Homogenizing Yarn Simulations

Large-scale mechanics, small-scale detail, and quantitative fitting

by

Georg Sperl

September, 2022

A thesis submitted to the
Graduate School
of the
Institute of Science and Technology Austria
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Committee in charge:

Carl Goodrich, Chair
Chris Wojtan
Nils Thuerey
Rahul Narain
Miguel A. Otaduy

The thesis of Georg Sperl, titled *Homogenizing Yarn Simulations*, is approved by:

Supervisor: Chris Wojtan, Institute of Science And Technology Austria, Klosterneuburg, Austria

Signature: _____

Committee Member: Nils Thuerey, Technical University of Munich, Munich, Germany

Signature: _____

Committee Member: Rahul Narain, Indian Institute of Technology Delhi, Delhi, India

Signature: _____

Committee Member: Miguel A. Otaduy, Universidad Rey Juan Carlos, Madrid, Spain

Signature: _____

Defense Chair: Carl Goodrich, Institute of Science And Technology Austria, Klosterneuburg, Austria

Signature: _____

Signed page is on file

ISTA Thesis, ISSN: 2663-337X

ISBN: 978-3-99078-020-6

I hereby declare that this thesis is my own work and that it does not contain other people's work without this being so stated; this thesis does not contain my previous work without this being stated, and the bibliography contains all the literature that I used in writing the dissertation.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution.

I certify that any republication of materials presented in this thesis has been approved by the relevant publishers and co-authors.

Signature: _____

Georg Sperl
September, 2022

Signed page is on file

Abstract

The complex yarn structure of knitted and woven fabrics gives rise to both a mechanical and visual complexity. The small-scale interactions of yarns colliding with and pulling on each other result in drastically different large-scale stretching and bending behavior, introducing anisotropy, curling, and more. While simulating cloth as individual yarns can reproduce this complexity and match the quality of real fabric, it may be too computationally expensive for large fabrics. On the other hand, continuum-based approaches do not need to discretize the cloth at a stitch-level, but it is non-trivial to find a material model that would replicate the large-scale behavior of yarn fabrics, and they discard the intricate visual detail. In this thesis, we discuss three methods to try and bridge the gap between small-scale and large-scale yarn mechanics using numerical homogenization: fitting a continuum model to periodic yarn simulations, adding mechanics-aware yarn detail onto thin-shell simulations, and quantitatively fitting yarn parameters to physical measurements of real fabric.

To start, we present a method for animating yarn-level cloth effects using a thin-shell solver. We first use a large number of periodic yarn-level simulations to build a model of the potential energy density of the cloth, and then use it to compute forces in a thin-shell simulator. The resulting simulations faithfully reproduce expected effects like the stiffening of woven fabrics and the highly deformable nature and anisotropy of knitted fabrics at a fraction of the cost of full yarn-level simulation.

While our thin-shell simulations are able to capture large-scale yarn mechanics, they lack the rich visual detail of yarn-level simulations. Therefore, we propose a method to animate yarn-level cloth geometry on top of an underlying deforming mesh in a mechanics-aware fashion in real time. Using triangle strains to interpolate precomputed yarn geometry, we are able to reproduce effects such as knit loops tightening under stretching at negligible cost.

Finally, we introduce a methodology for inverse-modeling of yarn-level mechanics of cloth, based on the mechanical response of fabrics in the real world. We compile a database from physical tests of several knitted fabrics used in the textile industry spanning diverse physical properties like stiffness, nonlinearity, and anisotropy. We then develop a system for approximating these mechanical responses with yarn-level cloth simulation, using homogenized shell models to speed up computation and adding some small-but-necessary extensions to yarn-level models used in computer graphics.

Acknowledgements

The research in this thesis has been accomplished thanks to the many people who supported me both professionally and otherwise, not only during the PhD but also on the way there.

First and foremost, I want to thank Chris, without whom I would not have had such a successful PhD and a great time. He has been an excellent mentor, boss, and friend to me. Always willing to help and give encouragement, his advice and continuous support shaped my efforts into publishable research projects.

I thank my committee for their time, effort, and valuable feedback: Rahul, who was my de facto co-supervisor during the first two papers, with his instant understanding of any problem; Miguel, who took a similar role during the third paper, encouraging and undeterred by the difficulty of real-life measurements; Bernd, who helped me get up to speed with continuum mechanics when I started from zero; and Nils, for discussions during joined group retreats, the qualifying exam and the defense.

Thank you to my other co-authors, Manwen and especially Rosa, for her dedication and efforts, unstoppable even with Covid. Thank you also to the anonymous reviewers for important feedback to add that final polish to our publications.

I'm grateful to my friends and co-workers at ISTA and in the Bickel and Wojtan groups, foremost Tomáš for constructive criticism and discussions on physics, code, and more; but also Ruslan, Christian, Mickaël, and many more for helpful scientific discussion.

I would like to also thank my friends outside ISTA: my "Wöd", for showing me that there is fun to be found away from my computer; as well as my family, for always supporting me; and in particular my brother Peter, for being my first and most creative role model and for enjoying (and "winning") several game jams together.

Finally, a big thank you to my wife Uschi, for her endless love and support during tight deadlines.

This research was supported by the Scientific Service Units (SSU) of ISTA, with the resources and help to set up hundreds of parallel simulations. Financially, the publications in this thesis were partially supported by a grant from the European Research Council (ERC) under grant agreements No. 638176 and No. 772738, as well as a Pankaj Gupta Young Faculty Fellowship and a gift from Adobe Inc.

About the Author

Georg Sperl completed a BSc and a "Diplom-Ingenieur" (equivalent to MSc) in Visual Computing at the Technical University of Vienna. He started his PhD at ISTA in September 2016 to pursue his research interests in physics-based animation and simulation. He later joined the group of Chris Wojtan, where he worked on research related to homogenization and the connection between small-scale vs. large-scale physics with a focus on yarn-based fabrics. During his PhD, he published and presented three papers at the top computer graphics conference SIGGRAPH, including an industry collaboration with SEDDI. He also interned as a simulation researcher at Weta Digital.

List of Collaborators and Publications

This thesis is based on three first-author publications of Georg Sperl. We list these publications below along with a summary of contributions from each author.

- Georg Sperl, Rahul Narain, and Chris Wojtan. Homogenized Yarn-Level Cloth. *ACM Transactions on Graphics (TOG)*, 39(4), 2020
  DOI: https://doi.org/10.1145/3386569.3392412
  Project Page: https://visualcomputing.ist.ac.at/publications/2020/HYLC/
  - This paper constitutes the core of Chapters 3 and 4.
  - Georg Sperl derived the boundary conditions for thin-shell homogenization and developed the method under guidance of Chris Wojtan and Rahul Narain. He designed, implemented, and executed experiments, wrote an initial paper draft and edited it.
  - Rahul Narain advised Georg Sperl throughout the project, providing important suggestions and discussion on methodology, experimental design, and paper writing. He also derived the computation of a rotation-matrix from fundamental forms, as well as the piecewise monotone bicubic interpolation algorithm.
  - Chris Wojtan advised Georg Sperl throughout the project and earlier exploration leading up to it. He provided important suggestions and discussion on methodology, experimental design, and paper writing.

- Georg Sperl, Rahul Narain, and Chris Wojtan. Mechanics-Aware Deformation of Yarn Pattern Geometry. *ACM Transactions on Graphics (TOG)*, 40(4), 2021
  DOI: https://doi.org/10.1145/3450626.3459816
  Project Page: https://visualcomputing.ist.ac.at/publications/2021/MADYPG/
  - This paper constitutes the core of Chapter 5.
  - Georg Sperl developed the method under guidance of Chris Wojtan and Rahul Narain. He designed, implemented, and executed experiments, wrote an initial paper draft and edited it.
  - Rahul Narain advised Georg Sperl throughout the project, providing important suggestions and discussion on methodology, experimental design, and paper writing.
  - Chris Wojtan advised Georg Sperl throughout the project, providing important suggestions and discussion on methodology, experimental design, and paper writing.

- Georg Sperl, Rosa M. Sánchez-Banderas, Manwen Li, Chris Wojtan, and Miguel A. Otaduy. Estimation of Yarn-Level Simulation Models for Production Fabrics. *ACM Transactions on Graphics (TOG)*, 41(4), 2022
  DOI: https://doi.org/10.1145/3528223.3530167
  Project Page: https://mslab.es/projects/YarnLevelFabrics/

  - This paper constitutes the core of Chapter 6.
  - Georg Sperl developed the yarn model and yarn parameter optimization under guidance of Miguel A. Otaduy and Chris Wojtan. He designed, implemented, and executed yarn-simulation-based experiments. He provided suggestions and discussion on the thin-shell fitting as well as data set preparation, and he contributed to paper writing.
  - Rosa M. Sánchez-Banderas developed and implemented the thin-shell optimization under guidance of Miguel A. Otaduy. She worked on data set preparation, including manual registration of all fabrics.
  - Manwen Li worked on collecting physical measurements and data set preparation.
  - Chris Wojtan advised Georg Sperl throughout the project, providing important suggestions and discussion on methodology, experimental design, and contributing to paper writing.
  - Miguel A. Otaduy advised Georg Sperl and Rosa M. Sánchez-Banderas throughout the project, providing important suggestions and discussion on methodology, experimental design. He wrote an initial paper draft and edited it.

Table of Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Knitted and woven fabrics can be constructed from simple yarns or wires, yet they span a wide range of material behavior. Fascinating material properties can arise entirely from the geometric structure. Figure 1.1 demonstrates how a *stockinette* knit (also called *plain knit* or *jersey*) is more compliant along one direction. This is caused by knit loops tightening and pulling in material from adjacent rows, bending and moving the yarns instead of stretching them, and so the fabric as a whole exhibits an *anisotropic* stiffness purely based on its geometry. The yarn topology of various patterns may result in dramatically different material behavior with respect to stiffness, anisotropy, area preservation, and more. See for example Figure 1.2, where a rib knit exhibits area-preservation behavior, and a stockinette knit curls on the boundary.

This variety in effective material behavior and the fact that these fabrics can be produced from simple yarns leads to knitted and woven fabric being ubiquitous in everyday life. Simultaneously, the large-scale behavior of yarn-based fabric is difficult to predict without a full simulation. Consequently, the simulation and analysis of such fabrics generated a great deal of research in the computer graphics [Kaldor et al. 2008; Cirio et al. 2014; Yuksel et al. 2012; Narayanan et al. 2018], materials science [Choi and Lo 2003; Fillep et al. 2017], and physics communities [Poincloux et al. 2018]. However, while simulating fabric as a collection of interacting threads can indeed both reproduce the inherently complex material behavior and also the beautiful visual complexity, this strategy is computationally expensive. Efficient algorithms for the handling of yarn-yarn contact [Kaldor et al. 2010], treating contacts as persistent [Cirio et al. 2014, 2016; Sánchez-Banderas et al. 2020], and GPU-based implementations [Leaf et al. 2018] can mitigate the cost substantially. Regardless, these methods still discretize a fabric on the scale of yarn crossings or knit loops and are thus fundamentally limited in terms of scalability.

On the other hand, spring-based [Choi and Ko 2005], constraint-based [Müller et al. 2007], or continuum-based [Terzopoulos et al. 1987; Baraff and Witkin 1998; Grinspun et al. 2003; Narain et al. 2012] thin-shell cloth solvers have been widely adopted in graphics research. These methods are free to discretize fabric at scales much larger than individual stitches, and as a result they are much more efficient. However, common continuum membrane and bending models fall short of reproducing the complex nonlinearities, area-preservation behavior, and even curling behavior that can arise naturally from yarn mechanics, and they also lose the visual detail carried by individual yarns.

Figure 1.1: Photo of a stockinette knit being pulled in different directions. The individual loops of the knit interact in such a way that the fabric as a whole is compliant along the weft direction (here: horizontal) and more stiff along other directions.

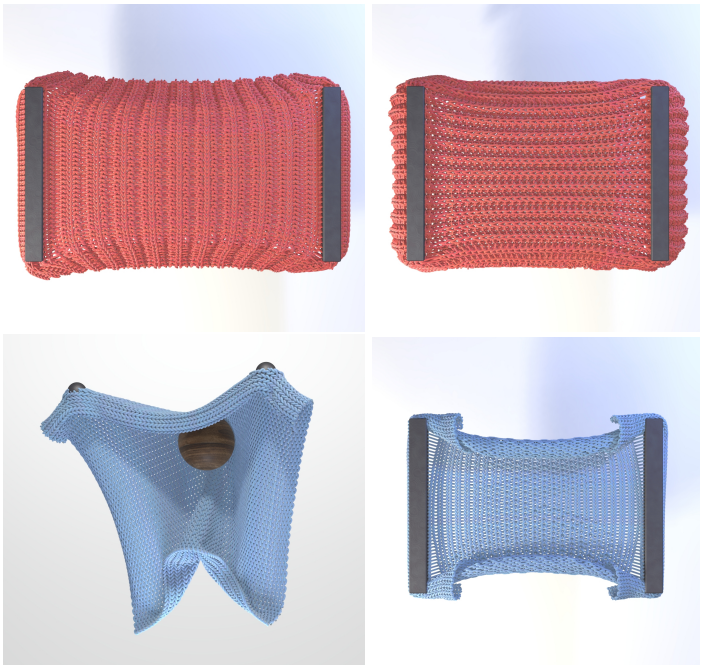


Figure 1.2: Large-scale phenomena can emerge from simulated yarn-level geometry. The rib pattern (top) exhibits anisotropy and a tendency to preserve area under tension, while the stockinette pattern (bottom) exhibits curling.

Thin-shell cloth simulation of knitted and woven fabric is effectively trying to predict the large-scale behavior emerging from the small-scale yarn structure. Abstracting the local detail into an *average* behavior is the essence of *homogenization*. This area of research has been adopted in the graphics community, e.g. for the purpose of analyzing rod structures [Schumacher et al. 2018], coarsening multi-material finite-element simulations [Kharevych et al. 2009; Chen et al. 2018a] and designing of metamaterials [Bickel et al. 2010; Schumacher et al. 2015].

1.1 Overview

In this thesis, we present three methods that combine yarn-level and thin-shell simulations via homogenization, see also Figure 1.3: fitting a continuum model to match the large-scale behavior of yarn-based fabrics [Sperl et al. 2020], adding back the local yarn-detail to match the visual complexity of knit loops tightening in real time [Sperl et al. 2021], and fitting yarn

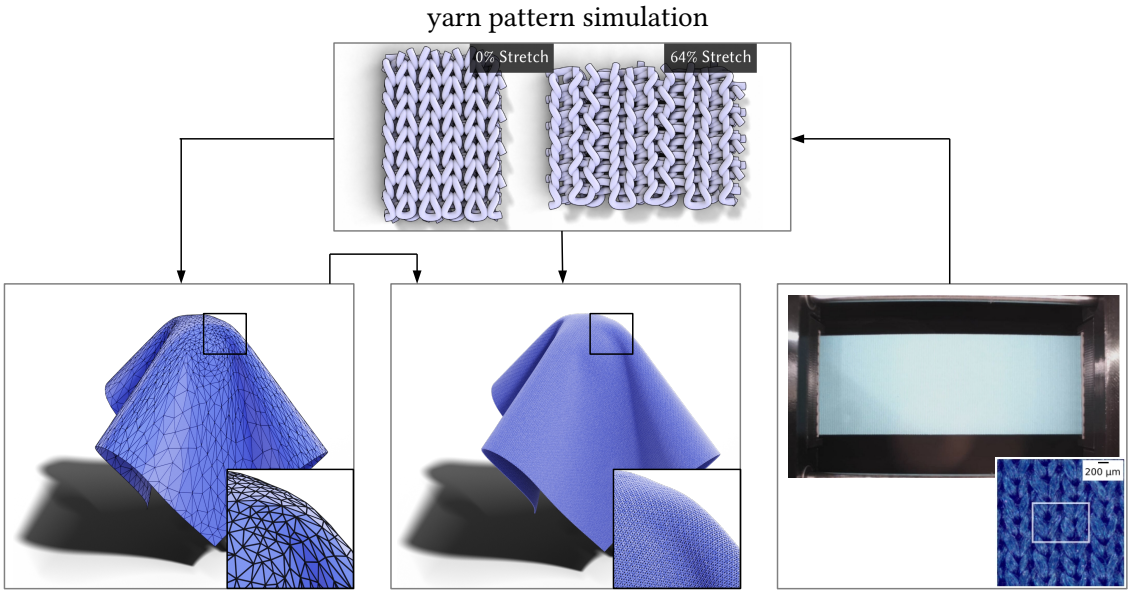


Figure 1.3: Overview of the methods included in this thesis. We use periodic yarn simulations (top) in three ways to analyze the behavior of yarn fabrics: First, we can compute an average large-scale elastic behavior by homogenizing a thin-shell model for cloth simulation (left). Second, we combine thin-shell simulation with data-driven yarn detail to animate millions of yarn loops in real-time (center). Third, we fit yarn parameters to real fabric measurements in a quantitative manner (right).

material parameters to match real physical measurements of fabric [Sperl et al. 2022].

Homogenized Yarn-Level Cloth We first address the topic of capturing the material properties of woven and knitted fabrics by fitting a custom continuum model to periodic yarn simulations. We introduce a procedure to homogenize the elastic energy of these yarn simulations under prescribed thin-shell strains using novel co-rotated periodic boundary conditions. Plugging the resulting continuum model into an off-the-shelf cloth solver, we manage to reproduce the complex macro-mechanical behavior of several fabrics, including effects such as anisotropic area-preservation, or even curling under tension. The code for yarn simulation is available at `https://git.ist.ac.at/gsperl/HYLC`, and for shell simulation at `https://git.ist.ac.at/gsperl/ARCSim-HYLC`.

Mechanics-Aware Deformation of Yarn Pattern Geometry While the previous method is able to capture the large-scale mechanics in terms of a thin-shell model, this abstraction loses the *visual* yarn-level detail. In the framework of homogenizing elastic properties of periodic yarn patterns, we also automatically capture the deformation-dependent displacements of the local yarn geometry. We therefore propose a method to interpolate yarn pattern deformations at run time to enrich a cloth simulation with detailed yarn geometry. The interpolated geometry rearranges in accordance with yarn-level mechanics and reproduces salient effects such as knit loops tightening under tension. It thus effectively approximates full yarn-level animation at negligible cost over an underlying mesh-based simulation. The code for this project is available at `https://git.ist.ac.at/gsperl/MADYPG`.

Estimation of Yarn-Level Simulation Models for Production Fabrics Finally, we propose a method to fit yarn material parameters such that yarn-level simulation matches physical measurements of real textiles used in production. Simulation of cloth at a yarn-level as discussed above has demonstrated its outstanding capabilities of matching the behavior of real fabrics *qualitatively*, producing animations of extreme detail and mechanical behavior exhibiting structural nonlinearity. However, real textiles often introduce additional mechanical complexity, e.g. by combining multiple types of yarns in one fabric. Previous work has not validated whether the yarn models used in computer graphics would be able to reproduce the behavior of real fabrics *quantitatively*. We compiled a library of production knit fabrics and developed a two-step fitting procedure, using an intermediate thin-shell model and homogenized yarn simulation, to also match the quantitative mechanical response. The data set is available at `https://mslab.es/projects/YarnLevelFabrics/`.

Modeling Assumptions Woven and knitted fabrics are complex materials with non-trivial elastic, plastic, hysteretic, and damping behaviors. As a first step toward data-driven yarn-level cloth simulation, the methods discussed in this thesis assume that these materials exhibit a purely hyperelastic response to deformation, and we do not explicitly model yarn friction. Although our current approaches are limited, we show in the following chapters that this hyperelastic assumption is sufficient to reproduce a number of qualitative and quantitative effects specific to yarn-level materials. We discuss future extensions in the directions of data-driven plasticity, hysteresis, and damping in the respective chapters and in Chapter 7.

The remainder of this thesis is organized as follows. In Chapter 2, we discuss related work. Chapter 3 introduces our approach for thin-shell homogenization and yarn simulation which subsequent chapters will use. In Chapter 4, we detail our method of fitting continuum shell models to periodic yarn simulations. Chapter 5 discusses how to use data-driven yarn detail to animate yarn-level geometry in real-time on top of a continuum cloth simulation. Chapter 6 describes our data set of production knit fabrics and our procedure to fit yarn-level parameters to fabric-level measurements. Chapter 7 concludes the thesis with a discussion on current limitations and future work.

CHAPTER 2

Related Work

In this chapter, we present a brief overview on related work. This thesis incorporates ideas form a variety of topics, ranging from yarn simulation and yarn-level cloth simulation, continuum cloth simulation, data-driven cloth, multiscale methods, as well as embedded and example-based deformation. We focus the discussion mostly on literature from computer graphics.

2.1 Yarn-Level Simulation

Yarn Simulation Simulators approximate the behavior of an individual strand of yarn or thread using the theory of elastic rods. Early work on simulation of elastic rods includes the work of Pai [2002], introducing Cosserat rods to the graphics community, as well as spring-based methods such as [Selle et al. 2008]. Bergou et al. [2008, 2010] present discrete elastic rods, a discrete model of Kirchhoff rods widely adopted in the graphics community [Kaldor et al. 2010; Fei et al. 2017; Pérez et al. 2015; Schumacher et al. 2018; Leaf et al. 2018]. Some works focused on interactive simulation of individual rods [Umetani et al. 2014; Kugelstadt and Schömer 2016; Angles et al. 2019; Soler et al. 2018] build on the frameworks of position-based dynamics [Müller et al. 2007; Macklin et al. 2016] or projective dynamics [Bouaziz et al. 2014]. Martin et al. [2010] propose a unified model for rods, shells and volumes.

Yarn-Level Cloth Simulation of fabric at the yarn level was pioneered in computer graphics by Remion et al. [1999] who simulated knitted fabrics as deformable rods in contact. Later, Kaldor et al. [2008] demonstrated the ability to simulate full garments at the yarn level, with subsequent work on improving the treatment of collision handling [Kaldor et al. 2010]. They showed that such models could reproduce qualitative macroscopic behavior of real-world knits. To accelerate computation, Cirio et al. [2014] introduced persistent sliding contacts to simulate woven fabric [Cirio et al. 2014], which was later extended to knitted fabrics [Cirio et al. 2015, 2016], an improved bending model [Pizana et al. 2020], and robust contact handling between multiple layers of cloth [Sánchez-Banderas et al. 2020]. Recently, Cirio and Rodríguez [2022] combined yarn-level garment simulation with mass-spring systems to accurately model realistic seams between fabric pieces. Jiang et al. [2017] modeled yarn-level simulations within the Material Point Method (MPM). In this thesis, we rely on the methods of Kaldor et al. [2008] and Bergou et al. [2010] for the simulation of our periodic yarn patterns. In Chapter 6 we extend these models to biphasic stretching and collision to fit yarn simulations to real fabric measurements with difficult nonlinearities.

As discussed in the introduction, an important gap in yarn-level modeling of fabrics is their connection to real-world materials. Leaf et al. [2018] propose a method for the interactive authoring and editing of small periodic yarn patches on GPUs. They showed that varying the stiffnesses and rest-shape parameters of yarns could lead to the design of yarn patterns that matched the geometry of complex real-world knits. However, they did not validate the mechanical behavior of the resulting fabric. In our works, we similarly examine periodic yarn patches but use them to judge both the qualitative and quantitative mechanical behavior. In textile engineering, recent works have looked at initializing the yarn geometry for volumetric finite-element simulation [Wadekar et al. 2020]. In our experience, this leaves many open unknowns, such as the rest-shape geometry of the yarns and the contact model.

Yarn Friction While we omit treatment of friction in our yarn simulations, recent works have investigated simulation of rods with friction. Daviet [2020] propose a unified approach for Coulomb friction of hair, cloth, and elastic bodies. Similarly, Li et al. [2021] extend Incremental Potential Contact (IPC) [Li et al. 2020] to codimensional objects for interpenetration-free and stable contact with friction. Other works propose an implicit penalty-based frictional contact for yarns with penetration [Choi et al. 2021; Tong et al. 2022] or differentiable persistent-contact models [Gong et al. 2022].

2.2 Continuum-Level Cloth

Researchers in computer graphics often simulate cloth based on continuum mechanics, i.e. treating it as an elastic solid with a potential energy that increases as it deforms from its rest state. Typical methods for discretizing such an elastic solid are mass-spring networks [Provot 1995; Choi and Ko 2005], discrete thin shells [Grinspun et al. 2003; Bridson et al. 2003; Wardetzky et al. 2007], finite differences [Terzopoulos et al. 1987], finite elements [Baraff and Witkin 1998; Kim 2020; Thomaszewski et al. 2007; Narain et al. 2012], and the material point method [Guo et al. 2018]. Chen et al. [2018b] develop a discrete Koiter model for thin-shell simulation. Further work focused on contact [Weidner et al. 2018; Buffet et al. 2019; Li et al. 2021] and on efficient simulation using multi-grid solvers [Wang et al. 2018; Xian et al. 2019], local-global schemes [Bouaziz et al. 2014; Overby et al. 2017], and GPU-based implementations [Schmitt et al. 2013; Wu et al. 2022]. Most recently, Lan et al. [2022] proposed a combined approach for penetration-free projective dynamics on the GPU. In Chapter 4, we use a finite-element thin-shell solver with adaptive remeshing to simulate our macroscale cloth (ArcSim [Narain et al. 2012, 2013]).

2.3 Estimating of Mechanical Cloth Parameters

Many of the methods above use analytically derived material models based on a somewhat straightforward relationship between deformation and potential energy. However, real materials can be exhibit hard-to-model nonlinearities, anisotropies and more. As a result, there is a long line of research trying to estimate thin-shell models for fabric in a data-driven way. In the textile engineering field early approaches designed mechanical tests that could elicit mechanical properties in a separable way [Peirce 1930; Kawabata 1980].

In computer graphics, we can distinguish two main directions in the estimation of cloth simulation models. One direction focuses on the accuracy of the estimation, in particular trying to address the nonlinear and anisotropic behavior of cloth. Works in this direction entail

the design of physical tests to produce force-deformation examples, parameterization of the cloth model, and estimation algorithms [Wang et al. 2011; Miguel et al. 2012]. Some works have considered the hysteresis behavior in cloth deformations, by estimating models of internal friction [Miguel et al. 2013]. Further research discusses incremental fitting of separable models for convex hyperelastic materials [Miguel et al. 2016] and an orthotropic model for woven fabric based on commercially available tests [Clyde et al. 2017]. When fabric is deformed, it is challenging to apply uniform stress; therefore, multiple mechanical properties lead to a complex interplay. We face a similar challenge when trying to circumvent the computational cost of yarn-level simulations in Chapter 6 by leveraging periodic deformations. For each update to model parameters, the above methods typically need to recompute quasistatic cloth equilibria to compare to real-world measurements. They also mention difficulty in accurately capturing bending. For our homogenized materials (Chapter 4), data-gathering and fitting are decoupled. We precompute deformation responses once as an inexpensive preprocessing step and thus do not require simulations during fitting. We also do not require any real-world measurement setup. Additionally, our method can directly compute the bending resistance for applied curvatures, allowing for more controlled measurements.

Another direction focuses on estimating simulation models from more casual data, such as video. The pioneering work of Bhat et al. [2003] used optimization methods to estimate mass, elasticity and damping parameters from videos of cloth motion. Bouman et al. [2013] used a machine-learning technique, where they learned a mapping from cloth model parameters to video features, and then inverted this mapping to fit parameters to new video footage. This approach has received major thrust with the explosion of deep learning methods [Yang et al. 2017; Runia et al. 2020]. Recent works look at the design of semi-controlled setups where particular mechanical properties are exposed, e.g., contact friction [Rasheed et al. 2020].

Modern machine learning methods have also posed the problem of efficient simulation-in-the-loop optimization. To this end, research on differentiable simulation answers how to compute gradients of (dynamic) equilibrium constraints for cloth simulation [Liang et al. 2019; Li et al. 2022a].

2.4 Multiscale Modeling and Homogenization

In computer graphics, the concept of multiscale modeling covers a wide area of research such as analytic multiscale models [Fei et al. 2017, 2018, 2019], numerical coarsening [Kharevych et al. 2009; Chen et al. 2017, 2018a], metamaterials and digital fabrication [Bickel et al. 2010; Schumacher et al. 2015; Chen et al. 2015; Pérez et al. 2015; Rodriguez et al. 2022; Li et al. 2022b], sound simulation [Cirio et al. 2018], and rendering [Guarnera et al. 2016; Zhao et al. 2016].

Our work focuses on homogenization of periodic yarn patterns and is thus closely related to the work of Schumacher et al. [2018]. They investigate the elastic properties of isohedral tilings represented as planar rod patterns through numerical homogenization. They also provide a tool for exploring the various families of tilings and discuss emergent properties such as material symmetries in detail. Their tool examines material nonlinearities by fitting linear models at multiple magnitudes of deformation. Our work can be seen as an extension to fully nonlinear models for non-planar woven and knitted yarn patterns. In addition, our novel boundary conditions let us homogenize interaction between multiple modes of deformation, such as simultaneous stretching and bending.

Recently, Rodriguez et al. [2022] built on our work and developed an algorithm for inverse

design of bending-active structures. Chapter 6 similarly discusses inverse design using homogenization. It also relates to the inverse design of simulated fibers [Hadap 2006; Derouet-Jourdan et al. 2010] and hair [Derouet-Jourdan et al. 2013].

Another work has addressed the computational cost of full yarn-level simulations by enabling hybrid simulations [Casafranca et al. 2020], combining thin-shell and yarn-level simulation while focusing yarn-level computational effort only where needed.

Computational Homogenization Multiscale modeling has received a lot of attention also outside of computer graphics; this includes the technique of computational homogenization, where macroscopic material responses are computed based on representative microscale simulations [Renard and Marmonier 1987; Guedes and Kikuchi 1990]. Macroscopic strains are imposed on the representative microscale material sample through boundary conditions, and stresses can be computed through averaging. To this end, De Souza Neto et al. [2015] and Blanco et al. [2016] propose a generalized framework to derive microscale boundary conditions and averaging relations for homogenization in general. For more details, we refer to the reviews of Geers et al. [2010] and Matouš et al. [2017].

This method has been applied to the homogenization of thin shells [Geers et al. 2007] as well as textiles and fabrics [Mehnert et al. 2015; Fillep et al. 2017; Dinh et al. 2018; Liu et al. 2019]. However, these works discretize yarns with expensive volumetric finite-elements and limit their analysis to small stitches and deformations. They use a small-curvature assumption which is inadequate for large bending, as we will discuss in Section 3.2.2.

The nature of representative microscale computations in computational homogenization lends itself to data-driven approaches. Various approaches fit constitutive models from precomputed stress and energy data [Bessa et al. 2017; Le et al. 2015; Yvonnet et al. 2013]. However, the basic constitutive models used are either not descriptive enough for our data or do not provide any guarantees to ensure smooth animation.

Other Continuum Models for Fabric The physics and engineering communities have also developed continuum-level models for approximating the behavior of fabrics. Choi and Lo [2003] and Poincloux et al. [2018] propose mathematical models describing the rich material response of a stockinette pattern based on inextensible and incompressible yarns. However, their investigations are limited to a small set of extension tests. Researchers have also developed mesoscopic models of woven fabric using spring-based finite elements [King et al. 2005; Parsons et al. 2010, 2013].

2.5 Embedded Detail

A simple and effective way to give the illusion of detailed physics is to embed fine geometric detail into a coarser control mesh. After the idea of dynamic free-form deformations were introduced by Faloutsos et al. [1997], researchers embedded geometric detail into animations of elasticity [Sifakis et al. 2007], fracture [Muller et al. 2004], viscoplasticity [Wojtan and Turk 2008], fluids [Wojtan et al. 2009], and articulated characters [Rumman and Fratarcangeli 2016]. Skinning or cage-based approaches deform detailed geometry to follow skeletal poses or a surrounding cage respectively. For more detail, we refer to surveys of skinning techniques [Rumman and Fratarcangeli 2016] as well as the overview in Corda et al. [2020]. James [2020] introduced Phong Deformation to reduce artifacts caused by the resolution of a coarse embedding mesh.

Porumbescu et al. [2005] proposed shell maps to embed geometry within thin shells. Researchers also use embedding to deform woven yarn pattern geometry [Zhao et al. 2016; Montazeri et al. 2020]. Wu and Yuksel [2017] instantiate fiber-level detail using yarn curves. Hoffman et al. [2020] map detailed knit, crochet, and sequin geometry onto cloth meshes with support for fly-away fibers. Because these techniques apply coarse deformations to fine-scale geometry, they cannot reproduce small-scale physical effects. Our method for mechanics-aware yarn detail (Chapter 5) addresses this in the context of yarn-level cloth, by displacing the yarn geometry before mapping it onto the deformed mesh such that the combined deformation actually captures the yarn-level physical effects.

2.6 Example-Based Deformation

Our work in Chapter 5 animates yarn physics in real-time by interpolating from precomputed examples. Researchers also use example geometry to bias physics simulators toward preferred results [Martin et al. 2011; Schumacher et al. 2012; Koyama et al. 2012; Wampler 2016; Gao et al. 2019]. Similar to Ma et al. [2008]'s interpolation of displacement textures based on the deformation of a face mesh, we interpolate yarn geometry based on the deformation of a cloth mesh. Montazeri et al. [2019] deform yarn cross-sections by learning a model from precomputed simulations; our approach operates at one scale higher, by animating the reconfiguration of yarn *patterns* based on a database of precomputed examples.

2.7 Mechanics-Aware Detail & Wrinkling

Researchers use similar ideas to add fine-scale wrinkle details to a coarse cloth or flesh simulation. Procedural wrinkle methods rely on an underlying strain field [Hadap et al. 1999; Rohmer et al. 2010; Zuenko and Harders 2019; Chen et al. 2021] explicit simulation of detail [Müller and Chentanez 2010], while data-driven wrinkle methods instead train local operators or pose-dependent systems [Wang et al. 2010; Kavan et al. 2011; Zurdo et al. 2012], with recent works building on recurrent or convolutional neural networks [Santesteban et al. 2019; Chentanez et al. 2020; Jin et al. 2020; Vidaurre et al. 2020]. Our technique (Chapter 5) assumes that the input cloth mesh animation already contains all cloth-scale features including wrinkles, and then adds detail purely on a yarn scale.

2.8 Machine Knitting & Garment Authoring

The fact that knitted and woven fabric can be produced from simple yarns while having both great visual and mechanical appeal has spurred the development of authoring tools for these types of fabric. These tools have also been one of the main targets of yarn-level models in computer graphics.

Yuksel et al. [2012] introduce the stitch mesh representation for authoring yarn-level models of knitted garments. This work has later been extended to guarantee knittability [Wu et al. 2019; Narayanan et al. 2019] and for crocheting [Guo et al. 2020]. Several other representations for machine-knittable fabric have been developed for sketch-based authoring [Kaspar et al. 2021], focusing more on high-level design aspects [Nader et al. 2021; Jones et al. 2021], or more on low-level topology [Kapllani et al. 2021]. Similarly, design tools have also been developed for 3D woven fabrics [Wu et al. 2020a,b].

CHAPTER 3

Preliminaries

In this chapter, we will introduce our method for the homogenization of periodic yarn patterns to thin-shell cloth. We will first briefly discuss notation and terms, then review homogenization of elastic solids, before detailing thin-shell homogenization and yarn pattern simulation that subsequent chapters rely on.

3.1 Notation and Terms

In the following and throughout this thesis, we use the terms *microscopic* and *microscale* when referring to small local (yarn-level) effects, and we use the terms *macroscopic* and *macroscale* when referring to average (continuum-level) behaviors of the bulk material.[1]

We will refer to the dominant directions of both knitted and woven fabric as *weft*, for horizontal rows of knit loops or woven threads, and *warp* for vertical columns, see also Figure 3.1.

The notation we use during the exposition of our homogenization approach is as follows. We write matrix-valued quantities $\boldsymbol{A}$ and vector-valued quantities $\boldsymbol{x}$ in bold, as compared to scalar-valued quantities Ψ or k. We write macroscopic quantities $\overline{\boldsymbol{x}}$ with a bar and microscopic quantities $\boldsymbol{x}$ without. We use Latin indices i, j to iterate dimensions $1, 2, 3$, and Greek indices α, β to iterate only the first two dimensions $1, 2$. We use indices preceded by a comma as shorthand for derivatives, e.g. $x_{i,j}$ is the derivative of element x_i with respect to parameter j.

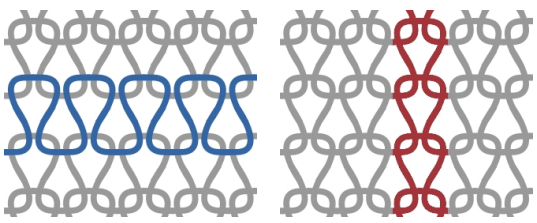


Figure 3.1: The main directions in a knit: weft (rows of knit loops, left) and warp (columns of knit loops, right).

[1] As an aside, for several of our examples the term *mesoscale* (in between micro and macro) might be more appropriate, given that the deformation is on a somewhat similar scale to the yarn geometry. For simplicity, we will continue referring to the yarn scale as microscopic.

3.2 Homogenization

We begin by summarizing the "kinematic averaging" theory of computational homogenization for volumetric solids, and we extend these concepts to the homogenization of thin shells in the second part of this section. For further details, we recommend the following reviews on computational homogenization and multiscale modeling [Geers et al. 2010; Matouš et al. 2017].

