not have bounding volumetric mesh for each frame, the material parameters over mesh are of high dimension, and the optimization is sensitive to the initial condition. Therefore, we decompose the problem into two fitting steps. First, we formulate an optimization problem to find the best-fitting mesh shape matching the yarn-level deformation (Sec. 5). With the fitted shape, we employ an adjoint Gauss-Newton formulation[Zehnder et al. 2021] that estimates a quasi-second-order descent direction of the high-dimension material parameter (Sec. 6).

Adjoint Gauss-Newton synergizes with a harmonic initialization algorithm to progressively explore a good initial guess. To utilize the homogenized mesh generated from our fitting pipeline, we introduce a novel GPU-based forward simulator (Sec. 7). The global PD matrix is partitioned into domains corresponding to different knit patterns, and we leverage component mode synthesis (CMS) [Craig Jr and Hale 1988] to build a subspace preconditioner for the global stage solve, followed by a full-space GPU-based Jacobi iteration.

4 MESH CONSTRUCTION

Given a yarn-level model discretized into piece-wise line segments at the rest pose, we construct a volumetric mesh to encapsulate the yarn structure and assign the mass for each nodal point so it can facilitate the subsequent fitting steps and mesh-level simulation.

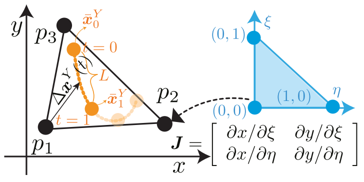

y
x
p_1
p_2
p_3
$\bar{x}_0^Y$
$\bar{x}_1^Y$
$t = 0$
$t = 1$
L
$\Delta x^Y(t)$
ξ
η
$(0, 1)$
$(1, 0)$
$(0, 0)$
$J = \begin{bmatrix} \partial x/\partial\xi & \partial y/\partial\xi \\ \partial x/\partial\eta & \partial y/\partial\eta \end{bmatrix}$

Fig. 3. ***Mass lumping.*** *We estimate lumped mass at each mesh node via the line integral of the shape function over the embedded yarn segment.*

In mesh construction, we employ the 3D Bresenham algorithm [Zhang et al. 2018; Žalik et al. 1997] to create a volumetric grid, such that each voxel encapsulates a non-zero portion of a yarn thread. To ensure connectivity and eliminate dangling voxels, additional voxels are included in rare cases where the thread passes through a voxel corner. To prevent numerical locking of the hexahedral element [Bathe 2006], each voxel is further split into six tetrahedrons. Let n^Y and n^V be the total DOF for the yarns and the volume mesh, respectively. Typically, n^V is around half of n^Y. In other words, meshing the yarn model is unable to lower the DOF count by orders like in Sperl et al. [2020]. As we discuss later, our method excels in a more stable energy formulation and highly parallelizable GPU solver after the volumetric homogenization. This overcomes the cost of high DOF and makes the entire pipeline much more efficient than prior methods [Feng et al. 2024; Sperl et al. 2020].

The next step is to distribute the yarn curves to the surrounding nodal points of the mesh. A 2D illustration is shown in Fig. 3, where the triangle element encloses a piece of yarn thread of three linearized segments. Let $\boldsymbol{x}$ and m be the position and mass of a material particle in the element. It is assumed that

$$m = \sum_i N_i(\xi, \eta) \cdot m_i \,, \tag{1}$$

where m_i represents the lumped mass of the i-th nodal point. $N_i(\xi, \eta)$ is the shape function defined with the natural coordinates

$$N_1(\xi, \eta) = 1 - \xi - \eta \,, \qquad N_2(\xi, \eta) = \xi \,, \qquad N_3(\xi, \eta) = \eta \,.$$

We compute the lumped mass for each nodal point as

$$m_i = \sum_j L_j \int_0^1 N_i \left(\boldsymbol{J}^{-1} \Delta \boldsymbol{x}_j^Y(t) \right) \rho_j(t) \mathrm{d}t \,, \tag{2}$$

where the summation index j iterates over all linearized yarn segments (three segments in this example), L_j is the length of the j-th segment, $\boldsymbol{J}$ is the element Jacobi, which relates the natural coordinates with the material coordinates, and $\Delta \boldsymbol{x}_j^Y(t)$ is the vector in the material space from the natural origin to the parameterized yarn-level position on j-th segment. For instance, the first segment in Fig. 3 has two endpoints $\bar{\boldsymbol{x}}_0^Y$ and $\bar{\boldsymbol{x}}_1^Y$, and we have

$$\Delta \boldsymbol{x}^Y(t) = \bar{\boldsymbol{x}}_0^Y + t\left(\bar{\boldsymbol{x}}_1^Y - \bar{\boldsymbol{x}}_0^Y \right) - \bar{x}_0 \,. \tag{3}$$

Note that we use the overbar $\bar{(\cdot)}$ to denote the rest-shape position.

5 SHAPE FITTING

With the mesh constructed at the rest pose, our next step is to estimate its shape based on the simulated yarn-level data for each frame. This boils down to finding out the mesh position given the displacement of embedded yarn threads. We use superscripts $(\cdot)^Y$ and $(\cdot)^V$ to differentiate variables defined on yarns or on the volume mesh. Let $\boldsymbol{x}^V \in \mathbb{R}^{3n^V}$ be mesh-level position, stacking x, y, and z coordinates of all the nodes on the tetrahedral mesh. The mesh-to-yarn (V2Y) transfer $\phi^{V2Y} : \mathbb{R}^{3n^V} \rightarrow \mathbb{R}^{3n^Y}$ is conveniently established with shape-function-based interpolation

$$\phi^{V2Y}(\boldsymbol{x}^V) \triangleq \boldsymbol{N}\boldsymbol{x}^V \,, \tag{4}$$

where $\boldsymbol{N} \in \mathbb{R}^{3n^V \times 3n^Y}$ contains N_i values at yarn endpoints. However, the map along the other direction i.e., from yarn to mesh or Y2V, is less straightforward.

We formulate the Y2V transfer ϕ^{Y2V} by minimizing position and deformation inconsistency given an input yarn pose $\boldsymbol{x}^Y \in \mathbb{R}^{3n^Y}$. The deformation error measures the discrepancy of the deformation gradient tensor between yarn and the encapsulating volumetric mesh. The deformation gradient of a yarn segment is computed as:

$$\boldsymbol{F}^Y(\boldsymbol{x}^Y) = \left[\boldsymbol{x}_1^Y - \boldsymbol{x}_0^Y, \boldsymbol{n}_1^Y, \boldsymbol{n}_2^Y \right] \left[\bar{\boldsymbol{x}}_1^Y - \bar{\boldsymbol{x}}_0^Y, \bar{\boldsymbol{n}}_1^Y, \bar{\boldsymbol{n}}_2^Y \right]^{-1}$$

where $\bar{\boldsymbol{x}}_{0,1}^Y$ and $\boldsymbol{x}_{0,1}^Y$ are the rest-shape and deformed positions of two endpoints and $\bar{\boldsymbol{n}}_{0,1}^Y$ and $\boldsymbol{n}_{0,1}^Y$ are two mutually perpendicular material normals. The deformation gradient includes a rotation and stretch component. While the stretch component can be directly averaged to the corresponding element, the rotation component cannot. To manage the averaging of the rotation component, we apply polar decomposition to $\boldsymbol{F}^Y$, which yields a rotation tensor and a symmetric deformation tensor such that $\boldsymbol{F}^Y = \boldsymbol{R}^Y \boldsymbol{S}^Y$. We parameterize the rotation tensor using matrix exponential as $\boldsymbol{R}^Y = \exp(\Omega^Y)$. Note that Ω^Y is a skew-symmetric matrix, which can be directly averaged using weighted summation. This geometrically corresponds to averaging the rotation axis and rotation angle[Grassia 1998]. Based on this information, we estimate the deformation gradient of the element as

$$\boldsymbol{F}^{Y2V}(\boldsymbol{x}^Y) = \exp\left(\frac{1}{\sum_j L_j} \sum_j L_j \Omega_j^Y \right) \left(\frac{1}{\sum_j L_j} \sum_j L_j \boldsymbol{S}_j^Y \right) . \tag{5}$$

As in Eq. 2, the summation is for all the yarn segments in the element.

We define the Y2V transfer as the minimizer of the optimization

$$\phi^{Y2V}(\boldsymbol{x}^Y) \triangleq \arg\min_{x} \sum_e V_e \left\| \mathcal{D} \cdot (\boldsymbol{B}_e \boldsymbol{x}) - \boldsymbol{F}_e^{Y2V} \right\|_F^2 + \alpha \left\| \boldsymbol{N}\boldsymbol{M}^V \boldsymbol{x} - \boldsymbol{M}^Y \boldsymbol{x}^Y \right\|^2. \quad (6)$$

Here, $\boldsymbol{B}_e \in \mathbb{R}^{12\times 3n^Y}$ is a binary matrix picking 12 DOFs for the tetrahedron e from $\boldsymbol{x}$ i.e., $\boldsymbol{x}_e = \boldsymbol{B}_e \boldsymbol{x}$. $\mathcal{D} \in \mathbb{R}^{3\times 3\times 12}$ is a third-order tensor. It works as a differential operator to compute the deformation gradient of the element out of $\boldsymbol{x}_e$. $\|\cdot\|_F$ denotes the matrix Frobenius norm. α is a hyperparameter, and we set it as $\alpha = 0.1$. $\boldsymbol{M}^V$ is the diagonal lumped mass matrix of the mesh (i.e., using Eq. (2)), and $\boldsymbol{M}^Y$ is the mass matrix for the yarn. V_e is the volume of the element. In other words, Y2V transfer is essentially a volumetric Possion reconstruction [Kazhdan et al. 2006]–we expect $\phi^{Y2V}(\boldsymbol{x}^Y)$ to capture the yarn-level deformation while using $\left\|\boldsymbol{N}\boldsymbol{M}^V\boldsymbol{x} - \boldsymbol{M}^Y\boldsymbol{x}^Y\right\|^2$ as a mass-weighted regularization penalty.

Shape fitting retrieves the information of the mesh from YLS and allows us to estimate inertia forces as defined in Eq. 14, derived from an arbitrary yarn motion sequence. As a result, the yarn-level deformation can be understood as a quasi-static one under the non-inertia frame, and the volumetric homogenization only needs to focus on fitting the elastic material since mass is decoupled.

6 ELASTICITY FITTING

Elasticity fitting lies at the core of our volumetric homogenization pipeline. In a nutshell, elasticity fitting estimates spatially varying material parameters for each element given input YLS sequences. This is particularly challenging because 1) volumetric homogenization seeks a high-dimension material parameter with a large number of unknowns, and 2) the optimization takes consideration of all yarn poses. We employ several techniques to address these challenges including the use of second-order Gauss-newton, as described by Zehnder et al. [2021], within the framework of the adjoint method. Additionally, we introduce a novel harmonic initialization scheme for initial value guessing. We borrow the idea of stochastic descent [Bottou et al. 1991] to mitigate the dimensionality concern of space-time optimization.

Let n^E be the total number of elements on the mesh. The material vector $\boldsymbol{\gamma}$ has $2n^E$ freedoms such that an element e has two hypothesized material parameters, namely γ_e^s and γ_e^v. We define an intuitive elastic energy at each element e as

$$E_e = \gamma_e^s V_e \left\|\boldsymbol{F}_e - \boldsymbol{R}(\boldsymbol{F}_e)\right\|_F^2 + \gamma_e^v V_e \left\|\boldsymbol{F}_e - \boldsymbol{V}(\boldsymbol{F}_e)\right\|_F^2, \quad (7)$$

where

$$\boldsymbol{R} = \arg\min_{\boldsymbol{A}\in \mathrm{SO}(3)} \|\boldsymbol{F} - \boldsymbol{A}\|_F^2 \quad \text{and} \quad \boldsymbol{V} = \arg\min_{\boldsymbol{A}\in \mathrm{SL}(3)} \|\boldsymbol{F} - \boldsymbol{A}\|_F^2. \quad (8)$$

Here, SL(3) is the special linear group that preserves the volume (i.e. $|\boldsymbol{V}| = 1$), SO(3) is the special orthogonal group that preserves the length, γ_e^s determines the strength of penalizing the strain magnitude, and γ_e^v gives the strength of preserving the volume.

The choice of energy. There exists a wide range of choices for energy formulation, and many commonly seen hyperelastic energies should serve the purpose well. Our primary argument is that the material complexity holds less significance compared to the material variation, which naturally reflects the spatial adaptivity inherent in knit patterns. The elastic model of Eq. 7 is simple–its constraint-based quadratic form eases the implementation efforts and produces good results in practice.