3.2.1 Computational Homogenization of Volumetric Solids

We describe the macroscale deformation of an elastic solid with reference coordinates $\overline{\boldsymbol{X}}$, deformed coordinates $\overline{\boldsymbol{x}}$, and deformation gradient $\overline{\boldsymbol{F}} = \frac{\partial \overline{\boldsymbol{x}}}{\partial \overline{\boldsymbol{X}}}$. Similarly, we have microscale quantities $\boldsymbol{X}$, $\boldsymbol{x}$, and $\boldsymbol{F} = \nabla \boldsymbol{x} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$. Homogenization theory assumes that the bulk material exhibits microscale variations, and thus we can zoom in at any macroscale point $\overline{\boldsymbol{x}}$ to find a volume of microscale material, called the *representative volume element* (RVE) [Hill 1963; Geers et al. 2010]. Mathematically, we can describe the RVE with a first-order expansion about a point $\overline{\boldsymbol{x}}$ [De Souza Neto et al. 2015]:

$$\boldsymbol{x}(\boldsymbol{X}) = \overline{\boldsymbol{x}} + \overline{\boldsymbol{F}}\boldsymbol{X} + \tilde{\boldsymbol{u}}(\boldsymbol{X}), \tag{3.1}$$

where $\tilde{\boldsymbol{u}}$ is a *microscale displacement fluctuation field* which encodes all of the non-affine local deformations around $\overline{\boldsymbol{x}}$. In other words, $\tilde{\boldsymbol{u}}$ encodes all of the detailed, high-frequency deformations of the microstructure geometry that are not accounted for by the large-scale deformation $\overline{\boldsymbol{F}}$. The holes in a spongy material, for example, may deform more than the stiffer elastic parts; $\tilde{\boldsymbol{u}}$ would define this difference in microscale deformation. See Figure 3.2 for an illustration.

The theory assumes that the macroscale quantities vary so slowly over the RVE that they are essentially constant at the microscale [De Souza Neto et al. 2015], i.e. $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{F}}$ do not depend on $\boldsymbol{X}$. We also assume without loss of generality that $\frac{1}{|\Omega|}\int_\Omega \boldsymbol{X}\, d\Omega = \boldsymbol{0}$, i.e. the centroid of the microscale reference domain Ω with volume $|\Omega|$ is located at the origin.[2] Here, $d\Omega$ denotes integration over Ω.

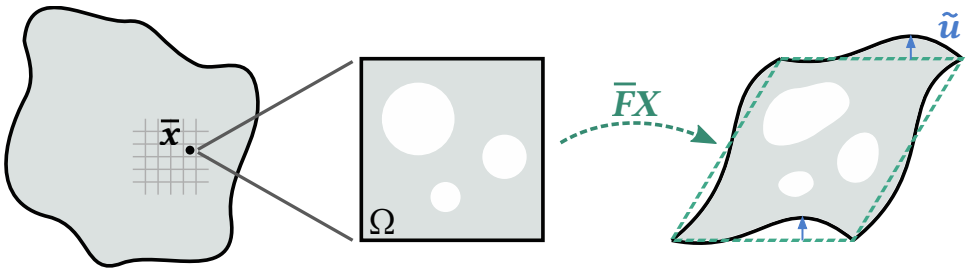


Figure 3.2: RVE for an elastic solid with microscopic holes. At any macroscale point $\overline{\boldsymbol{x}}$, we can observe a microscale RVE with reference domain Ω. The RVE is deformed through an affine transformation given by $\overline{\boldsymbol{F}}$ (dashed lines) and additional periodic fluctuations $\tilde{\boldsymbol{u}}$ (blue). Note that the deformation of the holes is described by a combination of $\overline{\boldsymbol{F}}$ and $\tilde{\boldsymbol{u}}$.

[2] Otherwise, the term $\overline{\boldsymbol{F}}\boldsymbol{X}$ in (3.1) would instead become $\overline{\boldsymbol{F}}(\boldsymbol{X} - \boldsymbol{C})$ for the centroid $\boldsymbol{C} = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{X}\, d\Omega$.

Next, macroscale quantities are defined to be averages over their microscale counterparts [Hill 1963; De Souza Neto et al. 2015]:

$$\overline{\boldsymbol{x}} = \frac{1}{|\Omega|} \int_\Omega \boldsymbol{x}(\boldsymbol{X})\, d\Omega, \tag{3.2}$$

$$\overline{\boldsymbol{F}} = \frac{1}{|\Omega|} \int_\Omega \boldsymbol{F}(\boldsymbol{X})\, d\Omega. \tag{3.3}$$

Plugging (3.1) into (3.2) and (3.3) and applying these assumptions gives us

$$\int_\Omega \tilde{\boldsymbol{u}}(\boldsymbol{X})\, d\Omega = \boldsymbol{0}, \tag{3.4}$$

$$\int_\Omega \nabla\tilde{\boldsymbol{u}}(\boldsymbol{X})\, d\Omega = \boldsymbol{0}. \tag{3.5}$$

In other words, the small-scale fluctuations in translation $\tilde{\boldsymbol{u}}$ and deformation $\nabla\tilde{\boldsymbol{u}}$ must average out over the RVE. In computer simulations with periodic micro-structures, (3.4) is satisfied by fixing the barycenter of $\tilde{\boldsymbol{u}}$, and (3.5) is commonly satisfied by requiring $\tilde{\boldsymbol{u}}$ to be periodic on the boundaries [Van der Sluis et al. 2000; De Souza Neto et al. 2015]:

$$\tilde{\boldsymbol{u}}^+ = \tilde{\boldsymbol{u}}^-, \tag{3.6}$$

where $\tilde{\boldsymbol{u}}^+$ is the value of the fluctuation field on one side of the domain, and $\tilde{\boldsymbol{u}}^-$ is its value on the corresponding opposite side. Finally, we compute the homogenized energy density as the averaged total energy in the RVE

$$\overline{\Psi} = \frac{1}{|\Omega|} \int_\Omega \Psi(\boldsymbol{X})\, d\Omega, \tag{3.7}$$

where Ψ and $\overline{\Psi}$ are the microscale and macroscale energy densities respectively.

We can also derive an equation for homogenized stresses $\overline{\boldsymbol{P}}$, either variationally or by taking the derivative of $\overline{\Psi}$ with respect to the deformation $\overline{\boldsymbol{F}}$:

$$\overline{\boldsymbol{P}} = \frac{1}{|\Omega|} \int_\Omega (\boldsymbol{P} - \boldsymbol{f} \otimes \boldsymbol{X})\, d\Omega, \tag{3.8}$$

where $\boldsymbol{P}$ are microscale first Piola-Kirchhoff stresses, and $\boldsymbol{f}$ are microscale inertial and body forces. We refer to the work of De Souza Neto et al. [2015] for more details.

To restate briefly, we expand a microscale RVE from a macroscopic deformation $\overline{\boldsymbol{F}}$ and with fluctuations $\tilde{\boldsymbol{u}}$ that describe local deformation. We then require that the microscale deformation *on average* equals $\overline{\boldsymbol{F}}$. This imposes the constraint that $\tilde{\boldsymbol{u}}$ should on average not induce any additional deformation, which can be enforced through periodicity. We solve for $\tilde{\boldsymbol{u}}$ at equilibrium subject to these constraints. For the purposes of macroscale simulation, we can then compute forces by taking the negative gradient of the homogenized potential energy or from the homogenized stresses.

This pipeline can be used to solve coarse problems *without an explicit coarse constitutive model* [Geers et al. 2010]. We can evolve macroscale and microscale at the same time, for example enriching a macroscale mesh with nested solves of RVEs, mapping local deformation to local elastic stresses. The downside is that we will have to solve for microscale equilibria for every new deformation, which negatively impacts performance. To address this, data-driven approaches have been developed [Bessa et al. 2017; Le et al. 2015; Yvonnet et al. 2013], and we will similarly instead precompute a constitutive model in Chapter 4.

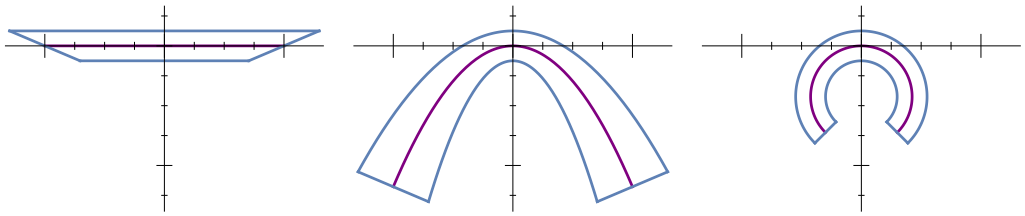


Figure 3.3: Comparison of a first order (left), second order (middle) and our nonlinear expansion (right) of thin-shell RVEs in a curved configuration. The lower order expansions show strong artifacts as bending modes are approximated through shearing (left) or stretching (middle).

3.2.2 Nonlinear Homogenization of Thin Shells

Next, we apply this rationale to the problem of homogenizing a yarn-level microscale to a thin-shell macroscale. The main challenge here is to find a suitable analogy to Equation (3.1) that works for thin shells instead of volumes. Previous work on thin-shell homogenization relies on a small curvature assumption and uses first or second order expansions for the RVE (e.g. [Geers et al. 2007]). This effectively replaces bending modes with shearing or stretching of the material, as illustrated in Figure 3.3. For microscale materials that resist stretching far more than bending, the erroneous stretching can introduce artificial stiffness in the homogenized response for macroscale bending. To support our goal of homogenizing highly flexible materials, this section proposes a novel nonlinear thin-shell expansion based on metrics from differential geometry.

On the macroscale, we have a thin shell $\overline{\boldsymbol{x}}$ that is defined through its midsurface $\overline{\boldsymbol{\varphi}}$, which is extruded along the normal $\overline{\boldsymbol{n}}$:

$$\overline{\boldsymbol{x}}(\overline{X}_1, \overline{X}_2, h) = \overline{\boldsymbol{\varphi}}(\overline{X}_1, \overline{X}_2) + h\, \overline{\boldsymbol{n}}(\overline{X}_1, \overline{X}_2), \tag{3.9}$$

where $\overline{X}_\alpha$ are the flat reference coordinates of the midsurface, and h is the thickness coordinate. The left side of Figure 3.4 illustrates this parametrization.

We locally define deformations with the first fundamental form $\overline{\mathbf{I}}$ [Do Carmo 2016] for in-plane deformation and the second fundamental form $\overline{\mathbf{II}}$ for bending modes. With surface tangents $\overline{\boldsymbol{a}}_\alpha = \overline{\boldsymbol{\varphi}}_{,\alpha}$ we have

$$\overline{\boldsymbol{n}} = \frac{\overline{\boldsymbol{a}}_1 \times \overline{\boldsymbol{a}}_2}{|\overline{\boldsymbol{a}}_1 \times \overline{\boldsymbol{a}}_2|}, \tag{3.10}$$

and we compute the components of the fundamental forms as

$$\overline{\mathrm{I}}_{\alpha\beta} = \quad \overline{\boldsymbol{a}}_\alpha \cdot \overline{\boldsymbol{a}}_\beta, \tag{3.11}$$

$$\overline{\mathrm{II}}_{\alpha\beta} = -\overline{\boldsymbol{n}}_{,\alpha} \cdot \overline{\boldsymbol{a}}_\beta. \tag{3.12}$$

We construct the RVE expansion similar to (3.1):

$$\boldsymbol{x}(X_1, X_2, h) = \boldsymbol{\varphi}(X_1, X_2) + h\, \boldsymbol{n}(X_1, X_2) + \tilde{\boldsymbol{u}}(X_1, X_2, h) \tag{3.13}$$

with microscale midsurface $\boldsymbol{\varphi}$, its normal $\boldsymbol{n}$, and fluctuation field $\tilde{\boldsymbol{u}}$. In an analogy to (3.1), which deforms the volumetric RVE based on the macroscale quantity $\overline{\boldsymbol{F}}$, we deform the thin-shell RVE with a midsurface $\boldsymbol{\varphi}$ *derived from the macroscale fundamental forms* $\overline{\mathbf{I}}$ and

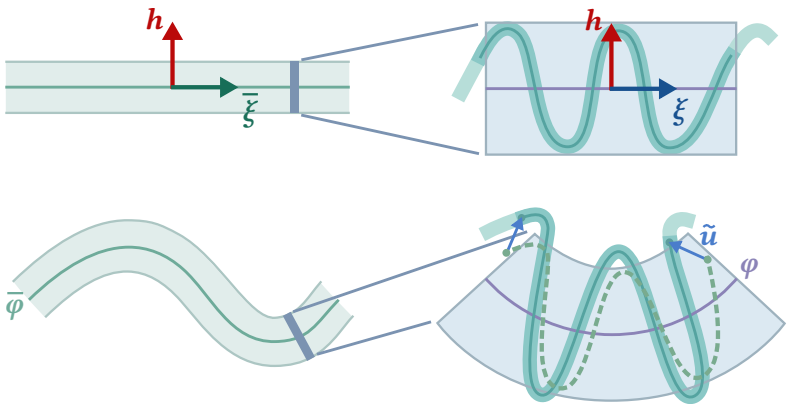


Figure 3.4: A macroscopic line-segment (left) is expanded in-plane into a curved microscale volume (right). Top and bottom show reference and deformed configurations respectively. This example uses a squiggly yarn for the microstructure, and we indicate the fluctuations $\tilde{\boldsymbol{u}}$ as offsets to the yarn deformed purely from its embedding (dashed).

$\overline{\mathbf{II}}$. The function $\varphi(X_1, X_2) + h\,\boldsymbol{n}(X_1, X_2)$ applies a low-resolution spatial deformation across the entire microstructure (illustrated by the dashed line in Figure 3.4), while $\tilde{\boldsymbol{u}}$ encodes the remaining high-frequency details of the thin-shell microgeometry. In a knitted microstructure, for example, $\tilde{\boldsymbol{u}}$ prescribes how the individual threads stretch, slide, twist, and bend relative to each other. Figure 3.4 illustrates a 2D schematic, and Figure 3.5 shows a 3D rendering of this expansion. Notice that the thickness coordinate h is shared between both micro- and macroscale since our thin-shell homogenization averages only the in-plane coordinate. Thus, we sometimes interchangeably write $h = X_3 = \overline{X}_3$.

Defining the Midsurface Our goal here is to create a midsurface φ in (3.13) with constant fundamental forms $\mathbf{I}, \mathbf{II}$ matching those of the macroscale. Although it is possible to derive such constant-fundamental-form surfaces analytically, the exact solutions are only compatible with a limited set of boundary conditions. Here, we present a more general least-squares solution to this surface-reconstruction problem.

Inspired by the rotation-strain decomposition for deformation extrapolation [Huang et al. 2011], we begin with the polar decomposition of the midsurface gradient

$$\nabla\varphi = \begin{pmatrix}\boldsymbol{a}_1 & \boldsymbol{a}_2\end{pmatrix} = \boldsymbol{R}\overline{\boldsymbol{S}}. \tag{3.14}$$

Here, the 3×2 matrix $\overline{\boldsymbol{S}}$ represents the constant in-plane deformation and $\boldsymbol{R}$ is a 3×3 rotation matrix that aligns $\overline{\boldsymbol{S}}$ with the tangent plane of the curved surface. Without loss of generality, we choose the macroscale frame of reference such that $\begin{pmatrix}\overline{\boldsymbol{a}}_1 & \overline{\boldsymbol{a}}_2\end{pmatrix} = \overline{\boldsymbol{S}}$ and $\overline{\boldsymbol{n}} = \begin{pmatrix}0 & 0 & 1\end{pmatrix}^\top$. Note that $\nabla\varphi$, $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, and $\boldsymbol{R}$ vary along the midsurface; we omit the (X_1, X_2) function notation when convenient for readability.

We want to match $\mathbf{I} = \bar{\mathbf{I}}$. With $\mathbf{I} = \nabla\varphi^\top\nabla\varphi$ and (3.14) we get

$$\overline{\boldsymbol{S}}^\top\overline{\boldsymbol{S}} = \bar{\mathbf{I}}, \tag{3.15}$$

allowing us to compute $\overline{\boldsymbol{S}}$ in (3.14) from the principal square root of the first fundamental form $\bar{\mathbf{I}}$:

$$\overline{\boldsymbol{S}} = \begin{pmatrix}\sqrt{\bar{\mathbf{I}}} \\ 0 \quad 0\end{pmatrix}. \tag{3.16}$$

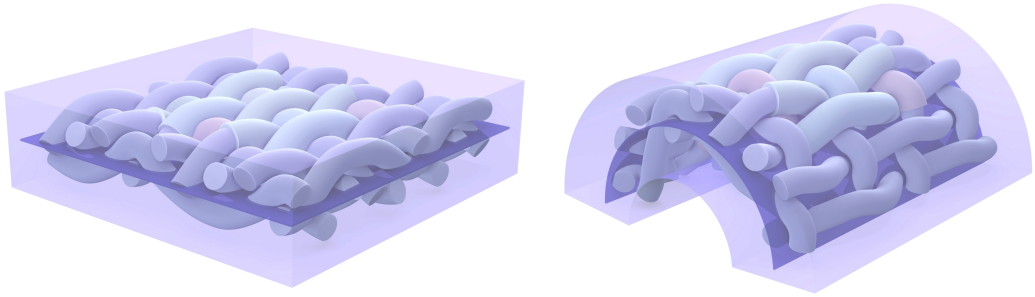
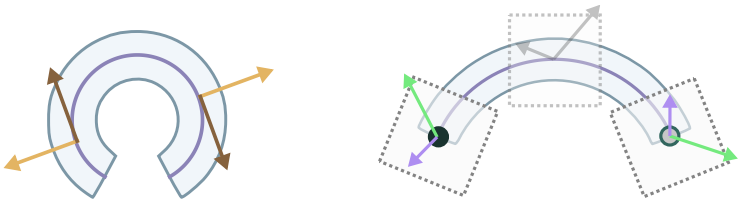


Figure 3.5: A periodic yarn pattern microstructure is shown with its associated midsurface in an undeformed (left) and deformed state (right).



(a) (b)

Figure 3.6: (a) Naive averaging creates a null-space of growing/shrinking cylinder radii as well as sliding along the surface. As an example, the displacements indicated as pairs of orange or brown arrows would cancel each other respectively, whereas with our co-rotated averaging they are treated as the same *rotated* displacement. (b) Our co-rotated periodicity compares fluctuations (arrows) by rotating them into a common frame (gray).

To match $\mathbf{II} = \overline{\mathbf{II}}$, we compute $\boldsymbol{R}(X_1, X_2)$ in (3.14) by integrating the normal curvatures $\overline{\boldsymbol{n}}_{,\alpha}$ outward from the RVE center $X_1 = X_2 = 0$. We perform this integration with an analytic expression for the exponential map, which we explain in detail in Appendix A.1.1.

Now that we know $\boldsymbol{R}$ and $\overline{\boldsymbol{S}}$, we solve (3.14) for $\boldsymbol{\varphi}$ in the least squares sense, giving us a vector Poisson equation with natural boundary conditions:

$$\nabla^2 \boldsymbol{\varphi} = \nabla \cdot \boldsymbol{R}\overline{\boldsymbol{S}} \text{ inside the domain,} \tag{3.17}$$

$$\boldsymbol{N} \cdot \nabla \boldsymbol{\varphi} = \boldsymbol{N} \cdot \boldsymbol{R}\overline{\boldsymbol{S}} \text{ on the boundary.} \tag{3.18}$$

This equation gives the exact solution for singly-curved surfaces and can generalize to solutions for non-constant $\mathbf{I}$ and $\mathbf{II}$. We solve the system numerically by discretizing the surface as a regular grid and using standard finite differencing. This midsurface can now be used in (3.13) to completely describe a highly deformed thin-shell microstructure, as illustrated in Figure 3.5.

Co-Rotated Boundary Conditions To complete our analogy with the homogenization strategy in Section 3.2.1, we must derive constraints on the fluctuation field $\tilde{\boldsymbol{u}}$ which make sense for thin shells. Unfortunately, as illustrated in Figure 3.6a, the simple averages proposed in (3.4) and (3.5) can lead to erroneous cancellation of fluctuations when applied to a highly deformed domain, leading to undesired nullspaces in the RVE.

To address this problem, we propose to average quantities by parallel transporting them to a common frame. The rotation $\boldsymbol{R}$ from earlier rotates $\overline{\boldsymbol{n}} = \boldsymbol{n}(0,0)$ to $\boldsymbol{n}(X_1, X_2)$ and thus describes orthogonal frames oriented along the midsurface normal. Therefore, we can use its

transpose to align local frames for fluctuations, resulting in the modified constraint

$$\int_\Omega \boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega = \boldsymbol{0}. \tag{3.19}$$

With a bit more work (explained in Appendix A.1.2), we can also derive a co-rotated constraint on the derivative of $\tilde{\boldsymbol{u}}$:

$$\int_\Omega \boldsymbol{R}^\top \tilde{\boldsymbol{u}}_{,\alpha}\, d\Omega = \boldsymbol{0}, \tag{3.20}$$

implying an analogous co-rotated version of (3.6):

$$(\boldsymbol{R}^\top \tilde{\boldsymbol{u}})^+ = (\boldsymbol{R}^\top \tilde{\boldsymbol{u}})^-, \tag{3.21}$$

which is satisfied by splitting the boundary of the midsurface domain Γ into opposing parts $\partial\Gamma^+$ and $\partial\Gamma^-$, and using this constraint as periodic boundary conditions. Figure 3.6b illustrates how our co-rotated periodicity aligns displacements.

Homogenized Energies and Stresses Finally, for macroscale thin-shell simulations, we are interested in homogenizing an elastic energy *area* density. Instead of dividing the total energy by the volume as in (3.7), we divide by the area of the RVE midsurface to get

$$\overline{\Psi} = \frac{1}{|\Gamma|}\int_\Omega \Psi(X_1, X_2, h)\, d\Omega, \tag{3.22}$$

with $|\Gamma|$ being the area of the midsurface domain.

We can also compute homogenized stresses by computing the derivative of (3.22) with respect to the deformation. Writing $\overline{\Psi} = \frac{1}{|\Gamma|}\int_\Omega \Psi(\boldsymbol{x}(\overline{\boldsymbol{x}}, \tilde{\boldsymbol{u}}))\, d\Omega$ with $\boldsymbol{x} = \overline{\boldsymbol{x}} + \tilde{\boldsymbol{u}}$, we compute the derivative with respect to some strain $\overline{\boldsymbol{s}}$ as:

$$\frac{\partial\overline{\Psi}}{\partial\overline{\boldsymbol{s}}} = \frac{1}{|\Gamma|}\int_\Omega \frac{\partial\overline{\Psi}}{\partial\boldsymbol{x}}\left(\frac{\partial\boldsymbol{x}}{\partial\overline{\boldsymbol{x}}}\frac{\partial\overline{\boldsymbol{x}}}{\partial\overline{\boldsymbol{s}}} + \frac{\partial\boldsymbol{x}}{\partial\tilde{\boldsymbol{u}}}\frac{\partial\tilde{\boldsymbol{u}}}{\partial\overline{\boldsymbol{s}}}\right) d\Omega \tag{3.23a}$$

$$= \frac{1}{|\Gamma|}\int_\Omega \left(\frac{\partial\overline{\Psi}}{\partial\boldsymbol{x}}\frac{\partial\overline{\boldsymbol{x}}}{\partial\overline{\boldsymbol{s}}} + \frac{\partial\overline{\Psi}}{\partial\tilde{\boldsymbol{u}}}\frac{\partial\tilde{\boldsymbol{u}}}{\partial\overline{\boldsymbol{s}}}\right) d\Omega \tag{3.23b}$$

$$= \frac{1}{|\Gamma|}\int_\Omega \frac{\partial\overline{\Psi}}{\partial\boldsymbol{x}}\frac{\partial\overline{\boldsymbol{x}}}{\partial\overline{\boldsymbol{s}}}\, d\Omega, \tag{3.23c}$$

where in (3.23c) we use that at equilibrium we have $\frac{\partial\overline{\Psi}}{\partial\tilde{\boldsymbol{u}}} = \boldsymbol{0}$.

Note that for volumetric homogenization (and similar for in-plane deformations for thin-shells) the derivative $\frac{\partial\overline{\boldsymbol{x}}}{\partial\overline{\boldsymbol{s}}}$ is quite simple. Similar to (3.8), we can compute a first Piola-Kirchhoff stress $\overline{\boldsymbol{P}} = \frac{\partial\overline{\Psi}}{\partial\overline{\boldsymbol{F}}} = \frac{1}{|\Omega|}\int_\Omega -\boldsymbol{f}\otimes\boldsymbol{X}\, d\Omega$, with forces $\boldsymbol{f} = -\frac{\partial\overline{\Psi}}{\partial\boldsymbol{x}}$. Due to equilibrium, net yarn forces $\frac{\partial\overline{\Psi}}{\partial\boldsymbol{x}}$ are zero in the interior of the tile; therefore, the computation of the coarse stress simplifies to a gathering of tile boundary forces [Schumacher et al. 2018]. These boundary forces are exactly the constraint forces maintaining the average RVE deformation on the boundary. In our implementation, the integral in (3.23) is weighted by a periodicity weight (see Appendix A.2), which after discrete summation aggregates only those forces.

Alternatively, stresses can simply be computed by finite-differencing energies homogenized with (3.22), although this does require multiple simulations, e.g. $\overline{\Psi}(s + \Delta s)$ and $\overline{\Psi}(s - \Delta s)$ for

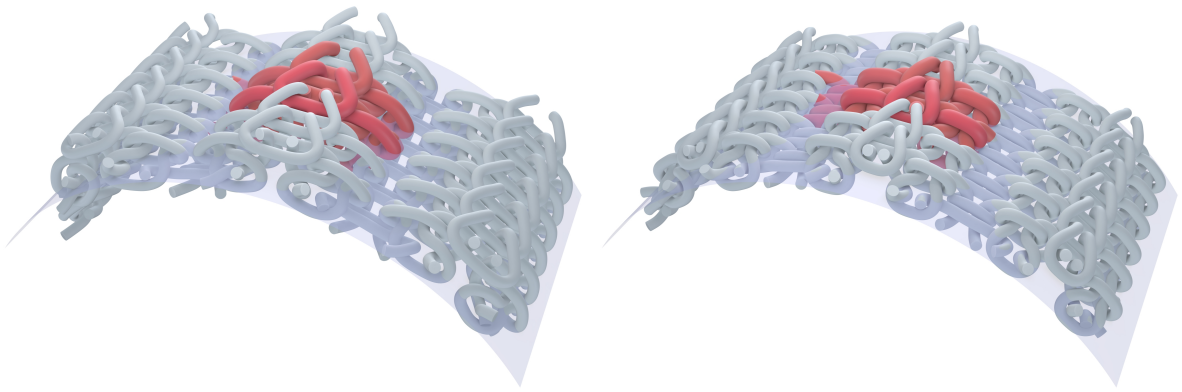


Figure 3.7: We show a stretched rib pattern before (left) and after (right) optimization with the periodic tile highlighted in red, periodic ghost segments in gray, and the midsurface in translucent blue. Notice how the rib pattern's yarn loops naturally tighten under tension while maintaining the curvature of the surface.

central differencing. In experiments for Chapter 6 we used finite-differencing for bending stresses.

To summarize, we are now able to take a macroscale deformation given by $\bar{\mathbf{I}}$ and $\overline{\mathbf{II}}$ and compute a midsurface φ from (3.17)–(3.18). This defines the fluctuation field $\tilde{\boldsymbol{u}}$ through (3.13), on which we can then enforce the translation and periodicity constraints (3.19) and (3.21), and compute homogenized energy area densities $\overline{\Psi}$ with (3.22) or homogenized stresses with (3.23).

3.3 Yarn Pattern Simulation

For any periodic yarn pattern, we aim to compute a mapping from deformation to homogenized energy densities. To minimize the dimensionality of the problem, we seek the energy at the elastostatic equilibrium configuration, subject to the macroscopic deformation. In the context of homogenization, the timescale of the microscale is often faster than that of the macroscale. As a result, microscale oscillations and motions can be considered damped and resolved when viewed on a macroscale. In our case, this equilibrium state corresponds to the physical state with yarn collisions resolved and the yarns being at rest with respect to bending, twisting, and stretching. The elastostatic assumption is common in many applications like animation [Teran et al. 2005], fracture simulation [Müller et al. 2001], and structural optimization [Liu et al. 2014], because it captures the overall behavior of a material without needing to compute dynamic effects.

To deform a microscale periodic yarn patch, we embed it into the RVE as shown in Figure 3.5. Finding the elastostatic equilibrium amounts to a constrained optimization problem of minimizing the homogenized energy with respect to the fluctuations $\tilde{\boldsymbol{u}}$ and subject to the translation and deformation constraints; i.e.,

$$\overline{\Psi} = \min_{\tilde{u}} \frac{1}{|\Gamma|} \int_{\Omega} \Psi(\tilde{\boldsymbol{u}})\, d\Omega \quad \text{s.t. (3.19) and (3.21).} \tag{3.24}$$

Figure 3.7 shows a yarn pattern before and after relaxing it into its optimized state.

The number of tiles within an RVE is a choice that determines which scales of buckling are handled by homogenization, and which ones are handled by the cloth simulator. In this work, we chose to use a small RVE size for each pattern primarily based on computational cost, and

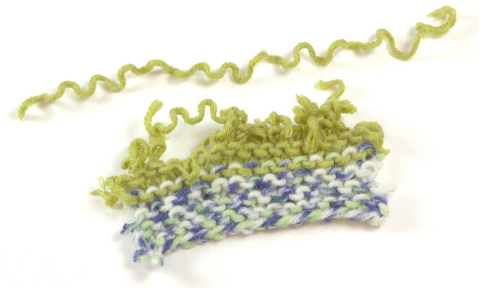


Figure 3.8: In this real-world example, we extracted the top strand of wool yarn from the knit pattern below, and allowed the yarn to come to rest. The yarn clearly has a bent rest shape related to the pattern it was knitted into.

have not explored larger sizes. We leave the study of RVE sizes and buckling frequencies as future work.

3.3.1 Yarn Model

We simulate yarns using discrete elastic rods [Bergou et al. 2008, 2010] with the yarn-level cloth collision forces of [Kaldor et al. 2008] modified for linear spline segments. Most real world yarns consist of many threads wound together, so they may resist bending and twisting much less than stretching. To add more flexibility to our yarn simulations, we therefore add an additional parameter γ to scale bending and twisting energies in relation to stretching energy. Thus, we compute the integral in (3.24) as the sum of stretching E_s, bending E_b, twisting E_t, and collision energies E_c of yarns in the periodic patch:

$$\int_{\Omega} \Psi \, d\Omega = E_s + \gamma E_b + \gamma E_t + E_c. \tag{3.25}$$

For the definition of the individual energies, see [Kaldor et al. 2008] for E_c and [Bergou et al. 2010] for the other terms. As discussed in Chapter 1, we omit inter-yarn friction in the microscale quasistatic optimization.

The elastic energy terms in this model require that we know the rest shape of each yarn. Because the act of knitting and weaving can actually change the rest shape of a yarn (as seen in Figure 3.8), obtaining it is a non-trivial task. In our experiments, we apply a heuristic that the rest pattern should be in equilibrium relative to the stretching energy; inspired by Leaf et al. [2018], we apply tension by shortening the yarns' rest lengths, and then we shrink the periodic lengths of the pattern to find an energy minimum relative to stretching. We explain this initialization process in detail in Appendix A.2.4.

Note that we use the above yarn model in both Chapter 4 to fit a continuum model and in Chapter 5 for computing deformation-aware yarn detail. In Chapter 6, however, we will use an extended yarn model and a slightly altered rest-shape heuristic for the problem of fitting yarn parameters to real fabric measurements.

3.3.2 Periodicity

Yarns on one side of the patch can interact with yarns on the opposite side through periodic collisions or by being periodically connected. Therefore, we have to consider periodic discrete elastic rod and collision forces. We introduce *ghost segments* that copy and tile the yarns along

the periodic field $\boldsymbol{R}^\top \tilde{\boldsymbol{u}}$ implied by the constraint (3.21). Ghost segments do not contribute to the energy in (3.25); they simply copy the motion of the primary yarns and act as colliders and boundaries for the yarn segments in the RVE. Figure 3.7 shows these ghost segments colored gray.

In addition to positional degrees of freedom, [Bergou et al. 2010] incorporates material frames and edge twists. We enforce periodicity on reference frame directors $\underline{\boldsymbol{d}}_\alpha$ and twist variables θ via

$$(\boldsymbol{R}^\top \underline{\boldsymbol{d}}_\alpha)^+ = (\boldsymbol{R}^\top \underline{\boldsymbol{d}}_\alpha)^-, \tag{3.26}$$

$$\theta^+ = \theta^-, \tag{3.27}$$

where + and − denote an original and copied edge respectively.

3.3.3 Homogenization Constraints

For the purposes of homogenization, we have to impose the translation constraint (3.19), periodic vertex positions (3.21), and periodic edge twists (3.27) on the microscale. Additionally, the yarn forces are invariant to a constant twist and to parametric sliding.

We remove the constant twist nullspace by requiring the total twist per periodically connected yarn to be zero:

$$\sum_i \theta_i = 0. \tag{3.28}$$

We found that the reference frames do not drift from their constraint manifold (3.26) over time, so we do not actively enforce this constraint after initialization.

We found that the optimization contains a nullspace that allows yarns to slide: a periodic yarn curve $\boldsymbol{x}(s)$ parametrized by s can slide by a parametric shift Δs without changing its elastic energy E, i.e. $E(\boldsymbol{x}(s)) = E(\boldsymbol{x}(s + \Delta s))$. Geometrically, such a shift corresponds to tangential sliding of yarn while maintaining the same periodic shape. To the optimizer any such state is equivalent, and the actual result may depend arbitrarily on numeric solver parameters. This nullspace does not affect the energies in the system by definition. However, as we'll see later in Chapter 5 (Figure 5.5), it may produce artifacts when trying to interpolate between two shifted parametrizations. We eliminate this parametric yarn sliding by adding a constraint to the optimization, effectively removing the nullspace. Specifically, we fix one vertex per periodic yarn to remain on the boundary of the pattern:

$$\tilde{\boldsymbol{u}} \cdot (\nabla \boldsymbol{\varphi}\, \boldsymbol{N}) = 0, \tag{3.29}$$

where $\boldsymbol{N}$ is the undeformed normal to the respective pattern boundary, either $\boldsymbol{N} = (1, 0)^\top$ or $\boldsymbol{N} = (0, 1)^\top$. Note that $\nabla \boldsymbol{\varphi}$ is by definition $\boldsymbol{RS}$.

Implementation We enforce the periodicity constraints by eliminating the copied degrees of freedom from the linear system in the Newton step. Exploiting the fact that any periodic vertex or twist relates linearly to exactly one other vertex or twist through (3.21) and (3.27), we can define reduced degrees of freedom $\boldsymbol{y}$ through

$$\tilde{\boldsymbol{C}}\boldsymbol{y} + \tilde{\boldsymbol{d}} = \boldsymbol{q}, \tag{3.30}$$

where $\boldsymbol{q}$ is the vector of all vertex positions and edge twists. Notably, $\tilde{\boldsymbol{C}}$ is sparse and will preserve the sparsity of the Newton system. This elimination of variables is based on parametrizing the nullspace of all periodicity constraints. We discuss its construction in Appendix A.2.2.

On the other hand, due to its density, enforcing the translation constraint (3.19) by parametrizing its nullspace would result in a dense $\tilde{\boldsymbol{C}}$. Instead, we enforce this constraint with Lagrange multipliers. We also use Lagrange multipliers for the nullspaces of constant twists and parametric sliding. We concatenate the translation, twist and sliding constraints to get

$$\boldsymbol{C}_L \boldsymbol{q} = \boldsymbol{d}_L. \tag{3.31}$$

Alternative Implementation Another way of implementing periodicity constraints is to expose only the original (non-ghost) degrees of freedom to the Newton solver as $\boldsymbol{g} = (\boldsymbol{R}^\top \tilde{\boldsymbol{u}}, \theta)$, i.e. using *unrotated* displacements. Internally, $\boldsymbol{g}$ will be copied to all periodic vertices with the displacements appropriately rotated by $\boldsymbol{R}$ at that point. While the result is the same as the implementation discussed in this thesis, this treatment simplifies constraint implementation and can produce more readable and separable code. This variant is used in the published code at `https://git.ist.ac.at/gsperl/HYLC`, which was used to generate the results in Chapter 5. Rodriguez et al. [2022], who build on our method, define an analytic mapping for cylindrical curvature and then similarly express the constraints in a computational domain based on the inverse mapping.

3.3.4 Optimization Step

We can now solve the constrained minimization problem in (3.24). Using Newton iteration, each step to solve for increments $\delta\boldsymbol{y}$ and Lagrange multipliers $\boldsymbol{\lambda}$ is given by

$$\begin{pmatrix} \tilde{\boldsymbol{C}}^\top \boldsymbol{H} \tilde{\boldsymbol{C}} + \alpha \boldsymbol{I} & \tilde{\boldsymbol{C}}^\top \boldsymbol{C}_L{}^\top \\ \boldsymbol{C}_L \tilde{\boldsymbol{C}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta\boldsymbol{y} \\ \boldsymbol{\lambda} \end{pmatrix} = - \begin{pmatrix} \tilde{\boldsymbol{C}}^\top \nabla E \\ \boldsymbol{C}_L \boldsymbol{q} - \boldsymbol{d}_L \end{pmatrix}, \tag{3.32}$$

where E is the total energy, $\boldsymbol{H} = \frac{\partial^2 E}{\partial \boldsymbol{q} \partial \boldsymbol{q}}$ is its Hessian, and α is an exponentially decaying regularizer to help convergence. We also limit the maximal vertex displacement per step to a fraction of a yarn radius to avoid missing collisions between iterations, and we observed improved numerical conditioning if we rescale positional degrees of freedom relative to twists. We provide these details, as well as initialization and stopping criteria for this optimization algorithm in Appendix A.2.3.

The following chapters will use the homogenization procedure discussed above, starting with a method for fitting continuum materials that reproduce the large-scale elastic behavior of yarn-based fabric.

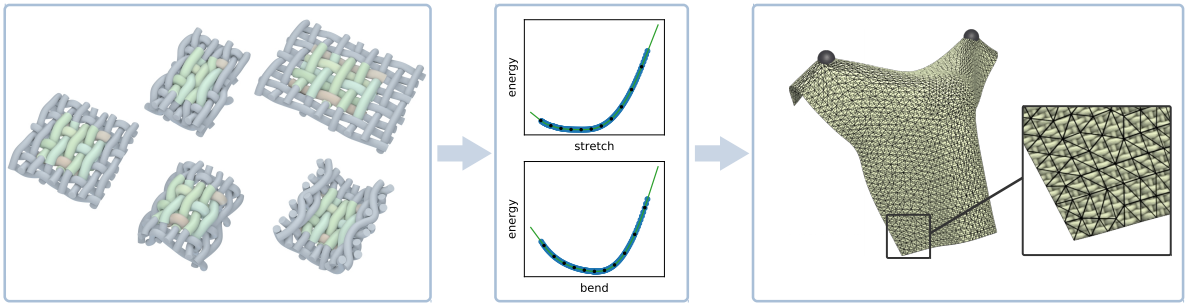
CHAPTER 4

Homogenized Yarn-Level Cloth

We have discussed that the simulation of knitted and woven materials directly as a collection of threads can faithfully reproduce the fascinating large-scale behavior caused by small-scale yarn interactions —for example, knit loops tighten and pull on adjacent rows to produce fabric-scale anisotropy and area-preservation. Can we reproduce these large-scale mechanics without needing to simulate hundreds or thousands of individual yarns in a garment?

In this chapter, we propose to approximate the *homogenized* fabric mechanics with a continuum model. To do so, we use our quasi-static yarn simulator with novel co-rotated periodic boundary conditions established in Chapter 3. With it, we sample the knitted or woven material's behavior in response to a number of different in-plane and bending deformations. We use regularized spline regression to fit a macroscopic energy density model to this data (Section 4.1), and use the new material model directly in a thin-shell cloth simulator (Section 4.2). This way, we are able to reproduce the expected effects like the stiffness of woven fabrics, and the highly deformable nature and anisotropy of knitted fabrics (Section 4.3). The method is based entirely on efficient simulations of only a small periodic sample of the underlying yarn pattern. As such, it can generate entirely new material models quickly without the need for testing apparatuses or human intervention. We will discuss later in Chapter 6 how these simulation models can be combined with real-world measurements to enhance their predictiveness and

Figure 4.1: Left: A comparison between direct yarn-level simulation (YLC) and simulation with our homogenized model (HYLC); our homogenized model accurately captures the non-trivial elastic stretching and bending response of the fabric. Middle and right: Results simulated with homogenized continuum models of woven and knitted patterns; our method allows us to efficiently compute large-scale simulations where direct yarn-level simulation would be prohibitively slow.



(a) Precomputed Homogenization (b) Fitting (c) Thin-shell Simulation

Figure 4.2: Our method takes a periodic yarn pattern and produces a homogenized cloth material model. (a) We impose macroscopic in-plane and bending deformations on a periodic pattern. (b) We compute homogenized energy density samples for ranges of deformations and fit them with regularized splines. (c) The resulting material model can be used to efficiently simulate cloth by computing elastic responses of the pattern to deformations.

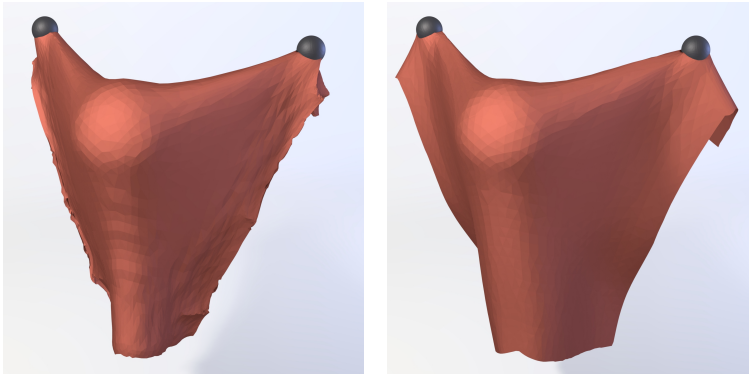


Figure 4.3: Insufficient regularization can negatively affect simulated rest shapes. Here, a draped rib knit shows noisy boundaries (left) compared to a fit with better regularization (right).

realism. Figure 4.1 shows some results achieved with our method, and Figure 4.2 provides an overview of our method.

4.1 Fitting

With the homogenization procedure discussed in the previous chapter, we are able to compute an energy density $\overline{\Psi}$ for a yarn pattern given an input deformation $\bar{\mathbf{I}}, \overline{\mathbf{II}}$. Our next step is to build a database of entries sampling this $\overline{\Psi}(\bar{\mathbf{I}}, \overline{\mathbf{II}})$ function, and then approximate the data by fitting a model to it. However, the energy landscape can be noisy due to multiple microscale equilibria — the yarn pattern can buckle, interacting yarns are generally multistable and slide over each other. Especially in compressive regimes, the pattern can buckle differently for similar strains, leading to noise in the energies. Local minima in the fit then introduce noisy restshapes and popping in the final macroscale simulation (see Figure 4.3). Additionally, our data is neither convex nor is it well-fit by polynomials. After experimenting with several fitting schemes, we settled on the strategy of first regularizing the input data, and then fitting a model as a sum of regularized splines while enforcing quasiconvexity and piecewise monotone interpolation. We will discuss the main ideas of the fitting procedure in this section, and we provide further details in Appendix A.3.

4.1.1 Parametrization and Sampling

We begin by choosing a reparametrization of the input strains $\bar{\mathbf{I}}$ and $\overline{\mathbf{II}}$ that is better suited to sampling and interpolation. We desire each input parameter to be valid over a fixed interval independent of other parameter values, so that we can use standard interpolation schemes over rectilinear grids. Furthermore, we wish to avoid sampling over the full n-dimensional space of possible strains, but still capture pairwise interactions such as the Poisson's ratio, influence of stretching on bending, and so on.

To start, we reparametrize the in-plane strains. Using the entries of $\bar{\mathbf{I}}$ is problematic as its off-diagonal entry $\boldsymbol{a}_1 \cdot \boldsymbol{a}_2$ not only encodes the shearing angle but is also influenced by the lengths of the $\boldsymbol{a}_\alpha$. Instead, we define weft-stretching s_x, shearing s_a, and warp-stretching s_y strains as

$$s_x = \sqrt{\bar{\mathrm{I}}_{11}} - 1, \qquad s_a = \frac{\bar{\mathrm{I}}_{12}}{\sqrt{\bar{\mathrm{I}}_{11}\bar{\mathrm{I}}_{22}}}, \qquad s_y = \sqrt{\bar{\mathrm{I}}_{22}} - 1, \tag{4.1}$$

and the combined in-plane strain $\boldsymbol{s} = \begin{pmatrix} s_x & s_a & s_y \end{pmatrix}^\top$. Here, we use the terms "weft" and "warp" to refer to the directions X_1 and X_2 respectively.

The difficulty with the bending strain $\overline{\mathbf{II}}$ is that it is not possible to construct a microscale patch with constant strain unless it is singly curved, i.e. the rank of $\overline{\mathbf{II}}$ is ≤ 1. We were further unable to find a satisfactory parametrization for the space of all singly-curved bending strains. Instead, we choose to only sample the response to bending along two orthogonal directions. That is, we collect one set of data with $\overline{\mathbf{II}}$ of the form $\mathrm{diag}(\lambda_x, 0)$, and another set with $\overline{\mathbf{II}} = \mathrm{diag}(0, \lambda_y)$.

The data then represent samples of the function along two subspaces: one with arbitrary $\boldsymbol{s}$ and bending only in x, and one with arbitrary $\boldsymbol{s}$ and bending only in y. As described the next section, we interpolate the data in each subspace to obtain fits $\overline{\Psi}_x(\boldsymbol{s}, \lambda_x)$ and $\overline{\Psi}_y(\boldsymbol{s}, \lambda_y)$. Finally, we describe how to interpolate between them to define the fitted energy density for arbitrary bending strain $\overline{\mathbf{II}}$.

Note that our choice of axis-aligned bending and stretching corresponds to the weft and warp directions that are dominant in the patterns we investigate, but in general the orientation of the bases is arbitrary.

Prior to fitting, we normalize all strains $(s_x, s_a, s_y, \lambda_x, \lambda_y)$ by their maximum absolute values in the data, which ensures that stretching and bending strains are treated as equally important. We have tried various strategies to mitigate the noise in the data induced by buckling, including prohibiting specific buckling modes through constraints and even penalizing yarn motion normal to the midsurface. However, we were unable to eliminate noise without affecting the overall elastic response and concluded that homogenization of microscale buckling is a difficult problem. As a first step, we settled on regularizing the data by re-sampling it using moving least squares interpolation.

4.1.2 Fitting and Interpolation

We define a fitting procedure for multidimensional data which captures pairwise interactions between parameters without requiring high-dimensional sampling. Consider a function f depending on many parameters $\theta_1, \theta_2, \ldots$. Inspired by Miguel et al. [2016], we additively split

it into the form

$$f(\theta_1, \ldots, \theta_n) = f_0 + \sum_i f_i(\theta_i) + \sum_{i<j} f_{ij}(\theta_i, \theta_j). \tag{4.2}$$

Without loss of generality, we may fix $f_i(0) = 0$ and $f_{ij}(0, \theta_j) = f_{ij}(\theta_i, 0) = 0$. Thus the one-dimensional term f_i encodes the response to θ_i holding other parameters at zero, and the two-dimensional term f_{ij} encodes the *residual* response to both θ_i and θ_j, i.e. the component of $f(\ldots, \theta_i, \ldots, \theta_j, \ldots)$ not explained by $f_0 + f_i(\theta_i) + f_j(\theta_j)$.

Therefore, the f_{ij} terms describe cross-modal material responses, including stretching in two directions or simultaneous stretching and bending. Notably, our homogenization method is capable of sampling these cross-modal deformations.

To fit the components of (4.2), we measure $f_0 = f(0, 0, 0, \ldots)$, we fit the one-dimensional f_i terms using piecewise monotone cubic splines [Fritsch and Carlson 1980], and we fit the two-dimensional residual f_{ij} terms using our novel extension of Carlson and Fritsch [1989] to spline patches (Appendix B). We also apply a heuristic outward marching algorithm to ensure quasiconvexity. We provide details for each of these steps in Appendix A.3. Miguel et al. [2016] enforce convexity in their fits. However, we found that this would not describe our data well, and we opted for quasi-convexity as the closest choice. Specifically, we want to enforce that individual terms f_i and f_{ij} have a single well-defined minimum. While the aggregate sum of terms can still produce multiple minima, we found that our heuristics as explained in Appendix A.3 produce reasonable fits and more stable simulations (see Figure 4.3) with the downside of not perfectly fitting the data. Outside of the sampled range, we linearly extrapolate the fitted splines. Figure 4.4 shows data and fit for the 1D splines. Figure 4.5 compares data, 1D fits, 2D residuals, and the cumulative fit.

Our method makes the simplifying assumption that there are only pairwise interactions between parameters. What this assumption buys us is a dramatic economy of sampling: even for arbitrarily high-dimensional parameter spaces, our procedure only needs samples along coordinate axes and 2D coordinate planes. When the assumption is violated, however, our approach may not preserve convexity. For example, $f(x, y, z) = \max(x^2, y^2, z^2)$ is a convex function for which our fit is nonconvex.

The above procedure is applied to the singly-curved data $\overline{\Psi}_x$ and $\overline{\Psi}_y$ defined previously. Of course, the zero-curvature data points and the 1D and 2D fitting terms not involving curvature will be shared between both. Finally, to define our fitted energy density for an arbitrary curvature $\overline{\mathbf{II}}$, we look at the eigenvalues of $\overline{\mathbf{II}}$, λ_1 and λ_2, and the squared cosine c^2 of the angle between the eigenvector corresponding to λ_1 and the x-axis. In Appendix A.4, we show how to robustly compute these values. Now we define $\overline{\Psi}(\boldsymbol{s}, \overline{\mathbf{II}})$ as

$$\begin{aligned} \overline{\Psi}(\boldsymbol{s}, \overline{\mathbf{II}}) = c^2 &\left(\overline{\Psi}_x(\boldsymbol{s}, \lambda_1) + \overline{\Psi}_y(\boldsymbol{s}, \lambda_2)\right) \\ &+ (1 - c^2)\left(\overline{\Psi}_x(\boldsymbol{s}, \lambda_2) + \overline{\Psi}_y(\boldsymbol{s}, \lambda_1)\right). \end{aligned} \tag{4.3}$$

Limitations We found the fitting problem particularly challenging due to the complex interactions between deformation modes, the numerical noise in the data, and especially the sensitivity of macroscale simulations to local minima in the energy density (Figure 4.3). We invested a great deal of effort to design a fitting scheme that works well for all the yarn patterns we tested, but we found a few cases unavoidable, which we summarize below. Firstly, to ensure

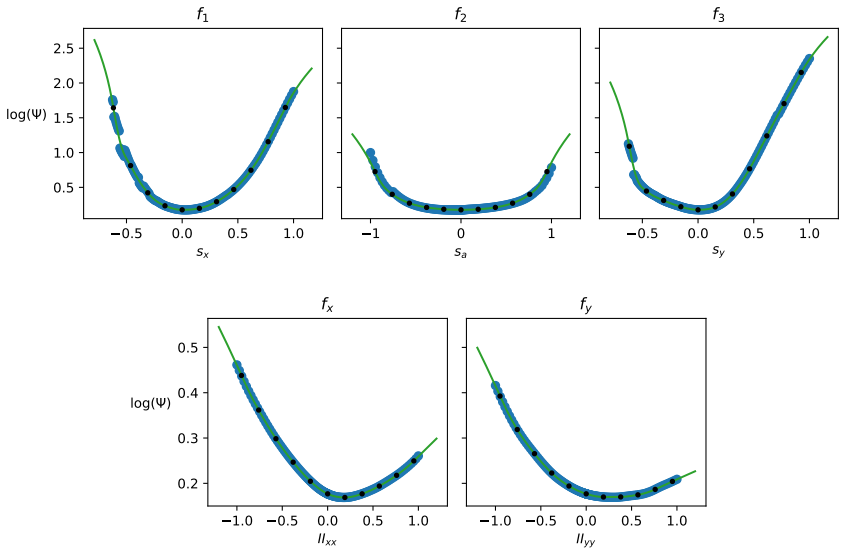


Figure 4.4: One-dimensional in-plane (left) and bending (right) terms for the honeycomb pattern. We show data in blue, the fit as a green line, and spline control points as black dots. Notice the off-center minimum for the bending terms, which corresponds to the pattern's curved rest shape.

a decent fit for the "stockinette" pattern, which features a strong tendency to curl, we found it necessary to concentrate spline control points for 2D residual terms involving bending strains more closely around the origin (Figure 4.6), and to apply a higher quasiconvexity parameter in the marching step. We believe that this may be caused by the far-off-center bending minimum and the additively split model thus creating local minima. Secondly, we observed yarn-level reference simulations to exhibit symmetric rest shapes with zero shear; to ensure that this behavior is preserved in our macroscale simulations, we symmetrized our data with respect to s_a. Finally, we disabled our heuristic quasiconvexity marching for the two-dimensional $f_{13}(s_x, s_y)$ term, which would otherwise prevent us from modeling Poisson's ratio.

4.2 Cloth Simulation

We now want to drive a thin-shell cloth simulator using the continuum models fit in the previous section. The cloth is discretized as a triangle mesh, which represents the macroscale thin-shell midsurface $\overline{\varphi}$. Similarly, we need to discretize $\overline{\mathbf{I}}$ and $\overline{\mathbf{II}}$ to compute in-plane and bending strains, (4.1) and $\lambda_1, \lambda_2, c^2$, on the triangle mesh. For robust simulation, we use implicit integration, which requires computing the Hessian of the energy. To improve stability, we enforce positive definiteness in the Hessian. Dynamic yarn friction is partially modeled via Rayleigh damping in the continuum simulations, but we leave the inclusion of friction into the homogenization procedure as future work.

For each triangle, we first compute its deformation gradient

$$\boldsymbol{F}_\triangle = \left(\overline{\boldsymbol{\varphi}}_1 - \overline{\boldsymbol{\varphi}}_0, \overline{\boldsymbol{\varphi}}_2 - \overline{\boldsymbol{\varphi}}_0\right)\left(\overline{\boldsymbol{X}}_1 - \overline{\boldsymbol{X}}_0, \overline{\boldsymbol{X}}_2 - \overline{\boldsymbol{X}}_0\right)^{-1}, \tag{4.4}$$

where $\overline{\boldsymbol{\varphi}}_j$ and $\overline{\boldsymbol{X}}_j$ are the world-space and material-space coordinates of vertex j, and the

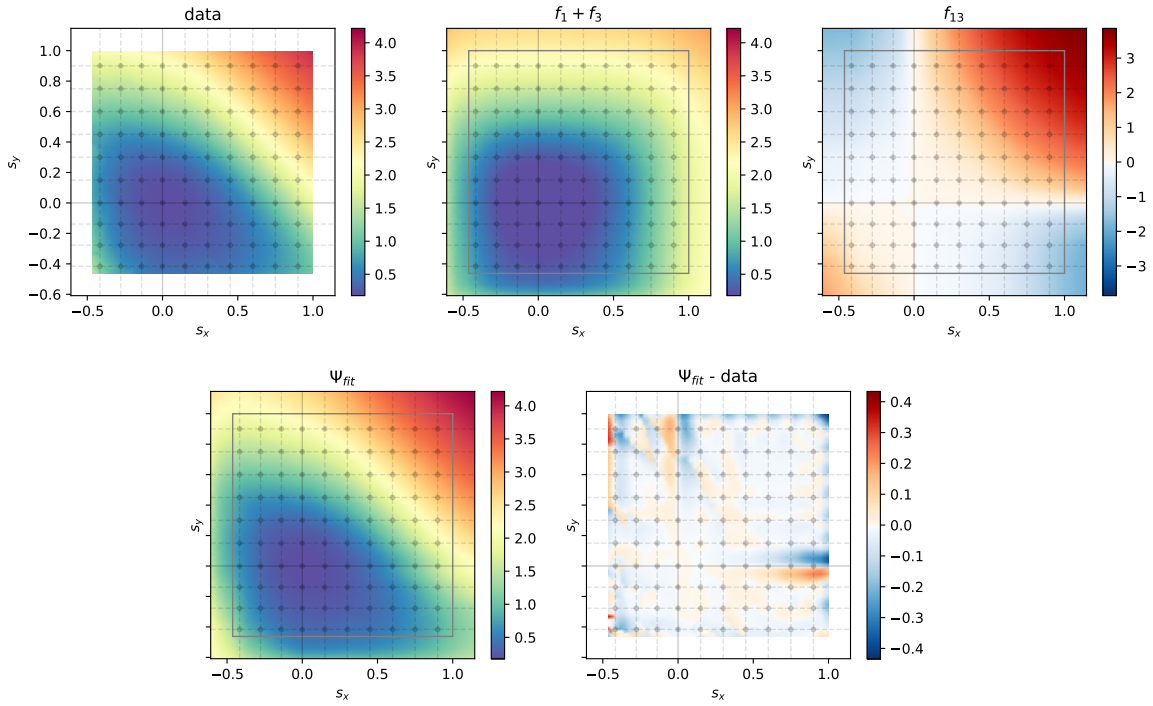


Figure 4.5: Fit of the 2D term $f_{13}(s_x, s_y)$ for the honeycomb pattern. From left to right: data, sum of 1D fits, 2D residual, cumulative fit, and fitting error. The colors show the magnitude on a symmetric-log scale, and we show extrapolated values outside of the data range (indicated as a rectangle). Note that this term shows the area preservation of the material; while increasing tension along one axis, the minimum along the other moves towards a compressed state. Crucially, this behavior is missing from just the sum of 1D terms.

triangle-averaged shape operator [Grinspun et al. 2006]

$$\boldsymbol{\Lambda} = \sum_i \frac{\theta_i}{2Al_i} \boldsymbol{t}_i \otimes \boldsymbol{t}_i, \tag{4.5}$$

where θ_i is the signed angle between this and the i-th neighboring triangle's normals, A is the triangle area, l_i are edge lengths, and $\boldsymbol{t}_i$ are vectors of length l_i perpendicular to each edge and the inner triangle normal. All quantities in (4.5) are computed in *world-space*. With this, we compute the discrete fundamental forms as

$$\bar{\mathbf{I}}_\triangle = \boldsymbol{F}_\triangle{}^\top \boldsymbol{F}_\triangle, \tag{4.6}$$

$$\overline{\mathbf{II}}_\triangle = \boldsymbol{F}_\triangle{}^\top \boldsymbol{\Lambda}\, \boldsymbol{F}_\triangle. \tag{4.7}$$

Because of (4.5), the degrees of freedom involved in a triangle's strain also include the triangle vertices of up to three neighboring triangles. Denoting the combined degrees of freedom as $\boldsymbol{q}_\triangle$ and the collected strains $\boldsymbol{z} = (s_x, s_a, s_y, \lambda_1, \lambda_2, c^2)$, the total energy of a triangle is given by

$$E_\triangle = A\, \overline{\Psi}(\boldsymbol{z}(\boldsymbol{q}_\triangle)). \tag{4.8}$$

Since our energies are nonconvex, their Hessians are not guaranteed to be positive definite, which negatively affects stability. Inspired by Teran et al. [2005], we enforce positive definiteness by clamping the eigenvalues of per-triangle sub-Hessians $\frac{\partial^2 E_\triangle}{\partial \boldsymbol{q}_\triangle \partial \boldsymbol{q}_\triangle}$ to be non-negative using an eigensolver for self-adjoint matrices in the library Eigen [Guennebaud et al. 2010]. The global system in the implicit timestep will then be positive definite as a sum of the positive semi-definite sub-Hessians and the positive definite global mass matrix.

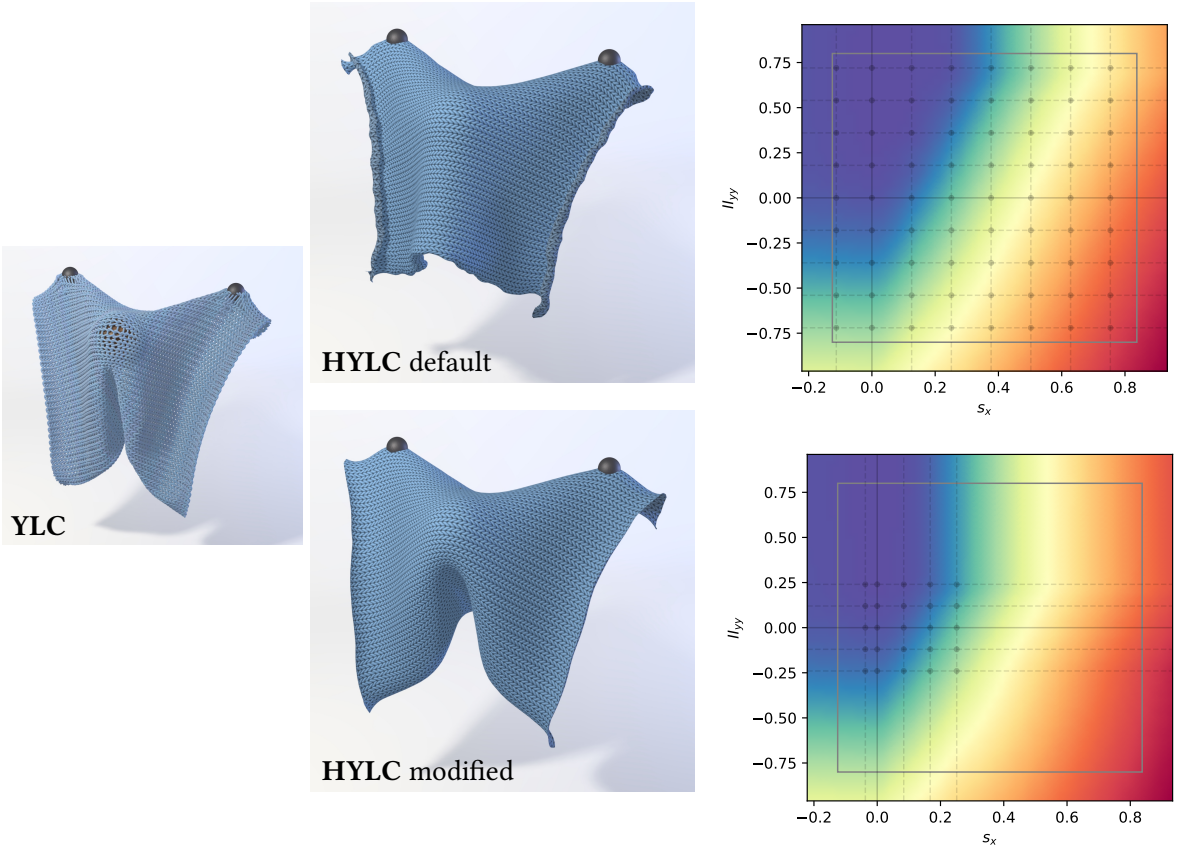
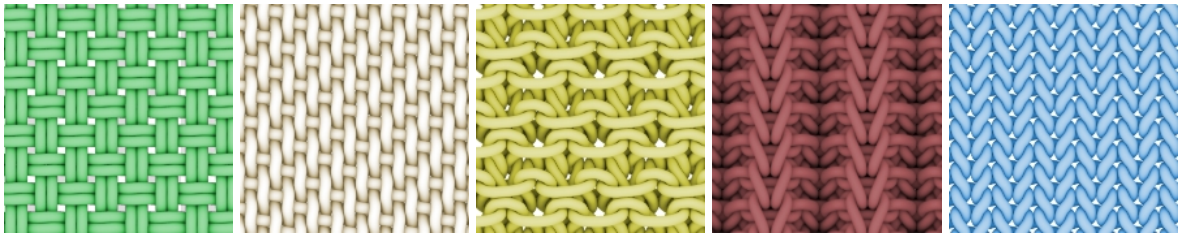


Figure 4.6: We show a simulated restshape and the plot for a representative bending residual for the homogenized stockinette with default and modified spline control points, and a yarn-level reference (YLC). Even though the fit is smooth, default control point locations create artifacts at the fabric boundary.

4.3 Results

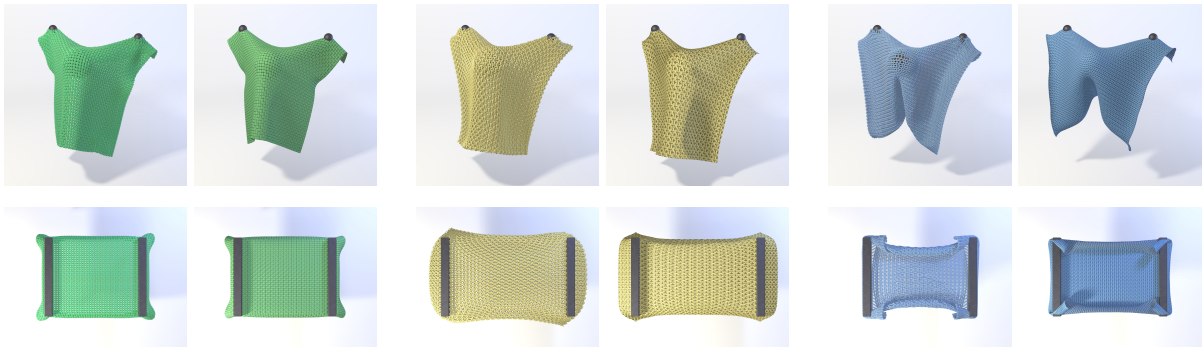
To summarize, our pipeline first takes in a periodic yarn pattern and elastic rod material properties, simulates the pattern subject to various deformed boundary conditions, and records the resulting potential energy density. We then create a data-driven strain-parameterized material model for each yarn pattern and simulate the material in an existing thin-shell finite element solver (ArcSim [Narain et al. 2012, 2013]). Code for periodic pattern simulation, the full fitting algorithm as well as the raw data are available at `https://git.ist.ac.at/gsperl/HYLC`, and our code for cloth simulation is available at `https://git.ist.ac.at/gsperl/ARCSim-HYLC`.

In our experiments, we wanted to model a variety of yarn patterns with notably different topologies and macroscale material effects. We drew several patterns from the yarn pattern database of Leaf et al. [2018] (`basket2_2`, `satin2_3`, `slip_stitch_honeycomb`, and `cartridge_belt_rib`), and implemented a custom stockinette knit pattern of our own. Figure 4.7 shows the five patterns. The knitted patterns are topologically quite different from each other and from the woven patterns, leading to significant variance in macroscopic effects like area preservation, resistance to stretching, and out-of-plane curling. We rescale the patterns to have a yarn radius of 1 mm and smaller variants of the satin and stockinette patterns to 0.1 mm. Table 4.1 lists the yarn-level parameters for each pattern; we choose parameters to achieve realistic-looking yarn-level simulations.



basket satin honeycomb rib stockinette

Figure 4.7: The patterns used in our results with abbreviated names.



YLC HYLC YLC HYLC YLC HYLC

(a) basket weave (b) honeycomb (c) stockinette

Figure 4.8: Comparison of direct yarn-level simulation (YLC) to simulation with our homogenized continuum models (HYLC) for drapes and stretching tests of three patterns. Our method is able to capture a wide array of phenomena such as the Poisson's ratio of the honeycomb pattern, or the more exotic restshape and curling under tension of the stockinette pattern at a fraction of the cost.

We render cloth simulated with our models using ambient occlusion and normal map textures, which we create by projecting the periodic yarn patterns. Thus, the results in presented in this chapter do not show visible gaps between yarns as seen in Figure 4.6, regardless of the quality of the homogenization. We attribute the differences between yarn-level and homogenized results in Figure 4.6 to both texture mapping as well as an imperfectly homogenized model. It is possible to drive the deformation of detailed yarn-level geometry using the coarser, simulated mesh, as we will see in the next chapter (Chapter 5).

4.3.1 Validation

To validate our homogenized macro-material models, we run side-by-side comparison simulations between our macro-material cloth simulator and a brute-force yarn-level cloth simulator. We compare the behavior of a 30 cm × 30 cm square patch of material when stretched in different directions and draped over a spherical obstacle. Some of these comparisons are displayed in Figure 4.8 and Figure 4.1, and all of them are included with the paper [Sperl et al. 2020] and openly available at `https://doi.org/10.1145/3386569.3392412`.

Our homogenized yarn-level cloth models generally agree well with the yarn-level cloth simulations, even though the various yarn patterns behave very differently from each other: the woven materials tend to be stiffer and exhibit no tendency to preserve area when stretched; the rib knit exhibits fairly extreme anisotropy when stretching; the stockinette stitch curls up

Table 4.1: Yarn-level parameters per pattern including Young's modulus E, the shear modulus G, the bending and twisting stiffness multiplier γ, the collision stiffness k_{contact}, and the density ρ.

Pattern	E (Pa)	G (Pa)	γ	k_{contact} $\left(\frac{\text{kg}}{\text{s}^2}\right)$	ρ $\left(\frac{\text{kg}}{\text{m}^3}\right)$
basket	1e5	4e4	0.1	1.2e1	1.2e2
honey	5e5	2e5	0.1	6e1	1.2e2
rib	5e5	2e5	0.001	6e1	6e1
satin	1e5	4e4	0.1	1.2e1	1.2e2
stock.	5e5	2e5	0.001	6e1	1.2e2
satin small	1e6	4e5	1	1e2	1.2e3
stock. small	1e6	4e5	1	1e2	1.2e3

on the boundaries when stretched or left to hang freely.

For the yarn-level simulations in this comparison, we used the non-rigid motion damping of Kaldor et al. [2008]. Because our material models are based on elastic properties of the cloth, we did not yet attempt to learn damping properties. Instead, we used the continuum Rayleigh damping model implemented in ArcSim, which we tuned to empirically match the yarn-level damping model.

Note that our material models are extracted from periodic yarn patterns, so they should be able to adequately reproduce the behavior of a yarn-level simulation near the interior of the cloth. However, knitted garments generally have different stitches or fasteners near boundaries, which disrupts this periodic structure; indeed, to model boundary effects in our yarn-level simulator, we effectively "glue" the yarns together with springs that are pre-stretched in the thickness direction. These boundary effects were not included in our periodic homogenization, so we do not expect our material to behave perfectly near boundaries. Nevertheless, our results do show relatively similar boundary behaviors to the yarn-level examples.

To illustrate the merits of our multidimensional fitting procedure described in Section 4.1, we also compared our method's behavior with and without two-dimensional energy terms. As seen in Figure 4.9, the materials with only one-dimensional stress response do a reasonable job of approximating the overall stretching and bending resistance, but they fail to capture more complex two-dimensional compensations. Notably, the 1D models cannot capture Poisson-like behaviors, where stretching in one direction causes the material to compress in the other.

Performance The computational complexity of a yarn-level cloth simulator scales with the number of yarn segments. In contrast, the performance of our macroscale material scales with the number of elements in a cloth simulator, multiplied by the cost of evaluating our potential energy function (or its gradient). Yarn-level simulations also invest computational resources into carefully handling persistent inter-yarn collisions, either through small time steps or more clever collision handling. Our method deals with those persistent contacts in its preprocessing phase, and only deals with large-scale self-collisions within the cloth solver. Because our method sidesteps most of the performance bottlenecks in a yarn-level cloth simulator, we expect our method to achieve a large speedup over a yarn-level cloth, especially when the yarn density is high. Additionally, using an implicit cloth solver allows us to take larger timesteps compared to the explicit yarn-level solver, where computing Hessians becomes infeasible. Although these side-by-side examples use a modest number of yarn segments, our simulator

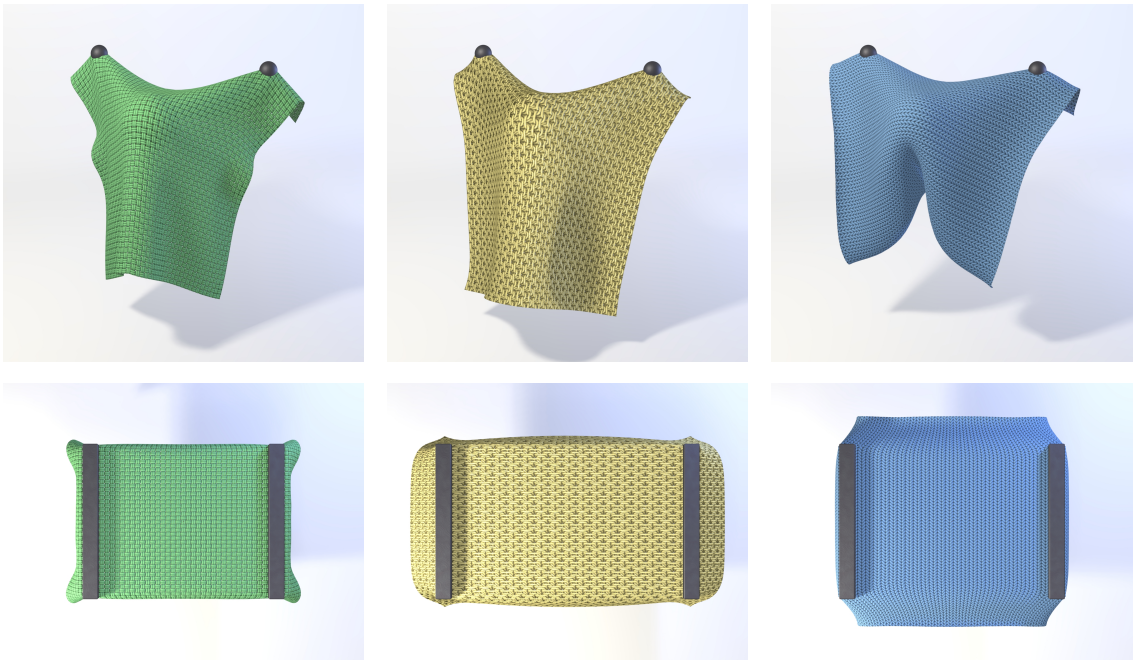


Figure 4.9: Only fitting the one-dimensional energy terms for the same models as in Figure 4.8 shows that overall draped shapes are still captured nicely and are arguably faster and easier to fit. However, area-preservation effects and curling under tension are modeled by two-dimensional terms and as a result are lost in the simpler model.