Elasticity fitting finds the optimal $\boldsymbol{\gamma}$ i.e., the value of γ_e^s and γ_e^v for all n^E elements, such that V2Y transfer (Eq. 4) of the resultant mesh simulation, $\boldsymbol{x}^V(\boldsymbol{\gamma})$, constitutes a similar yarn dynamics obtained from the full-scale YLS. For a YLS sequence of n^F frames

$$\left\{\boldsymbol{x}_1^Y, \boldsymbol{x}_2^Y, \boldsymbol{x}_3^Y, \cdots, \boldsymbol{x}_{n^F}^Y\right\},$$

we build an error metric or the loss function between $\boldsymbol{x}_i^Y$ and $\boldsymbol{x}_i^V(\boldsymbol{\gamma})$ for $i = 1, \cdots, n^F$ and minimize the accumulated error for the entire sequence

$$\arg\min_{\boldsymbol{\gamma}} \sum_{i=1}^{n^F} \epsilon_i(\boldsymbol{\gamma}), \quad \text{s.t. } \boldsymbol{\gamma} \geq 0, \quad (9)$$

where

$$\epsilon_i(\boldsymbol{\gamma}) = \sum_e V_e \left\| \mathcal{D} \cdot \left(\boldsymbol{B}_e \boldsymbol{x}_i^V(\boldsymbol{\gamma})\right) - \boldsymbol{F}_e^{Y2V}(\boldsymbol{x}_i^Y) \right\|_F^2 + \alpha \left\| \boldsymbol{N}\boldsymbol{M}^V \boldsymbol{x}_i^V(\boldsymbol{\gamma}) - \boldsymbol{M}^Y \boldsymbol{x}_i^Y \right\|^2. \quad (10)$$

Here, ϵ_i adopts the same error measure as in Y2V transfer (Eq. 6) but it now depends on $\boldsymbol{\gamma}$ since $\boldsymbol{x}_i^V(\boldsymbol{\gamma})$ relates to $\boldsymbol{\gamma}$ by the mesh-level simulation. The sub-index i here refers to the index of the input n^F YLS poses.

Eq. 9 describes a space-time optimization problem. It imposes a significant computational challenge since both n^F and n^E are big numbers, not to mention n^V and n^Y are also of high resolution. While first-order descent methods like gradient descent are often preferred due to their simplicity, they fail to deliver a good result even after a large number of iterations in this case. We have to resort to more sophisticated optimization techniques of a higher order for elasticity fitting of Eq. 9.

6.1 Adjoint Gradient Descent

Let us start with the gradient. Expanding the gradient of ϵ_i via the chain rule yields

$$\frac{\partial \epsilon_i}{\partial \boldsymbol{\gamma}} = \frac{\partial \epsilon_i}{\partial \boldsymbol{x}_i^V} \cdot \frac{\partial \boldsymbol{x}_i^V}{\partial \boldsymbol{\gamma}}. \quad (11)$$

The specific form of $\partial \boldsymbol{x}^V / \partial \boldsymbol{\gamma}$ is up to the mesh-level simulation result, which is often formulated as a variational optimization minimizing the sum of the inertial potential I and the elasticity potential $\sum E_e$

$$\arg\min_{\boldsymbol{x}_i^V} I + \sum_e E_e, \qquad \text{where} \qquad I = \frac{1}{2\Delta t^2} \boldsymbol{a}_i^{V\top} \boldsymbol{M}^V \boldsymbol{a}_i^V. \quad (12)$$

Shape fitting allows us to approximate $\boldsymbol{a}_i^V$ via

$$\boldsymbol{a}_i^V \approx \phi^{Y2V}\left(\boldsymbol{x}_i^Y - 2\boldsymbol{x}_{i-1}^Y + \boldsymbol{x}_{i-2}^Y\right) - \Delta t^2 \boldsymbol{M}^{V^{-1}} \boldsymbol{f}_i^V, \quad (13)$$

so that it becomes a known vector and depends on $\boldsymbol{x}_i^Y, \boldsymbol{x}_{i-1}^Y, \boldsymbol{x}_{i-2}^Y$, and the external force $\boldsymbol{f}_i^V$. In other words, the Y2V transfer function ϕ^{Y2V} and pre-computed nodal mass (Eq. 2) decouple I from the simulation, converting a dynamic problem to a quasi-static one.

The necessary optimality condition of Eq. 12 gives

$$g(x_i^V,\gamma) \triangleq \underbrace{M^V \phi^{Y2V}\left(x_i^Y - 2x_{i-1}^Y + x_{i-2}^Y\right) - f_i^V}_{\text{inertia force + external force}} + \frac{\partial \sum E_e}{\partial x_i^V} = 0. \tag{14}$$

It is easy to see that Eq. 14 is the requirement of force equilibrium, which is the constraint that should always be satisfied, making it an identity equation. Differentiating Eq. 14 at both sides w.r.t. γ yields

$$\frac{\mathrm{d}g}{\mathrm{d}\gamma} = \frac{\partial g}{\partial x_i^V}\cdot\frac{\partial x_i^V}{\partial \gamma} + \frac{\partial g}{\partial \gamma} = 0 \Rightarrow \frac{\partial x_i^V}{\partial \gamma} = -\left(\frac{\partial g}{\partial x_i^V}\right)^{-1}\cdot\frac{\partial g}{\partial \gamma}. \tag{15}$$

To avoid solving the linear system of $\partial g/\partial x_i^V$ for $2n^E$ times (recalling γ is a $2n^E$-dimension material vector and, therefore, $\partial g/\partial x_i^V \in \mathbb{R}^{3n^V\times 2n^E}$), the adjoint method [Givoli 2021; Tarantola 2005] leverages an adjoint state vector λ, and evaluates the gradient of the target loss function via

$$\frac{\partial \epsilon_i}{\partial \gamma} = -\lambda^\top \frac{\partial g}{\partial \gamma}, \tag{16}$$

where λ is obtained by solving the linear system of

$$\left(\frac{\partial g}{\partial x_i^V}\right)\lambda = \left(\frac{\partial \epsilon_i}{\partial x_i^V}\right)^\top. \tag{17}$$

The gradient of the loss function $\partial\epsilon_i/\partial\gamma$ allows us to employ first-order optimizers, e.g., gradient descent, to iterative refine the material vector. Unfortunately, it is known that first-order methods are less effective as γ approaches a local optimum. To improve the convergence, we employ a hybrid optimization scheme, which uses different optimizers at different stages of the elasticity fitting process.

6.2 Adjoint Gauss-Newton

A well-known strategy to improve the convergence of gradient descent is to regularize the gradient by a preconditioning matrix P_i, often an SPD (symmetric positive definite) matrix. For instance, Newton's method uses the inverse of the Hessian matrix

$$H_i = \frac{\partial^2 \epsilon}{\partial \gamma^2} = \left(\frac{\partial x_i^V}{\partial \gamma}\right)^\top \cdot \frac{\partial^2 \epsilon_i}{\partial x_i^{V^2}} \cdot \frac{\partial x_i^V}{\partial \gamma} + \frac{\partial \epsilon_i}{\partial x_i^V}\cdot\frac{\partial^2 x_i^V}{\partial \gamma^2}, \tag{18}$$

so that the preconditioned search direction becomes $H_i^{-1}(\partial\epsilon_i/\partial\gamma)^\top$. When $\|\partial\epsilon_i/\partial\gamma\|$ is small, Newton's method delivers locally second-order convergence [Nocedal and Wright 1999].

We observe that when ϵ_i is small and mesh deformation x_i^V starts aligning with the target yarn pose x^Y, $\partial\epsilon_i/\partial x_i^V$ is always close to zero. This allows an alternative precondition matrix P_i that discards the second term of $\mathbf{H}_i$, such that

$$P_i = \left(\frac{\partial x_i^V}{\partial \gamma}\right)^\top G_i \frac{\partial x_i^V}{\partial \gamma}, \tag{19}$$

where, $G_i = \partial^2\epsilon_i/\partial x_i^{V^2} \in \mathbb{R}^{3n^V\times 3n^V}$ and can be pre-computed. This Hessian simplification strategy is a.k.a. Gauss-Newton method, and the preconditioned search direction can be obtained by solving the linear system P_i:

$$P_i d_i^{GN} = -\left(\frac{\partial \epsilon_i}{\partial \gamma}\right)^\top, \tag{20}$$

where the loss gradient on the r.h.s. of Eq. 20 can be obtained via the adjoint solve (Eq. 16 and Eq. 17).

While the formulation is well known, Gauss-Newton is seldom used with an adjoint method in practice. The difficulty lies in the fact that $\partial x_i^V/\partial\gamma$ should never be explicitly evaluated via solving Eq. 15 and, therefore, we do not have the actual precondition matrix P_i. Even if we could compute P_i exactly, solving Eq. 20 remains prohibitive since P_i is likely a dense matrix.

Zehnder et al. [2021] demonstrate that this issue can be circumvented. Similar to how the adjoint method avoids explicit computation of $\frac{\partial x_i^M}{\partial \gamma}$, two additional adjoint state vectors are introduced to transform the dense system Eq. 20 into the following larger sparse linear system:

$$\begin{bmatrix} G_i & -\left(\frac{\partial g}{\partial x_i^V}\right)^\top & 0 \\ -\frac{\partial g}{\partial x_i^V} & 0 & \frac{\partial g}{\partial \gamma} \\ 0 & \left(\frac{\partial g}{\partial \gamma}\right)^\top & 0 \end{bmatrix}\begin{bmatrix} \mu \\ \nu \\ d_i^{GN}\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \left(\frac{\partial \epsilon_i}{\partial \gamma}\right)^\top \end{bmatrix}. \tag{21}$$

One can quickly verify that the solution of Eq. 21 coincides with the solution of Eq. 20: from the first two lines of Eq. 21, we have $(\partial g/\partial x_i^V)^{-\top}G_i\mu = \nu$ and $(\partial g/\partial x_i^V)^{-1}(\partial g/\partial\gamma)d_i^{GN} = \mu$, suggesting

$$\left(\frac{\partial g}{\partial x_i^V}\right)^{-\top} G_i \underbrace{\left(\frac{\partial g}{\partial x_i^V}\right)^{-1}\frac{\partial g}{\partial \gamma}}_{-\partial x_i^V/\partial\gamma} d_i^{GN} = \nu.$$

Left multiplying $(\partial g/\partial\gamma)^\top$ both sides and knowing $(\partial g/\partial\gamma)^\top\nu = \partial\epsilon_i/\partial x_i^V$, i.e. the third row of Eq. 21, restores the formula back to the vanilla Gauss-Newton of Eq. 20.

When γ is fixed, ϵ_i becomes a quadratic form of x_i^V, and G_i is therefore non-negative definite. To this end, we follow the Levenberg-Marquardt method [Moré 2006] that adds a small diagonal κI at l.h.s. of Eq. 21 to secure its positive definiteness. It is noteworthy that $\partial g/\partial x_i^V$ is a sparse matrix–it has non-zero entities only at the adjacent mesh elements (similar to FEM matrices), and $\partial g/\partial\gamma$ is also a sparse matrix (only the element's material parameter influences its residual force). Solving Eq. 21 is more efficient than solving Eq. 20 despite the increased matrix dimension.

6.3 Linear Search & Non-Negativity Constraint

Our optimization is two-stage. We use the vanilla gradient descent based on the adjoint method (Eq. 17) for the first batch of iterations (10 - 15 iterations in our implementation). Gradient descent remains a competitive option at the early stage of the optimization due to its efficiency. Adjoint Gauss-Newton (Eq. 21) then ensues, which offers a stronger descent direction. We observe that 20 - 30 Gauss-Newton iterations are more effective than over 1,000 gradient descent in the later phase of the optimization.

It should be noted that all the components i.e., $\partial \boldsymbol{g}/\partial \boldsymbol{x}_i^V$, $\partial \boldsymbol{g}/\partial \boldsymbol{\gamma}$, and $\partial \epsilon_i/\partial \boldsymbol{\gamma}$ that assemble the Gauss-Newton system are readily available after the gradient computation. Compared with gradient descent, the additional cost of our adjoint Gauss-Newton comes from the linear solve of Eq. 21. Thanks to the sparsity of the matrix, this step is *not* the bottleneck of the pipeline. Instead, the most expensive computation is always at finding $\boldsymbol{x}_i^V$ to satisfy the equilibrium condition i.e., Eq. 14. As to be discussed in Sec. 7, the constraint-based energy formulation (Eq. 7) allows the use of highly efficient parallel GPU procedures to accelerate this step, which is not only helpful for the forward simulation but also for the elasticity fitting.

A line search is needed for both gradient descent and Gauss-Newton phases to prevent overshooting. We use the default step size of 0.01 for gradient descent refinements and 1.0 for Gauss-Newton refinements. We shrink the step size by half if the refinement produces a higher loss value.

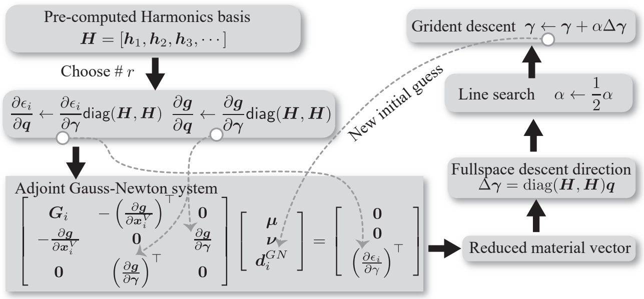
$\boldsymbol{\gamma}$ should be non-negative, and simulation becomes ill-defined if some components $[\gamma]_j$ in $\boldsymbol{\gamma}$ are smaller than zero. We use a mixed projection-pivoting strategy to enforce the non-negativity of $\boldsymbol{\gamma}$. Specifically, when a refinement $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} + \Delta\boldsymbol{\gamma}$ produces negative components ($[\gamma]_j < 0$), we correct their values to be a small positive quality of $1e-3$ and cache indices of those components/elements. If the next $\Delta\boldsymbol{\gamma}$ tries to further reduce the value of those parameters, we clamp them to zero ($[\Delta\gamma]_j \leftarrow 0$) to cancel such constraint-violating refinement. This is an easy but heuristic mechanism to enforce $\boldsymbol{\gamma} > \boldsymbol{0}$. Occasionally, the clamped search direction becomes non-descent even using adjoint Gauss-Newton. When this occurs, we opt for the pivoting method [Baraff 1994] that sets most negative $[\gamma]_j$ as equality-constrained ones (i.e., $[\gamma]_j = 1e-3$) and precludes them from the Gauss-Newton solve by removing corresponding rows and columns in the l.h.s. of Eq. 21. When pivoting is activated, we do not relax those equality-constrained DOFs. Fortunately, pivoting is rarely needed provided a reasonable initial guess of $\boldsymbol{\gamma}$.