Figure 4.10: We demonstrate the effectiveness and the rich behavior of our homogenized models for all of our patterns on simulations of sweaters and t-shirts. This freeze frame highlights: stronger stretching resistance of woven fabric (basket and satin), the anisotropy of the rib, curling of the stockinette, and the folds of the small-scale patterns.

shows significant speedups from ×3.3 to up to ×46, as seen in Table 4.2. Across the patterns, sampling the data for fitting takes from 15min to 76min, and the fitting itself takes less than a minute, further highlighting the cost benefit of precomputing inexpensive simulations. The stockinette examples in Table 4.2 have a higher "sec/frame" and number of vertices due to finer adaptive remeshing needed for resolving tight curls.

Our proposed constitutive model depends on the second fundamental form and thus requires more computation compared to standard bending models based on dihedral angles such as [Grinspun et al. 2003].

Table 4.2: Simulation timings for the comparisons of direct yarn-level simulation (YLC) and with our method (HYLC). Pattern names are abbreviated. All tests were performed for two orientations of the cloth (original and 90° rotation), and their videos can be found with the published paper.Δt denotes the timestep in seconds. sec/step denotes the average seconds per timestep. sec/frame denotes the average seconds per frame for a reference framerate of 30fps. # Vertices denotes the number of vertices in yarn-level simulations, and the average number of vertices for thin-shell simulations, which are subject to remeshing.

Simulation		HYLC				YLC			
		Δt	sec/step	sec/frame	# Vertices*	Δt	sec/step	sec/frame	# Vertices
basket drape	Fig. 4.8	2.09e-04	0.46	73.46	2276	1e-05	0.13	(×5.9) 430.43	65188
basket drape 90°		2.09e-04	0.46	72.74	2229	2e-05	0.23	(×5.2) 381.50	65188
basket stretch	Fig. 4.8	2.09e-04	0.50	80.58	2668	1e-05	0.17	(×7.0) 560.43	65188
basket stretch 90°		2.09e-04	0.52	82.96	2657	1e-05	0.24	(×9.6) 797.10	65188
honey drape	Fig. 4.8	1.67e-04	0.41	81.91	2091	1e-05	0.36	(×14.7) 1206.40	118140
honey drape 90°		1.28e-04	0.42	109.65	2127	1e-05	0.39	(×12.0) 1314.72	118140
honey stretch	Fig. 4.8	1.67e-04	0.49	98.67	2370	1e-05	0.31	(×10.3) 1017.60	118140
honey stretch 90°	Fig. 4.1	1.67e-04	0.44	87.50	2376	1e-05	0.29	(×10.9) 954.20	118140
rib drape		2.09e-04	0.48	76.86	2337	5e-06	0.39	(×34.2) 2625.12	157592
rib drape 90°		2.09e-04	0.48	77.32	2374	5e-06	0.53	(×46.0) 3559.10	157592
rib stretch		2.09e-04	0.48	76.35	2577	5e-06	0.38	(×33.3) 2542.47	157592
rib stretch 90°		2.09e-04	0.47	75.07	2541	5e-06	0.45	(×39.6) 2971.27	157592
satin drape	Fig. 4.1	2.09e-04	0.48	77.21	2297	1e-05	0.56	(×24.0) 1855.08	95040
satin drape 90°		2.09e-04	0.47	74.74	2246	1e-05	0.56	(×24.9) 1861.55	95040
satin stretch		2.09e-04	0.44	70.82	2500	1e-05	0.35	(×16.6) 1176.50	95040
satin stretch 90°		2.09e-04	0.50	79.66	2684	1e-05	0.30	(×12.4) 985.17	95040
stock. drape	Fig. 4.8	2.09e-04	0.96	152.96	3390	1e-05	0.19	(×4.2) 643.08	76156
stock. drape 90°		2.09e-04	1.03	165.04	3383	4e-06	0.08	(×4.0) 652.35	76156
stock. stretch	Fig. 4.8	2.09e-04	1.15	184.17	4415	1e-05	0.18	(×3.3) 615.83	76156
stock. stretch 90°		2.09e-04	0.79	126.91	3869	4e-06	0.08	(×5.4) 684.30	76156

4.3.2 Large-scale Simulations

Because our homogenized material's computational complexity is now independent of the number of yarns, we are able to approximate the behavior of large garments with a high density of yarns. Figure 4.11 shows draped cloth simulated with models of stockinette and satin patterns rescaled to 10% of their original size. The stitch density of these materials is one hundred times higher than those we were able to feasibly simulate with a yarn-level simulator, so we do not have any direct performance or behavioral comparisons to report here.

Similarly, we are able to simulate large garments such as sweaters and shirts (Figure 4.1, Figure 4.10). We note that these homogenized knitted materials retain their unique material properties, like stretchiness (honey), anisotropic effects (rib), or curling at the boundaries (stockinette), despite the fact that they were simulated with a continuum-mechanics based cloth solver. For comparison, a direct yarn-level simulation of a stockinette sweater would require over 1.7 million vertices, compared to the 76 thousand vertices in our yarn-level validation tests. The small-stockinette shirt would require 36 million vertices.

Because our homogenized materials rely on triangle meshes instead of knitted patterns to determine their geometry, it is straightforward to simulate garments with more exotic shapes using our method (Figure 4.1, Figure 4.12). We report the simulation timings for each of these results in Table 4.3.

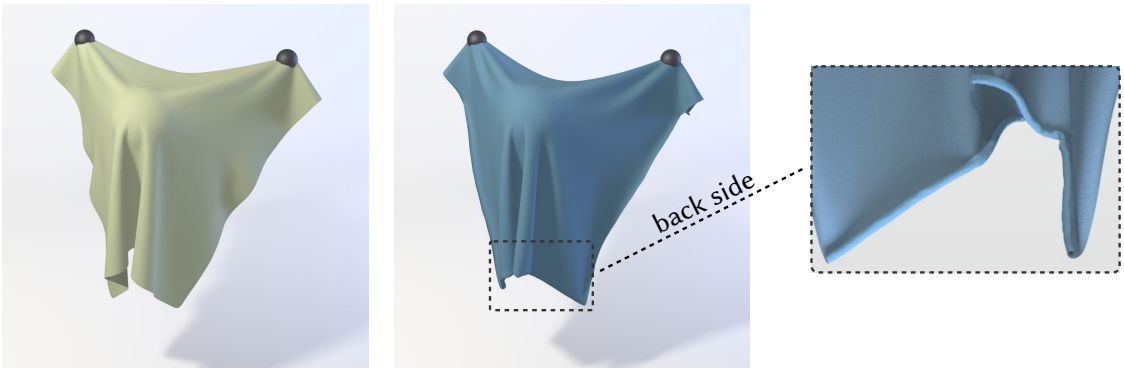


Figure 4.11: Models homogenized from higher density variants of the satin (left) and stockinette (right) patterns at a 10% scale naturally produce more folds when draped. Notably, the small stockinette shows small curls on the inside, similar to cut t-shirts. Besides the scale of the folds, the larger stitch density does not affect the performance of our method.

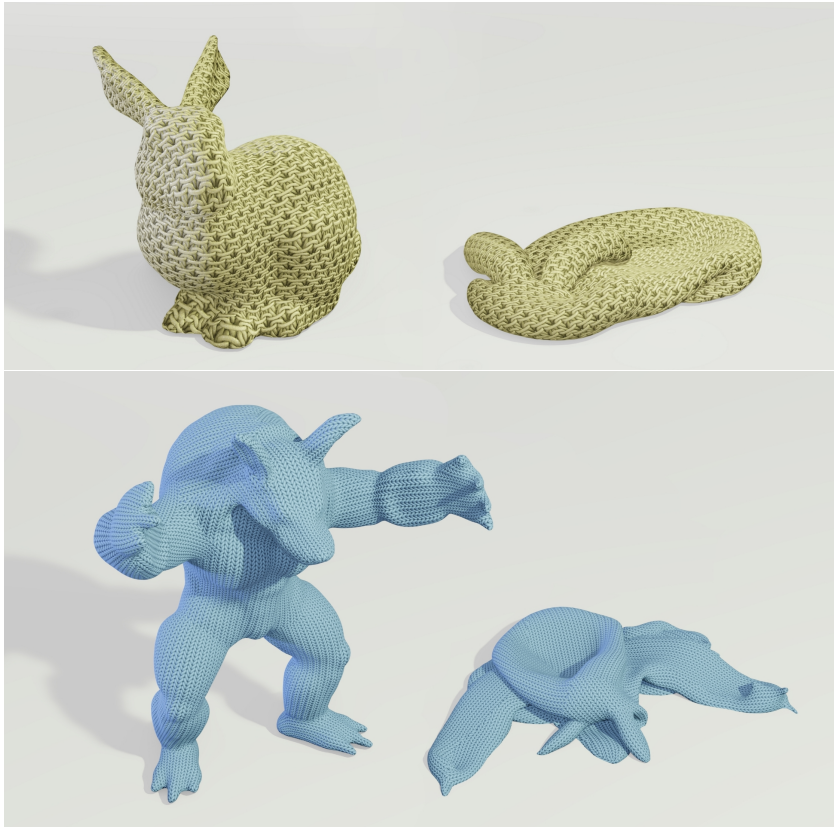


Figure 4.12: Before and after of a bunny and a *yarnmadillo* simulated with our models.

4.4 Discussion

In this chapter, we have proposed a method for computing homogenized models capable of simulating yarn-level effects in a thin-shell cloth solver. Through homogenization of a nonlinear shell, we are able to compute homogenized responses of periodic yarn patterns to macroscale deformations. We can then fit a regularized continuum model without the need for expensive measurement equipment. We compare our results with brute force simulations for multiple patterns on a series of stretching and draping tests. Our method is able to capture the

Table 4.3: Simulation timings for large-scale simulations with timestep Δt in seconds, average seconds per step, and average seconds per frame for a reference framerate of 30fps.

Simulation		Δt	sec/step	sec/frame
satin small drape	Fig. 4.11	3.34e-04	1.10	109.72
stock. small drape	Fig. 4.11	3.34e-04	2.01	200.32
sweater basket	Fig. 4.10	1.67e-03	7.40	147.62
sweater honey	Fig. 4.10	1.67e-03	7.43	148.36
sweater rib	Fig. 4.10	8.35e-04	5.79	231.15
sweater satin	Fig. 4.10	1.67e-03	7.45	148.79
sweater stock.	Fig. 4.10	1.67e-03	7.44	148.50
shirt stock. small	Fig. 4.10	8.35e-04	3.74	149.19
shirt satin small	Fig. 4.10	1.67e-03	4.96	99.06
scarf	Fig. 4.1	8.35e-04	0.91	36.31
yarn bunny	Fig. 4.12	6.68e-04	1.23	61.20
yarnmadillo	Fig. 4.12	5.57e-04	11.16	668.24

rich properties of knitted fabric such as general stretching and bending anisotropy, including Poisson's ratio, while being an order of magnitude faster even on moderate scales.

Our model is able to abstract the yarn-level interactions into an elastic continuum; however, this implies that we do not model localized effects such as tearing or pulling on single yarns. To this end, combining our continuum model with localized yarn simulation is worth investigating.

While our model captures elastic rest shapes well, we ignore yarn-level friction and hysteresis in our homogenization procedure. Although our method can be combined with other macroscale damping and plasticity models, we would like to explore homogenizing viscous and plastic effects from yarn-level simulations as well. We have left cloth-cloth and cloth-obstacle frictional contact entirely to the continuum solver; the more recent Argus simulator [Li et al. 2018] could be used in place of ArcSim for improved accuracy there.

Homogenization theory assumes a small RVE compared to the macroscale deformation. Although our co-rotated boundary conditions significantly loosen this limitation by allowing large highly-deformed configurations, the theory still imposes practical limitations on pattern size and thickness. For example, extreme curvatures at the macroscale may cause excessive self-intersections at the microscale. Similarly, approximating voluminous yarn patterns with a triangle-based cloth solver may make the garment look unrealistically thin. In the next chapter (Chapter 5), we address this issue by adding mechanics-aware yarn geometry that co-deforms alongside an underyling cloth simulation. This pairs especially well with the continuum models developed here since both elastic material and yarn-detail can be computed from the same periodic yarn pattern.

Our fitting procedure based on regularized splines aims to strike a balance between generality and robustness. Although we present a number of heuristics to increase the quality of the fit for nonconvex data, we do not offer any provable performance guarantees, and the approach is tailored to our application domain. One way to improve the stability of our cloth simulations would be to integrate it in the incremental potential method of Li et al. [2021].

Our focus in this work was to find one approach that yields stable simulations and reproduces the essential qualitative features of yarn-level cloth. Due to various approximations in fitting,

we do not expect a perfect quantitative match. The results in this chapter use yarn material parameters chosen to produce realistic-looking animations, and our system does manage to capture the realistic non-linearities and anisotropies induced by the yarn-scale geometry. However, we will see in Chapter 6 that for our system to quantitatively match real physical measurements of fabrics, we have to address additional complications such as the scale of physical tests and the parametrization of yarn models.

CHAPTER 5

Mechanics-Aware Deformation of Yarn Pattern Geometry

In the previous chapters we have established a method for fitting thin-shell continuum models that reproduce the large-scale elastic behavior of yarn-based fabrics. We have seen that we can produce satisfying animations of knitted and woven cloth. However, while these models are able to capture the large-scale mechanics, this abstraction loses the rich *visual* yarn-level detail. Simple texturing, as used in the results of the previous chapter, can make the fabric look unrealistically flat. We could instead embed the yarn geometry within the cloth triangle mesh and use barycentric coordinates to animate the yarn detail. While this does add some of the visual complexity back, this approach may miss certain critical *local* deformations. For example, knit loops tighten when the fabric is being stretched (Figure 5.1 right). Running full yarn-level simulations to reproduce such effects is prohibitively expensive at large scales. We would like the best of both worlds — the efficiency of the mesh-based solvers and the physical yarn detail.

In this chapter, we propose a method to interpolate yarn pattern deformations at run time to enrich a cloth simulation with detailed yarn geometry. In the framework of homogenizing elastic properties of periodic yarn patterns, we also automatically capture the deformation-dependent displacements of the local yarn geometry. At runtime, we use the triangle strains of the deforming cloth mesh to interpolate precomputed yarn displacements. This interpolated geometry then rearranges in accordance with yarn-level mechanics and reproduces salient effects such as knit loops tightening under tension. Our method thus effectively approximates full yarn-level animation at negligible cost over an underlying mesh-based simulation. Figure 5.2 highlights some examples achieved with our method.

We also introduce an efficient way of linearizing the bending response of yarn patterns in terms of stretching, and we propose a way to clamp compression, allowing the user to tune the extent of yarn buckling. We implement the yarn deformation procedure as compute shaders on the GPU.

Overview We start by discussing the two main stages of our method: data generation and real-time displacement. Section 5.1 explains the data generation step shown in Figure 5.3, where we optimize for the rest configuration of yarn patterns under a range of large-scale deformations, and compute local displacements relative to these deformations. Section 5.2 details

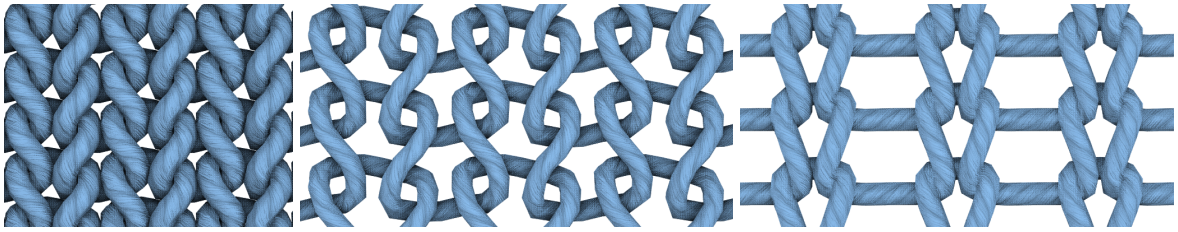


Figure 5.1: Deforming a yarn pattern (left) with embedded deformation causes it to stretch uniformly (center), which ignores yarn-level physics that should make the yarn loops tighten (right).

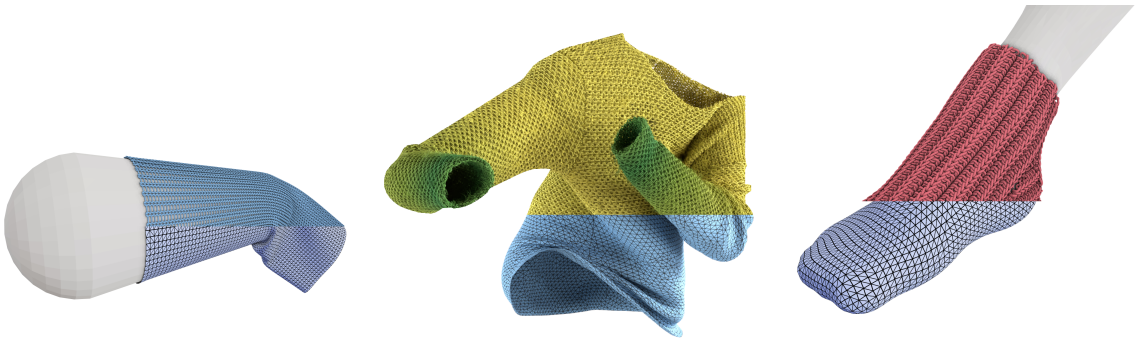


Figure 5.2: Our method uses an underlying cloth mesh (bottom) to animate yarn-level cloth (top) in real-time by approximating the yarn-level response in a data-driven fashion based on the deformation of the mesh. It reproduces the stretching behavior of knits (left), animates large garments with millions of yarn vertices (middle), and combines with real-time cloth simulation for end-to-end interactive animation of yarn-level cloth (right). We render the sweater (middle) using pathtracing and with hair particles.

the real-time displacement phase illustrated in Figure 5.4, where we map the precomputed yarn geometry onto a deforming triangle mesh. Section 5.3 explores our method's results. Section 5.4 discusses limitations and advantages, and concludes with a summary and future work.

5.1 Data Generation

We want our embedded yarn details to deform realistically, so we prescribe their motion based on yarn-level simulations. This section describes the physics precomputation phase of our algorithm, illustrated in Figure 5.3.

5.1.1 Deformation Optimization

We use the method presented in Chapter 3 to simulate how each yarn pattern deforms. This method takes as input a particular yarn pattern in an undeformed configuration (e.g. the "stockinette" pattern in Figure 5.3), and a large-scale surface deformation encoded as the first $\mathbf{I}$ and second $\mathbf{II}$ fundamental forms. [1] The method uses this large-scale deformation to define boundary conditions, and then optimizes for the elastostatic equilibrium configuration of the yarn pattern. Chapter 4 then reduced this geometric information down to a single scalar

[1] In this chapter, omit explicit bar notation (such as $\overline{\mathbf{II}}$) for mesh-level quantities and instead reserve it for the uniform part of the deformation mapping $\overline{\boldsymbol{x}}$ as compared to the local fluctuations $\tilde{\boldsymbol{u}}$.

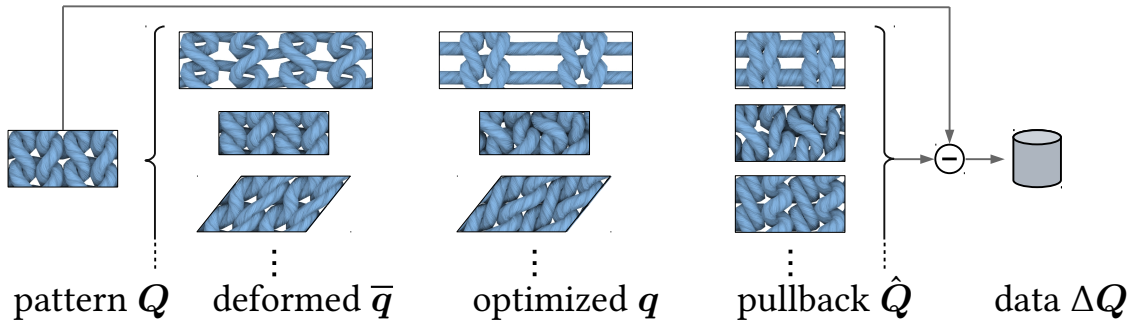


Figure 5.3: From left to right: We start with a periodic yarn pattern $\boldsymbol{Q}$, and we apply a range of large-scale surface deformations to get the deformed state $\overline{\boldsymbol{q}}$. Using the optimization method of Chapter 3, we optimize for the elastostatic rest shape $\boldsymbol{q}$ subject to the respective deformation. We then pull the resulting geometry back into the undeformed material space to get $\hat{\boldsymbol{Q}}$. Finally, we subtract the initial state to compute displacements $\Delta\boldsymbol{Q}$, building a mapping from large-scale deformation to the associated local yarn-level deformation.

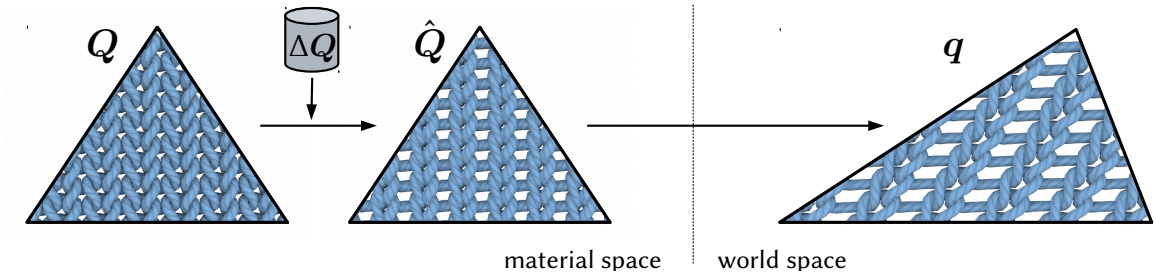


Figure 5.4: Our algorithm for animating yarn geometry in real-time: We start with undeformed yarn-level geometry $\boldsymbol{Q}$ tiled over a triangle in material space (left), and we apply the appropriate material displacement $\Delta\boldsymbol{Q}$ from the database to get deformed yarn geometry $\hat{\boldsymbol{Q}}$ (middle). This geometry is then mapped along with the triangle to get the current world-space deformation $\boldsymbol{q}$.

energy density used in a hyperelastic material model. In contrast, here we will use the yarn geometry directly.

We model yarns as discrete elastic rods [Bergou et al. 2010], represented as connected lists of vertices with positions $\boldsymbol{x}$ along the centerline. Each edge also stores a twist angle θ and a reference director $\underline{\boldsymbol{d}}_1$.[2] We concatenate positions and twists for each vertex into a four-dimensional vector $\boldsymbol{q} = \left(\boldsymbol{x}^\top, \quad \theta\right)^\top$, where θ corresponds to one incident edge. (Note that the reference directors $\underline{\boldsymbol{d}}_1$ are not degrees of freedom.)

We recall from Chapter 3 that during the optimization the kinematics of yarn vertex positions are defined as

$$\boldsymbol{x}(\boldsymbol{X}) = \overline{\boldsymbol{x}}(\boldsymbol{X}) + \tilde{\boldsymbol{u}}(\boldsymbol{X}), \tag{5.1}$$

$$\overline{\boldsymbol{x}}(\boldsymbol{X}) = \boldsymbol{\varphi}(X_1, X_2) + X_3\, \boldsymbol{n}(X_1, X_2). \tag{5.2}$$

Here, $\boldsymbol{X} = (X_1, X_2, X_3)^\top$ are the *material-space* coordinates of the undeformed yarn pattern, with (X_1, X_2) the orthogonal and periodic directions along the pattern and X_3 the height coordinate. $\overline{\boldsymbol{x}}$ denotes the *large-scale* deformation constructed from the input fundamental

[2]The subscript in $\underline{\boldsymbol{d}}_1$ refers to its definition as one of two reference directors: one for the edge normal, and one for the edge binormal.

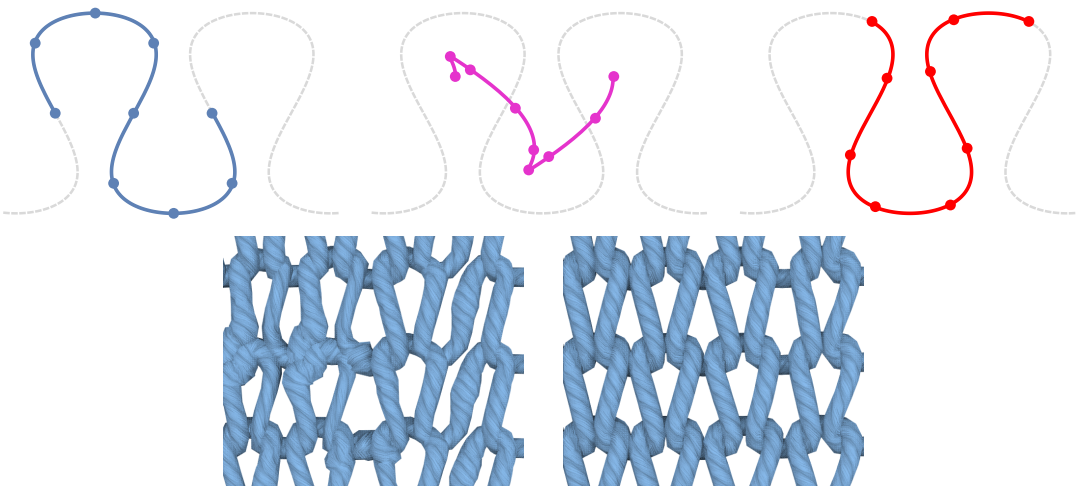


Figure 5.5: Top: Interpolating two periodic curves (left and right) that are identical up to a shift in parametrization can result in arbitrary interpolated shapes which look very different from the original data (center).
Bottom: Interpolating yarn geometry without accounting for sliding creates unrealistic artifacts like self-collisions and floating loops (left); our sliding constraint eliminates these artifacts (right).

forms **I** and **II**, which encode in-plane and bending deformations respectively. From these fundamental forms, we construct the mid-surface $\boldsymbol{\varphi}$ with normal $\boldsymbol{n}$, as described in Chapter 3, such that $\mathbf{I} = \nabla\boldsymbol{\varphi}^\top\nabla\boldsymbol{\varphi}$ and $\mathbf{II} = -\nabla\boldsymbol{\varphi}^\top\nabla\boldsymbol{n}$. We then solve for the yarn configuration which minimizes elastic energy. The optimization variables are the fluctuations $\tilde{\boldsymbol{u}}$, which describe local displacements relative to the large-scale deformation, and the twists θ. Using Θ for undeformed twists, the concatenated coordinates are undeformed $\boldsymbol{Q} = (\boldsymbol{X}^\top, \Theta)^\top$, large-scale-deformed $\overline{\boldsymbol{q}} = (\overline{\boldsymbol{x}}^\top, \Theta)^\top$, and optimized $\boldsymbol{q} = (\boldsymbol{x}^\top, \theta)^\top$. Here, we assume that twists Θ are unaffected by the large-scale mapping, which is true for pure in-plane deformations. In general, bending may induce local twists that depend on **II** and the orientation of yarns relative to the curvature direction. In our experiments, we assume that this effect is small.

Crucially for our application, we found it important to restrict parametric sliding, i.e. periodic sliding of the yarn that does not affect the conceptual *periodic* geometry, but its representative simulated geometry. While this sliding does not affect the energy of the simulated yarn pattern, *interpolating* between two shifted parameterizations may produce completely different shapes, as shown in Figure 5.5. In our experience, this nullspace creates distracting interpolation artifacts even for nearly-identical deformations. The problem remains as we sample the deformations more densely, manifesting as sharp discontinuities in deformation space. We recall from Section 3.3.3 that we can easily remove the parametric yarn sliding by effectively constraining one vertex per periodic yarn to remain on the boundary of the pattern:

$$\tilde{\boldsymbol{u}} \cdot (\nabla\boldsymbol{\varphi}\,\boldsymbol{N}) = 0, \tag{5.3}$$

where $\boldsymbol{N}$ is the undeformed normal to the respective pattern boundary, either $\boldsymbol{N} = (1, 0)^\top$ or $\boldsymbol{N} = (0, 1)^\top$. This sparse set of vertex constraints efficiently and effectively removes the aforementioned interpolation artifacts. This strategy allows us to find a physically realistic yarn shape for a given large-scale deformation described by **I** and **II**.

5.1.2 Sampling

Now that we can produce yarn geometry for a given deformation, we want to precompute a series of representative samples and interpolate them at runtime. **I** and **II** are each symmetric 2×2 tensors, so parameterizing deformation by **I** and **II** directly yields a 6-dimensional function which is expensive to precompute, store, and read at runtime. We aim to reduce this dimensionality by parameterizing the deformation with as few variables as possible.

Identically to Chapter 4, we reparameterize **I** as a 3-dimensional function using the in-plane strains

$$s_x = \sqrt{\mathrm{I}_{11}} - 1, \qquad s_a = \frac{\mathrm{I}_{12}}{\sqrt{\mathrm{I}_{11}\mathrm{I}_{22}}}, \qquad s_y = \sqrt{\mathrm{I}_{22}} - 1. \tag{5.4}$$

II can be parameterized similarly by introducing additional curvature variables and increasing the dimensionality of the data set. However, we show in Section 5.2 that bending deformation can be reasonably approximated in terms of stretching variables alone. This strategy lets us sample the entire large-scale deformation space with only three variables s_x, s_a, and s_y, significantly reducing memory and computation overhead. We sample these deformations on a regular 3D grid, and we discuss the performance, data size, and quality for different numbers of samples in Section 5.3.

5.1.3 Material-Space Displacements

At runtime, our interpolated yarn geometry will be mapped onto a deformed triangle mesh, where it will naturally inherit the deformation of its triangle (the mapping from material-space to world-space in Figure 5.4). Thus, we want to store all of the deformation *except* the large-scale deformation in our precomputation as material-space displacements (the "pullback" column in Figure 5.3).

To do this, we need to find the modified material space coordinates $\hat{\boldsymbol{X}}$ which give us our desired world-space deformations $\boldsymbol{x}$ when deformed by only the large-scale deformation $\overline{\boldsymbol{x}}(\hat{\boldsymbol{X}})$: in other words, find $\hat{\boldsymbol{X}}$ s.t. $\boldsymbol{x} = \overline{\boldsymbol{x}}(\hat{\boldsymbol{X}})$. We perform this solve using Newton's method and provide more details in Appendix C.1.

Concatenating $\hat{\boldsymbol{Q}} = (\hat{\boldsymbol{X}}^\top, \theta)^\top$, we subtract the initial material state to get displacements

$$\Delta\boldsymbol{Q} = \hat{\boldsymbol{Q}} - \boldsymbol{Q}. \tag{5.5}$$

Note that at the rest pose, $\mathbf{I} = \mathrm{Id}$, $\mathbf{II} = 0$, and $\Delta\boldsymbol{Q} = 0$.

To summarize, we can now build a database of material-space yarn displacements $\Delta\boldsymbol{Q}$ for each vertex i of a pattern and for a range of in-plane deformations j: $\Delta\boldsymbol{Q}_i(s_{xj}, s_{aj}, s_{yj})$. This database can be interpreted as a grid of example deformations, or similarly as a 3D displacement texture per yarn vertex. Interpolating between these samples takes a planar deformation s_{xj}, s_{aj}, s_{yj} and maps it to a yarn-level displacement map. Note that this will recover the exact deformed yarn pattern for the strains sampled in the database and approximate patterns for intermediate strains.

5.2 Real-time Displacement

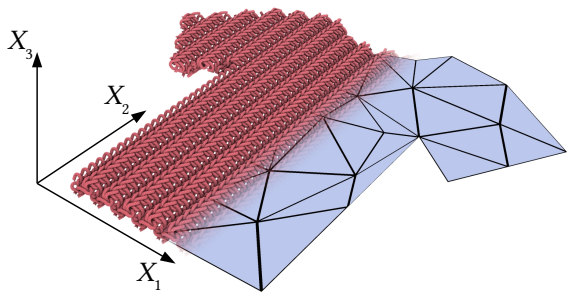
Now that we have a database of yarn pattern displacements for a range of deformations, we apply them to a yarn pattern tiled over an animating triangle mesh. In the following discussion,

Algorithm 1 Real-time yarn animation

Input: mesh animation, yarn pattern, displacement data $\Delta \boldsymbol{Q}$
Output: deformed yarn geometry $\boldsymbol{q}$

procedure Animate Yarns
  $\boldsymbol{Q} \leftarrow$ yarn pattern tiled over mesh
  Each Frame:
    compute **I**, **II** per face
    interpolate **I**, **II** to mesh vertices
    for each yarn vertex **do** ▷ on the GPU
      interpolate **I**, **II** from mesh
      compute linearized bending $\mathbf{I}(X_3)$ ▷ (5.7)
      clamp compression ▷ Section 5.2.3
      compute strains s_x, s_a, s_y ▷ (5.4)
      look up displacements $\Delta \boldsymbol{Q}$ ▷ Section 5.1.3
      displace: $\hat{\boldsymbol{Q}} \leftarrow \boldsymbol{Q} + \Delta \boldsymbol{Q}$ ▷ (5.8)
      map: $\boldsymbol{q} \leftarrow (\boldsymbol{x}(\hat{\boldsymbol{X}})^\top, \hat{\Theta})^\top$ ▷ Section 5.2.5
    end for
    tessellate and render $\boldsymbol{q}$ ▷ Appendix C.4
end procedure

we denote $\boldsymbol{Q}$ as the undeformed (material space) coordinates, $\hat{\boldsymbol{Q}}$ as the coordinates deformed by *local* displacements, and $\boldsymbol{q}$ as the final world-space coordinates, as illustrated in Figure 5.4. Algorithm 1 outlines our procedure.



As a precomputation, we first create the initial undeformed yarn mesh corresponding to the undeformed triangle mesh (inset). We generate a 2D background grid in the mesh's UV-coordinates with cells the size of the periodic pattern, and we copy the yarn geometry into every cell that overlaps an undeformed triangle. We then remove yarn vertices that do not lie within a triangle and delete yarn fragments shorter than a user-specified length for aesthetic purposes as described in the Appendix C.3. Finally, we precompute the material-space barycentric coordinates for each yarn vertex.

5.2.1 Mesh Strains

Each animation frame, we compute discrete fundamental forms for each triangle as in Chapter 4:

$$\mathbf{I} = \boldsymbol{F}^\top \boldsymbol{F}, \qquad \mathbf{II} = \boldsymbol{F}^\top \boldsymbol{\Lambda}\, \boldsymbol{F}, \tag{5.6}$$

where $\boldsymbol{F}$ is the triangle deformation gradient and $\boldsymbol{\Lambda}$ is the triangle-averaged shape operator from [Grinspun et al. 2006]. We then distribute them to triangle vertices using modified Shepard weights from Phong interpolation [James 2020], and finally interpolate them to each yarn vertex.

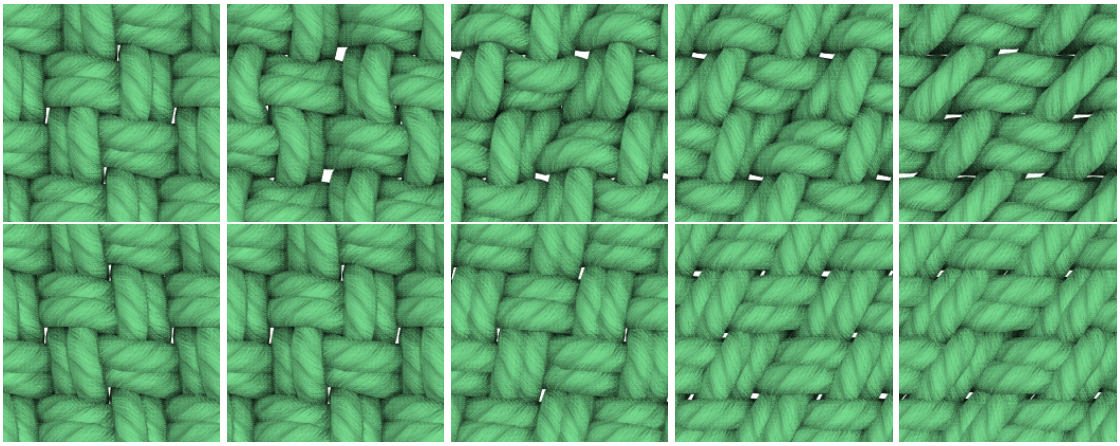


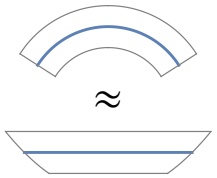
Figure 5.6: Top: From left to right, a yarn pattern buckles into different configurations under increasing shearing deformation. Bottom: Adding a lower bound $\lambda_{\min}$=0.7 to the eigenvalues of the in-plane deformation $\mathbf{I}$ allows a user to tune the extent of buckling during animation.

5.2.2 Linearized Bending

As alluded to in Section 5.1.2, we can approximate the effect of bending behavior by adding stretching and compression depending on the surface curvature. We show in Appendix C.2.1 that we can enhance the first fundamental form with one that varies along the surface normal:

$$\mathbf{I}(X_3) \approx \mathbf{I} - 2\,X_3\,\mathbf{II}, \tag{5.7}$$

We illustrate this idea in the inset figure to the right, which approximates the extruded volume around a curved blue midsurface (top) with a linearized volume that is stretched above and compressed below the midsurface (bottom). We compare this idea to other bending models in Section 5.3.