6.4 Sample-by-Sample Fitting

The discussion so far has focused on elasticity fitting for one input yarn pose $\boldsymbol{x}_i^Y$. Recall that the global loss function accumulates across all n^F poses (i.e., see Eq. 9). This leads to a very high-dimension space-time optimization problem, and solving it in its original form is infeasible. To ease the computational cost, we optimize $\boldsymbol{\gamma}$ sequentially in a frame-by-frame manner. Doing so is similar to stochastic optimization widely used in deep learning [Kingma and Ba 2015]. Specifically, we start with solving the sub-problem of Eq. 9 for the first pose $\boldsymbol{x}_i^Y$ as: $\boldsymbol{\gamma}_1 \leftarrow \arg\min_{\boldsymbol{\gamma}} \epsilon_1(\boldsymbol{\gamma}, \boldsymbol{x}_1^Y)$. The resulting $\boldsymbol{\gamma}_1$ is saved as the current solution of the material vector such that $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}_1$. Once $\boldsymbol{\gamma}_k$ is computed for the k-th pose, we update $\boldsymbol{\gamma}$ via by convexly averaging $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}_k$ as

$$\boldsymbol{\gamma} \leftarrow \frac{w}{w+w_k}\boldsymbol{\gamma} + \frac{w_k}{w+w_k}\boldsymbol{\gamma}_k, \tag{22}$$

Here, $w, w_k > 0$ suggest the importance of $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}_k$. We find the elasticity energy stored in $\boldsymbol{x}_k^Y$ to be a good choice of w_k, which is invariant under rigid body movements and always non-negative. The weight of $\boldsymbol{\gamma}$ is initially set as zero. As the optimization moves forward, we keep tracking the most deformed yarn pose and use the corresponding elastic energy as w.



Pre-computed Harmonics basis
$\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, \cdots]$
Choose # r
$\frac{\partial \epsilon_i}{\partial \boldsymbol{q}} \leftarrow \frac{\partial \epsilon_i}{\partial \boldsymbol{\gamma}}\mathrm{diag}(\boldsymbol{H},\boldsymbol{H})$ $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}} \leftarrow \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\gamma}}\mathrm{diag}(\boldsymbol{H},\boldsymbol{H})$
Adjoint Gauss-Newton system
New initial guess
Grident descent $\gamma \leftarrow \gamma + \alpha\Delta\gamma$
Line search $\alpha \leftarrow \frac{1}{2}\alpha$
Fullspace descent direction $\Delta\boldsymbol{\gamma} = \mathrm{diag}(\boldsymbol{H},\boldsymbol{H})\boldsymbol{q}$
Reduced material vector

$$\begin{bmatrix} \boldsymbol{G}_i & -\left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i^V}\right)^\top & \boldsymbol{0} \\ -\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i^V} & \boldsymbol{0} & \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\gamma}} \\ \boldsymbol{0} & \left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\gamma}}\right)^\top & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\nu} \\ \boldsymbol{d}_i^{GN} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \left(\frac{\partial \epsilon_i}{\partial \boldsymbol{\gamma}}\right)^\top \end{bmatrix}$$

Fig. 4. ***Harmonic initialization.*** *We design a progressive initialization strategy based on Harmonic bases of the volume mesh. By projecting the material vector into the Harmonic subspaces of different ranks, volumetric homogenization always finds a reliable initial value for the two-stage elasticity fitting procedure.*

Theoretically, such an online optimization strategy takes multiple epochs to converge. However, we note that a single pass over all frames always generates high-quality results in practice. Unlike in deep learning, where training data are considered equally important, most YLS poses are redundant or repeating, e.g. poses close in time often have a similar geometry because the motion trajectory of the fabric often remains smooth. This observation motivates us to further sparse the computation by applying elasticity fitting at a subset of fewer sample poses and finding the overall $\boldsymbol{\gamma}$ sample by sample. In practice, elasticity fitting over handful poses yields reasonably good results.

6.5 Harmonic Initialization

Another challenge is the high dimensionality of the material vector $\boldsymbol{\gamma}$. By assigning each element two independent freedoms, the material versatility is enhanced. On the downside, it also makes the fitting process sensitive to the initial guess of $\boldsymbol{\gamma}$. An unlucky guess easily strands the optimization at local minima. To make our per-element homogenization reliable, our pipeline includes a harmonic initialization scheme, as shown in Fig. 4.

The core idea is progressively exploring the material space from low frequency to high frequency to build the material complexity incrementally. To this end, we split $\boldsymbol{\gamma}$ into $\boldsymbol{\gamma}^s \in \mathbb{R}^{n^E}$ and $\boldsymbol{\gamma}^v \in \mathbb{R}^{n^E}$, which collect γ_e^s and γ_e^v for all the elements on the mesh respectively. After that, we construct a set of Harmonic bases $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_r]$ by computing r eigenvectors of the mesh Laplacian [Nasikun et al. 2018; Vallet and Lévy 2008] corresponding to the r smallest eigenvalues (see Fig. 5). The low-frequency material variation is assumed to be well captured by this set of basis vectors, and we require

$$\boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{\gamma}^s \\ \boldsymbol{\gamma}^v \end{bmatrix} = \mathrm{diag}(\boldsymbol{H},\boldsymbol{H}) \underbrace{\begin{bmatrix} \boldsymbol{q}^s \\ \boldsymbol{q}^v \end{bmatrix}}_{\boldsymbol{q}}, \tag{23}$$

where $\boldsymbol{q} \in \mathbb{R}^{2r}$ is the generalized material vector of much lower dimension. Since $\boldsymbol{H}$ is constant, all the derivatives w.r.t. $\boldsymbol{\gamma}$ can be

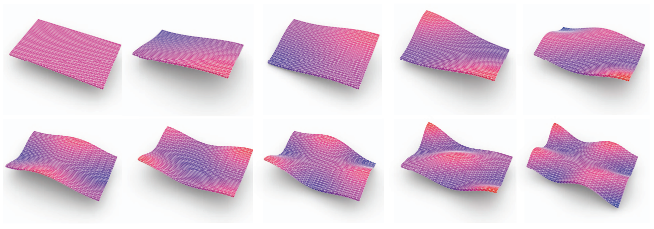


Fig. 5. ***Harmonic material bases.*** *We leverage mesh Harmonics to explore a good initial material variation from low frequency to high frequency. This figure visualizes the first ten basis vectors.*

conveniently transferred to $\boldsymbol{q}$ by the chain rule:

$$\frac{\partial(\cdot)}{\partial \boldsymbol{q}} = \frac{\partial(\cdot)}{\partial \boldsymbol{\gamma}} \cdot \frac{\partial \boldsymbol{\gamma}}{\partial \boldsymbol{q}} = \frac{\partial(\cdot)}{\partial \boldsymbol{\gamma}} \mathrm{diag}(\boldsymbol{H}, \boldsymbol{H}).$$

For instance, to switch the optimization target to $\boldsymbol{q}$ in Eq. 21, we right multiply $\mathrm{diag}(\boldsymbol{H}, \boldsymbol{H})$ to $\partial \boldsymbol{g}/\partial \boldsymbol{\gamma}$ and $(\partial \epsilon_i/\partial \boldsymbol{\gamma})^\top$, and sandwich $\boldsymbol{G}_i$ with $\mathrm{diag}(\boldsymbol{H}, \boldsymbol{H})$ as

$$\boldsymbol{G}_i \leftarrow \mathrm{diag}(\boldsymbol{H}^\top, \boldsymbol{H}^\top)\boldsymbol{G}_i \mathrm{diag}(\boldsymbol{H}, \boldsymbol{H}). \tag{24}$$

The harmonic initialization starts with $r = 1$. In this case, $\boldsymbol{H}$ has a single constant-value basis vector $\boldsymbol{h}_1$. Projecting $\boldsymbol{\gamma}$ into this basis is equivalent to requiring $\boldsymbol{\gamma}^s$ and $\boldsymbol{\gamma}^v$ to be constant across all the elements. This result is then used as the initial guess for a bigger r. The harmonic initialization moves forward progressively with $r = 1$, $r = 10$, and $r = 30$. Afterward, we fit the un-reduced $\boldsymbol{\gamma}$ based on the low-rank initialization.

7 DOMAIN-DECOMPOSED PROJECTIVE DYNAMICS

After $\boldsymbol{\gamma}$ is obtained, we can simulate the volume mesh at the runtime, which is used to drive the deformation of the underlying yarns via ϕ^{M2Y}. Our volume embedding implicitly deals with yarn-yarn contacts and collisions. This section focuses on the mesh-level simulation, and we omit the superscript M for simplicity of notation.

As in Eq. 12, we aim to minimize the total variational energy. Instead of using the approximation in Eq. 13, $\boldsymbol{a}$ is now defined as

$$\boldsymbol{a} = \boldsymbol{x} - \boldsymbol{x}^* - \Delta t \dot{\boldsymbol{x}}^* - \Delta t^2 \boldsymbol{M}^{-1} \boldsymbol{f}, \tag{25}$$

where $\boldsymbol{x}^*$ and $\dot{\boldsymbol{x}}^*$ are the mesh-level position and velocity in the previous time step. Instead of solving unknown position $\boldsymbol{x}$ using existing methods, such as Newton's method, we present an efficient domain-decomposed projective dynamics for our constraint-based energy formulation.

7.1 Projective Dynamics

Projective dynamics (PD) [Bouaziz et al. 2014] splits the variational optimization into global and local steps. At the local step, PD considers the elasticity energy of each element as the shortest square distance to a constraint manifold on which the constraint is exactly satisfied. The key operation is to identify such spot on the constraint manifold i.e., the *target position*. One may notice that our elasticity energy (Eq. 7 and Eq. 8) is formulated exactly in this way. Therefore, the goal of the local step is to find $\boldsymbol{R}$ and $\boldsymbol{V}$ for each mesh element. The best-fitting rotation can be obtained by polar decomposing the element's deformation gradient $\boldsymbol{F}_e$. To obtain the best-fitting volume-preserving transformation $\boldsymbol{V}$, we first compute its SVD (singular value decomposition) as $\boldsymbol{F}_e = \boldsymbol{U}\Sigma\boldsymbol{W}^\top$. Σ is a diagonal matrix of three singular values σ_1, σ_2, and σ_3, and the volume-preserving constraint becomes $\sigma_1\sigma_2\sigma_3 = 1$. If the element is not inverted or degenerated, σ_1, σ_2, and σ_3 should be positive. We then transfer this constraint into an optimization procedure of three singular values:

$$\begin{aligned} \arg \min_{\Delta\sigma_1, \Delta\sigma_2, \Delta\sigma_3} \quad & \Delta\sigma_1^2 + \Delta\sigma_2^2 + \Delta\sigma_3^2, \\ \text{s.t.} \quad & (\sigma_1 + \Delta\sigma_1)(\sigma_2 + \Delta\sigma_2)(\sigma_3 + \Delta\sigma_3) = 1, \\ \text{and} \quad & \sigma_1 + \Delta\sigma_1, \sigma_2 + \Delta\sigma_2, \sigma_3 + \Delta\sigma_3 > 0. \end{aligned} \tag{26}$$

To solve this nonlinear programming problem, we form its KKT (Karush–Kuhn–Tucker) system only involving the equality constraint using the Lagrange multiplier method. If any of $\Delta\sigma_1$, $\Delta\sigma_2$ or $\Delta\sigma_3$ violates the inequality constraint, we clamp it to $[0.01, +\inf]$ and fix its value at the next iteration. While Eq. 26 is highly nonlinear, it only has three unknowns. In practice, we always find a good local projection for the volume-preserving constraint in very few iterations.

The global step is a standard linear solve in the form of $\boldsymbol{K}\boldsymbol{x} = \boldsymbol{b}$ where we have

$$\boldsymbol{K} = \left(\frac{\boldsymbol{M}}{\Delta t^2} + \sum_e V_e (\gamma_e^s + \gamma_e^v) \boldsymbol{B}_e^\top (\mathcal{D}^\top : \mathcal{D}) \boldsymbol{B}_e \right) \boldsymbol{x}, \tag{27}$$

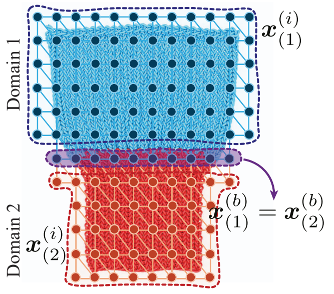
and

$$\begin{aligned} \boldsymbol{b} = \frac{\boldsymbol{M}}{\Delta t^2} \boldsymbol{a} &+ \sum_e V_e \gamma_e^s \boldsymbol{B}_e^\top \left(\mathcal{D}^\top : \boldsymbol{R}(\boldsymbol{B}_e \boldsymbol{x}^*) \right) \\ &+ \sum_e V_e \gamma_e^v \boldsymbol{B}_e^\top \left(\mathcal{D}^\top : \boldsymbol{V}(\boldsymbol{B}_e \boldsymbol{x}^*) \right). \end{aligned} \tag{28}$$

The global step stands as the most expensive computation along the pipeline , as the Cholesky decomposition of K may yield a large dense matrix that is not GPU-friendly.

7.2 Domain Decomposition

To accelerate the global step computation, we exploit the component mode synthesis (CMS) [Craig Jr 1985; Craig Jr and Hale 1988] to decompose the system into multiple domains (or components) and build a linear subspace at each domain to analyze the dynamic responses of complex structures. CMS was originally designed for linear structural analysis, and its generalization to nonlinear simulation remains an open research problem. Fortunately, the unique modality of PD allows us to apply CMS just at the global stage solve, so we can partition the mesh into multiple domains without aligning the decomposition with the variation of the underlying knit patterns.


Domain 1
$\boldsymbol{x}_{(1)}^{(i)}$
Domain 2
$\boldsymbol{x}_{(1)}^{(b)} = \boldsymbol{x}_{(2)}^{(b)}$
$\boldsymbol{x}_{(2)}^{(i)}$
Fig. 6. ***DOF types with domain decomposition.*** *The fabric is decomposed into two domains corresponding to two different knitting patterns. It is required that duplicated boundary DOFs at domains must always be equal to each other.*

Unknown DOFs at each domain can be now grouped into internal DOFs and boundary DOFs. As the name suggests, the boundary DOFs interface with the neighbor domains while the internal DOFs are isolated by the boundary DOFs.

Without loss of generality, let us assume the mesh is decomposed into two domains. If we extract rows and columns corresponding to DOFs of the d-th domain for $d = 1, 2$, a domain-level global system can be built as

$$\begin{bmatrix} \boldsymbol{K}_{(d)}^{(ii)} & \boldsymbol{K}_{(d)}^{(ib)} \\ \boldsymbol{K}_{(d)}^{(bi)} & \boldsymbol{K}_{(d)}^{(bb)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{(d)}^{(i)} \\ \boldsymbol{x}_{(d)}^{(b)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{(d)}^{(i)} \\ \boldsymbol{b}_{(d)}^{(b)} \end{bmatrix}. \tag{29}$$

Here, subscripts $(\cdot)^{(i)}$ and $(\cdot)^{(b)}$ denote the DOF type i.e., either internal or boundary; the subscript $(\cdot)_{(d)}$ indicates the domain index. Switching the order of subscripts applies the matrix transpose i.e., $\boldsymbol{K}_{(d)}^{(ib)} = \boldsymbol{K}_{(d)}^{(bi)\top}$. To analyze the internal response of the domain, we prescribe a unit displacement at each boundary DOF, and pre-compute its internal response for some unknown boundary stimuli

$$\begin{bmatrix} \boldsymbol{K}_{(d)}^{(ii)} & \boldsymbol{K}_{(d)}^{(ib)} \\ \boldsymbol{K}_{(d)}^{(bi)} & \boldsymbol{K}_{(d)}^{(bb)} \end{bmatrix} \begin{bmatrix} \Psi_{(d)}^{(i)} \\ \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{F}_{(d)}^{(b)} \end{bmatrix}, \tag{30}$$

where $\boldsymbol{I}$ is an identity matrix corresponding to the prescribed boundary displacements, and $\boldsymbol{F}_{(d)}^{(b)}$ is the external stimuli (they are not the "forces" but a type of system load in a more general sense). We do not really care about the value of $\boldsymbol{F}_{(d)}^{(b)}$ but are more interested in the system response at internal DOFs, i.e., $\Psi_{(d)}^{(i)}$. It can be computed via expanding the first row of Eq. 30:

$$\boldsymbol{K}_{(d)}^{(ii)}\Psi_{(d)}^{(i)} + \boldsymbol{K}_{(d)}^{(ib)} = \boldsymbol{0} \Rightarrow \Psi_{(d)}^{(i)} = -\left(\boldsymbol{K}_{(d)}^{(ii)}\right)^{-1}\boldsymbol{K}_{(d)}^{(ib)}. \tag{31}$$

Due to the linearity of this problem, $\Psi_{(d)}^{(i)}$ encodes all the possible internal responses induced by boundary stimuli. When the domain does not undertake any non-boundary loads, $\Psi_{(d)}^{(i)}$ relates boundary DOFs and internal DOFs as $\boldsymbol{x}_{(d)}^{(i)} = \Psi_{(d)}^{(i)}\boldsymbol{x}_{(d)}^{(b)}$. They are a.k.a. *boundary modes* in CMS, and serve as subspace basis vectors for our global solve. For non-boundary responses, we compute a compact set of eigenvectors (e.g., 20) of $\boldsymbol{K}_{(d)}^{(ii)}$ corresponding to the smallest eigenvalues

$$\left(\Phi_{(d)}^{(i)}\right)^{\top} \boldsymbol{K}_{(d)}^{(ii)}\Phi_{(d)}^{(i)} = \Lambda_{(d)}\ , \tag{32}$$

where $\Lambda_{(d)}$ is the diagonal matrix of eigenvalues. The composition of $\Psi_{(d)}^{(i)}$ and $\Phi_{(d)}^{(i)}$ constitutes a linear subspace for the internal DOFs of the domain:

$$\boldsymbol{x}_{(d)}^{(i)} = \left[\Psi_{(d)}^{(i)}, \Phi_{(d)}^{(i)}\right] \begin{bmatrix} \boldsymbol{p}_{(d)}^{(i)} \\ \boldsymbol{x}_{(d)}^{(b)} \end{bmatrix}. \tag{33}$$

Here, $\boldsymbol{p}_{(d)}^{(i)}$ is the generalized coordinate of the d-th domain for its non-boundary-driven deformation. The internal boundary-driven deformation is fully prescribed by its boundary deformation of $\boldsymbol{x}_{(d)}^{(b)}$. This allows us to construct a global subspace basis matrix, which has a block-wise structure, to relate the reduced coordinate and the fullspace coordinate as

$$\begin{bmatrix} \boldsymbol{x}_{(1)}^{(i)} \\ \boldsymbol{x}_{(1)}^{(b)} \\ \boldsymbol{x}_{(2)}^{(i)} \\ \boldsymbol{x}_{(2)}^{(b)} \end{bmatrix} = \begin{bmatrix} \Phi_{(1)}^{(i)} & \boldsymbol{0} & \Psi_{(1)}^{(i)} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \Phi_{(2)}^{(i)} & \Psi_{(2)}^{(i)} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_{(1)}^{(i)} \\ \boldsymbol{p}_{(2)}^{(i)} \\ \boldsymbol{x}^{(b)} \end{bmatrix}. \tag{34}$$

It is known that a domain-decomposed global system must include extra boundary constraints to make sure domains are seamlessly connected such as

$$\begin{bmatrix} \boldsymbol{K}_{(1)}^{(ii)} & \boldsymbol{K}_{(1)}^{(ib)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{K}_{(1)}^{(bi)} & \boldsymbol{K}_{(1)}^{(bb)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{K}_{(2)}^{(ii)} & \boldsymbol{K}_{(1)}^{(ib)} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{K}_{(2)}^{(bi)} & \boldsymbol{K}_{(1)}^{(bb)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{(1)}^{(i)} \\ \boldsymbol{x}_{(1)}^{(b)} \\ \boldsymbol{x}_{(2)}^{(i)} \\ \boldsymbol{x}_{(2)}^{(b)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{(1)}^{(i)} \\ \boldsymbol{b}_{(1)}^{(b)} \\ \boldsymbol{b}_{(2)}^{(i)} \\ \boldsymbol{b}_{(2)}^{(b)} \end{bmatrix} \tag{35}$$

$$\text{s.t.} \quad \boldsymbol{x}_{(1)}^{(i)} = \boldsymbol{x}_{(2)}^{(i)}. \tag{36}$$

CMS often uses the Lagrange multiplier method to enforce the boundary constraint [Yang et al. 2013]. Our novel subspace structure of Eq. 34 eliminates duplicated boundary DOFs at the domains' interface as $\boldsymbol{x}_{(1)}^{(b)}$ and $\boldsymbol{x}_{(2)}^{(b)}$ are now uniformly written as $\boldsymbol{x}^{(b)}$. The global step of PD can be more efficiently solved by projecting Eq. 35 into the subspace of Eq. 34. The reduced global matrix remains block-sparse, and the pre-computations of Eq. 30 can be carried out in parallel at domains.

Domain decomposition and CMS-like subspace construction techniques make our pipeline less sensitive to global matrix change e.g., due to material update or the presence of new constraints of collisions. After the reduced global solve is obtained, we use Eq. 34 to convert the generalized coordinate to the fullspace result. This result is then sent to a GPU-based full-space solver to make sure the global solution is accurate. In our implementation, we use the aggregated Jacobi method (A-Jacobi) proposed in [Lan et al. 2022b]. A-Jacobi combines two or three Jacobi iterations into one iteration to fully exploit the parallel capacity of modern GPUs. As a result, mesh-level simulation is efficient is nearly interactive in many examples reported in this paper.

Note that a fast and numerically stable mesh simulation is critical to the entire pipeline–it is not only helpful for simulating the volumetric homogenized knitwear materials but also an indispensable module for elasticity fitting. Recall that our adjoint Gauss-Newton is only feasible when Eq. 14 is satisfied. This requirement suggests we need to find the equilibrium configuration of the mesh given the current material vector $\boldsymbol{\gamma}$. Whenever $\boldsymbol{\gamma}$ are updated during the elasticity fitting process, we have to re-assemble and factorize $\boldsymbol{x}_i^M$, which is the most expensive computation along the pipeline. Without a fast solver, elasticity fitting at such a high-dimension material space is infeasible.

8 EXPERIMENTAL RESULTS

We implemented our volumetric homogenization framework on a desktop computer with an Intel i7-12700 CPU and an NVIDIA

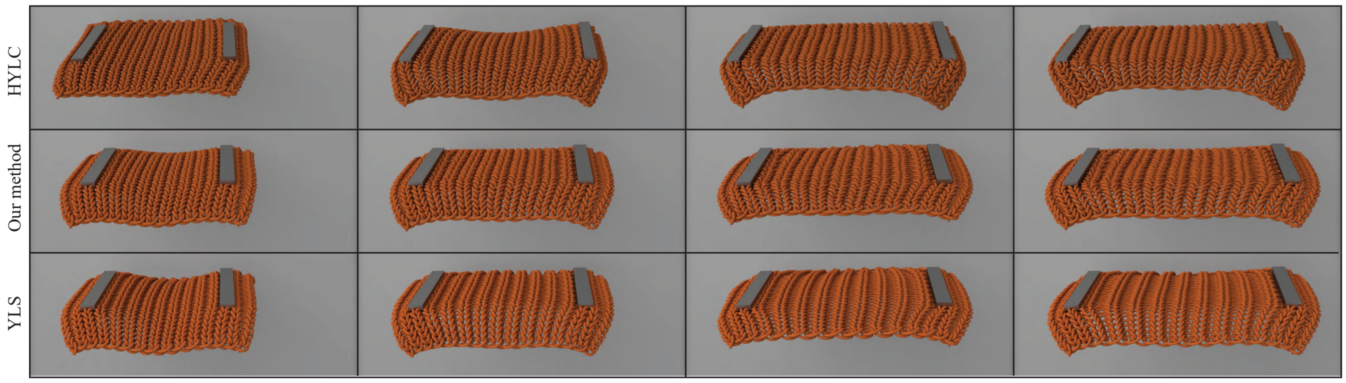


Fig. 7. ***Comparison with HYLC (1×1 rib).*** *We compare our method with HYLC and homogenize a square knitted fabric of the periodic 1×1 rib pattern. The full-scale YLS results are shown at the bottom for reference. In general, both our method and HYLC yield plausible results, and both are visually similar to YLS results. Volumetric homogenization uses a mesh of* 38K *DOFs, while HYLC only has* 9K *DOFs. Nevertheless, our method takes 85ms per frame and is* $\sim$ 180× *faster than HYLC.*

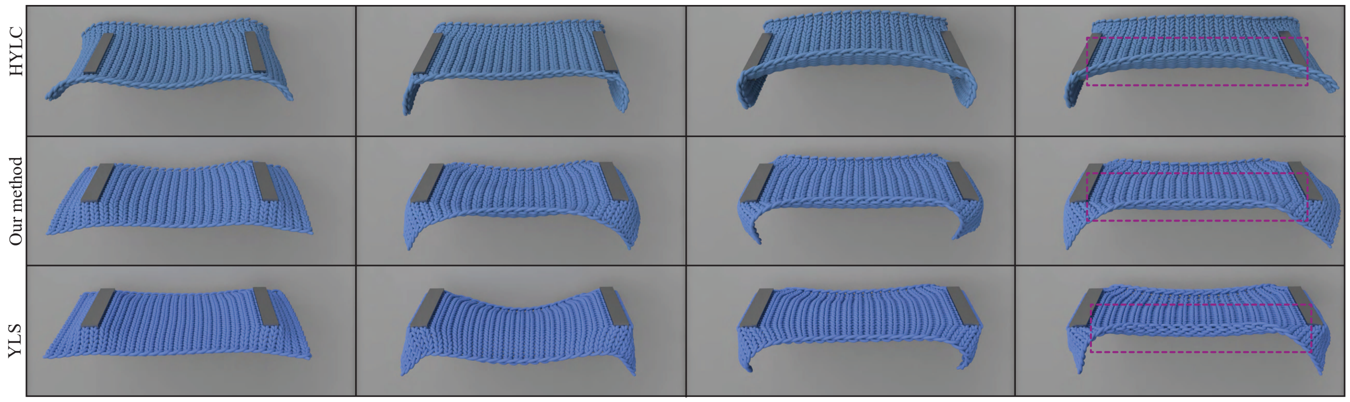


Fig. 8. ***Comparison with HYLC (stockinette).*** *Given fabric with a periodic stockinette pattern, our method effectively captures spatially varying deformation leveraging material heterogeneity, particularly under significant stretch as highlighted. In the example, volumetric homogenization uses a mesh of* 42K *DOFs, while HYLC has* 12K *DOFs. Our method is* $\sim$ 350× *faster than HYLC.*