5.2.3 Compression Clamping

Like most elastic materials, cloth *buckles* out of plane when compressed. The chaotic nature of this buckling can make our yarn optimization reach multiple visually distinct configurations that do not vary smoothly over deformation space (Figure 5.6, top). For the continuum materials in Chapter 4, we dealt with this issue by regularizing the fit of the output energies. Here, we similarly regularize compression by clamping the eigenvalues of $\mathbf{I}$ to a lower bound before looking up the yarn displacement, which reduces buckling in a user-tunable way.

Using the method of Deledalle et al. [2017], we set a minimum value $\lambda_{\min}$ for the eigenvalues of $\mathbf{I}(Z)$. (Unless stated otherwise, all of our experiments use $\lambda_{\min}$=0.8, where λ<1 is compression.) Because $\Delta\boldsymbol{Q}\to 0$ as $\mathbf{I}\to\mathrm{Id}$, our technique for clamping compression will only reduce local deformations, but it will still deform due to the triangle embedding. Figure 5.6 (bottom) shows how this clamping reduces buckling while preserving the large-scale deformation.

5.2.4 Local Displacement

After clamping $\mathbf{I}$, we convert it to strains s_x, s_a, s_z (Equation (5.4)), trilinearly interpolate the yarn displacement $\Delta\boldsymbol{Q}(s_x, s_a, s_z)$, and compute the deformed material space yarn coordinates

$$\hat{\boldsymbol{Q}} = \boldsymbol{Q} + \Delta\boldsymbol{Q}. \tag{5.8}$$

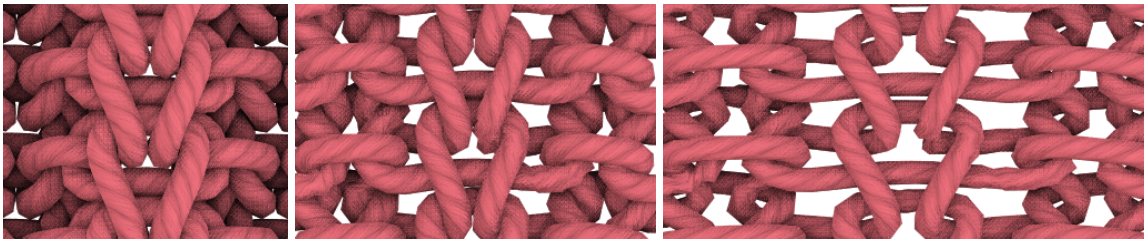


Figure 5.7: The yarns of a rib pattern adapt to increased stretching from an initial state (left) until the limits of the precomputed data are reached (middle). Deformation beyond the sample limits (right) do not result in further local displacements, but instead fall back to purely embedded deformation.

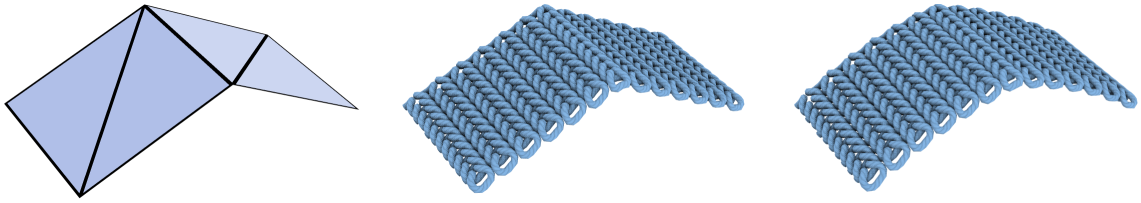


Figure 5.8: Barycentric embedding of yarn geometry may create sharp edges (center), whereas Phong deformation smooths out obvious mesh resolution artifacts (right).

We clamp strains outside of the sampled range to their nearest neighbor in the $\Delta\boldsymbol{Q}(s_x, s_a, s_z)$ data set. Similar to compression clamping, this constant extrapolation will limit the local deformations while still inheriting large-scale deformations from the mesh embedding (See Figure 5.7).

5.2.5 World-Space Mapping

To map the yarn vertices to world space $\boldsymbol{x}$, we use:

$$\boldsymbol{x}(\hat{\boldsymbol{X}}) = \boldsymbol{\varphi}(\hat{X}_1, \hat{X}_2) + \hat{X}_3\, \boldsymbol{n}(\hat{X}_1, \hat{X}_2), \tag{5.9}$$

which corresponds to extruding the world-space mesh surface $\boldsymbol{\varphi}$ along its normal $\boldsymbol{n}$. To avoid piecewise linear embedding artifacts, we employ Phong deformation [James 2020] and interpolated vertex normals to generate a smoother surface $\boldsymbol{\varphi}$ and shell-volume (Figure 5.8).

To map yarn twists, we simply copy the updated twist values θ=$\hat{\Theta}$, and we co-transform the edge normals $\underline{\boldsymbol{d}}_1$ using an approximate mapping of the Jacobian of (5.9), as detailed in Appendix C.4.1.

The computation of yarn vertex deformation and world-space mapping are trivially parallel, so we implement them as GPU compute shaders. The interpolation of $\Delta\boldsymbol{Q}$ results in a single 3D texture interpolation per yarn vertex.

5.2.6 Real-Time Rendering

Computer graphics researchers have developed a number of algorithms for rendering yarns and fibers [Wu and Yuksel 2017; Montazeri et al. 2019, 2020]. For the examples in this paper, we tessellate the deformed yarns as cylindrical meshes in a geometry shader. We approximate ply- and fiber-level detail with procedurally twistable normal maps and ambient occlusion maps, and we approximate volume conservation by locally rescaling yarn radii when they are stretched. We provide the full rendering details in the Appendix C.4.

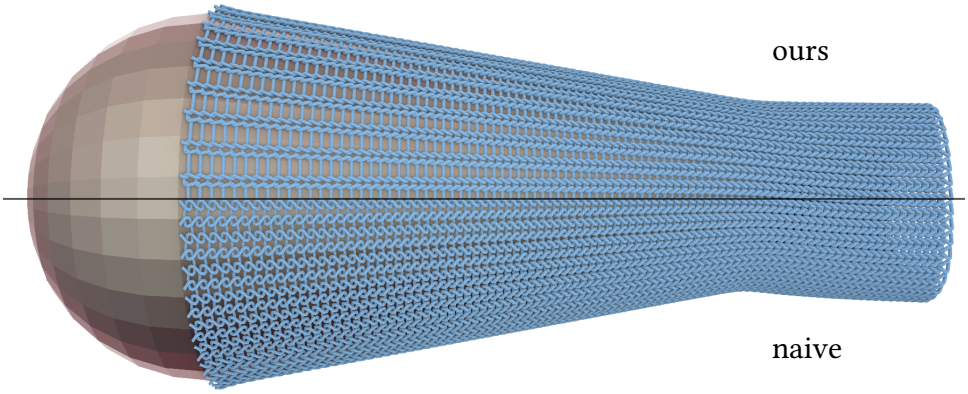


Figure 5.9: Comparison of our mechanics-aware yarn animation (top) against naive embedding (bottom). The yarn geometry differs substantially on the left, where the deformation is large.

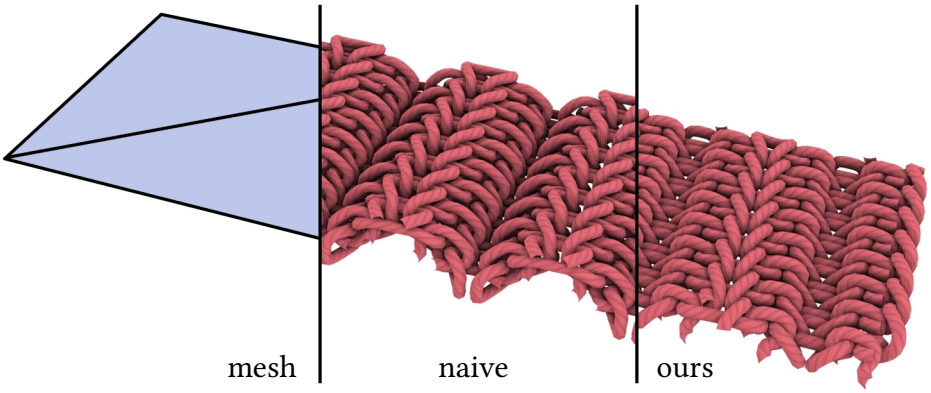


Figure 5.10: Compared to naive embedding, our method reproduces the ribs of knitted cloth flattening under tension.

5.3 Results

We will now discuss examples generated by our mechanics-aware yarn animation technique. Figure 5.9 shows how our algorithm compares to a more typical embedded approach to animating yarn-level geometry detail. Our method naturally reacts to the deformation of the underlying cloth by causing loops to rearrange and tighten up as the tension is increased. As a natural consequence, our approach also reproduces the tendency of knitted and mesh fabrics to become more transparent when stretched. Figure 5.10 highlights that thicker patterns similarly exhibit flattening under tension. Figure 5.11 shows how we can easily apply different yarn-patterns to any cloth mesh, producing visually distinct geometry which depends on both the deformations of the mesh and the precomputed yarn mechanics. Our method can add yarn-level details onto *any* deforming triangle mesh: examples in this paper use deforming cloth meshes from ArcSim [Narain et al. 2012, 2013], position-based dynamics [Müller et al. 2007], and Blender [2020].

We enhance the offline cloth simulation solver from Chapter 4, which reproduces the large-scale effects of knitted garments. Figure 5.12 compares this approximation to ground-truth yarn-level simulations, yielding visually plausible recreations of the local pattern deformation. We note that the accuracy of our method highly depends on the accuracy of the underlying triangle deformation, so we see higher discrepancy where the FEM cloth simulation deviates from yarn-level simulation.

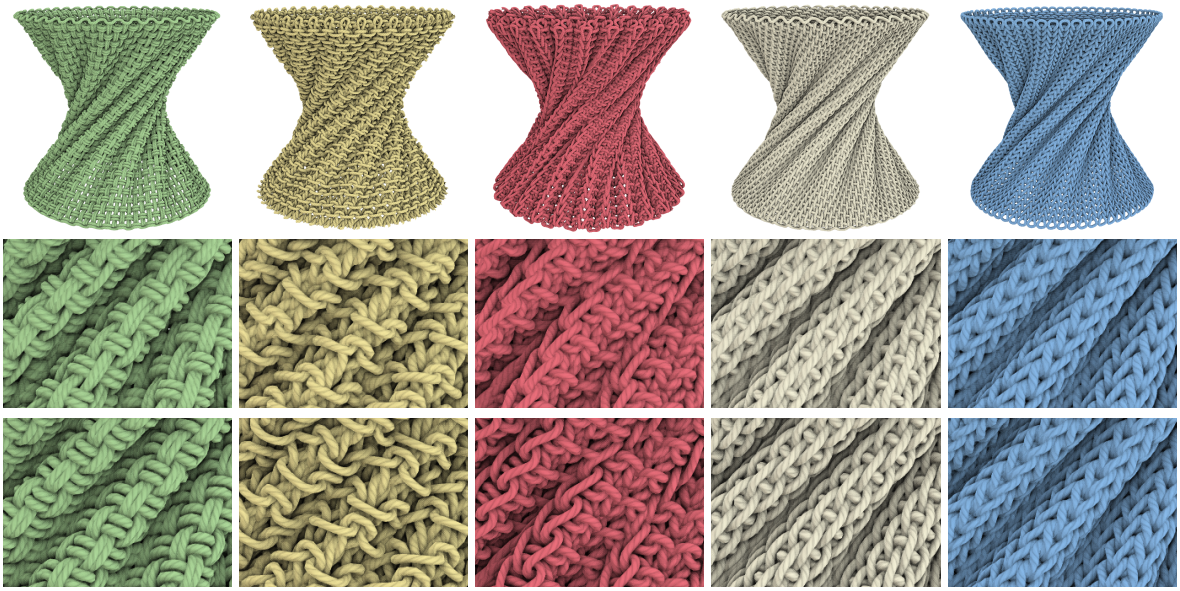


Figure 5.11: Different yarn patterns mapped onto a twisted cloth with our method react differently to the mesh deformation. The second row shows a zoom and compares it to naive embedding in the third row. Our method captures the subtle tightening in all cases, while the naive method results in unrealistic gaps between yarns.

We also used our approach to add yarn-level details to a position-based dynamics cloth solver [Bender et al. 2015], to approximate yarn-level cloth simulation in real-time. Figure 5.13 shows how a user can perform an interactive dressing operation by pulling a knitted sock onto a foot. Note that our method still produces plausible yarn-level deformations, despite the fact that the default position-based elastic material model is quite far from a model derived from yarn physics.

Figure 5.14 shows we can render our yarn geometry offline to produce higher quality path-traced scenes with "fuzz" from a procedural particle system. Figure 5.15 shows how our GPU-based yarn displacement algorithm allows for the animation of garments with millions of individual yarn vertices at interactive rates.

We animated most of the figures and concepts in this paper. For these and additional highly detailed results, please refer to the submission and supplementary videos included with the paper [Sperl et al. 2021], which is openly available at `https://doi.org/10.1145/3450626.3459816`. For reproducibility, we release the source code and data used to generate our results (`https://git.ist.ac.at/gsperl/MADYPG`).

Performance We ran our method on a desktop computer with an Intel Core i7-7820X processor and an NVIDIA GeForce GTX 1080 Ti graphics card and gathered performance statistics. Table 5.1 breaks down the computational cost of our algorithm for each of the various examples in this paper. Almost all examples in this paper perform both geometry generation and rendering in real-time (well over 60 fps). To stress-test our method, we created the knitted sweater in Figure 5.15 (right), which has more than 40 million yarn vertices and generates yarn geometry at over 14 fps. The total time including rendering (depending on the complexity of the renderer and the amount of geometry per pixel due to camera zoom) is around 4 fps for this example.

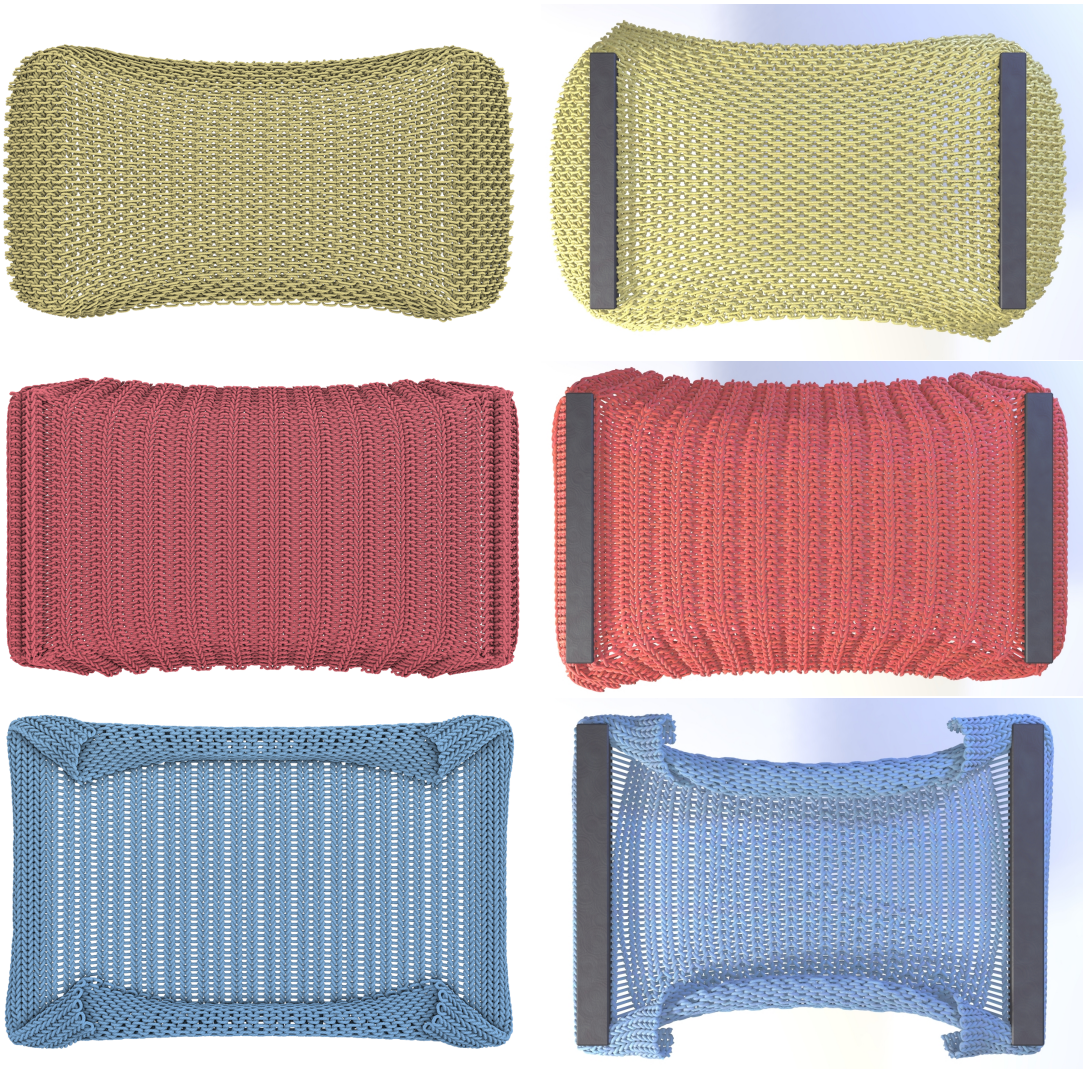


Figure 5.12: Comparison of yarns animated with our method applied to a precomputed cloth simulation (left) against full yarn-level simulation (right). Our yarn level deformations differ the most where the triangle-level cloth simulation is least accurate.

Data Set Size The number of samples in our precomputed yarn database only modestly affects the visual quality of the animation. Figure 5.16 shows a deformed rib pattern with a varying number of database samples and a visualization of the associated interpolation error. In practice, we found that a small number of samples was sufficient to capture strong effects like loops tightening under tension, and we noticed surprisingly few obvious interpolation errors (like interpolated threads leading to self-collisions).

On the other hand, a large number of database samples led to unwanted noise in the animations, like sudden 'pops' from one configuration to another, instead of gradual ones. We believe our observations are consistent with those presented in Chapter 4, where we noted that a dense sampling rate (especially in the compressive regime where fabric buckles chaotically) required explicit filtering to remove high-frequency noise. Here, simply removing samples from the database acts as an effective low-pass filter, and our compression clamping technique strongly reduces popping.

The complexity of our yarn database has only a small affect on runtime. Table 5.2 relates the number of precomputed yarn-level simulations in our database to the memory and runtime

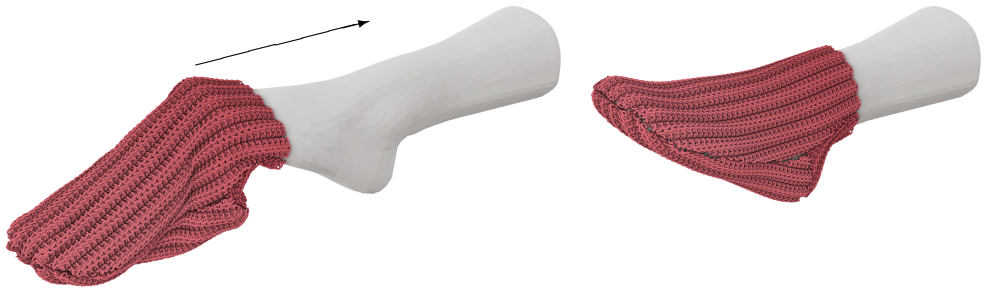


Figure 5.13: We use position-based dynamics to simulate a sock being pulled over a foot in real-time, where the user can interactively control the force.

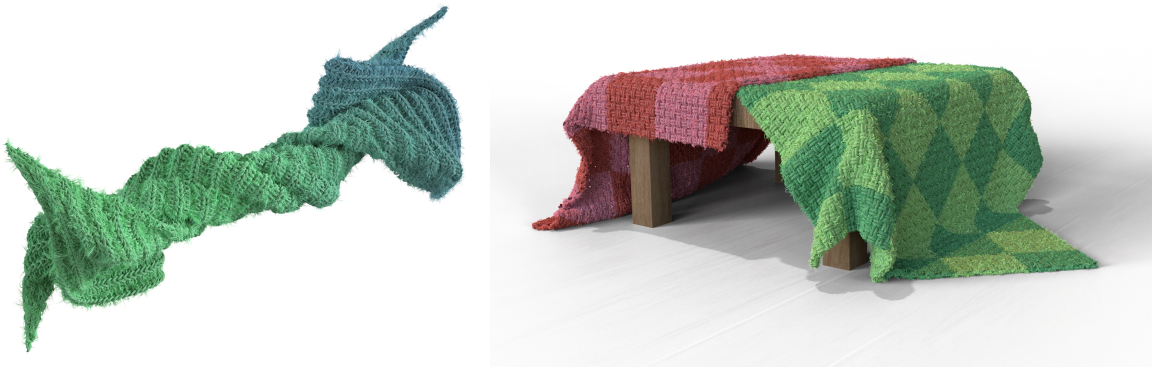


Figure 5.14: Two examples of our yarn geometry that were rendered offline using path tracing and hair particles.

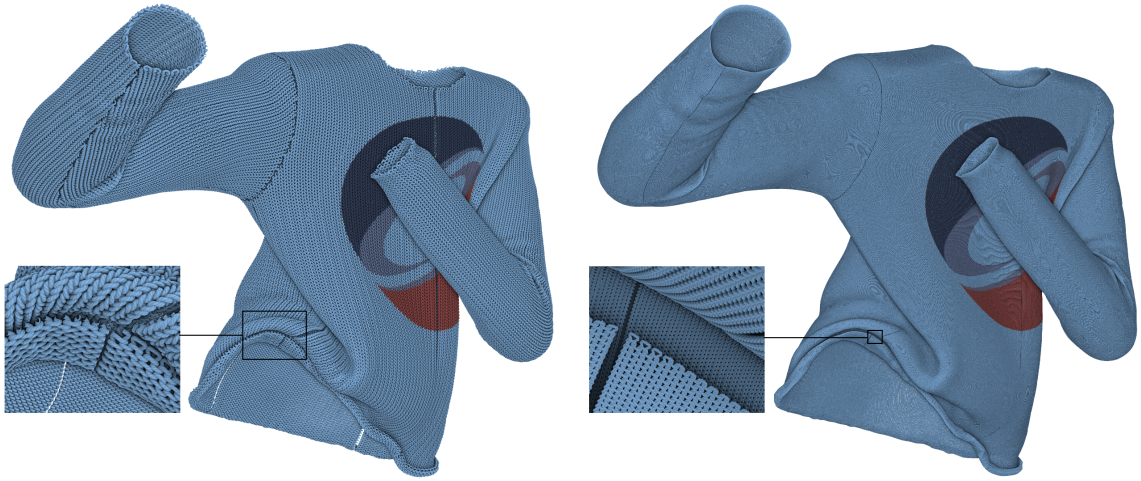


Figure 5.15: Our method scales favorably with increasing yarn density. We deform yarn geometry for a sweater with 0.8 million vertices at well over 100 FPS (left), and a sweater with 42.7 million vertices at over 14 FPS (right).

of our method. Notably, a 238× increase in database size caused a proportional increase in memory but only a 2.8× increase in runtime. Unless stated otherwise, the results shown in this paper use data sets of 9^3 samples sampled uniformly over the ranges $s_x, s_y \in [-0.2, 1.0]$ and $s_a \in [-0.7, 0.7]$.

Bending Models Finally, we compare the effect of our linearized bending approximation in Figure 5.17. We compare our linearized model described in Section 5.2.2 to a bending model

Table 5.1: Performance breakdown of results in this paper. From left to right, we list: the respective figure, the number of animated yarn vertices and average per-frame times of mesh strain computation (Section 5.2.1), data look-up and local yarn displacements (Sections 5.2.2 to 5.2.4), and embedded world-space mapping (Section 5.2.5). The mesh stage is implemented on the CPU, whereas yarn displacement and mapping are implemented on the GPU. In comparison, the CPU-implemented position-based dynamics step for the sock example averaged to 19.24 ms.

animation		# vertices	strains^{CPU}	displacement^{GPU}	mapping^{GPU}
sleeve	Fig. 5.9	94k	1.02 ms	0.13 ms	0.10 ms
patterns (averaged)	Fig. 5.11	82k	0.82 ms	0.09 ms	0.07 ms
honeycomb stretch	Fig. 5.12 top	74k	0.61 ms	0.08 ms	0.07 ms
rib stretch	Fig. 5.12 center	154k	0.77 ms	0.16 ms	0.12 ms
stockinette stretch	Fig. 5.12 bottom	119k	0.59 ms	0.12 ms	0.11 ms
stockinette sweater	Fig. 5.15 left	862k	2.59 ms	0.81 ms	0.66 ms
fine stockinette sweater	Fig. 5.15 right	42.7M	2.59 ms	35.34 ms	27.68 ms
rib twist	Fig. 5.14 left	433k	0.38 ms	0.51 ms	0.38 ms
table cloth	Fig. 5.14 right	130k	1.50 ms	0.16 ms	0.13 ms
sock	Fig. 5.13	139k	7.5 ms	0.14 ms	0.11 ms

Table 5.2: Comparison of rib pattern displacement data for different data set sizes. "generation" refers to the time needed to precompute the data, and "displacement" refers to the per-frame time of computing $\hat{\boldsymbol{Q}} = \boldsymbol{Q} + \Delta\boldsymbol{Q}(s_x, s_a, s_z)$ on a representative sweater animation with 1.8 million yarn vertices.

# samples	memory	generation	displacement
$5^3 = 125$	0.7 MB	2.2 min	1.58 ms
$9^3 = 729$	4.2 MB	12.7 min	1.82 ms
$15^3 = 3375$	19.2 MB	59.8 min	2.30 ms
$31^3 = 29791$	169.7 MB	533.6 min	4.35 ms

that explicitly captures combined stretching and bending. (See Appendix C.2 for details on these comparison models.) First of all, we found that the importance of bending depends on the knit pattern: "thick" patterns like the rib (Figure 5.11 red, center) exhibit more differences than nearly planar ones like the stockinette (Figure 5.11 blue, right). The further away the geometry is from the surface $\boldsymbol{\varphi}$, the more it is locally stretched or compressed by bending. Second, even for a thick pattern, the differences between the two models are minor and localized to regions of strong curvature. Most importantly, the linearized model is much more efficient — when comparing CPU-based implementation, our linearized model ran roughly 8× faster.

An extra benefit of the linearized bending model is that it transforms bending-induced buckling into compression, which can be filtered with our compression clamping algorithm (Section 5.2.3). Otherwise, it would be non-trivial to filtering both compression- and bending-related buckling in a compatible and coherent manner.

5.4 Discussion

Because our method is based on geometric interpolation of yarn vertices rather than an exact simulation of yarn-level geometry, it cannot exactly reproduce all yarn-level effects. Our

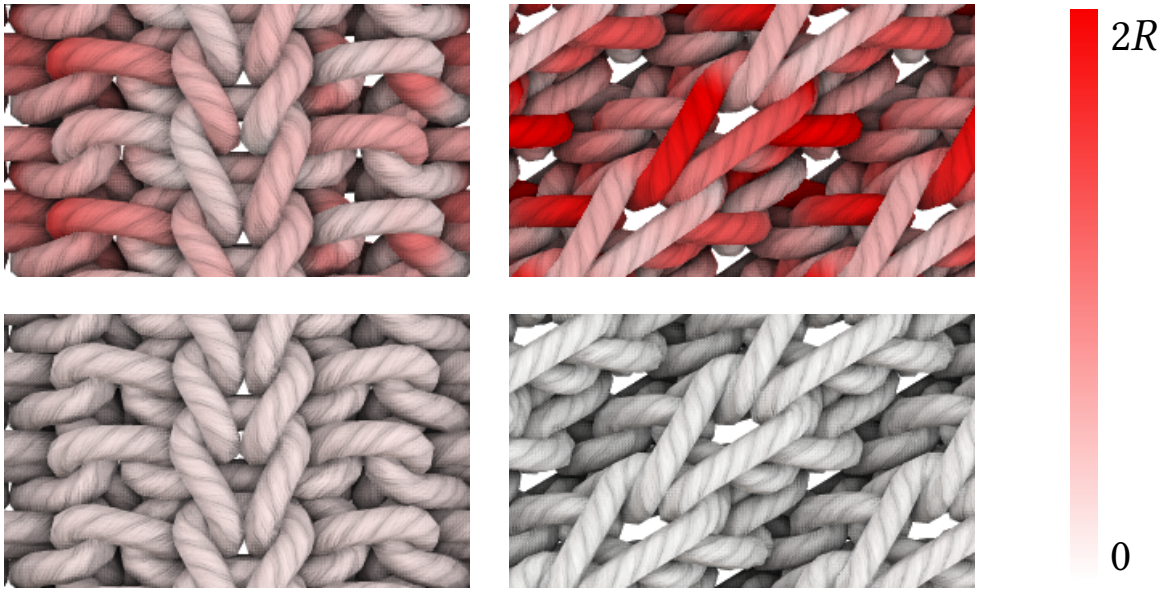


Figure 5.16: Comparison of different data sampling densities. The top and bottom rows show results generated with $5 \times 5 \times 5$ and $31 \times 31 \times 31$ samples respectively. In the top images, we color-code the L^2 error in vertex positions between the coarse and fine model, compared to the yarn radius R; red color corresponds to a larger difference. The coarse model produces plausible geometry even in the presence of larger errors.

method fundamentally assumes that the deformations sampled in the database are representative of the deformations in the online simulation. Because our per-vertex mesh strains only communicate deformations on the scale of a single triangle, the method will not be able to accurately react to fine-scale collision events like pulling on an individual thread. Similarly, our method relies on the mesh simulation to handle collisions; it will not exactly resolve object collisions on the level of the individual thread, and it will not exactly resolve collisions for very thick fabrics if the mesh is modeled as infinitely thin.

Our database currently also ignores time-dependent effects like hysteresis and damping, so repeating a mesh deformation will yield exactly the same fine-scale yarn arrangements. Our method interpolates the precomputed behaviors of a periodic yarn pattern, so we cannot yet simulate clothes consisting of completely aperiodic or disorganized threads, and it will be significantly more expensive to simulate ornate patterns with a large number of yarn vertices. We also do not yet handle non-periodic connections between different patches of cloth, so our method cannot yet sew together seams on the individual thread level. However, we *can* approximate larger seams by folding a piece of thin fabric over itself (Figure 5.18) similar to the "True Seams" technique recently proposed by Cirio and Rodríguez [2022].

On the other hand, our geometric approximation affords us a number of advantages and leads to a few novel challenges. The data-driven approach completely avoids the expense of online collision handling, and its exploitation of periodic structures avoids redundant computations resulting from structurally similar yarn patterns. The method cuts the complexity of brute-force yarn level cloth by several orders of magnitude, and as far as we know, our method is the only way to simulate millions of yarn vertices at interactive rates. The interpolative nature of the approach also guarantees unconditional numerical stability, so the method cannot blow up even in unrealistic video game environments. The speed and detail of our approach for animating knitted garments also creates new research challenges: zooming out from a pattern made of millions of threads can cause aliasing patterns when many of them occupy a single

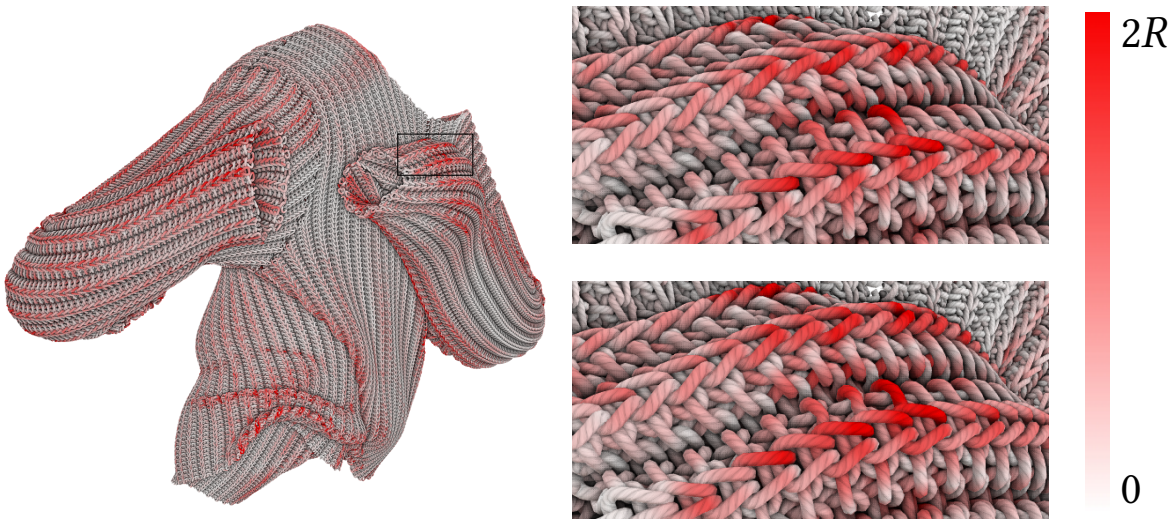


Figure 5.17: We compare our linearized bending model against a model incorporating bending strains explicitly on an animated knit sweater. We measure the difference between the models as the L^2 norm of the difference in yarn vertex positions, and we color code them relative to the yarn radius R; red color corresponds to a larger difference. The cut-outs show how knit loops appear to tighten more with the linearized model (top) than the explicit bending model (bottom).

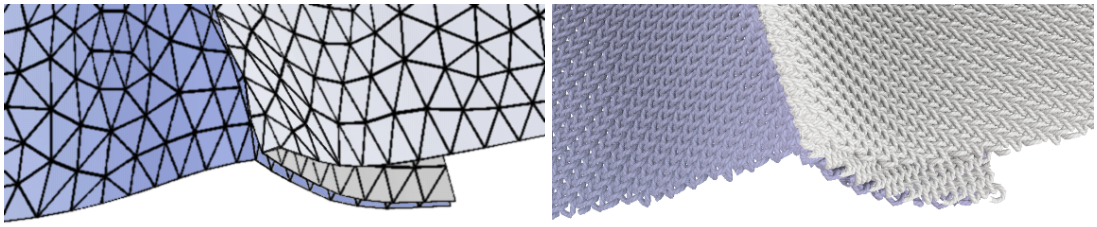


Figure 5.18: Approximating a seam by folding one piece of cloth over another on the mesh-level, rather than the thread level.

pixel, potentially requiring novel geometry anti-aliasing techniques in the future.

Conclusion We have presented a method for deforming yarn patterns in a mechanics-aware manner. It reproduces characteristic yarn-level cloth behaviors like knitted loops that tighten when the fabric is stretched. We introduced practical heuristics such as linearizing bending and limiting buckling, which make the method significantly more efficient and tunable by artists. The method is lightweight and GPU-parallelizable, so it is capable of animating millions of yarn vertices at real-time rates.

In the future, we are interested in new research challenges introduced by the massive scale of these yarn simulations. The method could further benefit from level-of detail approaches simplifying yarn geometry where it is not visible. Our technique might also be useful for research into deformation-dependent microfacet rendering, anti-aliasing techniques, or even neural rendering, for smoothly replacing extremely dense yarn geometry with an analytic or data-driven shading model.

CHAPTER 6

Estimation of Yarn-Level Simulation Models for Production Fabrics

Thanks to yarn-level models, we can produce animation results with extreme detail, as well as mechanical behavior that exhibits structural nonlinearity [Kaldor et al. 2008; Cirio et al. 2014]. Yarn-level models also enable visual design of complex knit patterns [Leaf et al. 2018]. Previous works have shown that computer-graphics yarn-level models can provide a qualitative match to the mechanical response of real-world fabrics; however, they have not tried to match this mechanical response in a *quantitative* manner.

This chapter introduces a methodology for inverse-modeling of yarn-level mechanics of cloth, based on the mechanical response of fabrics in the real world. We have documented, scanned, and tested a library of knit fabrics from real textile production. These fabrics span different types of complex knit patterns, yarn compositions, and fabric finishes, and they demonstrate

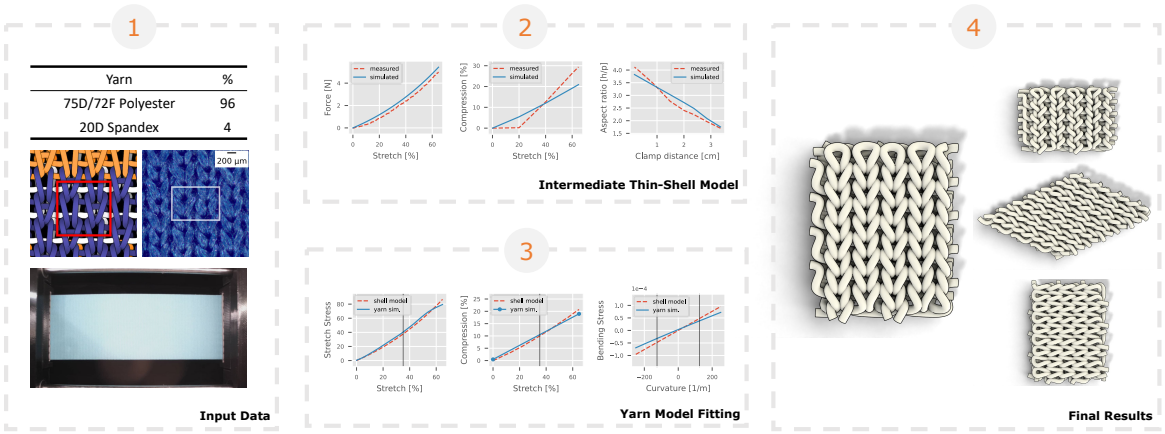


Figure 6.1: The figure shows our pipeline to fit yarn-level mechanical parameters to real-world knits, applied to a double knit pique fabric (DKP). (1) We take as input the fabric composition, knit schematics, high-resolution photographs, and swatch-level physical tests. (2) We fit a thin-shell model to the non-uniform physical data, and we use it to generate target uniform data. (3) Then, we fit the yarn-level model to the uniform data, leveraging periodic simulations to reduce the computational cost. (4) The images show the yarn model for DKP, under uniform stretch on the weft, bias, and warp directions.

diverse physical properties like stiffness, nonlinearity, and anisotropy. We then develop a system for fitting yarn-level simulation models that match their large-scale mechanical response.

In doing this, we have faced two major challenges. First, we demonstrate that real-world yarns exhibit complex deformation behavior that is not sufficiently captured by the Kirchhoff rod models typically used in computer animation and numerical simulation. Second, the best real-world data available is in the form of physical tests *at the swatch-level*, which is computationally prohibitive to simulate efficiently *at the level of individual yarns* — straightforward simulation-in-the-loop parameter estimations are intractable for this problem.

We introduce the first technique for the modeling and estimation of yarn-level fabric mechanics that succeeds to capture the macroscopic (swatch-level) response of textile-production knitted fabrics. We achieve this through three main contributions that address the challenges discussed above.

Data set of real-world fabrics We compiled a data set of physical test data from 33 different knitted fabrics used by industry professionals in the production of casual and sports garments. The fabrics span different knit patterns (e.g., multiple layers), yarn compositions (e.g., plated yarns), and yarn finishes. On a macroscopic level, they show diverse stiffness, nonlinearity, and anisotropy. The data set consists of physical information about each yarn type, manually registered yarn geometry, high-resolution photographic scans, and physical measurements from experiments for each fabric type. The data can be found at `http://mslab.es/projects/YarnLevelFabrics`.

Efficient fitting procedure To estimate yarn-level parameters from swatch-level physical tests, we have designed a two-step procedure that circumvents the computational cost of simulating full fabric swatches at yarn level: we first fit a thin-shell model to swatch-level data [Wang et al. 2011; Miguel et al. 2012], then we generate analytical stress-strain data for uniform deformations using the thin-shell model, and we finally use this data to fit yarn models using periodic simulations.

Practical and versatile simulation models Basic models for yarns and thin shells cannot capture the diversity of behaviors of real-world knitted fabrics, while complex models with a large number of parameters are vulnerable to overfitting. After experimenting with many of these models and observing how well they fit real-world data, we propose a few minimal extensions to typical models used in computer graphics to help strike the balance between simplicity and expressive power: an anisotropic area-preserving thin-shell model, and a yarn model with two-phase stretching and contact energies.

In this work, we focus on the major aspects of macroscopic mechanical response, including nonlinearity and anisotropy of stretch, shear, and bending deformation. We leave for future work more complex aspects such as extreme nonlinearity, hysteresis, or curling. Under these limitations, we maximize parallelism between the data, parameterization, and estimation processes of thin-shell and yarn-level fitting; we do this to minimize the error introduced by using the thin-shell model as an intermediate representation, while circumventing the challenge of simulating full-swatch non-uniform deformations at yarn level.

After discussing related work, we give an overview of our approach in Section 6.1. We then describe the input data to the yarn-model estimation process in Section 6.2. Section 6.3

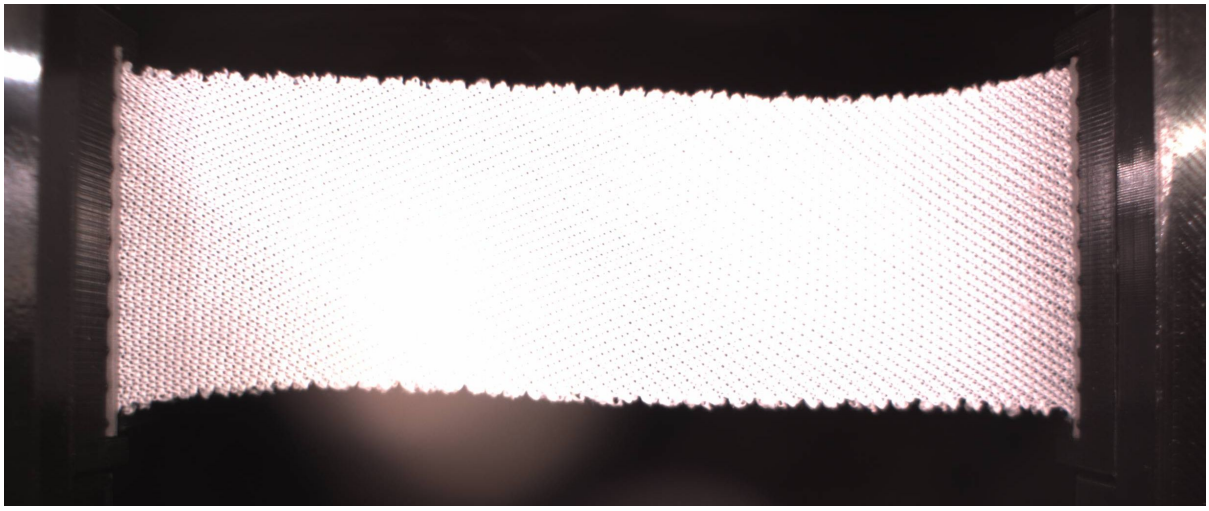


Figure 6.2: This image shows an all-needle fabric (A2) stretched along the bias direction. Notice the spatially non-uniform shear and curved shape near the clamps.

describes the thin-shell model, its estimation, and how it is used as an intermediate target for the yarn-level model. We continue with the description of the yarn-level model in Section 6.4 and the estimation of its parameters in Section 6.5. We discuss our results in Section 6.6, and we conclude with a discussion of limitations and future work in Section 6.7.

6.1 Overview

The main goal of our system is to efficiently solve for the yarn-level simulation parameters that will reproduce the large-scale in-plane responses measured by real-world knitted fabrics. The brute-force approach—simulating every yarn in a swatch of fabric, comparing the results to the measured data, and looping until the correct parameters are found—is computationally infeasible, as a single swatch can easily contain tens of thousands of knit loops. To avoid full-scale simulation, we approximate the fabric swatch as a periodically repeating pattern; this allows us to take advantage of the method for simulating knits with periodic boundary conditions as established in Chapter 3.

However, simply tiling these periodic yarn physics over a patch the size of the physical knitted fabric swatch is insufficient to reproduce our physical test data. In particular, it cannot model the spatially non-uniform deformations that occur when fabrics are sheared (Figure 6.2). To ensure that our method performs well even in the presence of spatially non-uniform deformations, we introduce an intermediate thin-shell model and solve for a physical model that reproduces the test data. Once we have this model, we generate spatially uniform deformations that can finally be modeled with periodic yarn-level simulation.

Our parameter-fitting pipeline is illustrated in Figure 6.1. We first solve for a thin-shell model that reproduces the real world data samples. Next, we use this thin-shell model to generate spatially uniform data samples. We then fit yarn-level simulation parameters to this new uniform data.

We want our simulation models to be complex enough to model important features in the data, while simultaneously minimizing additional complexity to avoid overfitting. We discuss in Section 6.3 how a simple StVK thin-shell model is insufficient to adequately capture the array of behaviors observed in our physical tests, but we are able to reproduce the data well

with the simple addition of anisotropy and an area-preservation term. Similarly, although we found the basic discrete elastic rod model insufficient for reproducing the behaviors of real-world textiles, Section 6.4 explains how we are able to reproduce the real-world data with the addition of two-phase models for contact and stretching to capture the behavior of plated yarns.

6.2 Input Data

This section discusses the different types of data that can be gathered from real-world fabrics, and how the data can be used in practice for achieving our goals.

The first type of data we consider is the information used for fabricating each material. Examples of such data are the type(s) of yarn used to create the fabric, and the knitting/weaving instructions. Another type of data is experimental measurements, such as yarn-level mechanical tests, swatch-level mechanical tests, and high-resolution images.

We could follow a number of approaches for using this data in simulation. A "bottom-up" modeling approach could take the fabrication settings and yarn-level mechanical tests, and simulate the mechanics of the knitting/weaving process that yield the final fabric pattern and its macroscopic response. A "top-down" approach, on the other hand, could take high-resolution images and swatch-level tests, and estimate both the pattern geometry and the yarn-level mechanical response. Unfortunately, both approaches suffer serious challenges. In the bottom-up approach, the fabrication process entails many unknowns, such as the forces applied by the needles or the plastic deformation of yarns, and there is no known procedure to measure the mechanical response of yarns in tight contact. In the top-down approach, the yarn pattern is not fully observable due to severe occlusion, especially in multi-layer fabrics, and the strains of individual yarns are unknown.

To work around these challenges, our project follows a hybrid approach, by combining fabrication settings and experimental measurements to design and fit yarn-level models. We use the yarn topology defined by knitting instructions, together with high-resolution photographs, to initialize the geometry of the yarn pattern. We also use swatch-level mechanical tests to estimate the mechanical parameters of the yarn-level model.

Section 6.2.1 describes the composition of the yarns used in our experiments; Section 6.2.2 describes the initialization of yarn geometry; and Section 6.2.3 describes the mechanical tests. Later in Section 6.7 we discuss additional yarn data that we considered but did not use in our system.

6.2.1 Yarn Composition

For each fabric swatch in our data set, we list the associated yarn types, the number of filaments, the denier of each yarn type (i.e., mass in grams per 9,000 meters of unstretched yarn), the stiffness of each yarn type (measured in response to a force of 5cN), and the mass percentage of each yarn type. For example, 72D/75F polyester 90%, 20D spandex 10%, indicating that 90% of the fabric mass corresponds to a 75-filament 72-denier polyester, and 10% of the mass corresponds to 20-denier spandex. This information allows us to assign basic parameters like the density of each simulated yarn, and we have the option to derive additional information, as discussed in Section 6.7.

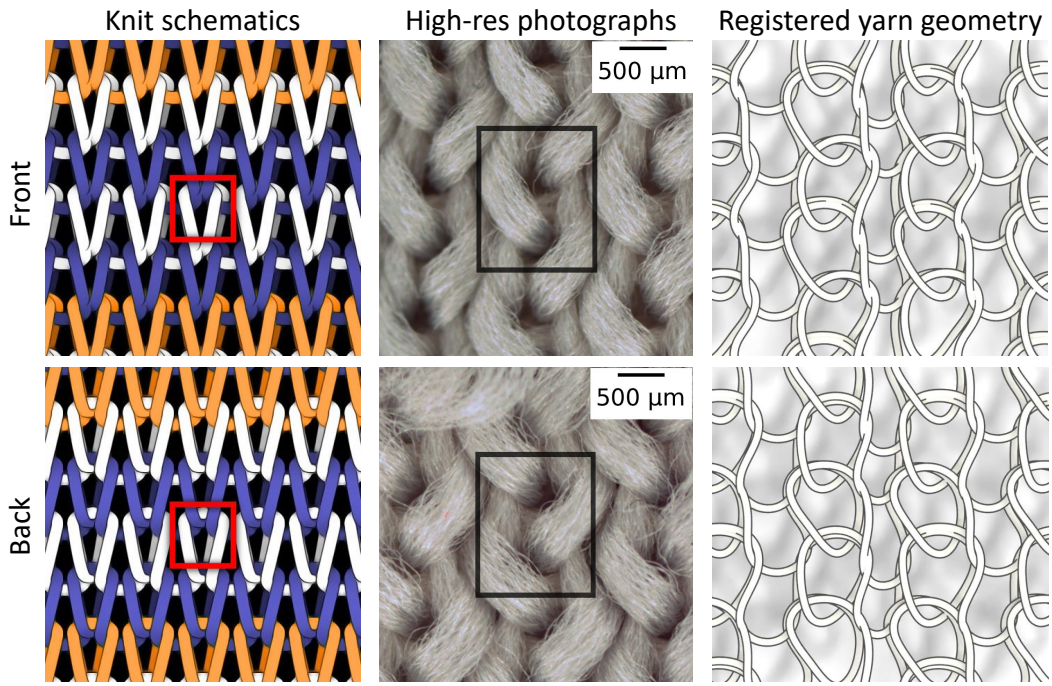


Figure 6.3: Initialization of the yarn geometry for an all-needle fabric (A2). We take as input a schematic visualization that depicts the topology of the yarn construction (left) and high-res photographs that show the yarn geometry on the visible layers (middle). The squares highlight the repeat of the yarn pattern. We initialize the yarn geometry (right) by manually registering the yarn curves to the photographs, and smoothly interpolating to unobserved regions.

6.2.2 Initialization of Yarn Geometry

To define the yarn topology from knitting instructions, we leverage existing tools by knitting machine providers. Specifically, we have used the M1plus by STOLL, which outputs a schematic representation of the knit pattern (Figure 6.3-left). Note that this schematic defines topology, but it does not accurately define where yarn contacts occur. We also use this data to identify a representative repeating periodic tile of the pattern.

We obtain front and back high-resolution photographs of the knit pattern using a custom-built optical system, which captures photographs of 4912×3684 pixels at a resolution of 1.8 μm per pixel, with close-to-uniform illumination (Figure 6.3-middle). We first use these images to estimate a yarn radius, which we use for visualization and to help with initial yarn registration. Next, we identify the corresponding periodic tile on the photographs, and manually register the yarn topology to the visible layer on both front and back photographs. Manual registration took 1 to 10 hours per fabric, depending on pattern complexity. This step assigns 2D coordinates of the portions of the yarns visible in the photographs, so we still need to approximate the geometry of the occluded yarns, as well as the depth coordinates. We approximate this missing geometry by constraining the centerlines of yarns that overlap in the images to be two yarn radii apart in depth, and by smoothly interpolating the yarn geometry in unobserved regions. We include these manually registered 3D yarn geometries in our data set, and we use them as input to our yarn model parameter estimation (which is free to optimize the initial guess further, as in Section 6.5.2).

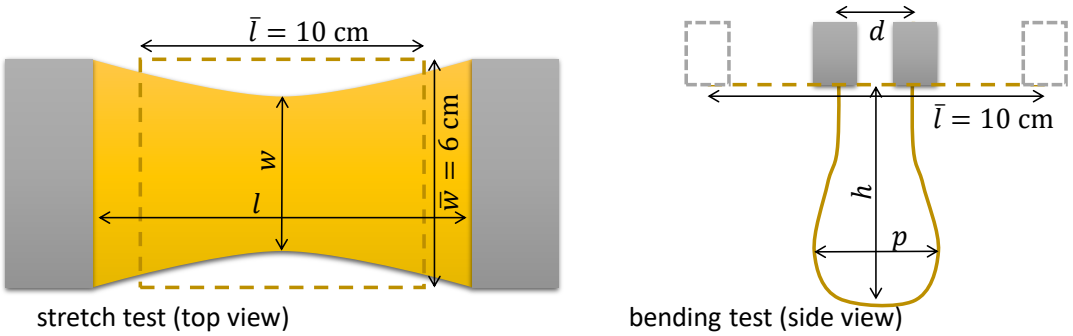


Figure 6.4: Testing principles and fabric measurements for swatch-level stretch and bending tests. We have performed these tests along three directions (weft, bias, warp) to all 33 fabrics in our library.

6.2.3 Physical Tests

To fit accurate yarn-level parameters, we seek physical test data that captures the force-deformation response of the fabric under stretch and bending in different directions, accounting also for nonlinearities and transverse behavior. We have developed a custom-built test rig capable of executing two state-of-the-art experiments: a clamped stretch test [Kawabata 1980] and a pear-loop bending test [Peirce 1930]. The principles of the tests and the size of the fabric swatches are shown in Figure 6.4. We chose the pear-loop test vs. the cantilever bending test [Wang et al. 2011] because in our early experiments it showed better sensitivity at low bending stiffness.

In the stretch test, a swatch of length $\bar{l} = 10$ cm (excluding the clamped region) and width $\bar{w} = 6$ cm is clamped on two ends and stretched horizontally under known forces. In practice, the stretch test is controlled by displacement, while force is measured. For each stretch force f_s, we compute the stretch $s = l/\bar{l} - 1$ (with l the deformed length) and the orthogonal compression $c = 1 - w/\bar{w}$ (with w the deformed width). In the bending test, the same swatch is bent by bringing the clamps together. For each inter-clamp distance d, we measure the aspect ratio $r = h/p$ of the pear-shaped loop (with h the height of the loop and p its width).

One of the important decisions for the estimation of yarn-level models is the definition of a working range, which in turn affects the range of the physical test data. In garment design, fabric stretch is quantified for the warp direction, which is typically the stiff direction of the fabric, and is usually vertically aligned with the subject's body direction along the torso and limbs. Weft is often too compliant to be aligned with the vertical direction, as clothes would hang loose; instead it allows a comfortable fit along the body's circumference and flexibility for (un)dressing. Fashion ergonomics studies suggest a comfort stretch (i.e., stretch necessary in the warp direction for casual wear) of 5 to 30%, and a power stretch (i.e., stretch necessary for active wear) of 30 to 50% [Wang et al. 2008; Lyle 1977]. For the fabrics in our library, we have observed that the average warp stretch at 2 N stretch force is 31%, and at 5 N it is 52%. This suggests that 2 N and 5 N are rough upper bounds for comfort stretch and power stretch, respectively. Therefore, we have executed stretch tests up to 5 N. We have done this, together with bending tests, on three directions: weft (0 deg), bias (45 deg) and warp (90 deg). In the weft direction we often fall short of 5 N, as we reach the rig's stretch limit (160%).

In this project, we have not addressed the hysteresis of force-deformation tests, i.e., the force difference between loading and unloading regime, produced by inter-yarn friction [Miguel et al. 2013]. We leave this phenomenon as future work, which requires estimating a model of

the inter-yarn friction forces. In the physical tests, we only consider stretch under loading conditions, from rest to 5 N.

6.2.4 Summary of Data

To summarize, our data set consists of the following information for each set of 33 different fabric samples:

- Physical characteristics of the yarn used in each fabric, including the material used, its density, its stiffness, and information about any special coating on the fibers
- Schematics describing the topology of each knitted pattern
- High resolution photographs of both sides of the fabric
- Initial yarn geometry and topology derived from knitting instructions and photographs
- Measurements resulting from mechanical tests of stretching and bending properties of the fabric

6.3 Intermediate Thin-Shell Model

As outlined in Section 6.1, we use a thin-shell model as an intermediary between mechanical tests on full-scale swatches (which can exhibit spatially non-uniform deformations) and periodic yarn-level simulations (which assume spatially uniform deformation). To solve for the material parameters in the thin-shell model, we focus our efforts on in-plane scenarios where bending plays no role (Section 6.3.1). Additionally, although it is not the main result of our work, we offer a preliminary method for solving for bending parameters in Section 6.3.2.

6.3.1 In-Plane Deformation Model

We seek a deformation model that captures the anisotropic and nonlinear behavior of knitted fabrics, while minimizing the number of parameters. Note that the input data (Section 6.2.3) approximates uniaxial deformations, and lacks information about the fabric's response to biaxial deformations. Adding high-order strain dependency to the parameters, as done in previous works [Wang et al. 2011; Miguel et al. 2012], could over-parameterize the model and lead to overfitting. While limited, we opt for the robustness provided by a strain-independent parameterization.

A simple choice to represent both anisotropy and nonlinearity is the anisotropic Saint Venant-Kirchhoff (StVK) model [Volino et al. 2009]. However, we have observed that anisotropic StVK might exhibit a high directional Poisson's ratio (up to 2 in some cases), and inversion within the tested stretch range. Note that this does not make the model unstable, but it is of course unrealistic. To address this, we augment the anisotropic StVK model with a Neo-Hookean area-preservation term [Bonet and Wood 2008; Smith et al. 2018]. As shown in the plots in Figure 6.5 for a uniaxial stretch deformation with zero orthogonal stress, the stretch-aligned stress of the Neo-Hookean-augmented model remains the same as the original anisotropic StVK model, but inversion no longer occurs.

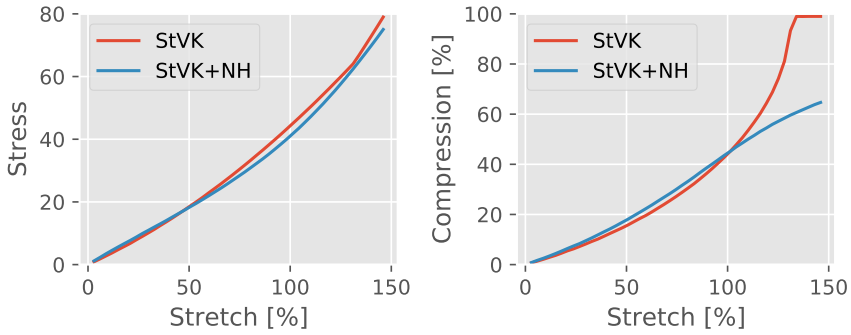


Figure 6.5: Our thin-shell formulation augments the anisotropic StVK model with a Neo-Hookean area-preservation term [Smith et al. 2018]. This term does not affect the stress-stretch behavior (left), but it eliminates inversion problems (right). Both models, with and without the Neo-Hookean term, are fitted to a double-knit interlock fabric (DKIN1). Without the Neo-Hookean term, the fitted model is stable but suffers inversion (i.e., it reaches 100% compression).

Formally, the Neo-Hookean-augmented anisotropic StVK model is formulated as follows. Given the deformation gradient $\mathbf{F}$, Green strain $\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^\top \mathbf{F} - \mathbf{I}\right)$ written in Voigt notation as $\boldsymbol{\varepsilon} = (E_{11}, E_{22}, 2E_{12})$, and area ratio $J = \det(\mathbf{F})$, the strain energy density is:

$$\Psi_{\text{inplane}} = \frac{1}{2}\boldsymbol{\varepsilon}^\top \begin{pmatrix} k_{xx} & k_{xy} & 0 \\ k_{xy} & k_{yy} & 0 \\ 0 & 0 & k_{ss} \end{pmatrix} \boldsymbol{\varepsilon} + k_{n1}\left(J-1\right)^2 + k_{n2}\log^2 J. \tag{6.1}$$

To estimate the parameters ($k_{\text{inplane}} = \{k_{xx}, k_{xy}, k_{yy}, k_{ss}, k_{n1}, k_{n2}\}$ in (6.1)), we follow a simulation-in-the-loop optimization strategy [Miguel et al. 2012]. We search for the parameters that produce the best match to the stretch deformation data described in Section 6.2.3, subject to static equilibrium of the simulated cloth swatch. Formally, this is

$$\begin{aligned} k_{\text{inplane}} = \arg\min \sum_i & w_s \left\| f_s(k_{\text{inplane}}, s_i) - f_{s,i} \right\|^2 + \\ & w_c \left\| c(k_{\text{inplane}}, s_i) - c_i \right\|^2. \end{aligned} \tag{6.2}$$

Specifically, we use 6 target deformations for each weft, bias, and warp direction, distributed evenly along the stretch range. For each i target deformation, we apply the measured stretch s_i, simulate the fabric swatch to equilibrium, and evaluate the error with respect to measured stretch force $f_{s,i}$ and orthogonal compression c_i. The weights w_s and w_c normalize stretch force and compression error using the average measured values.

Figure 6.6 shows a representative fit of the in-plane thin-shell model for an all-needle fabric (A2). Notice the extreme anisotropy of the fabric. Results are discussed in more detail in Section 6.6, but the proposed model provides an accurate overall fit across all fabrics (avg. 17.59%±8.33% error for stretch force, and avg. 16.84%±8.11% error for orthogonal compression). We have noticed that the fabrics in our data set exhibit higher nonlinearity than the StVK model, but the fit quality is sufficient to act as intermediate representation for the yarn-level model.

We use this thin-shell model to generate target data for our yarn-level model by reproducing spatially uniform versions of the mechanical test scenarios in our data-set. Notably, the stretch tests clamp two ends of the fabric swatch but leave the other two sides free, which

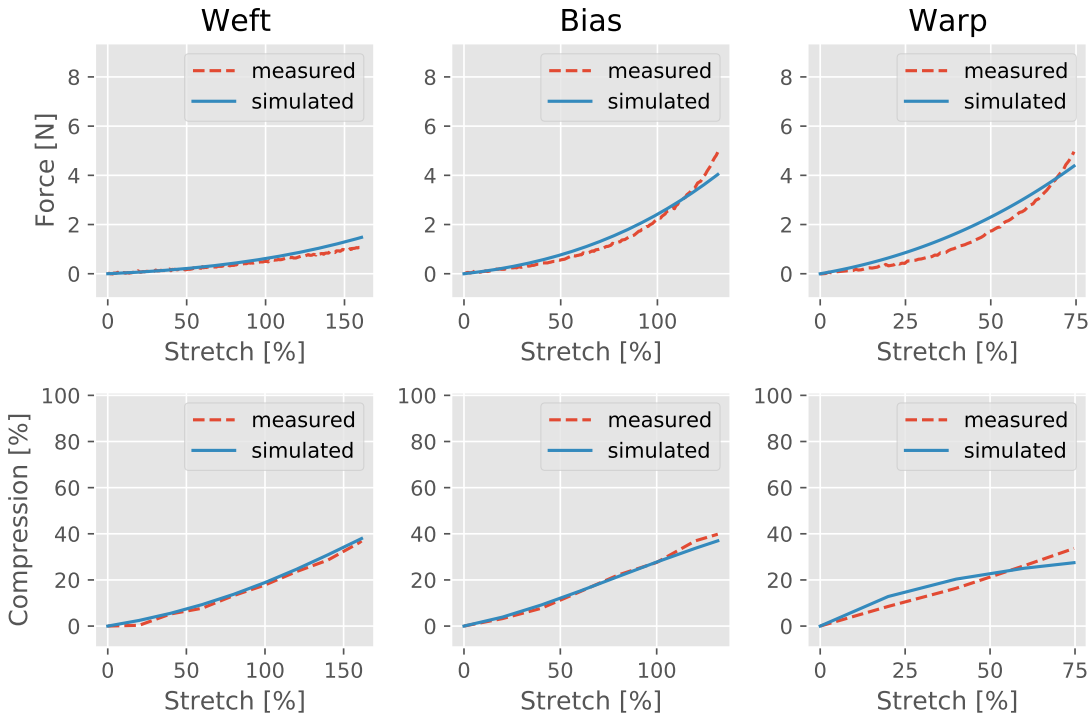


Figure 6.6: Fitting of the in-plane thin-shell model to the physical test data, for an all-needle fabric (A2). Top: force vs. stretch fits. Bottom: orthogonal compression vs. stretch fits. Notice the extreme anisotropy of the fabric.

leads to a minimization of stress in the direction orthogonal to the stretch. To reproduce this behavior, we compute uniaxial stretch deformations with zero orthogonal stress of the strain energy density (6.1). Denoting the known applied stretch as s and the unknown orthogonal compression as c, the stretching direction can be defined by a unit vector $\mathbf{u}$ and the compression direction by an orthogonal vector $\mathbf{v}$, or alternatively by a rotation matrix $\mathbf{U} = (\mathbf{u}\ \ \mathbf{v})$. The resulting deformation gradient is $\mathbf{F} = \mathbf{U}\,\mathrm{diag}(1+s, 1-c)\,\mathbf{U}^\top = (1+s)\,\mathbf{u}\,\mathbf{u}^\top + (1-c)\,\mathbf{v}\,\mathbf{v}^\top$. We then compute the orthogonal compression as:

$$c = \arg\min \Psi_{\text{inplane}}(\mathbf{F}(s, c, \mathbf{U})). \tag{6.3}$$

Once the minimum-energy compression is known, we evaluate the stretch stress $\sigma_s = \frac{\partial \Psi_{\text{inplane}}}{\partial s} = \mathbf{u}^\top \frac{\partial \Psi_{\text{inplane}}}{\partial \mathbf{F}} \mathbf{u}$, with $\frac{\partial \Psi_{\text{inplane}}}{\partial \mathbf{F}}$ the first Piola-Kirchhoff stress (recall (3.23)). We generate analytical stretch data $\{s, c, \sigma_s\}$ for weft, bias, and warp directions for each fabric, using the stretch range measured on the real fabric along each direction.

Because this thin-shell model acts as a translator from the potentially non-uniform data to the perfectly uniform periodic yarn simulator, we should avoid encoding additional noise from numerical errors into the thin-shell results. To verify its accuracy, we recomputed our results on meshes that were uniformly subdivided two times and found the average difference in output between the original and refined meshes to be only 2%.

6.3.2 Bending Model

The fabrics in our database exhibit a wide range of complex behaviors when subject to our physical bending test; for example, some fabrics curl out of plane when stretched (Figure 6.7) or break symmetry during bending tests. These complications make it challenging to isolate

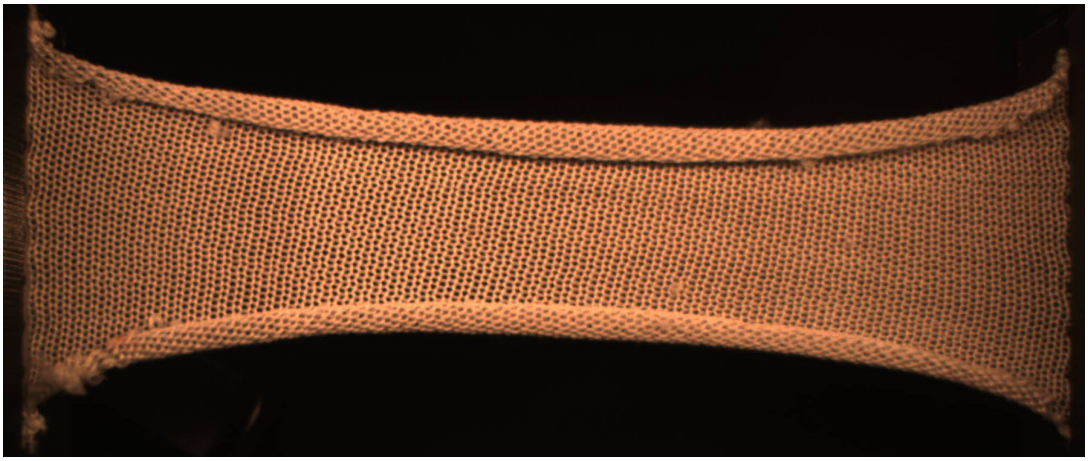


Figure 6.7: This single-jersey fabric (SJ14) has a tendency to strongly curl out of plane, even during in-plane stretching tests. Such behaviors pose difficulties to the accurate measuring and modeling of bending properties.

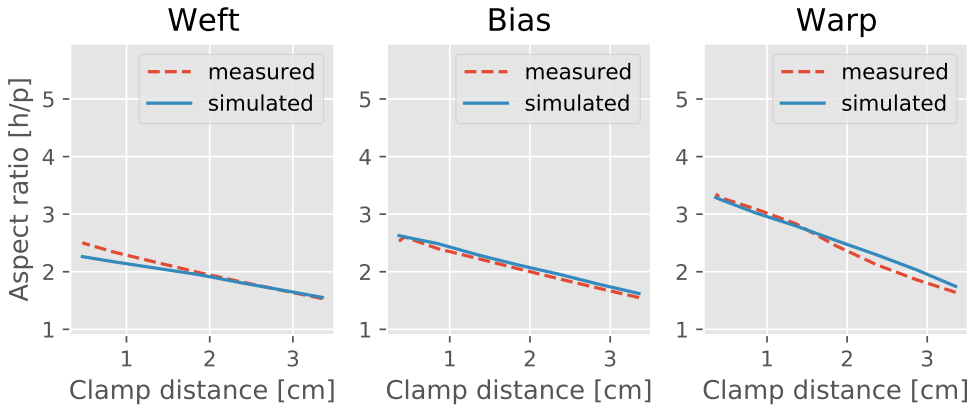


Figure 6.8: Fitting of the thin-shell bending model to the physical test data, for a links fabric (L1). The plots show the aspect ratio of the pear loop vs. inter-clamp distance.

simple bending relationships from our data and to accurately reproduce these results in simulation. Nevertheless, we document here our first efforts toward fitting the bending behavior of the materials in our data set.

We first note that, although the bending behavior of our fabrics appears more isotropic than the in-plane behavior, some fabrics (like the links fabric in Figure 6.8) exhibit noticeable bending anisotropy. To ensure our model generalizes to these scenarios, we fit these behaviors with a discrete-shell bending model with anisotropic stiffness.

In the computer graphics literature, there are multiple choices for discrete curvature models [Grinspun et al. 2003; Wardetzky et al. 2007]. We opt for an edge-based curvature metric, as this allows simple parameterization of anisotropy using the rest-shape orientation of mesh edges. Given an edge with bend angle θ, and incident triangles with average altitude from base to vertex H, we define the edge curvature as $\kappa = 3\,\theta/H$. This curvature metric converges to the mean curvature of a cylinder with edges aligned with the cylinder axis. This is particularly important for estimating the yarn-level model using analytical deformation data, as periodic yarn deformations will be designed following cylindrical bending (see Section 6.5.1).

Based on the curvature metric above, the bending energy density is:

$$\Psi_{\text{bending}} = k_\theta\,\kappa^2. \tag{6.4}$$

We multiply this energy density by the area of the incident triangles to obtain the discrete edge energy. We define bending stiffness values k_θ for the weft, bias, and warp directions, and interpolate linearly between them. The bending parameters are then $k_{\text{bending}} = \{k_{\theta,\text{weft}}, k_{\theta,\text{bias}}, k_{\theta,\text{warp}}\}$.

Similar to the estimation of in-plane model parameters, we use a simulation-in-the-loop optimization approach to estimate the bending parameters. We search for the parameters that produce the best match to the bending deformation data described in Section 6.2.3, subject to static equilibrium of the simulated cloth swatch. Formally, this is

$$k_{\text{bending}} = \arg\min \sum_i \|r(k_{\text{bending}}, d_i) - r_i\|^2.$$

Specifically, we use 7 target deformations for each weft, bias, and warp direction, distributed evenly between inter-clamp distances of 0.3 and 3.4 cm. For each i target deformation, we impose the inter-clamp distance d_i, simulate the fabric swatch to equilibrium, and measure the aspect ratio r of the bending loop.

Figure 6.8 shows a representative fit of the bending thin-shell model for a links fabric (L1). Again, results are discussed in more detail in Section 4.3, but the proposed model provides an accurate overall fit across all fabrics (avg. 5.61% ± 2.21% error).

To generate analytical target data for yarn-level estimation, we simply evaluate the bending stress $\sigma_\kappa = \frac{\partial \Psi_{\text{bending}}}{\partial \kappa} = 2k_\theta\,\kappa$. We obtain data $\{\kappa, \sigma_\kappa\}$ for weft, bias, and warp directions for each fabric, using the curvature range observed on the real fabric along each direction.

6.4 Yarn Model Parameterization

Our next goal is to find the parameters of a periodic yarn-level simulation so that it reproduces the same response to deformation as the thin-shell model described in the previous section (and thus, the fabric-level deformation tests in our data set).

We seek a yarn model with a minimal number of parameters, and which inherently captures the complexity of the coarse-scale behavior. In this regard, we start with rod models used previously for yarn-level cloth simulation [Bergou et al. 2008; Kaldor et al. 2008], and we introduce the minimal extensions necessary to capture the behavior of real fabrics with potential complications like plated yarns made of multiple materials. Note that this is a deviation from the model discussed in Chapter 3. We describe in turn the models we use for yarn *stretch*, *bending*, and *contact*. We observe in our use case that *twist* forces are small and do not affect the overall mechanical response. Our current model does not account for inter-yarn friction, and we leave this to future work.

6.4.1 Stretch

Many fabrics blend yarns of different types to achieve complex mechanical and/or aesthetic behavior. One common example in our fabric library is plating, where two or more yarns are knitted side by side. Figure 6.9 shows a plated fabric consisting of both a flexible spandex yarn and a stiff polyester yarn. At rest, the flexible yarn is stretched, while the stiff yarn is compressed. As a result, the fabric is flexible under low forces, and then it turns stiff under high forces, once the stiffer polyester yarn is stretched. The complex interplay of plated yarns cannot be captured by modeling each yarn type separately.

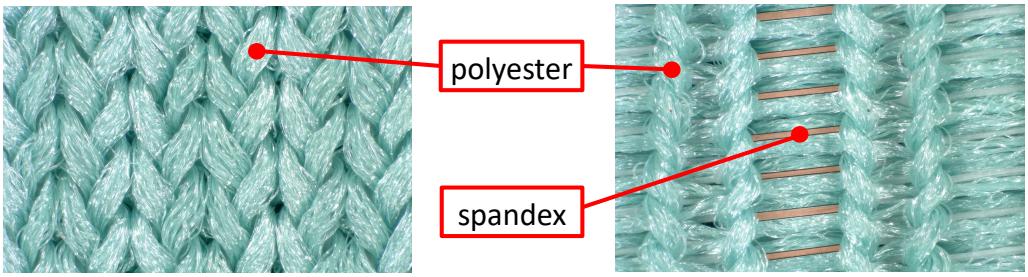
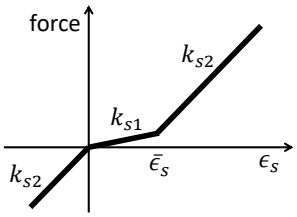


Figure 6.9: High-resolution photographs of a plated double-knit interlock fabric (DKIN8) at 20% warp stretch (left) and 150% weft stretch (right). A multi-filament stiff yarn, polyester, provides texture and stiff response under high forces. A single-filament flexible yarn, spandex (partially highlighted), provides flexible response under low forces. The flexible yarn is stretched during knitting, and then it compresses the stiff yarn as it retracts and relaxes into the stitch structure.