RTX 3090 GPU. We used `Spectra` library for computing the eigen-decomposition of the mesh Laplacian, and $\boldsymbol{K}_d^{ii}$ for each domain, and `Eigen` [Guennebaud et al. 2010] as the primary interface of linear algebra computations. The domain-decomposed PD procedure consists of two steps. The first step solves the global matrix in the subspace, and we employed the `PARDISO` solver [Schenk et al. 2001] shipped with Intel `MKL` [Wang et al. 2014]. Fullspace A-Jacobi iterations [Lan et al. 2022b] are followed to make sure the global step system is well solved. This step was implemented on the GPU with `CUDA`. We used Chebyshev weight to improve the convergence of A-Jacobi method [Wang 2015]. The local step is parallelized on the GPU at each element with `CUDA`. Adjoint gradient descent and adjoint Gauss-Newton are sensitive to the residual of Eq. 14 i.e., Eq. 14 must be strictly satisfied. Therefore, during the elasticity fitting, we applied a few (three to five) Newton iterations after the PD solve to keep the residual smaller than $1E-5$. We generate full-scale YLS results by simulating each yarn as the Cosserat rod model [Spillmann and Teschner 2007]. The detailed statistics of experiments are reported in Tab. 1.

8.1 Comparison with HYLC

Homogenized yarn-level cloth (HYLC) [Sperl et al. 2020] provides an excellent paradigm by homogenizing the "yarn material" to the mid-surface of the fabric, represented as a triangle mesh. Our volumetric homogenization method expands the dimensionality to a 3D volume with more DOFs and optimizes spatially varying material properties instead of assuming a uniform material as in HYLC. Our first experiment involves three side-by-side comparisons to thoroughly understand the differences between our method and HYLC [1]. In HYLC, the Discrete Elastic Rod energy is replaced by Cosserat Rod energy for our comparison. The visualization of both our method and HYLC is achieved by embedding the yarn geometry into the mesh.

[1]We used the implementation of HYLC at https://git.ista.ac.at/gsperl/HYLC

Table 1. ***Statistics.*** *We report detailed time statistics for experiments mentioned in the paper. n^Y is the total number DOFs on the yarn model. n^M and n^E are the numbers of mesh DOFs and mesh elements (and the deformable objects e.g., in Fig. 15). # d is the number of domains used. # GN reports the average number of adjoint Gauss-Newton iterations needed for elasticity fitting for each YLS pose. n^F is the total number of sample poses used in the example. Fit. gives the total time used for elasticity fitting. Δt is the time step size of the forward simulation, and Sim. gives the total time to simulate the homogenized knitwear for one time step.*

		n^Y	n^M	n^E	# d	# GN	n^F	Fit.	Δt	Sim.
Comp. HYLC (1×1 rib)	(Fig. 7)	66K	38K	79K	1	15	2	0.5 hours	1/150 sec	85 ms
Comp. HYLC (stockinette)	(Fig. 8)	66K	42K	83K	1	42	3	3.1 hours	1/150 sec	92 ms
Comp. HYLC (draping)	(Fig. 9)	66K	42K	83K	1	26	2	1.2 hours	1/150 sec	103 ms
Basketweave	(Fig. 10)	885K	107K	204K	6	17	4	2.2 hours	1/150 sec	137 ms
Montague	(Fig. 10)	85K	48K	90K	9	10	4	0.7 hours	1/150 sec	96 ms
Flame	(Fig. 10)	88K	60K	92K	7	11	4	0.8 hours	1/150 sec	117 ms
Mixed	(Fig. 11)	118K	77K	151K	2	16	6	0.6 hours	1/150 sec	121 ms
Make a knot	(Fig. 13)	277K	78K	91K	4	37	5	4.2 hours	1/200 sec	335 ms
Twisting	(Fig. 14)	885K	107K	204K	6	16	4	3.7 hours	1/200 sec	672 ms
Puffer ball on the knit	(Fig. 15)	31K	120K(207K)	140K(414K)	1	13	2	0.6 hours	1/250 sec	1132 ms
Yangge dance	(Fig. 1)	3.0M	342K	389K	14	47	10	41.2 hours	1/150 sec	604 ms
Short-sleeve Yangge	(Fig. 16)	2.7M	329K	361K	10	36	10	35.3 hours	1/150 sec	588 ms
Long-sleeve Yangge	(Fig. 16)	2.6M	493K	420K	8	52	10	65.0 hours	1/150 sec	872 ms

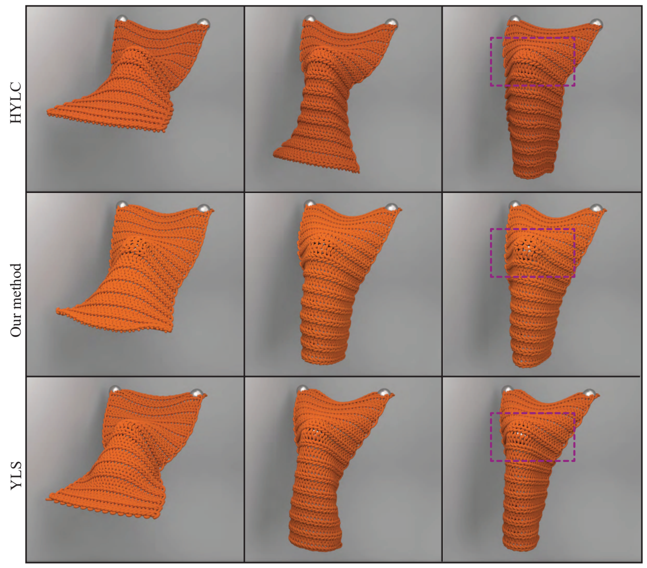


Fig. 9. ***Comparison with HYLC (draping).*** *We drape the pre-stretched ribbing patch on a sphere. Our method better replicates local deformation in the middle than HYLC. YLS simulation results are not used for elasticity fitting, showing our method's generalizability.*

As shown in Fig. 7, both HYLC and our method yield visually similar and plausible results for a knitted patch with a 1×1 rib pattern compared to YLS as the ground truth. The 1×1 rib pattern is highly stretchable due to the alternating knit and purl columns, allowing the fabric to expand and contract significantly across its width. However, HYLC, utilizing a single material across the entire mesh, causes the patch to narrow in the perpendicular directions when stretched, exhibiting a rubber-like behavior (see the right-most column in Fig. 7). The inconsistency between our method and HYLC becomes more apparent for the stockinette patch, which is less stretchy than the rib but tends to curl at the edges. While both sheet-based (HYLC) and volume-based (ours) materials display curly edges under stretch, HYLC is less expressive in capturing subtle deformation variations across the fabric due to the lack of material heterogeneity. In contrast, our method better replicates this phenomenon. We further test by draping a pre-stretched 1×1 rib patch over a sphere, demonstrating a similar difference to that shown in Fig. 8. Both HYLC and our method produce reasonable global deformation. However, our method, with a spatially varying material, better captures the inhomogeneous deformation at the contacting region. In this set of experiments, only two or three $\boldsymbol{x}_i^Y$ are used for elasticity fitting, and the draping simulation results are not observed during the elasticity fitting.

In terms of performance, the tetrahedron mesh used in our method consists of approximately 80K elements. This is a bigger number compared with HYLC, which simulates a triangle mesh of around 10K triangles. However, our constraint-based material model (Eq. 7) allows for a more stable simulation with a larger time step. The domain-decomposed PD solver completes these experiments at $\Delta t = 1/150$ sec, while HYLC must reduce to $\Delta t = 1/500$ sec (or even smaller) to ensure stability and prevent divergence due to material nonlinearity. Additionally, HYLC adaptively subdivides the mesh using `ArcSim` [Narain et al. 2013] to improve stability. Overall, our method is two orders faster than HYLC. The total training time is 0.5 hours, 3.1 hours, and 1.2 hours for Fig. 7, Fig. 8, and Fig. 9, respectively.

8.2 More patterns

We further compare our results with yarn-level simulation (YLS) using three additional knit patterns: basketweave, montague, and flame ribbing. Particularly, flame ribbing patterns exhibit anisotropic spatial variation with non-periodic yarn structures, making it challenging to fit a homogeneous material model using HYLC. In contrast,

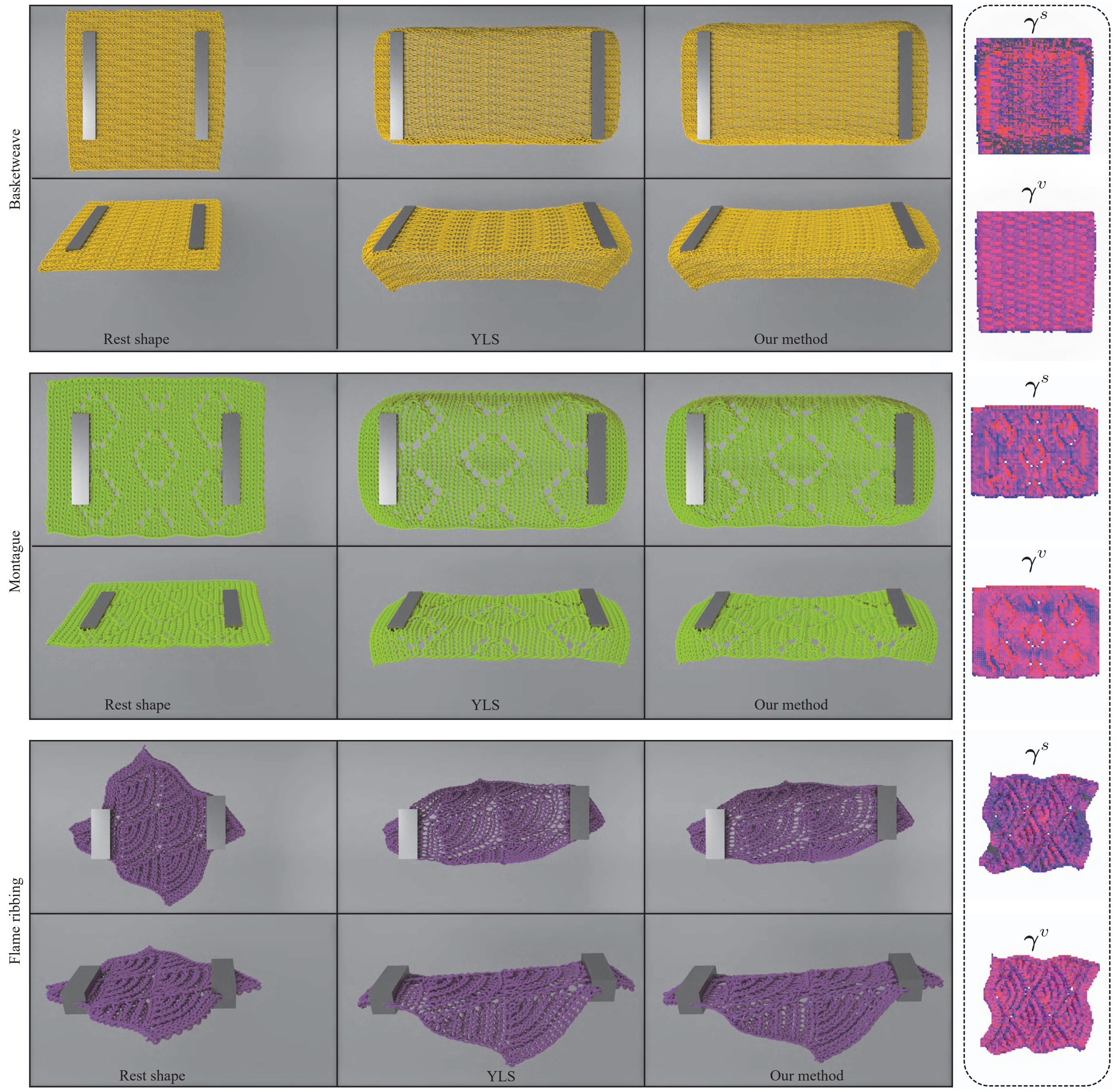


Fig. 10. ***More patterns.*** *We compare our results with YLS on basketweave, montague, and flame ribbing patterns. The leftmost column displays the rest shape of the knit patch, while the middle and right columns show the simulation results when the fabric is stretched using our method and YLS. We provide renderings from top and side views on the top and bottom rows, respectively. Our method accurately captures desired yarn-level deformations that vary across the patch and exhibit anisotropic behaviors. The material distributions of* $\boldsymbol{\gamma}^s$ *and* $\boldsymbol{\gamma}^v$ *are visualized on the right.*

our volumetric homogenization does not rely on any assumptions about the underlying arrangement of yarn structures. As shown in Fig. 10, our method produces high-fidelity knit stretching results that are nearly identical to YLS. As visualized on the right, the fitted material distributions align well with the underlying yarn structure and capture location-dependent deformations as expected.

Fig. 11 presents another experiment featuring a fabric composed of two distinct patterns: stockinette and 1×1 rib. With the same

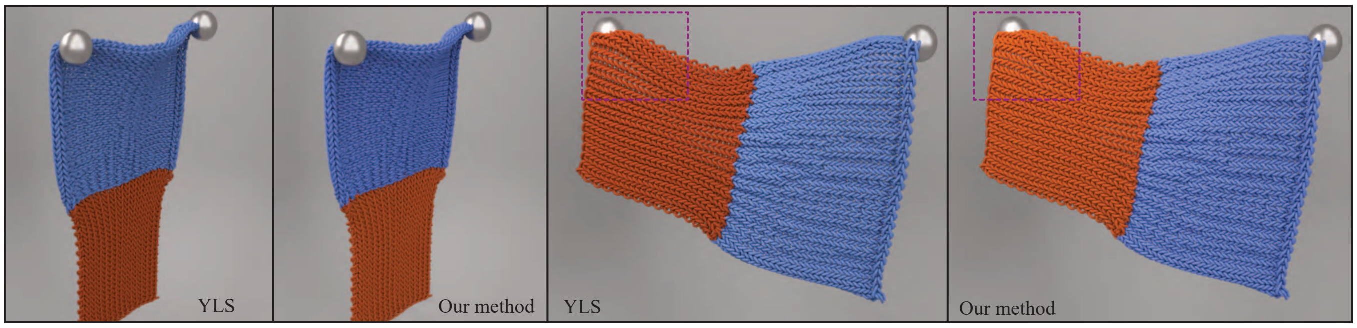


Fig. 11. ***Mixed patterns.*** *Our domain decomposed forward simulation is naturally compatible with garments made of combined patterns. In this example, we simulate a hanging fabric consisting of two patterns: knit and rib. One can still observe nuanced differences between our method and YLS, e.g., see highlighted areas. Nevertheless, homogenized simulation yields good animation results.*

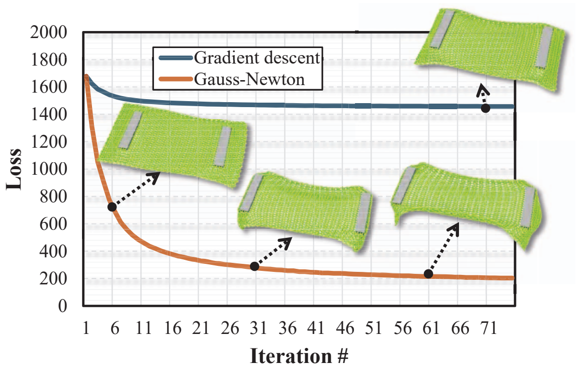


Fig. 12. ***Convergence plots.*** *We plot the converge curves using adjoint Gauss-Newton and gradient descent. The first-order method does not only converge properly, while second-order information in $\boldsymbol{G}_i$ (Eq. 21) effectively helps find a better descent direction and lower the loss function.*

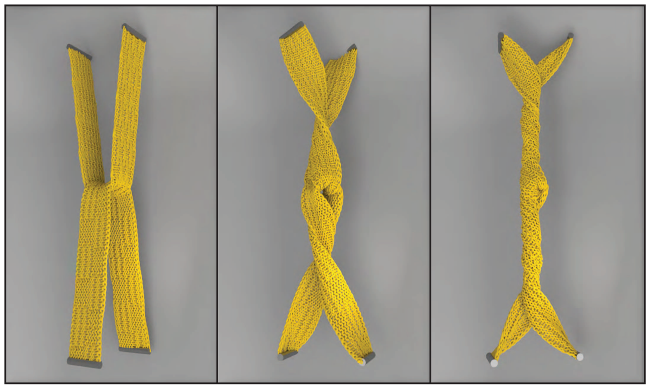


Fig. 13. ***Make a knot.*** *We rotate two knitted scripts (garter block pattern for the top and basketweave pattern for the bottom) for* 1, 920 *degrees to make a knot. There are* 78K *DOFs on the mesh, and homogenized simulation takes* 335 *ms for each time step ($\Delta t = 1/200$).*

number of stitches per row, the rib pattern shrinks due to the alternating knit and purl stitches, making it highly elastic and capable of significant expansion and contraction, while the stockinette is smooth and flat. This combined pattern poses a challenge to classic homogenization theory. However, our method remains effective in this case. Our domain-decomposed forward simulator naturally accommodates such pattern combinations. While YLS provides richer local details in this example, our method still produces reasonably good results. In this case, we have 77K DOFs on the mesh, and the simulation takes 121 ms for one time step, which is $\sim 80\times$ faster than running the simulation at the yarn level.

8.3 Convergence

The key to a successful volumetric homogenization is to solve the inverse problem using high-order optimization techniques i.e., the adjoint Gauss-Newton method discussed in Sec. 6. We note that most existing gradient-based methods, either using adjoint method or using AD (automatic differentiation) fail to converge in our case. We plot the convergence curves using adjoint Gauss-Newton and gradient descent for a representative elasticity fitting instance and show the result in Fig. 12. We also visualize the deformed mesh that satisfies Eq. 14 at three different iterations. A consistent observation is that first-order methods are never going to work. They frequently get trapped at local minima. A more severe issue is the step size of the gradient-based method is not stable. Performing line search is expensive in elasticity fitting – any proposed material update $\Delta\boldsymbol{\gamma}$ can only be checked when Eq. 14 is satisfied, and therefore a forward simulation procedure is invoked. This makes the first-order method prohibitive in practice. For instance, it could take over one week if we choose to use the first-order adjoint method or differentiable simulation to optimize the rib pattern (Fig. 7). On the other hand, Gauss-Newton finishes the training in about three hours.

8.4 Comparision at Different Resolution

The resolution of our volumetric material significantly influences the fitting capability of our method. We compare our method's performance on basketweave material at various resolutions as shown in Fig. 17. The center of the basketweave patch was fixed with a sphere. While the boundary condition exhibits rotational symmetry, the basketweave demonstrates distinct bending behaviors between the course and wale directions. When the mesh resolution is close

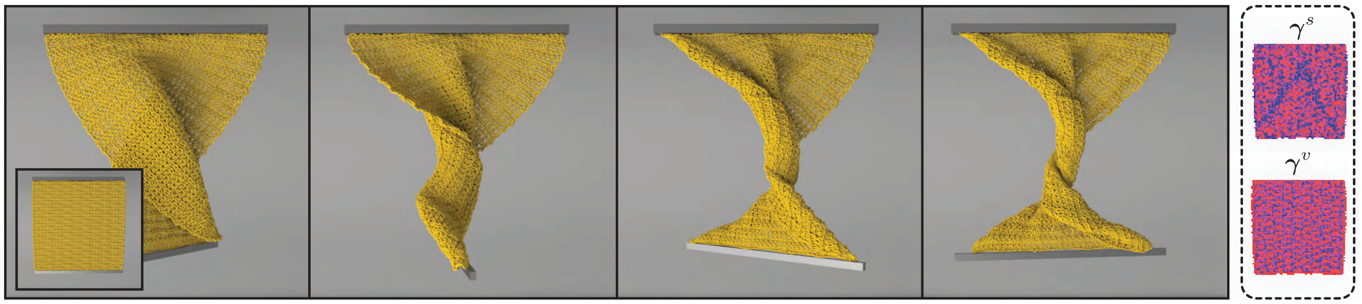


Fig. 14. ***Twisting.*** *We twist a square basketweave fabric for 900 degrees. In this example, the final deformed shape of our method does not perfectly align with the YLS result (please refer to the supplementary video for the animated simulation result). This is because the shape discrepancy accumulates at each frame, leading to different fabric-fabric collisions during the twisting. While not exactly same as YLC, our results are natural and realistic. There are 107K DOFs on the mesh. The homogenized simulation takes 672 ms for each time step ($\Delta t = 1/200$) and is $\sim 120\times$ faster.*

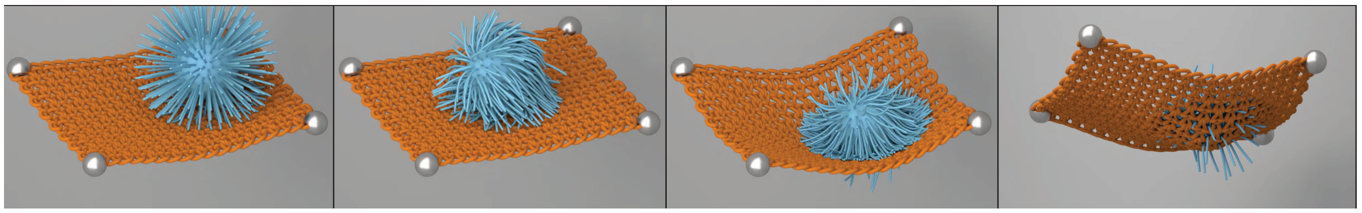


Fig. 15. ***Puffer ball on the knit.*** *We separate the processing of energy constraint (on the mesh) and collision constraint (at the yarn level) so that the homogenized model interacts as a yarn-level model. In this example, a puffer ball falls on a stockinette fabric. The hairs on the puffer ball pass through the gaps on the knit. They are two-way coupled under collision constraints and can be conveniently processed with our simulation.*

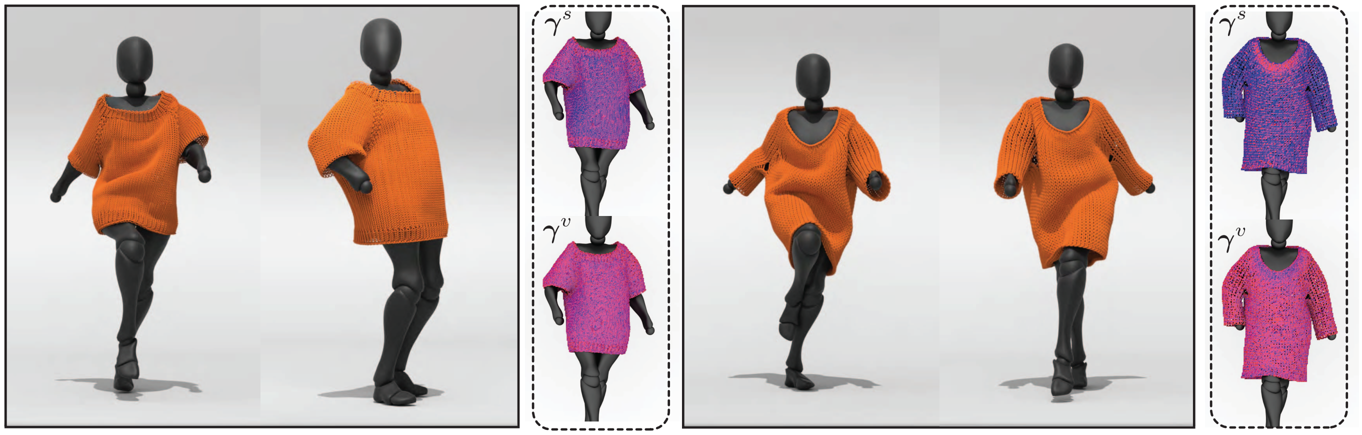


Fig. 16. ***Short- and long-sleeve Yangge.*** *In addition to Fig. 1, we show two more examples of full garment animations. On the left, the character wears a knitted short-sleeve sweater. It has a different pattern (stockinette pattern) than the one shown in Fig. 1, which yields different garment dynamics. On the right, the character wears a long-sleeve sweater in a stockinette pattern. The material distributions are visualized on the right.*

to or higher than that of the knit pattern, our approach produces results nearly identical to the ground truth. When the mesh resolution is significantly lower than that of the knit pattern, we still observe anisotropic bending behavior, but the distinction between the bending behaviors in the course and wale directions is not as sharp as in the ground truth.