Motivated by this complex stretch behavior of multi-yarn fabrics, we have designed a yarn stretch model that represents the combined response of multiple yarns. The force profile includes three linear regimes: one for low stretch with stiffness k_{s1}; another one for stretch larger than $\bar{\varepsilon}_s$ with stiffness k_{s2}; and the compression regime again with stiffness k_{s2}. The inset shows the force profile as a function of yarn stretch ε_s. Roughly, k_{s2} represents the stiffness of the stiffer yarn in a plated fabric, and $\bar{\varepsilon}_s$ the onset of stretch for this yarn. k_{s1} represents the combined response at low stretch. The stretch energy of a yarn segment with rest length $\bar{L}$ is formally:



$$W_s = \begin{cases} \frac{1}{2}\,\bar{L}\,k_{s2}\,{\varepsilon_s}^2 & \varepsilon_s \le 0 \\ \frac{1}{2}\,\bar{L}\,k_{s1}\,{\varepsilon_s}^2 & 0 \le \varepsilon_s \le \bar{\varepsilon}_s \\ \frac{1}{2}\,\bar{L}\,\left(k_{s2}\;(\varepsilon_s - \bar{\varepsilon}_s)^2 + k_{s1}\,\bar{\varepsilon}_s\;(2\,\varepsilon_s - \bar{\varepsilon}_s)\right) & \bar{\varepsilon}_s \le \varepsilon_s \end{cases} \tag{6.5}$$

Surprisingly, we found that this proposed nonlinear stretching behavior is important even for simulating fabrics made only of a single stiff yarn (e.g., polyester). We speculate that this could be due to uncertainty in rest lengths and/or friction state, as well as inherent stretch nonlinearity under low forces (e.g., due to filament realignment). For this reason, we use the nonlinear stretch model for all fabrics in the project.

6.4.2 Bending

We choose a yarn bending model following the formulation of Bergou et al. [2008, 2010], but similar to the model in Chapter 3 we disconnect the bending stiffness and stretch stiffness of the yarn, making them independent model parameters. In addition, under the uncertainty about the yarn's cross-section shape, we choose an isotropic bending model in our project, and we leave the design of richer bending models to future work.

Following the discrete curvature vector $\boldsymbol{\kappa}$ proposed by Bergou et al. [2008], we define the bending energy at a yarn vertex as

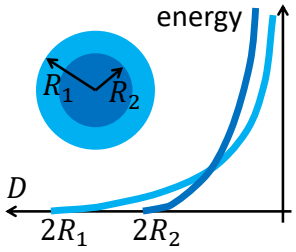
$$W_b = \frac{1}{2}\,\frac{2}{\bar{L}_a + \bar{L}_b}\,k_b\,\|\boldsymbol{\kappa} - \bar{\boldsymbol{\kappa}}\|^2, \tag{6.6}$$

where $\bar{\boldsymbol{\kappa}}$ is the rest curvature, and $\bar{L}_a$ and $\bar{L}_b$ are the rest lengths of the incident yarn edges.

6.4.3 Contact

The cross-section of yarns deforms in a complex and nonlinear way when yarns are compressed into contact. For a multi-filament yarn, fibers may be loose and voluminous under low stretch, and they may realign anisotropically under combined stretch and contact. This phenomenon is even more complex for plated yarns (Figure 6.9) when viewed as a single composite yarn.

For these reasons, we choose to model both moderately soft contacts over a large distance and stiff contacts over a small distance. This models both the forgiving collision response of knit loops gently touching when the fabric is relaxed, as well as a strong resistance when the space within braided fibers collapses and there is no more room to compress. We model this two-phase contact force by combining two barrier potentials, one modeling softer large-radius contact, and another one modeling stiffer small-radius contact, as shown in the inset. Each barrier potential is parameterized by its radius R_i and stiffness k_{ci}, $i \in 1, 2$.



We build on the yarn-yarn contact model of Kaldor et al. [2008], but we substitute their barrier term with the one proposed by Li et al. [2020]. However, we slightly modify it to use relative distances, as this improved the scale of the stiffness for parameter optimization. Given a yarn-yarn centerline distance D, contact radius R_i, contact stiffness k_{ci}, and a barrier function $f(x) = -(x-1)^2 \log x$, the contact energy is

$$W_c = k_{ci}\, \bar{L}_a\, \bar{L}_b \int_0^1 \int_0^1 f\left(\min\left(\frac{D}{2\,R_i}, 1\right)^2\right) da\, db, \tag{6.7}$$

with $\bar{L}_a$ and $\bar{L}_b$ the rest lengths of the two colliding yarn edges, and the double integral over colliding edges a and b is evaluated with Simpson's rule.

6.5 Yarn Model Estimation

We now discuss how to estimate yarn-level parameters in order to best fit the data produced by the thin-shell model. Section 6.5.1 summarizes the simulation of yarn-level fabrics under periodic boundary conditions, which are key to compare to the thin-shell data.

The yarn-level model contains two types of unknown parameters: yarn rest shapes and mechanical parameters. We follow an optimization procedure that alternates the estimation of these two parameter subsets. Section 6.5.2 describes the estimation of yarn rest shapes, and Section 6.5.3 the optimization of mechanical parameters.

6.5.1 Periodic Yarn-Level Simulations

We build on the periodic yarn simulations from Chapter 3. We separate two sets of degrees of freedom on a periodic yarn tile: a uniform coarse deformation $\boldsymbol{\chi} = (s, c, \kappa, \mathbf{U})$, which gathers stretch s, orthogonal compression c, bending curvature κ, and a 2D rotation matrix $\mathbf{U}$ which defines the direction of stretch and/or bending; and a vector of nodal yarn displacements $\tilde{\boldsymbol{u}}$, expressed relative to the coarse deformation.[1] Together, these degrees of freedom define full

[1] We can convert the coarse deformations s, c, κ, and $\mathbf{U}$ into the first and second fundamental forms we are used to from Chapter 3. Specifically, we compute $\mathbf{F}$ from s, c and $\mathbf{U}$ as in (6.3). Then, $\bar{\mathbf{I}} = \mathbf{F}^\top \mathbf{F}$. Similarly, $\bar{\mathbf{II}} = \mathbf{U}\,\mathrm{diag}(\kappa, 0)\,\mathbf{U}^\top$.

nodal yarn positions $\mathbf{x}(\boldsymbol{\chi}, \tilde{\boldsymbol{u}})$. To make the overall behavior of the yarn simulation match the prescribed coarse scale behavior, the nodal yarn displacements must satisfy periodic boundary conditions and must be absent of yarn sliding, yarn twist, and net rigid motion. We express all these constraints together as $C(\tilde{\boldsymbol{u}}) = 0$.

The various deformation components described in Section 6.4 (stretch, bending, and contact) compile a set of discrete energy elements $\{W_i\}$ over a periodic tile of rest area $\bar{A}$. We compute the overall energy density of the tile as

$$\Psi_{\text{yarns}}(\mathbf{x}(\boldsymbol{\chi}, \tilde{\boldsymbol{u}})) = \frac{1}{\bar{A}} \sum_i W_i(\mathbf{x}(\boldsymbol{\chi}, \tilde{\boldsymbol{u}})). \tag{6.8}$$

Given a coarse deformation $\boldsymbol{\chi}$, the yarn-level deformation can be obtained as the minimum-energy configuration (i.e., equilibrium) that satisfies the constraints:

$$\tilde{\boldsymbol{u}} = \arg\min \Psi_{\text{yarns}}(\mathbf{x}(\boldsymbol{\chi}, \tilde{\boldsymbol{u}})), \quad \text{s.t. } C(\tilde{\boldsymbol{u}}) = 0. \tag{6.9}$$

We solve this optimization using the constrained Newton approach from Chapter 3. As each yarn tile contains only tens to hundreds of yarn nodes, the optimizations are fast in practice.

To compare yarn simulations to the thin-shell analytical target data, we need two additional ingredients. First, we need to compute stretch deformations under minimum-energy orthogonal compression. To this end, given a stretch s, we find the compression c that minimizes Ψ_{yarns} in (6.8), subject to equilibrium conditions (6.9) on the yarn deformation. We implement this optimization using the COBYLA method [Powell 1994b], solving (6.9) before every energy evaluation.

Second, we need to compute homogenized coarse stress, in particular the stretch stress $\sigma_s = {}^{\mathrm{d}\Psi_{\text{yarns}}}/_{\mathrm{d}s}$. Applying the chain rule to (6.8) for a coarse stretch s and simplifying yields:

$$\frac{\mathrm{d}\Psi_{\text{yarns}}}{\mathrm{d}s} = \frac{\partial \Psi_{\text{yarns}}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial s}. \tag{6.10}$$

Recall also (3.23) from Chapter 3. Note that we compute stretching stress analytically and compute bending stress by finite differencing Ψ_{yarns}.

6.5.2 Yarn Rest-Shape Estimation

The rest shape of individual yarns, i.e., their rest length and rest curvature, is unknown, as it is heavily influenced by plasticity during the knitting process.One way to address this is to estimate both the yarn rest shape and mechanical parameters together, to best fit the thin-shell mechanical data. However, we have seen that the error function with respect to mechanical parameters alone is plagued with local minima, requiring the use of global optimization methods. Global optimization of both the rest shape and mechanical parameters together appears intractable, and we did not find a suitable low-rank parametrization of rest shapes; therefore, we have devised a different procedure for rest-shape estimation, motivated by the stability of the fabric's coarse-level rest state.

At the fabric's rest state, net yarn forces are zero, due to equilibrium between all force components. However, the input yarn geometry (Section 6.2.2) is not at rest, due to unbalanced inter-yarn contact. If we let the yarns relax to reach equilibrium under no coarse deformation, the fabric suffers a non-zero coarse stress $\frac{\partial \Psi_{\text{yarns}}}{\partial \boldsymbol{\chi}}$, i.e., the fabric's expected rest state is actually not stable and wants to deform (e.g. contract).

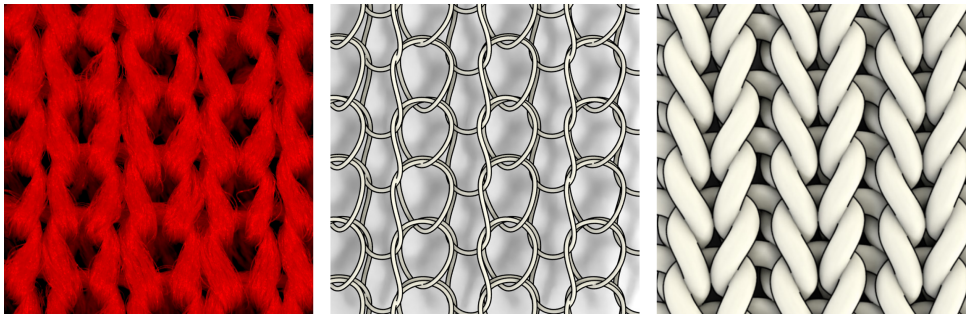


Figure 6.10: High-resolution photograph (left), hand-registered yarn geometry (center), and simulated yarn rest-shapes (right) for an all-needle fabric (A1).

Based on these observations, we separate the full parameter estimation into two problems. We let yarn rest shapes ensure stability of the fabric's rest state, and we let mechanical parameters fit the coarse mechanical response. In our experience, finding a stable but slightly pre-tensed rest state (i.e., with non-zero though balanced yarn forces) was key for obtaining good fits of the mechanical response with a small mechanical parameter set.

We pose the problem of yarn rest-shape estimation as follows: We seek rest shapes such that the coarse stress at the fabric's expected rest state is small, and the equilibrium configuration of the yarns deviates little from the input yarn geometry. To solve this problem, we follow a heuristic approach similar to the one proposed in Section 3.3.1 and Appendix A.2.4. We alternate equilibrium solves (6.9) and resetting of yarn rest shapes at the current configuration. While doing this, and to bound the deviation from the initial geometry, we bound the change in yarn rest lengths to 20%. Every time rest shapes are reset, contact forces push the yarns away. At initial steps, these contact forces may be very strong and too localized, therefore we run only a few simulation steps before resetting rest shapes. At later steps, contacts become smooth, and we let simulations run toward convergence. See Figure 6.10 for an example comparing the real-world fabric, the hand-registered initial yarn geometry, and the result of our rest-shape optimization.

Yarn rest-shape estimation must be executed after every change to the mechanical parameters, as the yarn equilibrium configuration is changed. For this reason, we alternate rest-shape estimation and the mechanical parameter optimization described next.

6.5.3 Mechanical Parameter Estimation

To estimate the mechanical parameters of the yarn model, we pose and solve a numerical optimization problem. The remainder of this section discusses the objective function, the optimized parameters, and the solvers we use.

Objective function

We have designed various error metrics between the yarn model and the target thin-shell data. The stretch error component err_{stretch} measures error in stretch stress. The default error, based on the first Piola-Kirchhoff stress σ_s, ramps up at high stretches, due to the high nonlinearity of the stress function. Instead, we measure error in the second Piola-Kirchhoff stress, obtained as $\frac{\sigma_s}{1+s}$. The compression error component err_{compress} measures simply the difference in orthogonal compression. To generate the stretch and compression values, we computed zero-orthogonal-stress deformations on both the thin-shell and yarn models, sampled over the stretch range

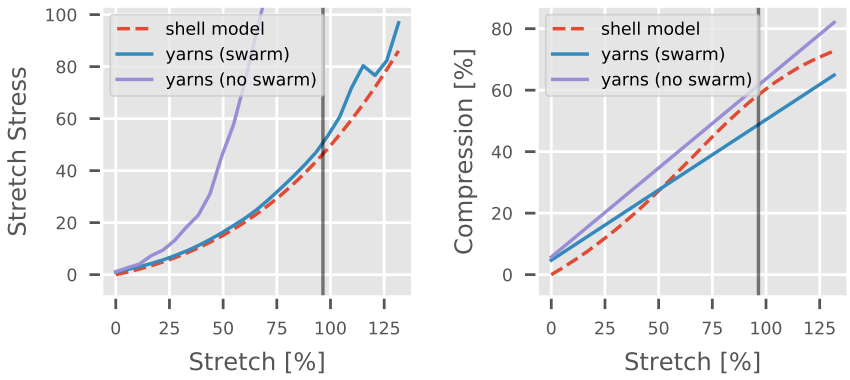


Figure 6.11: The performance of our system with a local nonlinear solver (purple) compared to initializing with a particle swarm optimization (blue), on an all-needle fabric (A2). Without the swarm optimization, the fitting gets stuck in a local minimum, causing the stretching error to blow up. The black bar denotes the transition from the "comfort" to the "power" range of stretches.

of the physical test data, for all three directions weft, bias, and warp. The yarn-level zero-orthogonal-stress computation is a slowly-varying function that is expensive to evaluate on-demand but well-approximated by simple interpolation; therefore, we compute the optimal orthogonal compression only on both ends of the stretch range, and linearly interpolate in between. As a positive side effect, this interpolation helps in smoothing the noise caused by buckling instabilities found near the optimal compression (see also Chapters 4 and 7).

For the estimation of the yarn model, we have used the following objective function, with stretch and compression error components:

$$g = \sum_i \alpha_i \, err_{\text{stretch},i}{}^2 + \sum_j \beta_j \, err_{\text{compress},j}{}^2. \tag{6.11}$$

To set the weights $\{\alpha_i, \beta_j\}$ of the error components, we follow these heuristics. First, we normalize each error component based on the maximum target value in the comfort stretch range. Second, we apply a decaying weight $v_{\text{comfort}}/\max(v_{\text{comfort}}, v_{\text{target}})$ on both stretch and compression error components to favor high-quality fitting of the comfort stretch range vs. the power stretch range (see Section 6.2.3 for the definitions). v_{comfort} is the target value at the maximum comfort stretch. As such, this weight is 1 within the comfort range, and decays under further stretching.

We found it hard to robustly incorporate bending energy into the objective function without hurting the quality of in-plane fitting. Therefore, we opted to fit stretch and compression only, and use bending data for post-hoc test of the results. This strategy gave us good qualitative fitting of bending in some cases, but the design of a good error metric for quantitative fitting appeared challenging. We provide a full discussion in Section 6.7.1.

Optimized parameters

The mechanical parameters of the yarn model are (Section 6.4): low-stretch stiffness k_{s1}, high-stretch stiffness k_{s2}, high-stretch onset $\bar{\varepsilon}_s$, bending stiffness k_b, outer contact radius R_1, outer contact stiffness k_{c1}, inner contact radius R_2, and inner contact stiffness k_{c2}.

Three of our parameters can be set according to a general heuristic and removed from the optimization procedure without negatively affecting our results. First, we set the high-stretch

stiffness k_{s2} as the yarn-stretch stiffness (which is provided as input) of the stiffest yarn in the fabric. Next, the inner contact acts as a non-penetration constraint, and we found that the fabric's mechanical response is barely sensitive to its actual parameter values. Therefore, we fix the inner contact stiffness k_{c2} to 1e3, and the inner contact radius R_2 to be 25% of a base radius R_{est}. We define R_{est} geometrically as the minimum required radius such that all yarn segments are in contact in the registered geometry.

After pruning these parameters, the final set of 5 optimized parameters is $\mathbf{p} = \{k_{s1}, \bar{\varepsilon}_s, k_b, R_1, k_{c1}\}$.

We also use R_{est} to define the optimization range of the outer contact radius, which we optimize in $R_1 \in [0.5\,R_{\text{est}}, 1.5\,R_{\text{est}}]$. Similarly, we define a base bending stiffness as the default stiffness in discrete rod simulation $k_{b,\text{base}} = k_{s2}\frac{R_{\text{est}}^2}{4}$, to optimize $k_b \in [10^{-3}k_{b,\text{base}}, 10k_{b,\text{base}}]$. We further fit $k_{s1} \in [10^{-3}k_{s2}, k_{s2}]$, $\bar{\varepsilon}_s \in [0.0, 0.15]$, $k_{c1} \in [10^{-2}, 10^2]$. k_{s1}, k_b, and k_{c1} are fit as log-space parameters. Finally, we specifically allow $\bar{\varepsilon}_s$ up to 0.20 for DKIN10, to mitigate extreme stiffening under stretching.

Optimization solvers

We found that our optimization landscape features numerous local minima. Therefore, we combine global optimization for initialization and local optimization for refining of optimized material parameters. Figure 6.11 gives an example of how local optimization can get stuck and result in strong errors.

For the global step, we found naïve grid sampling to be infeasible. Instead, we use 10 steps of swarm optimization [Bonyadi and Michalewicz 2017] with 64 particles. Each particle corresponds to a set of candidate material parameters and will compute stress and compression error using 7 samples per direction. For performance, we compute everything as parallel as possible; first, the rest-shape heuristic (Section 6.5.2) per particle, then the optimal compression optimizations (Section 6.5.3) per particle and direction for interpolation, and finally the individual stress samples to evaluate the errors. Our implementation uses a decaying inertia weight from 0.9 to 0.1 for each particle with heuristic repulsion enabled.

For local optimization we use COBYLA [Powell 1994a], starting from the best parameters found by the swarm optimization. Here, we first compute the two optimal compressions (for the purpose of quick evaluation via linear interpolation as described in Section 6.5.3) and then 20 simulations per direction for error evaluation.

6.6 Results

We collected results for 33 diverse fabrics, exhibiting a range of different knits and yarn compositions used in real-world industrial applications. The supplementary material accompanying the paper [Sperl et al. 2022][2] details the following fabric samples: all-needle fabrics made of polyester fiber (labeled A1 - A3); double-knit interlock fabric made with high-gauge polyester fiber (DKIN1 - DKIN7), spandex/polyester plated yarn (DKIN8), and low-gauge polyester fiber (DKIN9 - DKIN11); double-knit pique fabric made of spandex/polyester plated yarn (DKP); links fabric made of spandex/polyester plated yarn (L1 - L3); and single jersey fabric made of high-gauge multi-yarn (SJ1 - SJ12) and low-gauge polyester fiber (SJ13 - SJ15). Within each family of fabrics, the samples vary in yarn composition, gauge, and fabric finish. Figure 6.12 displays some rendered examples. This data set offers a unique level of access to industrial

[2]The paper and supplementary are openly available at `https://doi.org/10.1145/3528223.3530167`.

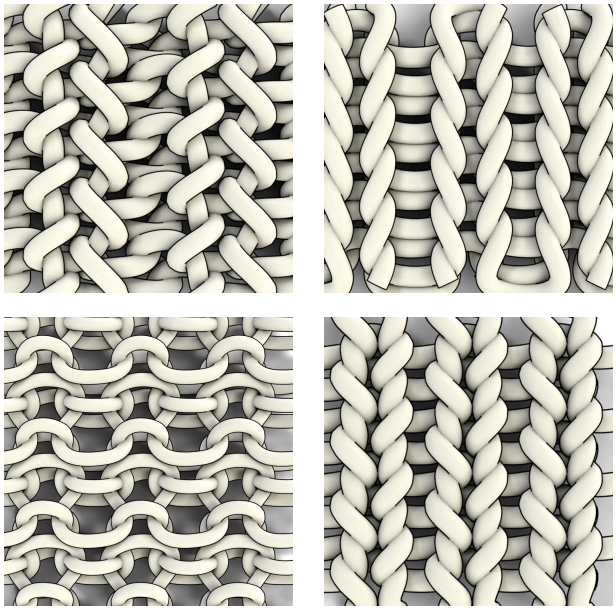


Figure 6.12: Some examples of the diversity of fabrics in our data set. The fabrics are (clockwise from top-left): all-needle (A1), double-knit interlock (DKIN9), single jersey (SJ9), and links (L3). Each is stretched in the weft direction for better visibility.

quality yarns and knitted fabrics; we consider the collection and publication of this data to be an important contribution of our work.

Figure 6.13 shows how well our thin-shell model reproduces the behavior of the fabrics in our database (which are color-coded based on knit family) after we optimize for its parameters. We display results for all 33 fabrics, measuring the percentage error relative to the average measured data in stretching force, orthogonal compression, and the pear-loop ratio for quantifying bending discussed in Section 6.2. We test against a wide range of fabrics with different material behaviors and find that, although some fabric families have a wider standard deviation of error (SJ vs A, for example), our shell model fits each fabric family with roughly the same magnitude of error.

The main result of our work is a system for producing a periodic yarn-level solver with the exact same topology as the original fabric, but with yarn-level parameters chosen such that the yarn-level simulation approximately matches the physical tests of the real fabric. Figure 6.14 shows the range of the estimated parameters for all 33 fabrics in the database. Figure 6.15 shows the fitting errors, indicating how well our yarn-level simulation is able to reproduce the fabrics. The leftmost plot shows the ability of our method to accurately model the behavior of materials within a "comfortable" range of stretching motions considered in fashion ergonomics, as defined in Section 6.2.3. We plot here the percentage differences in stress, so smaller errors imply a more accurate modeling of the material. As discussed in Section 6.5.3, our optimization emphasizes the accuracy within this "comfort" range more heavily, due to its importance in industrial applications. The next adjacent plot shows the accuracy of our yarn-level solver over the "power stretch" range of motion expected of an athletic garment, but normalized using the maximum target value in the comfort stretch range. Although many fabrics still produce remarkably accurate results for the full range (particularly the "A1-3" fabrics), the accuracy is lower overall due to our solver's intentional bias toward accuracy within the "comfort" region. The rightmost plot in Figure 6.15 shows how accurately our yarn-level model matches the orthogonal compression experienced by garments within the stretch tests. We discuss our

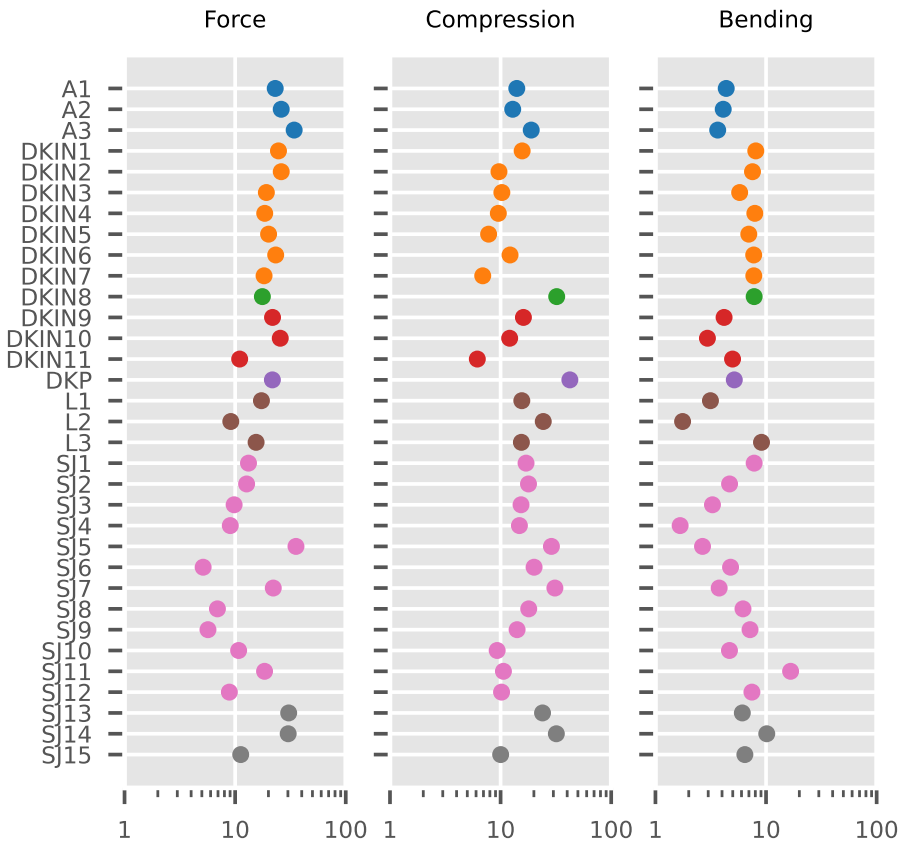


Figure 6.13: Overview over our thin-shell fitting results. The x-axis is error percentage, and the y-axis lists specific types of fabric in our database. Specifically, the error is relative to the average ground-truth measured stretching, compression, or bending datum per pattern as discussed in Section 6.3.1.

yarn simulation's ability to model fabric bending in Section 6.7. Overall, out of our data set of 33 fabrics, 24 of our yarn-level simulations are accurate to within 10% of the target data in the most important "comfort stretch" range of forces. The error tends to increase as we enter highly nonlinear behaviors with larger stretches with very large stretches, with one optimization (DKIN10) failing to converge.

We tested our optimization (with a periodic yarn simulation in the loop) with a combination of a local derivative-free solver and an optimizer based on particle swarms on a number of machines, the most powerful of which was an AMD EPYC 7662 server with 256 cores and 1TB of RAM; all optimizations were run simultaneously in parallel on different cores of the same machine. The local solver averaged 11m49s per fabric, with a minimum of 2m15s and maximum of 46m11s. The swarm-based solver took about twice as long, averaging 25m19s per solve, with a minimum time of 5m18s and maximum of 1h15m25s.

Figure 6.16 shows the effect of our biphasic yarn stiffness model discussed in Section 6.4.2. The naive elastic rod stiffness model works reasonably well for most fabrics, but it can fail to find suitable parameters, especially in cases with plated yarns made from multiple types of fabric. Our new two-phase yarn stiffness model, in contrast succeeds to map the behavior without a blow-up in fitting error. Figure 6.17 illustrates the importance of the yarn rest-shape estimation discussed in Section 6.5.2. The black bar in these figures denotes the transition from the "comfort" to the "power" range of stretches.

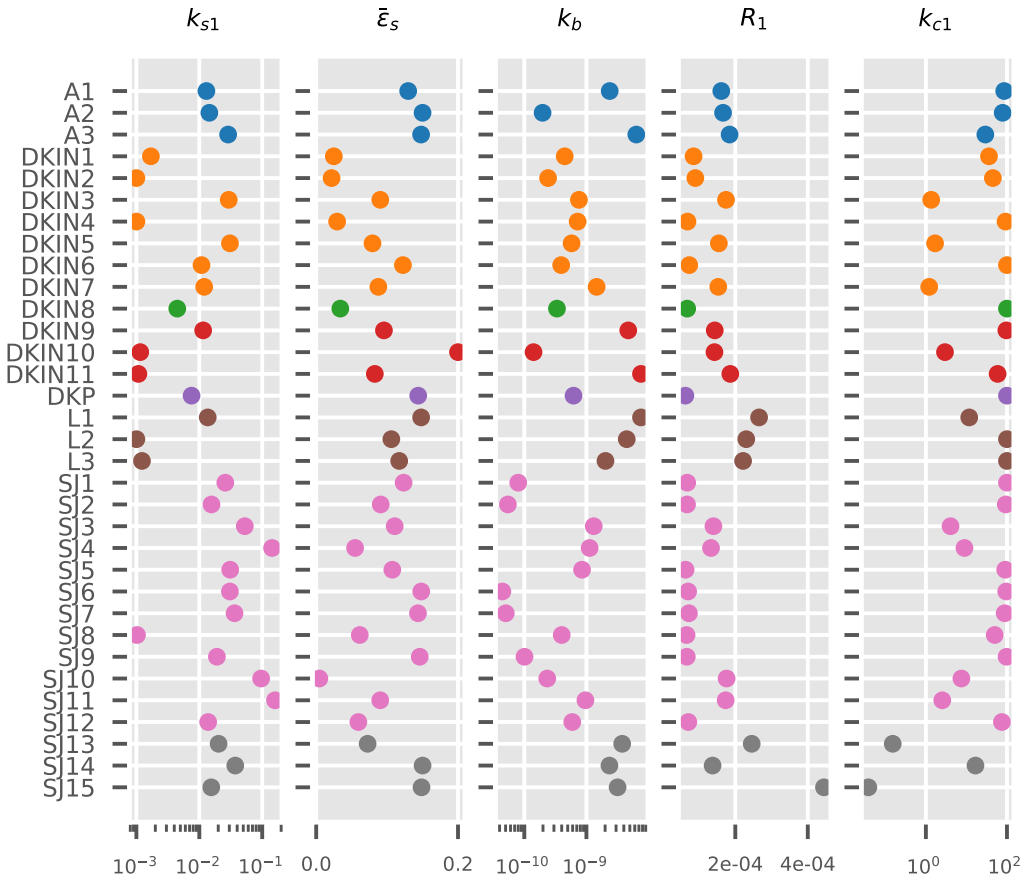


Figure 6.14: Estimated parameter values of the yarn-level model for all 33 fabrics in the test database. A linear (resp. log) scale is used when the parameters are estimated in linear (resp. log) scale.

The thin-shell simulator obviously approximates fabric at a completely different level than the periodic yarn-based model, so we should expect some differences in fitting error between the two. The thin-shell model is in a sense "closer to the data" in that we treat both the model and the real-world samples as geometric surfaces. The fabric samples are also much larger than individual stitches, and they occasionally exhibit features (like non-uniform deformation) that are not possible to model with our yarn-level solver. For these reasons, it is reasonable to expect that the thin-shell model might provide a more accurate fit than the yarn level model. We also note that the thin-shell model has almost twice as many degrees of freedom as our yarn-level solver (9 vs 5 DOFs), so it should also have more representation power.

6.7 Discussion and Future Work

This work marks the first demonstration that yarn-level simulation is capable of approximating the mechanical stretching response of real-world fabrics. We compiled a database from physical tests of several different knitted fabrics used in the textile industry, which spans several complex knit patterns, yarn compositions, and yarn coatings, resulting in diverse physical properties like stiffness, nonlinearity, and anisotropy.

We developed a system for optimizing yarn-level parameters in order to match this real-world data, and we offer a few novel extensions to make yarn-level simulation models more capable of replicating the biphasic stiffness behavior of plated yarns and real-world contact scenarios.

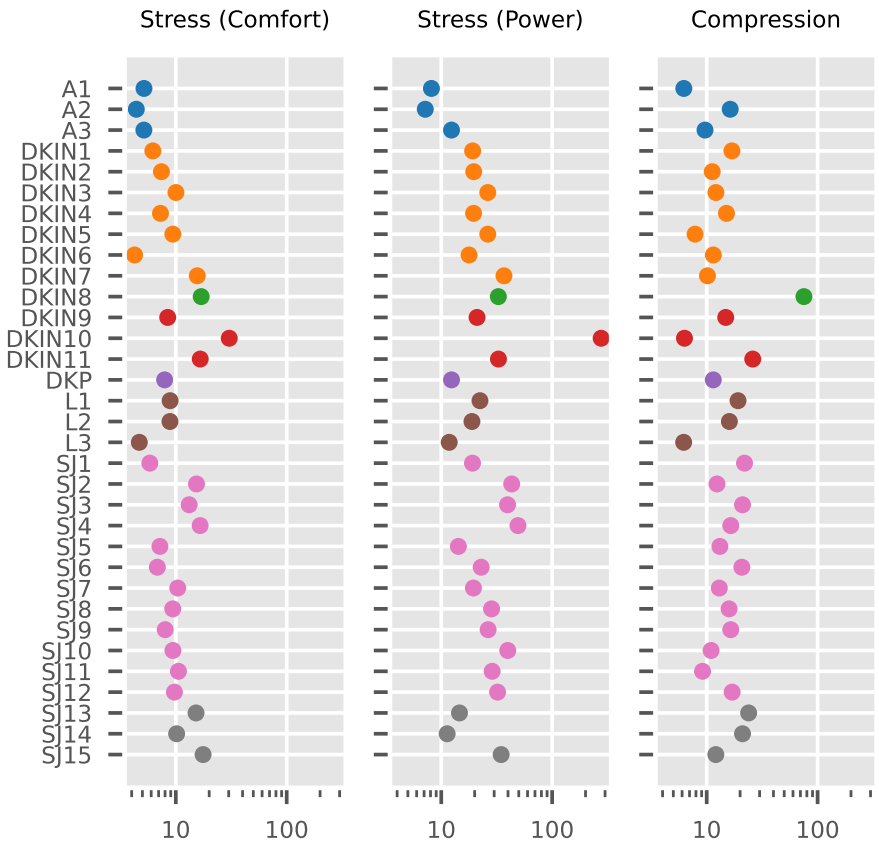


Figure 6.15: Overall ability of our yarn-level solver to reproduce the corresponding real-world behaviors of materials in our database. The x-axis is error percentage, and the y-axis lists specific types of fabric in our database. The error is relative to the maximum ground-truth datum in the comfort stretch per pattern (see Section 6.5.3).

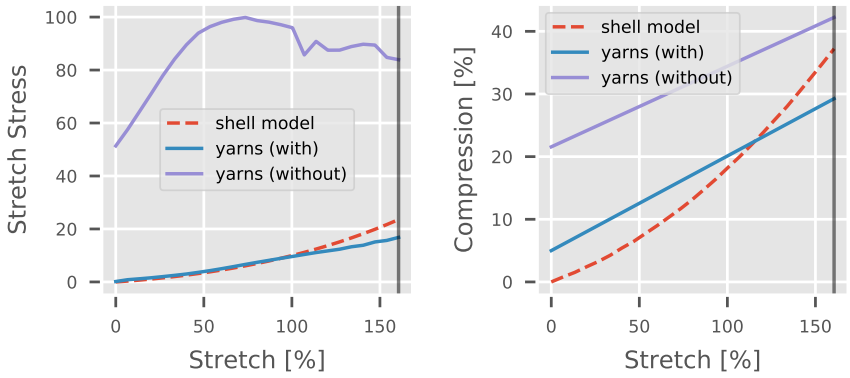


Figure 6.16: The performance of our optimization with and without our biphasic yarn-stretching model, on an all-needle fabric (A2). Omitting the more complex yarn model massively increases the overall stretching error and adds to the compression error (purple line).

We are releasing our data set to the public research community, in hopes that it inspires future work and acts as a potential benchmark for yarn-level cloth research. In particular, we hope that future scholars use our data set and results as an inspiration for potentially finding a reliable connection between yarn-level parameters and large-scale material behavior. Finding such a connection will address a long-standing problem in material-science and directly aid in the fabrication of novel fabrics.

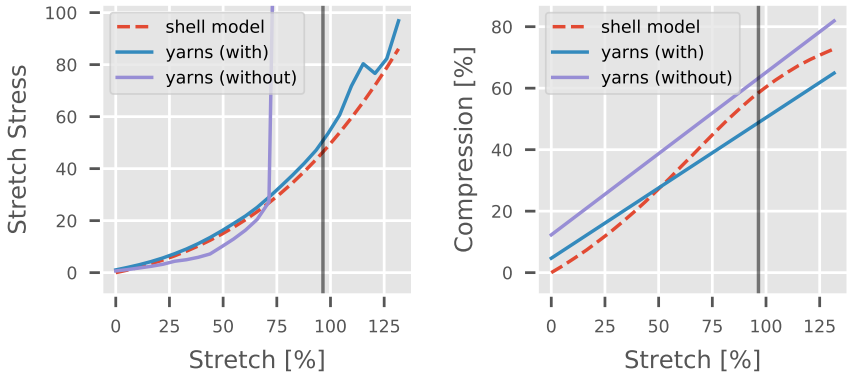


Figure 6.17: The performance of our optimization with and without our rest-shape heuristic, on an all-needle fabric (A2). Omitting the rest-shape heuristic causes the stretching error to blow up and adds to the compression error (purple line).