8.5 More Results

We test our method in several complex scenes. Figs. 13 and 14 report two challenging simulation scenarios involving intensive twisting and bending. In Fig. 13, we rotate two fabric strips (basketweave) for about 1, 920 degrees. They are tightly intertwined to form a knot. In this case, volumetric homogenization yields a stronger volume-preserving penalty than examples shown in Fig. 10. We would like

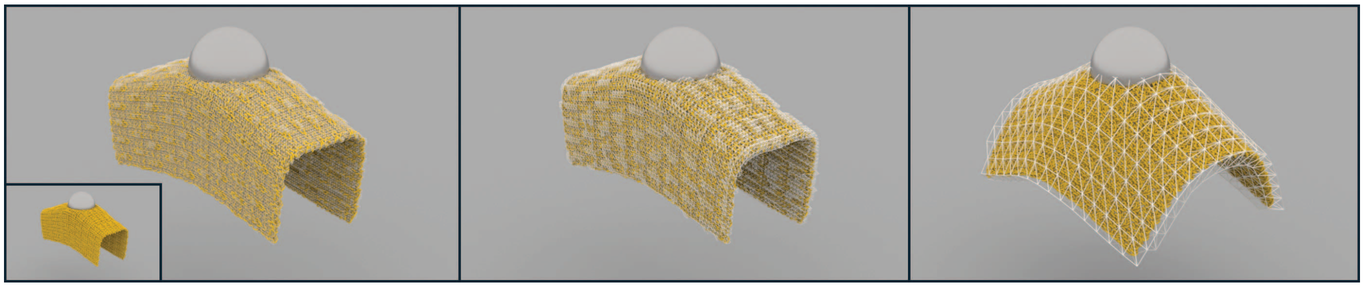


Fig. 17. ***Comparison at different resolution.*** *We compare our method's fitting capabilities across varying mesh resolutions by presenting an instance of a basketweave fabric draped over a sphere. Each panel displays the fabric fitting result at a different level of resolution. The smallest thumbnail inset in the bottom left panel provides the ground truth. The middle and the leftmost panels, where each cell's size is similar to or smaller than the size of each loop structure of threads, show fitting results that are essentially consistent with the ground truth. However, the rightmost panel, with a much coaser resolution, displays anisotropic bending effects macroscopically but significantly loses sharp bending details.*

to mention that while the homogenized simulation uses the tetrahedron mesh for energy integration and elasticity constraint projection, we can still process collision at yarn segments if needed (e.g., see Fig. 15). As most yarn-yarn contacts are inelastic, we follow the collision projection strategy as in vanilla PD [Bouaziz et al. 2014] for each edge-edge collision. The collision impulse is then interpolated to the corresponding tetrahedron [Lan et al. 2022a]. For collision detection, to enhance performance, we adopt the DCD (Discrete collision detection) strategy, noting that edge-edge collision pairs that exist in the same cell or adjacent cells can be excluded in advance. Of course, if we aim for strict non-penetration, we can adopt CCD (Continuous collision detection) with the interior point method as in [Li et al. 2021]. Although in our experiments we used the DCD collision detection method, which does not have guarantees and may eventually result in inter-penetration, we find that our homogenized animations remain visually plausible. Fig. 14 showcases an animation of twisting a wider fabric. Similar to Fig. 13, our method faithfully captures the twisting behaviors of such a complicated yarn structure using our volume-preserving energy. The YLS result can be found in the supplementary video. There are 91K elements in Fig. 13 and 204K elements in Fig. 14. The simulation uses 335 ms for one time step on average in Fig. 13 and 672 ms in Fig. 14.

Fig. 16 and Fig. 1 show the animation of three different knit garments on a virtual character performing Yangge dance. Different patterns on the front panels of the garments lead to interesting and distinctive animation effects even under the same character motion. Our homogenized simulations well replicate the YLS results but are two orders faster. In these three examples, we use ten poses for the elasticity training. The resulting material distributions are also visualized in the figures.

8.6 Generalizability

Most of our experiment's training data is directly sampled from the animation sequence. However, in Fig. 9, the draping test of the pre-stretched patch uses training data based on lateral stretching deformation. The final result shows a similar draping behavior to YLS, demonstrating the generalizability of our method to some extent. Here, we provide additional experiments to further explore the

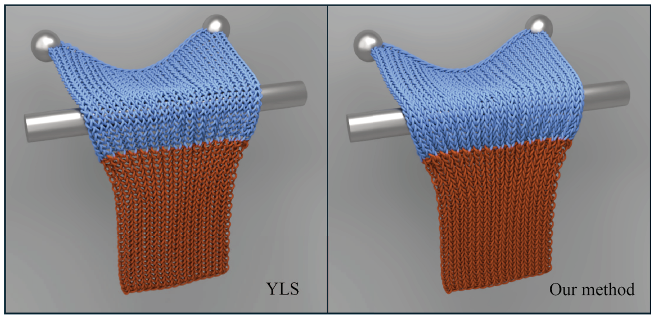


Fig. 18. ***Mixed patterns (draping).*** *We drape the mixed pattern on a cylinder, reusing the material of the mixed pattern trained in Fig. 11 Our method exhibits almost the same behavior as YLS.*

generalizability of our method. Fig. 18 demonstrates the draping of mixed patterns on a cylindrical form, using the material of the mixed pattern trained as shown in Fig. 11, the homogenized simulation exhibits almost the same behavior. Besides sample training data from the animation sequence, we can also follow the data generation protocol as in HYLC to sample possible deformations. In Fig. 19, we generate five types of deformation poses: Panel 1 shows a simple upward bend along the central axis, creating a smooth concave shape; Panel 2 presents a diagonal bend, where the material curves along the diagonal axis; Panel 3 illustrates a fan-like stretch configuration; Panel 4 depicts a twisting deformation that forms a helicoidal structure; and Panel 5 features a more complex, multi-folded bend with several folds along the material. Each of these poses, along with their symmetric counterparts, progressively composes the training set used to train the material distribution, as shown in Fig. 20. As the training data set increases, the behavior becomes closer to that of YLS. It is important to note that the training dataset does not include any data from the YLS animation sequence shown in Fig. 20.

9 CONCLUSION & LIMITATION

This paper explores a different perspective on improving the simulation of knitwear with complex knitting patterns. We name our method volumetric homogenization because it enables a volumetric and spatially heterogeneous material synthesis. The advantages

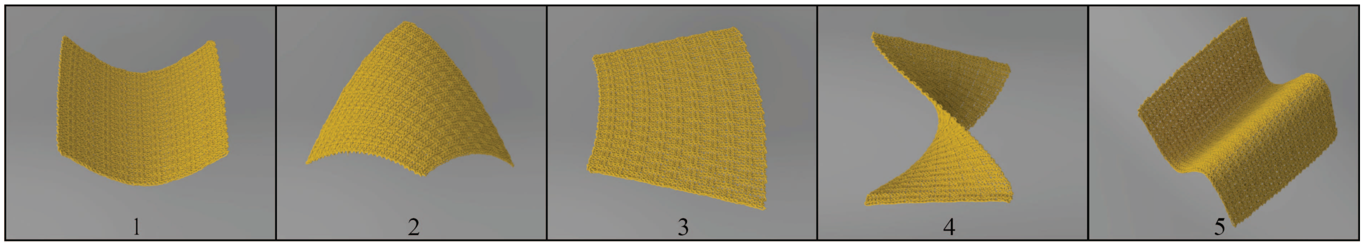


Fig. 19. ***More poses.*** *We follow the data generation protocol outlined in HYLC to sample five types of deformation, including their symmetric counterparts. The first two types are bending along the central axis and diagonal axis. The third deformation is a fan-like expansion. For the fourth type, the fabric is subject to a twist that forms a helix, we include the multi-folded deformation featuring several folds along the material.*

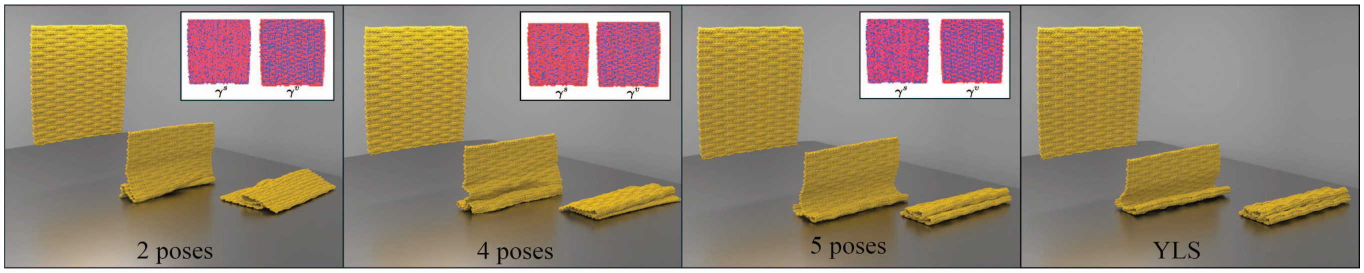


Fig. 20. ***Dropping to ground.*** *We utilize the training data generated from Fig. 19 and compare the dropping behavior with different training sets. Initially, only the first two bending types are used as our training data. In the subsequent panel, the fan-like expansion and twist types are also included. All deformation types in Fig. 19 are added in the third panel. The corresponding material distributions are shown in the top-right corner of each panel. The YLS is displayed in the rightmost column.*

of this strategy are multifold. Volume materials implicitly handle the bending and twisting and reduce the nonlinearity of the material property. It allows a faster simulation algorithm and, in turn, benefits the fitting efficiency. Assigning each volume element an independent set of material parameters effectively enhances the versatility of simulation so that our method is able to capture subtle and visually pleasing local deformations of complex knittings. With a domain-decomposed PD solver, our method is orders of magnitude faster than full-scale YLS.

While volumetric homogenization shows make some non-trivial improvements over existing methods, it still has drawbacks and limitations. We leverage shape fitting to extract the acceleration of YLS sequences and to isolate the inertia effects during the material learning. For fast-moving scenes however, the estimated inertia and lumped mass may still differ from the reference. Yarn-level cloth models are often highly dampened. This is because the friction and contacts among yarn threads dissipate inertia energy quickly. It is challenging to calibrate a homogenized garment with full-scale YLS using commonly seen macroscopic damping models e.g., Rayleigh damping. Being a data-driven method, our method could generate different results of the same knitwear given different YLS inputs. It is possible to learn a more sophisticated strain-stress model as in [Sperl et al. 2020] at each element. Doing so requires significant computations, and the resulting material becomes strongly nonlinear (again). In other words, it remains an open question to find the right compromise between material nonlinearity, the level of visual realism, and computational efficiency.

ACKNOWLEDGMENTS

We thank reviewers for their detailed and constructive comments. Yin Yang is partially supported by NSF under grant numbers 2301040, 2008915, 2244651, 2008564. Chenfanfu Jiang is supported in part by NSF CAREER 2153851, CCF 2153863, ECCS-2023780.

REFERENCES

Grégoire Allaire and Robert Brizzi. 2005. A multiscale finite element method for numerical homogenization. *Multiscale Modeling & Simulation* 4, 3 (2005), 790–812.

Erik Andreassen and Casper Schousboe Andreasen. 2014. How to determine composite material properties using numerical homogenization. *Computational Materials Science* 83 (2014), 488–495.

David Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*. Association for Computing Machinery, New York, NY, USA, 23–34.

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.

David Baraff, Andrew Witkin, and Michael Kass. 2003. Untangling cloth. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 862–870.

Klaus-Jürgen Bathe. 2006. *Finite element procedures.* Klaus-Jurgen Bathe, MA, USA.

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete viscous threads. *ACM Transactions on graphics (TOG)* 29, 4 (2010), 1–10.

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.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete elastic rods. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 63, 12 pages.

Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2021. PBNS: physically based neural simulation for unsupervised garment pose space deformation. *ACM Trans. Graph.* 40, 6, Article 198 (dec 2021), 14 pages.

Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2022. Neural cloth simulation. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14.

Pablo J Blanco, Pablo J Sánchez, Eduardo A de Souza Neto, and Raúl A Feijóo. 2016. Variational foundations and generalized unified theory of RVE-based multiscale models. *Archives of Computational Methods in Engineering* 23 (2016), 191–253.

Jeff Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröder. 2003. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *ACM transactions on graphics (TOG)* 22, 3 (2003), 917–924.

Léon Bottou et al. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes* 91, 8 (1991), 12.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33, 4, Article 154 (jul 2014), 11 pages.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.* 21, 3 (jul 2002), 594–603.

R. Bridson, S. Marino, and R. Fedkiw. 2003. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Goslar, DEU, 28–36.

Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. A multiresolution framework for dynamic 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, 41–47.

Juan J. Casafranca, Gabriel Cirio, Alejandro Rodríguez, Eder Miguel, and Miguel A. Otaduy. 2020. Mixing Yarns and Triangles in Cloth Simulation. *Computer Graphics Forum* 39, 2 (2020), 101–110.

Desai Chen, David IW Levin, Wojciech Matusik, and Danny M Kaufman. 2017. Dynamics-aware numerical coarsening for fabrication design. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–15.

Desai Chen, David IW Levin, Shinjiro Sueda, and Wojciech Matusik. 2015. Data-driven finite elements for geometry and material design. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

Jiong Chen, Hujun Bao, Tianyu Wang, Mathieu Desbrun, and Jin Huang. 2018. Numerical coarsening using discontinuous shape functions. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Yunuo Chen, Tianyi Xie, Cem Yuksel, Danny Kaufman, Yin Yang, Chenfanfu Jiang, and Minchen Li. 2023. Multi-Layer Thick Shells. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 25, 9 pages.

Kwang-Jin Choi and Hyeong-Seok Ko. 2002. Stable but responsive cloth. *ACM Trans. Graph.* 21, 3 (jul 2002), 604–611.

Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. 2014. Yarn-level simulation of woven cloth. *ACM Trans. Graph.* 33, 6, Article 207 (nov 2014), 11 pages.

Gabriel Cirio, Jorge Lopez-Moreno, and Miguel A. Otaduy. 2015. Efficient simulation of knitted cloth using persistent contacts. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '15)*. Association for Computing Machinery, New York, NY, USA, 55–61.

David Clyde, Joseph Teran, and Rasmus Tamstorf. 2017. Modeling and Data-Driven Parameter Estimation for Woven Fabrics. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '17)*. Association for Computing Machinery, New York, NY, USA, Article 17, 11 pages.

Roy R Craig Jr. 1985. A review of time-domain and frequency-domain component mode synthesis method. (1985).

Roy R Craig Jr and Arthur L Hale. 1988. Block-Krylov component synthesis method for structural model reduction. *Journal of Guidance, Control, and Dynamics* 11, 6 (1988), 562–570.

Eduardo Alberto De Souza Neto, Pablo Javier Blanco, Pablo Javier Sánchez, and Raúl Antonino Feijóo. 2015. An RVE-based multiscale theory of solids with micro-scale inertia and body force effects. *Mechanics of Materials* 80 (2015), 136–144.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (TOG)* 41, 2 (2021), 1–21.

Elliot English and Robert Bridson. 2008. Animating developable surfaces using nonconforming elements. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–5.

Hans A Eschenauer and Niels Olhoff. 2001. Topology optimization of continuum structures: a review. *Appl. Mech. Rev.* 54, 4 (2001), 331–390.

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.

Xudong Feng, Weiwei Xu, Huamin Wang, and Yin Yang. 2024. Neural-Assisted Homogenization of Yarn-Level Cloth. *ACM Transactions on Graphics (TOG)* 34, 4 (2024), 1–11.

Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: A Practical Gauss-Seidel Method for Stable Soft Body Dynamics. *ACM Trans. Graph.* 35, 6, Article 214 (Nov. 2016), 9 pages.

Akash Garg, Eitan Grinspun, Max Wardetzky, and Denis Zorin. 2007. Cubic shells. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 91–98.

Marc GD Geers, Varvara G Kouznetsova, and WAM1402 Brekelmans. 2010. Multi-scale computational homogenization: Trends and challenges. *Journal of computational and applied mathematics* 234, 7 (2010), 2175–2182.

Dan Givoli. 2021. A tutorial on the adjoint method for inverse problems. *Computer Methods in Applied Mechanics and Engineering* 380 (2021), 113810.

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient simulation of inextensible cloth. *ACM Trans. Graph.* 26, 3 (jul 2007), 49–es.

F. Sebastin Grassia. 1998. Practical parameterization of rotations using the exponential map. *J. Graph. Tools* 3, 3 (mar 1998), 29–48. https://doi.org/10.1080/10867651.1998.10487493

Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Goslar, DEU, 62–67.

Eitan Grinspun, Petr Krysl, and Peter Schröder. 2002. CHARMS: A simple framework for adaptive simulation. *ACM transactions on graphics (TOG)* 21, 3 (2002), 281–290.

Gaël Guennebaud, Benoit Jacob, et al. 2010. Eigen. *URl: http://eigen. tuxfamily. org* 3, 1 (2010).

Qi Guo, Xuchen Han, Chuyuan Fu, Theodore Gast, Rasmus Tamstorf, and Joseph Teran. 2018. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15.

David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. 2008. Robust Treatment of Simultaneous Collisions. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 23, 4 pages.

Florian Hecht, Yeon Jin Lee, Jonathan R Shewchuk, and James F O'Brien. 2012. Updated sparse cholesky factors for corotational elastodynamics. *ACM Transactions on Graphics (TOG)* 31, 5 (2012), 1–13.

Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2008. Simulating knitted cloth at the yarn level. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–9.

Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2010. Efficient yarn-based cloth with adaptive contact linearization. *ACM Trans. Graph.* 29, 4, Article 105 (jul 2010), 10 pages.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) *(SGP '06)*. Eurographics Association, Goslar, DEU, 61–70.

Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Transactions on graphics (TOG)* 28, 3 (2009), 1–8.

Theodore Kim. 2020. A Finite Element Formulation of Baraff-Witkin Cloth. *Computer Graphics Forum* 39, 8 (2020), 171–179.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. ICLR, San Diega, CA, USA, 15 pages.

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.

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.

Jonathan Leaf, Rundong Wu, Eston Schweickart, Doug L James, and Steve Marschner. 2018. Interactive design of periodic yarn-level cloth patterns. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15.

Yongjoon Lee, Sung-eui Yoon, Seungwoo Oh, Duksu Kim, and Sunghee Choi. 2010. Multi-Resolution Cloth Simulation. *Computer Graphics Forum* (2010). https://doi.org/10.1111/j.1467-8659.2010.01811.x

Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E. Brown, and Laurence Boissieux. 2018. An Implicit Frictional Contact Solver for Adaptive Cloth Simulation. *ACM Trans. Graph.* 37, 4, Article 52 (July 2018), 15 pages.

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional incremental potential contact. *ACM Trans. Graph.* 40, 4, Article 170 (jul 2021), 24 pages.

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2022. DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact. *ACM Trans. Graph.* 42, 1, Article 2 (oct 2022), 20 pages.

Junbang Liang, Ming Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., Vancouver, Canada.

Chenchen Liu and Celia Reina. 2016. Discrete averaging relations for micro to macro transition. *Journal of Applied Mechanics* 83, 8 (2016), 081006.

Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–7.

E. Miguel, D. Bradley, B. Thomaszewski, B. Bickel, W. Matusik, M. A. Otaduy, and S. Marschner. 2012. Data-Driven Estimation of Cloth Simulation Models. *Comput. Graph. Forum* 31, 2pt2 (May 2012), 519–528.

Eder Miguel, Rasmus Tamstorf, Derek Bradley, Sara C. Schvartzman, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Steve Marschner, and Miguel A. Otaduy. 2013. Modeling and Estimation of Internal Friction in Cloth. *ACM Trans. Graph.* 32, 6, Article 212 (Nov. 2013), 10 pages.

Jorge J Moré. 2006. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*. Springer Berlin, Heidelberg, Heidelberg, Germany, 105–116.

Rahul Narain, Tobias Pfaff, and James F O'Brien. 2013. Folding and crumpling adaptive sheets. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–8.

Rahul Narain, Armin Samii, and James F O'brien. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM transactions on graphics (TOG)* 31, 6 (2012), 1–10.

Vidya Narayanan, Kui Wu, Cem Yuksel, and James McCann. 2019. Visual knitting machine programming. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–13.

Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. 2018. Fast Approximation of Laplace-Beltrami Eigenproblems. *Computer Graphics Forum* 37, 5 (2018), 121–134.

Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. 2009. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph.* 28, 3, Article 52 (jul 2009), 9 pages.

Jorge Nocedal and Stephen J Wright. 1999. *Numerical optimization.* Springer, New York, NY, USA.

Miguel A. Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. 2009. Implicit Contact Handling for Deformable Objects. *Computer Graphics Forum* 28, 2 (2009), 559–568.

Dinesh K. Pai. 2002. STRANDS: Interactive Simulation of Thin Solids using Cosserat Models. *Computer Graphics Forum* 21, 3 (2002), 347–352.

Julian Panetta, Qingnan Zhou, Luigi Malomo, Nico Pietroni, Paolo Cignoni, and Denis Zorin. 2015. Elastic textures for additive fabrication. *ACM Trans. Graph.* 34, 4 (2015), 135–1.

Yue Peng, Bailin Deng, Juyong Zhang, Fanyu Geng, Wenjie Qin, and Ligang Liu. 2018. Anderson acceleration for geometry optimization and physics simulation. *ACM Trans. Graph.* 37, 4, Article 42 (jul 2018), 14 pages. https://doi.org/10.1145/3197517.3201290

Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J Black. 2017. ClothCap: Seamless 4D clothing capture and retargeting. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–15.

Xavier Provot. 1995. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. In *Proceedings of Graphics Interface '95* (Quebec, Quebec, Canada) *(GI '95)*. Canadian Human-Computer Communications Society, Toronto, Ontario, Canada, 147–154.

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.

Rosa M. Sánchez-Banderas, Alejandro Rodríguez, Héctor Barreiro, and Miguel A. Otaduy. 2020. Robust eulerian-on-lagrangian rods. *ACM Trans. Graph.* 39, 4, Article 59 (aug 2020), 10 pages.

Igor Santesteban, Miguel A. Otaduy, and Dan Casas. 2019. Learning-Based Animation of Clothing for Virtual Try-On. *Computer Graphics Forum* 38, 2 (2019), 355–366.

Igor Santesteban, Miguel A Otaduy, and Dan Casas. 2022. Snug: Self-supervised neural dynamic garments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 8140–8150.

Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner, and Andreas Stricker. 2001. PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems* 18, 1 (2001), 69–78.

Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. 2015. Microstructures to control elasticity in 3D printing. *ACM Transactions on Graphics (Tog)* 34, 4 (2015), 1–13.

Georg Sperl, Rahul Narain, and Chris Wojtan. 2020. Homogenized yarn-level cloth. *ACM Trans. Graph.* 39, 4 (2020), 48.

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 Trans. Graph.* 41, 4, Article 65 (jul 2022), 15 pages.

J. Spillmann and M. Teschner. 2007. CoRdE: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 63–72.

Rasmus Tamstorf, Toby Jones, and Stephen F McCormick. 2015. Smoothed aggregation multigrid for cloth simulation. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–13.

Albert Tarantola. 2005. *Inverse Problem Theory and Methods for Model Parameter Estimation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically deformable models. *SIGGRAPH Comput. Graph.* 21, 4 (aug 1987), 205–214.

Rosell Torres, Alejandro Rodríguez, José M Espadero, and Miguel A Otaduy. 2016. High-resolution interaction with corotational coarsening models. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–11.

Bruno Vallet and Bruno Lévy. 2008. Spectral Geometry Processing with Manifold Harmonics. *Computer Graphics Forum* 27, 2 (2008), 251–260.

Pascal Volino and Nadia Magnenat Thalmann. 2000. Implementing Fast Cloth Simulation with Collision Response. In *Proceedings of the International Conference on Computer Graphics (CGI '00)*. IEEE Computer Society, USA, 257.

Endong Wang, Qing Zhang, Bo Shen, Guangyong Zhang, Xiaowei Lu, Qing Wu, Yajuan Wang, Endong Wang, Qing Zhang, Bo Shen, et al. 2014. Intel math kernel library. *High-Performance Computing on the Intel® Xeon Phi™: How to Fully Exploit MIC Architectures* (2014), 167–188.

Huamin Wang. 2015. A Chebyshev Semi-Iterative Approach for Accelerating Projective and Position-Based Dynamics. *ACM Trans. Graph.* 34, 6, Article 246 (Oct. 2015), 9 pages.

Huamin Wang. 2018. Rule-free sewing pattern adjustment with precision and efficiency. *ACM Trans. Graph.* 37, 4, Article 53 (jul 2018), 13 pages. https://doi.org/10.1145/3197517.3201320

Huamin Wang, James O'Brien, and Ravi Ramamoorthi. 2010. Multi-resolution isotropic strain limiting. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 1–10.

Huamin Wang, James F. O'Brien, and Ravi Ramamoorthi. 2011. Data-Driven Elastic Models for Cloth: Modeling and Measurement. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 71, 12 pages.

Huamin Wang and Yin Yang. 2016. Descent Methods for Elastic Body Simulation on the GPU. *ACM Trans. Graph.* 35, 6, Article 212 (Nov. 2016), 10 pages.

Zhendong Wang, Longhua Wu, Marco Fratarcangeli, Min Tang, and Huamin Wang. 2018. Parallel Multigrid for Nonlinear Cloth Simulation. *Computer Graphics Forum* 37, 7 (2018), 131–141.

Zhendong Wang, Yin Yang, and Huamin Wang. 2023. Stable Discrete Bending by Analytic Eigensystem and Adaptive Orthotropic Geometric Stiffness. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–16.

Nicholas J Weidner, Kyle Piddington, David IW Levin, and Shinjiro Sueda. 2018. Eulerian-on-lagrangian cloth simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

Kui Wu, Hannah Swan, and Cem Yuksel. 2019. Knittable stitch meshes. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 1–13.

Kui Wu, Marco Tarini, Cem Yuksel, Jim McCann, and Xifeng Gao. 2021. Wearable 3D Machine Knitting: Automatic Generation of Shaped Knit Sheets to Cover Real-World Objects. *IEEE Transactions on Visualization & Computer Graphics* 1, 01 (feb 2021), 1–1.

Longhua Wu, Botao Wu, Yin Yang, and Huamin Wang. 2020. A Safe and Fast Repulsion Method for GPU-based Cloth Self Collisions. *ACM Trans. Graph.* 40, 1, Article 5 (dec 2020), 18 pages.

Zangyueyang Xian, Xin Tong, and Tiantian Liu. 2019. A scalable galerkin multigrid method for real-time simulation of deformable objects. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–13.

Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbič. 2015. Nonlinear material design using principal stretches. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

Yin Yang, Weiwei Xu, Xiaohu Guo, Kun Zhou, and Baining Guo. 2013. Boundary-aware multidomain subspace deformation. *IEEE transactions on visualization and computer graphics* 19, 10 (2013), 1633–1645.

Cem Yuksel, Jonathan M Kaldor, Doug L James, and Steve Marschner. 2012. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–12.

Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2021. SGN: Sparse Gauss-Newton for Accelerated Sensitivity Analysis. *ACM Trans. Graph.* 41, 1, Article 4 (sep 2021), 10 pages.

Yumin Zhang, Steven Garcia, Weiwei Xu, Tianjia Shao, and Yin Yang. 2018. Efficient voxelization using projected optimal scanline. *Graphical Models* 100 (2018), 61–70.

Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Transactions on Graphics (TOG)* 29, 2 (2010), 1–18.

Simon Zimmermann, Roi Poranne, James M. Bern, and Stelian Coros. 2019. PuppetMaster: robotic animation of marionettes. *ACM Trans. Graph.* 38, 4, Article 103 (jul 2019), 11 pages. https://doi.org/10.1145/3306346.3323003

Borut Žalik, Gordon Clapworthy, and Črtomir Oblonšek. 1997. An Efficient Code-Based Voxel-Traversing Algorithm. *Computer Graphics Forum* 16, 2 (1997), 119–128.