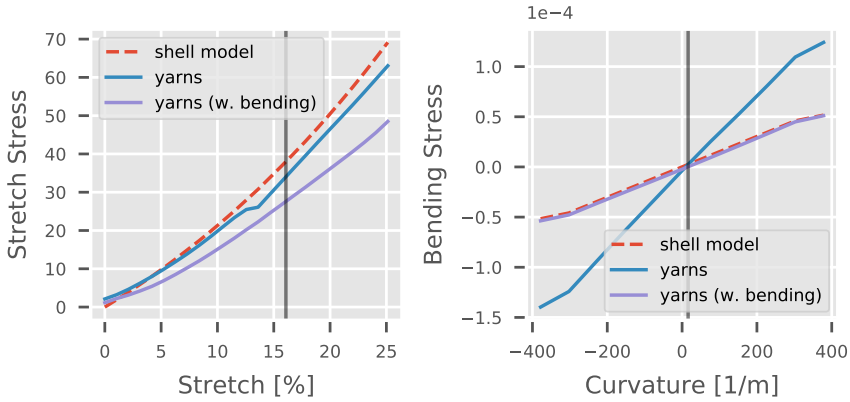


Figure 6.18: The performance of our yarn-level model with (blue) and without (purple) training on bending data in the objective function, on a double-knit interlock fabric (DKIN1). Including bending data makes the stretching error worse (left) but dramatically improves the bending error (right).

6.7.1 Bending

Our database includes data from mechanical stretching tests as well as bending tests. Although, the main effort of our work is to accurately reproduce the stretching tests, we can also consider matching the bending data. Unfortunately, matching the bending data is challenging or a number of practical and theoretical reasons. The bending data itself is difficult to capture using the processes we proposed — by fitting the fabric to a single curve and measuring curvature information from it. This single-curve assumption breaks down when the fabrics curl dramatically (Figure 6.7), or when sheared fabrics asymmetrically bulge out of plane. (Note that the tendency of fabrics to curl makes it difficult to measure orthogonal compression as well.)

We started our investigation with a yarn-level model that was optimized to match stretching data for a given fabric, as explained in the previous section. We wondered whether such a model could reproduce the *bending* behavior of that fabric, even though it was not trained on its bending data. Perhaps unsurprisingly, the fits to the bending data were not nearly as precise as the fits to the stretching data, but we did find that the bending behaviors actually match fairly well *qualitatively*. Essentially, our solver correctly exhibits weaker bending stiffness for weak fabrics, and stronger bending stiffness for strong fabrics.

Table 6.1: Effect on various fitting errors for all fabrics if we omit bending energy ("No bending" column) or include it ("Bending" column) in the yarn-level parameter optimization.

Regime	No bending		Bending	
	avg err	std dev	avg err	std dev
Stress Comfort	10.40%	±5.27%	15.05%	±5.12%
Stress Full	31.69%	±44.51%	41.41%	±50.39%
Compression	16.28%	±11.58%	24.32%	±15.03%
Bending	97.78%	±98.26%	64.03%	±39.37%

To quantify bending error, we considered a bending error component err_{bend} that measures the difference in bending stiffness k_θ between the thin-shell and yarn models. On the yarn model, we compute this stiffness through central-difference approximation of the second derivative of the yarn energy Ψ_{yarns} at ±20% of the curvature range in the physical test data. We considered measuring error on stiffness, and not on stress, because the thin-shell bending model does not account for curling effects, and it assumes a flat rest state. Some of the fabrics, e.g., jersey knits, showed evident curling on the yarn model, which manifests as an offset bending stress that cannot (and probably should not) be removed by the parameter optimization.

If we actually add the bending data into the objective function (6.11), our yarn-level solver matches bending behavior much better in several cases, but compromises stretching quality. (Figure 6.18 shows a representative example, and Table 6.1 compiles the total effect on all fabrics.) This trade-off behavior is somewhat unsurprising given the weighted least-squares form of our objective function, but we also wonder whether the bending metric can be improved. The bending stiffnesses of the fabrics in our database span several orders of magnitude, so it may be the case that the absolute difference from the input data is the wrong metric to use in the future.

6.7.2 Other Yarn Data

We use yarn density and average stiffness data measured from physical tests to inform our simulated yarn-models, as discussed in Section 6.2.1. We also considered using yarn mass and yarn-level stretch tests for improving fidelity; by combining the yarn's physical mass with the fabric's mass density, it is possible to estimate each yarn's rest length. However, when we compared the estimated values to the actual yarn lengths measured from the yarn geometry (Section 6.2.2), we found that the resulting pre-stretch values would require unreasonable stretch forces for stiff yarns such as polyester.

We also obtained individual yarns of all the tested fabrics, and we performed yarn-level stretch tests using a commercial device. With this data, we are able to estimate the stretching stiffness of individual yarns. Specifically, we linearly approximate the yarn stretch response in the 5 cN range, which is an upper bound of per-yarn forces under 5 N of swatch-level force. However, we do not use this yarn stretch stiffness alone in the simulation model (only as large-stretch and compression stiffness k_{s2}) for multiple reasons: (i) the fitting procedure is sensitive to experimental noise in the low-stretch regime, yielding an unrealistically large stiffness fit in the 5 cN range; (ii) many of the tested fabrics combine multiple yarn types, and the composite stiffness is a complex combination of the stiffness of individual yarn types; and (iii) production fabrics undergo finishes that could change the mechanical response of yarns.

6.7.3 Future Work

We presented two novel extensions to our yarn-level simulation: two-phase stiffness (for modeling plated yarns) and two-phase contact modeling. During this project, we also considered a number of other phenomenologically plausible extensions to the yarn level model. We could model the yarn's anisotropic cross-section [Montazeri et al. 2019], and extend that to an anisotropic bending model. Our current model ignores friction and hysteresis in models, but we can consider this in the future. We note that our yarn simulations are well conditioned without friction because they are subject to periodic boundary conditions, but in a non-periodic simulation, a fabric full of frictionless fibers may unravel. Adding friction or even cohesion to model "fuzz" may also increase the realism of our yarn contact and fabric modeling.

As noted in Section 6.7.1, some fabrics have a tendency to curl. We do not yet model this curly, non-flat rest shape in our thin-shell model. We could do this in the future by adding non-zero rest angles to the edges and solving for these additional degrees of freedom.

When fitting real-world materials with computer models, validation is an important task [Oberkampf and Roy 2010]. As we use the thin-shell model as an intermediate representation, the final estimated yarn-level model is not validated against the measured ground-truth data, and the final results may suffer higher error than the one reported. However, comparing full, non-periodic yarn-level simulations to the swatch-level mechanical tests comes with challenges. We would need to model the free edges of the swatches, including critical items such as yarn-yarn contact friction mentioned above. The error added by these modeling aspects would introduce high uncertainty to the validation. Romero et al. [2021] recently developed a protocol for validating rod simulation models, but it does not support the complex contact interactions of knitted yarns.

While our two-step procedure made optimization feasible by circumventing the scale of real fabric, it is based on derivative-free solvers. Here, future work includes estimating the derivative of our losses (and thus continuum stress) also with respect to the material parameters and then utilizing gradient-based optimization. This could drastically improve the convergence speed of our system and enable more research without the need for powerful servers. Differentiable yarn simulation can be a useful tool in computing these homogenized derivatives automatically.

Finally, we would like to consider more complicated fabrics in the future, notably those composed of multiple layers, or those fabricated by 3D weaving [Wu et al. 2020b] or warp knitting. These fabrics would challenge our geometry initialization procedure, because we would need to do more work to estimate the location of the yarns at the start of the simulation.

CHAPTER 7

Conclusion

7.1 Summary

In this thesis, we presented three methods based on homogenized yarn simulation. We first developed a method to analyze the elastostatic behavior of periodic yarn patterns through numeric simulation, imposing large-scale uniform deformation as periodic boundary conditions, and reading out averaged energies or stresses. With this, we were able to predict the large-scale elastic behavior of yarn-based fabric and fit a regularized continuum model to use in thin-shell cloth solvers. The resulting thin-shell simulations were able to match the quality of full yarn-level simulation at a fraction of the cost, and they scaled up favorably to large garments. Next, we introduced an algorithm to animate yarn detail that adapts to the underlying cloth deformation in real time. To do so, we also stored the local displacements of the periodic yarn patterns, sampling a grid of large-scale deformations, and interpolating the data on the GPU with efficient linearized bending and tunable buckling. In combination with the thin-shell simulations, we were able to animate garments with millions of yarn vertices in real time and with mechanics-aware yarn detail that properly tightens under stretching. Finally, as a first step towards predictive yarn simulation, we used homogenization as a tool to efficiently fit yarn models to physical stretching and bending measurements of real fabric used in production. We also discussed modifications to yarn models and heuristics for yarn rest shapes.

These three methods can even be combined into one system. We could fit predictive yarn models to real data, use them to learn a continuum model for efficient mesh-based simulation, and add the missing yarn detail with our real-time displacement technique. In fact, Figure 1.3 in the introduction was created in exactly this way. In this thesis, we thus developed a prototype of a complete system for efficient yarn-level cloth that is based on homogenization and can reproduce some of the rich elastic and visual properties of real knitted fabrics.

7.2 Limitations & Future Work

Buckling We have seen that, as yarn patterns are compressed in-plane, the yarns may unpredictably buckle into different configurations. These configurations are not only geometrically different but also exhibit different levels of stress. We found it necessary to deal with buckling in each of our methods. For our continuum materials we developed strongly regularized fitting

against spurious local minima that caused unstable cloth simulations. In the detail animation, we limit the lookup to less-compressed deformations for smoother interpolation. While fitting yarn parameters to real data, we subsampled and linearly interpolated the optimal orthogonal compression, which also served as a regularizer against noisy homogenized stresses. It is unclear which buckling frequencies should be modeled on which scale. We arbitrarily pick a small number of periodic repeats and expect low frequencies to be handled on the macroscale. In real fabric, buckled yarn configurations also depend on the history of deformation, but the problem is also mitigated by yarn friction, both of which we leave as future work.

Friction and Plasticity Our results show that even without yarn friction we manage to reproduce several key qualities of yarn-based fabric. However, friction between yarns is arguably the main cause for cloth hysteresis and may also strongly affect cloth damping. As such, integrating recent models for differentiable yarn friction [Li et al. 2021; Gong et al. 2022] in our pipeline would be a natural and interesting next step. This would involve homogenizing path-dependent plasticity for hysteresis, or strain-rate-dependent friction forces for dynamic damping. Both of these topics require additional research to develop suitable techniques for homogenization and to deal with the increased dimensionality of the deformation space. Moreover, yarn friction could also play a more dramatic role in stabilizing the fabric at rest, where we currently modify the yarn rest shapes using heuristics instead.

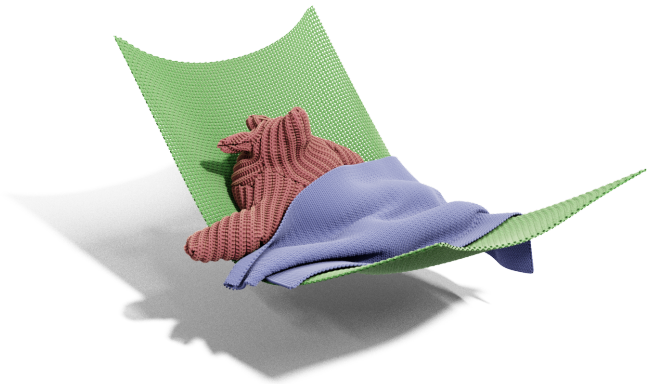
Non-periodic Geometry The periodic boundary conditions we use for homogenization fundamentally limit our methods to periodic geometry. As such, we currently cannot accurately model localized effects such as tearing or pulling on individual yarns, or seams between different regions. One way to solve this is to integrate our method with blended-in localized yarn simulations similar to Casafranca et al. [2020]. A homogenized treatment of non-periodic geometry comes with several issues. First, the periodic boundary conditions would have to be relaxed to allow such geometry. Because periodic conditions are a specialization of the average-deformation constraint, this could be solved with a more general averaging constraint. Second, to accurately model seams, one might need to solve the combinatorial explosion that comes with connecting different aperiodic regions.

Neural Methods Our methods fit relatively simple functions to data extracted from small simulations. It would be interesting to design an approach that instead takes several big simulations and takes local snapshots to extract similar data as before. Such an approach would also lend itself to the use of neural networks and together could attempt to *automatically* find a good representation of fabrics with aperiodic geometry, seams, or more — both in terms of a continuum material and detail interpolation.

Knitting and Weaving Our work could be used to investigate more types of yarn-based textiles, such as warp-knitted or 3D woven fabrics [Wu et al. 2020b]. It would be interesting to integrate our approaches into mesh-based machine-knitting approaches [Yuksel et al. 2012; Jones et al. 2021]; homogenized pattern-based materials could improve the predictiveness of the mesh-based relaxation step in stitch meshes, or aid in physics-based garment authoring in general, although this may require extending our method to non-rectangular tiles or seams. Liu et al. [2021] propose inverse-designed knitted garments with controlled elasticity by mixing a stiff and a compliant yarn. Our pipeline for predictive yarn models based on real data could enable exciting research in inverse mechanical designing of yarn fabrics.

Other Materials and Phenomena On the other hand, our thin-shell homogenization procedure is not limited to yarn-level cloth and could be useful for animating other complicated multi-physics materials like layered quilts, layered elastic materials [van Rees et al. 2017], skin tissue, and layered deployable shells [Guseinov et al. 2017]. Outside of computer graphics, our technique may be applicable to the homogenization of composite materials, micro-structured shells, and finite-element simulations. In fact, Rodriguez et al. [2022] have applied our method to inverse design of shell-based bending-active structures. We believe that there is a lot of untapped potential in extending our methods to different phenomena. This could include different dimensions, such as thick shells including transversal shearing, or one-dimensional phenomena such as strands of hair. Homogenization could be applied to the animation of further materials like granular materials, fluids, or even multiphase materials.

Thank you for reading.



Bibliography

Baptiste Angles, Daniel Rebain, Miles Macklin, Brian Wyvill, Loic Barthe, JP Lewis, Javier Von Der Pahlen, Shahram Izadi, Julien Valentin, Sofien Bouaziz, et al. 2019. VIPER: Volume invariant position-based elastic rods. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 2 (2019).

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 43–54.

Jan Bender, Matthias Müller, and Miles Macklin. 2015. Position-Based Simulation Methods in Computer Graphics.. In *Eurographics (tutorials)*. 8.

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

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete elastic rods. *ACM Transactions on Graphics (TOG)* 27, 3 (2008).

MA Bessa, R Bostanabad, Z Liu, A Hu, Daniel W Apley, C Brinson, Wei Chen, and Wing Kam Liu. 2017. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Computer Methods in Applied Mechanics and Engineering* 320 (2017), 633–667.

Kiran S. Bhat, Christopher D. Twigg, Jessica K. Hodgins, Pradeep K. Khosla, Zoran Popović, and Steven M. Seitz. 2003. Estimating Cloth Simulation Parameters from Video. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Goslar, DEU, 37–51.

Bernd Bickel, Moritz Bächer, Miguel A Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Transactions on Graphics (TOG)* 29, 4 (2010).

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, 2 (2016), 191–253.

Jim Blinn. 1996. Consider the lowly 2 x 2 matrix. *IEEE Computer Graphics and Applications* 16, 2 (1996), 82–88.

Javier Bonet and Richard D. Wood. 2008. *Nonlinear Continuum Mechanics for Finite Element Analysis* (2 ed.). Cambridge University Press.

Mohammad Reza Bonyadi and Zbigniew Michalewicz. 2017. Particle swarm optimization for single objective continuous space problems: a review. *Evolutionary computation* 25, 1 (2017), 1–54.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014).

Katherine L. Bouman, Bei Xiao, Peter Battaglia, and William T. Freeman. 2013. Estimating the Material Properties of Fabric from Video. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

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.

Thomas Buffet, Damien Rohmer, Loic Barthe, Laurence Boissieux, and Marie-Paule Cani. 2019. Implicit untangling: A robust solution for modeling layered clothing. *ACM Transactions on Graphics (TOG)* 38, 4 (2019).

Ralph E Carlson and Frederick N Fritsch. 1989. An algorithm for monotone piecewise bicubic interpolation. *SIAM J. Numer. Anal.* 26, 1 (1989), 230–238.

Juan J Casafranca, Gabriel Cirio, Alejandro Rodríguez, Eder Miguel, and Miguel A Otaduy. 2020. Mixing yarns and triangles in cloth simulation. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 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).

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).

Hsiao-Yu Chen, Arnav Sastry, Wim M van Rees, and Etienne Vouga. 2018b. Physical simulation of environmentally induced thin shell deformation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018).

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

Zhen Chen, Hsiao-Yu Chen, Danny M Kaufman, Mélina Skouras, and Etienne Vouga. 2021. Fine Wrinkling on Coarsely Meshed Thin Shells. *ACM Transactions on Graphics (TOG)* 40, 5 (2021).

Nuttapong Chentanez, Miles Macklin, Matthias Müller, Stefan Jeschke, and Tae-Yong Kim. 2020. Cloth and skin deformation with a triangle mesh based convolutional neural network. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library.

Andrew Choi, Dezhong Tong, Mohammad K Jawed, and Jungseock Joo. 2021. Implicit contact model for discrete elastic rods in knot tying. *Journal of Applied Mechanics* 88, 5 (2021).

Ka-Fai Choi and Tien-Yu Lo. 2003. An energy model of plain knitted fabric. *Textile research journal* 73, 8 (2003), 739–748.

Kwang-Jin Choi and Hyeong-Seok Ko. 2005. Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses*. ACM, 1.

Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A Otaduy. 2014. Yarn-level simulation of woven cloth. *ACM Transactions on Graphics (TOG)* 33, 6 (2014).

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*. ACM, 55–61.

Gabriel Cirio, Jorge Lopez-Moreno, and Miguel A Otaduy. 2016. Yarn-level cloth simulation with sliding persistent contacts. *IEEE Transactions on Visualization and Computer Graphics* 23, 2 (2016), 1152–1162.

Gabriel Cirio, Ante Qu, George Drettakis, Eitan Grinspun, and Changxi Zheng. 2018. Multi-scale simulation of nonlinear thin-shell sound with wave turbulence. *ACM Transactions on Graphics (TOG)* 37, 4 (2018).

Gabriel Cirio and Alejandro Rodríguez. 2022. True Seams: Modeling Seams in Digital Garments. *ACM Transactions on Graphics (TOG)* 41, 4 (2022).

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*. ACM.

Blender Online Community. 2020. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. `http://www.blender.org`

Fabrizio Corda, Jean-Marc Thiery, Marco Livesu, Enrico Puppo, Tamy Boubekeur, and Riccardo Scateni. 2020. Real-Time Deformation with Coupled Cages and Skeletons. In *Computer Graphics Forum*. Wiley Online Library.

Gilles Daviet. 2020. Simple and scalable frictional contacts for thin nodal objects. *ACM Transactions on Graphics (TOG)* 39, 4 (2020).

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.

Charles-Alban Deledalle, Loic Denis, Sonia Tabti, and Florence Tupin. 2017. Closed-form expressions of the eigen decomposition of 2 x 2 and 3 x 3 Hermitian matrices. (2017).

Alexandre Derouet-Jourdan, Florence Bertails-Descoubes, Gilles Daviet, and Joëlle Thollot. 2013. Inverse dynamic hair modeling with frictional contact. *ACM Transactions on Graphics (TOG)* 32, 6 (2013).

Alexandre Derouet-Jourdan, Florence Bertails-Descoubes, and Joëlle Thollot. 2010. Stable inverse dynamic curves. *ACM Transactions on Graphics (TOG)* 29, 6 (2010).

Tien Dung Dinh, Oliver Weeger, Sawako Kaijima, and S-K Yeung. 2018. Prediction of mechanical properties of knitted fabrics under tensile and shear loading: Mesoscale analysis using representative unit cells and its validation. *Composites Part B: Engineering* 148 (2018), 81–92.

Manfredo P Do Carmo. 2016. *Differential geometry of curves and surfaces: revised and updated second edition.* Courier Dover Publications.

Petros Faloutsos, Michiel Van De Panne, and Demetri Terzopoulos. 1997. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics* 3, 3 (1997), 201–214.

Yun Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2019. A multi-scale model for coupling strands with shear-dependent liquid. *ACM Transactions on Graphics (TOG)* 38, 6 (2019).

Yun Raymond Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2018. A multi-scale model for simulating liquid-fabric interactions. *ACM Transactions on Graphics (TOG)* 37, 4 (2018).

Yun Raymond Fei, Henrique Teles Maia, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2017. A multi-scale model for simulating liquid-hair interactions. *ACM Transactions on Graphics (TOG)* 36, 4 (2017).

Sebastian Fillep, Julia Mergheim, and Paul Steinmann. 2017. Towards an efficient two-scale approach to model technical textiles. *Computational Mechanics* 59, 3 (2017), 385–401.

Frederick N Fritsch and Ralph E Carlson. 1980. Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal.* 17, 2 (1980), 238–246.

Lin Gao, Yu-Kun Lai, Jie Yang, Zhang Ling-Xiao, Shihong Xia, and Leif Kobbelt. 2019. Sparse data driven mesh deformation. *IEEE transactions on visualization and computer graphics* (2019).

Marc GD Geers, Erica WC Coenen, and Varvara G Kouznetsova. 2007. Multi-scale computational homogenization of structured thin sheets. *Modelling and Simulation in Materials Science and Engineering* 15, 4 (2007), S393.

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

Deshan Gong, Zhanxing Zhu, Andrew J Bulpitt, and He Wang. 2022. Fine-grained differentiable physics: a yarn-level model for fabrics. *arXiv preprint arXiv:2202.00504* (2022).

Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. 2006. Computing discrete shape operators on general meshes. In *Computer Graphics Forum*, Vol. 25. Wiley Online Library, 547–556.

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*. Eurographics Association, 62–67.

Darya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. 2016. BRDF representation and acquisition. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 625–650.

JoséMiranda Guedes and Noboru Kikuchi. 1990. Preprocessing and postprocessing for materials based on the homogenization method with adaptive finite element methods. *Computer methods in applied mechanics and engineering* 83, 2 (1990), 143–198.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. `http://eigen.tuxfamily.org`

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).

Runbo Guo, Jenny Lin, Vidya Narayanan, and James McCann. 2020. Representing crochet with stitch meshes. In *Symposium on Computational Fabrication*. 1–8.

Ruslan Guseinov, Eder Miguel, and Bernd Bickel. 2017. CurveUps: Shaping objects from flat plates with tension-actuated curvature. *ACM Transactions on Graphics (TOG)* 36, 4 (2017).

Sunil Hadap. 2006. Oriented strands: dynamics of stiff multi-body system. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 91–100.

Sunil Hadap, E Bongarter, Pascal Volino, and Nadia Magnenat-Thalmann. 1999. Animating wrinkles on clothes. In *Proceedings Visualization'99 (Cat. No. 99CB37067)*. IEEE.

Rodney Hill. 1963. Elastic properties of reinforced solids: some theoretical principles. *Journal of the Mechanics and Physics of Solids* 11, 5 (1963), 357–372.

Jonathan Hoffman, Matt Kuruc, Junyi Ling, Alex Marino, George Nguyen, and Sasha Ouellet. 2020. Hypertextural Garments on Pixar's Soul. In *ACM SIGGRAPH 2020 Talks*. ACM, Article 75.

J. Huang, Y. Tong, K. Zhou, H. Bao, and M. Desbrun. 2011. Interactive Shape Interpolation through Controllable Dynamic Deformation. *IEEE Transactions on Visualization and Computer Graphics* 17, 7 (July 2011), 983–992.

Doug L James. 2020. Phong deformation: a better C0 interpolant for embedded deformation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020).

Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)* 36, 4 (2017).

Ning Jin, Yilin Zhu, Zhenglin Geng, and Ronald Fedkiw. 2020. A Pixel-Based Framework for Data-Driven Clothing. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library.

Benjamin Jones, Yuxuan Mei, Haisen Zhao, Taylor Gotfrid, Jennifer Mankoff, and Adriana Schulz. 2021. Computational Design of Knit Templates. *ACM Transactions on Graphics (TOG)* 41, 2 (2021).

Jonathan M Kaldor, Doug L James, and Steve Marschner. 2008. Simulating knitted cloth at the yarn level. *ACM Transactions on Graphics (TOG)* 27, 3 (2008).

Jonathan M Kaldor, Doug L James, and Steve Marschner. 2010. Efficient yarn-based cloth with adaptive contact linearization. *ACM Transactions on Graphics (TOG)* 29, 4 (2010).

Levi Kapllani, Chelsea Amanatides, Genevieve Dion, Vadim Shapiro, and David E. Breen. 2021. TopoKnit: A Process-Oriented Representation for Modeling the Topology of Yarns in Weft-Knitted Textiles. *Graphical Models* 118 (2021), 101114.

Alexandre Kaspar, Kui Wu, Yiyue Luo, Liane Makatura, and Wojciech Matusik. 2021. Knit sketching: from cut & sew patterns to machine-knit garments. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).

Ladislav Kavan, Dan Gerszewski, Adam W Bargteil, and Peter-Pike Sloan. 2011. Physics-inspired upsampling for cloth simulation in games. In *ACM SIGGRAPH 2011 papers*.

S. Kawabata. 1980. *The standardization and analysis of hand evaluation.* Textile Machinery Soc. Japan.

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

Josef Kiendl, Ming-Chen Hsu, Michael CH Wu, and Alessandro Reali. 2015. Isogeometric Kirchhoff–Love shell formulations for general hyperelastic materials. *Computer Methods in Applied Mechanics and Engineering* 291 (2015), 280–303.

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

Michael James King, P Jearanaisilawong, and S Socrate. 2005. A continuum constitutive model for the mechanical behavior of woven fabrics. *International journal of solids and structures* 42, 13 (2005), 3867–3896.

Yuki Koyama, Kenshi Takayama, Nobuyuki Umetani, and Takeo Igarashi. 2012. Real-time example-based elastic deformation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Tassilo Kugelstadt and Elmar Schömer. 2016. Position and orientation based Cosserat rods.. In *Symposium on Computer Animation*.

Lei Lan, Guanqun Ma, Yin Yang, Changxi Zheng, Minchen Li, and Chenfanfu Jiang. 2022. Penetration-Free Projective Dynamics on the GPU. *ACM Transactions on Graphics (TOG)* 41, 4, Article 69 (2022).

BA Le, Julien Yvonnet, and Q-C He. 2015. Computational homogenization of nonlinear elastic materials using neural networks. *Internat. J. Numer. Methods Engrg.* 104, 12 (2015), 1061–1084.

Jonathan Leaf, Rundong Wu, Eston Schweickart, Doug L James, and Steve Marschner. 2018. Interactive design of periodic yarn-level cloth patterns. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 202.

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 Transactions on Graphics (TOG)* 37, 4 (2018).

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection-and Inversion-Free, Large-Deformation Dynamics. *ACM Transactions on Graphics (TOG)* 39, 4, Article 49 (jul 2020).

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Transactions on Graphics (SIGGRAPH)* 40, 4 (2021).

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2022a. DiffCloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics (TOG)* (2022).

Yue Li, Juan Montes, Bernhard Thomaszewski, and Stelian Coros. 2022b. Programmable Digital Weaves. *IEEE Robotics and Automation Letters* 7, 2 (2022).

Junbang Liang, Ming C. Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Dani Liu, Seid Koric, and Antonios Kontsos. 2019. A multiscale homogenization approach for architectured knitted textiles. *Journal of Applied Mechanics* 86, 11 (2019).

Ligang Liu, Ariel Shamir, Charlie CL Wang, and Emily Whiting. 2014. 3D printing oriented design: geometry and optimization.. In *SIGGRAPH ASIA Courses*.

Zishun Liu, Xingjian Han, Yuchen Zhang, Xiangjia Chen, Yu-Kun Lai, Eugeni L Doubrovski, Emily Whiting, and Charlie CL Wang. 2021. Knitting 4D garments with elasticity controlled for body motion. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).

D. S. Lyle. 1977. *Performance of Textiles*. John Wiley & Sons, New York.

Wan-Chun Ma, Andrew Jones, Jen-Yuan Chiang, Tim Hawkins, Sune Frederiksen, Pieter Peers, Marko Vukovic, Ming Ouhyoung, and Paul Debevec. 2008. Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Transactions on Graphics (TOG)* 27, 5 (2008).

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*.

Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. 2010. Unified simulation of elastic rods, shells, and solids. *ACM Transactions on Graphics (TOG)* 29, 4 (2010).

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. In *ACM SIGGRAPH 2011 papers*.

Karel Matouš, Marc GD Geers, Varvara G Kouznetsova, and Andrew Gillman. 2017. A review of predictive nonlinear theories for multiscale modeling of heterogeneous materials. *J. Comput. Phys.* 330 (2017), 192–220.

Markus Mehnert, Sebastian Fillep, Julia Mergheim, and Paul Steinmann. 2015. Computational Homogenization of Micromechanically Resolved Textile Materials. *PAMM* 15, 1 (2015), 461–462.

Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A Otaduy, and Steve Marschner. 2012. Data-driven estimation of cloth simulation models. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 519–528.

Eder Miguel, David Miraut, and Miguel A Otaduy. 2016. Modeling and Estimation of Energy-Based Hyperelastic Objects. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 385–396.

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 Transactions on Graphics (TOG)* 32, 6 (2013).

Zahra Montazeri, Søren B Gammelmark, Shuang Zhao, and Henrik Wann Jensen. 2020. A practical ply-based appearance model of woven fabrics. *ACM Transactions on Graphics (TOG)* 39, 6 (2020).

Zahra Montazeri, Chang Xiao, Yun Fei, Changxi Zheng, and Shuang Zhao. 2019. Mechanics-Aware Modeling of Cloth Appearance. *IEEE transactions on visualization and computer graphics* 27, 1 (2019), 137–150.

Matthias Müller and Nuttapong Chentanez. 2010. Wrinkle Meshes.. In *Symposium on Computer Animation*. Madrid, Spain.

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

Matthias Müller, Leonard McMillan, Julie Dorsey, and Robert Jagnow. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Computer Animation and Simulation 2001*. Springer, 113–124.

Matthias Muller, Matthias Teschner, and Markus Gross. 2004. Physically-based simulation of objects represented by surface meshes. In *Proceedings Computer Graphics International, 2004.* IEEE.

Georges Nader, Yu Han Quek, Pei Zhi Chia, Oliver Weeger, and Sai-Kit Yeung. 2021. KnitKit: A flexible system for machine knitting of customizable textiles. *ACM Transactions on Graphics (TOG)* (2021).

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

Rahul Narain, Armin Samii, and James F O'Brien. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012).

Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James McCann. 2018. Automatic machine knitting of 3D meshes. *ACM Transactions on Graphics (TOG)* 37, 3 (2018).

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

W Oberkampf and Christopher Roy. 2010. *Verification and Validation in Scientific Computing*. Cambridge University Press.

Matthew Overby, George E Brown, Jie Li, and Rahul Narain. 2017. ADMM ⊇ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017).

Dinesh K Pai. 2002. Strands: Interactive simulation of thin solids using cosserat models. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 347–352.

Ethan M Parsons, Michael J King, and Simona Socrate. 2013. Modeling yarn slip in woven fabric at the continuum level: Simulations of ballistic impact. *Journal of the Mechanics and Physics of Solids* 61, 1 (2013), 265–292.

Ethan M Parsons, Tusit Weerasooriya, Sai Sarva, and Simona Socrate. 2010. Impact of woven fabric: Experiments and mesostructure-based continuum-level simulations. *Journal of the Mechanics and Physics of Solids* 58, 11 (2010), 1995–2021.

F. T. Peirce. 1930. The "Handle" of Cloth as a Measurable Quantity. *Journal of the Textile Institute Transactions* 21, 9 (1930), T377–T416.

Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A Canabal, Robert Sumner, and Miguel A Otaduy. 2015. Design and fabrication of flexible rod meshes. *ACM Transactions on Graphics (TOG)* 34, 4 (2015).

José M Pizana, Alejandro Rodríguez, Gabriel Cirio, and Miguel A Otaduy. 2020. A Bending Model for Nodal Discretizations of Yarn-Level Cloth. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 181–189.

Samuel Poincloux, Mokhtar Adda-Bedia, and Frédéric Lechenault. 2018. Geometry and elasticity of a knitted fabric. *Physical Review X* 8, 2 (2018), 021075.

Serban D Porumbescu, Brian Budge, Louis Feng, and Kenneth I Joy. 2005. Shell maps. *ACM Transactions on Graphics (TOG)* 24, 3 (2005).

Michael JD Powell. 1994a. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*. Springer, 51–67.

M. J. D. Powell. 1994b. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*. Springer Netherlands, Dordrecht, 51–67.

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

Abdullah Haroon Rasheed, Victor Romero, Florence Bertails-Descoubes, Stefanie Wuhrer, Jean-Sebastien Franco, and Arnaud Lazarus. 2020. Learning to Measure the Static Friction Coefficient in Cloth Contact. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yannick Remion, Jean-Michel Nourrit, and Didier Gillard. 1999. Dynamic Animation Of Spline Like Objects. In *Proc. of WSCG*.

J Renard and MF Marmonier. 1987. Study of damage initiation in the matrix of a composite material by an homogenization method. *Rech. Aerosp.* 24 (1987), 43–51.

Emmanuel Rodriguez, Georges-Pierre Bonneau, Stefanie Hahmann, and Mélina Skouras. 2022. Computational Design of Laser-Cut Bending-Active Structures. *Computer-Aided Design* (2022).

Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. 2010. Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Transactions on Graphics (TOG)* 29, 6 (2010).

Victor Romero, Mickaël Ly, Abdullah Haroon Rasheed, Raphaël Charrondière, Arnaud Lazarus, Sébastien Neukirch, and Florence Bertails-Descoubes. 2021. Physical Validation of Simulators in Computer Graphics: A New Framework Dedicated to Slender Elastic Structures and Frictional Contact. *ACM Transactions on Graphics (TOG)* 40, 4, Article 66 (2021).

Wulf Rossmann. 2006. *Lie Groups: An Introduction Through Linear Groups*. Oxford University Press.

Nadine Abu Rumman and Marco Fratarcangeli. 2016. State of the Art in Skinning Techniques for Articulated Deformable Characters.. In *VISIGRAPP (1: GRAPP)*. 200–212.

Tom F. H. Runia, Kirill Gavrilyuk, Cees G. M. Snoek, and Arnold W. M. Smeulders. 2020. Cloth in the Wind: A Case Study of Physical Measurement Through Simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rosa M Sánchez-Banderas, Alejandro Rodríguez, Héctor Barreiro, and Miguel A Otaduy. 2020. Robust eulerian-on-lagrangian rods. *ACM Transactions on Graphics (TOG)* 39, 4 (2020).

Igor Santesteban, Miguel A Otaduy, and Dan Casas. 2019. Learning-Based Animation of Clothing for Virtual Try-On. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library.

Nikolas Schmitt, Martin Knuth, Jan Bender, and Arjan Kuijper. 2013. Multilevel Cloth Simulation using GPU Surface Sampling. *VRIPHYS* 13 (2013).

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).

Christian Schumacher, Steve Marschner, Markus Cross, and Bernhard Thomaszewski. 2018. Mechanical characterization of structured sheet materials. *ACM Transactions on Graphics (TOG)* 37, 4 (2018).

Christian Schumacher, Bernhard Thomaszewski, Stelian Coros, Sebastian Martin, Robert Sumner, and Markus Gross. 2012. Efficient simulation of example-based materials. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*. Citeseer.

Andrew Selle, Michael Lentine, and Ronald Fedkiw. 2008. A Mass Spring Model for Hair Simulation. In *ACM SIGGRAPH 2008 Papers (SIGGRAPH '08)*. Article 64.

Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ronald Fedkiw. 2007. Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*.

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Transactions on Graphics (TOG)* 37, 2, Article 12 (2018).

Carlota Soler, Tobias Martin, and Olga Sorkine-Hornung. 2018. Cosserat rods with projective dynamics. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library.

Georg Sperl, Rahul Narain, and Chris Wojtan. 2020. Homogenized Yarn-Level Cloth. *ACM Transactions on Graphics (TOG)* 39, 4 (2020).

Georg Sperl, Rahul Narain, and Chris Wojtan. 2021. Mechanics-Aware Deformation of Yarn Pattern Geometry. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).

Georg Sperl, Rosa M. Sánchez-Banderas, Manwen Li, Chris Wojtan, and Miguel A. Otaduy. 2022. Estimation of Yarn-Level Simulation Models for Production Fabrics. *ACM Transactions on Graphics (TOG)* 41, 4 (2022).

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

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

Bernhard Thomaszewski, Markus Wacker, Wolfgang Straßer, Etienne Lyard, C. Luible, Pascal Volino, M. Kasap, V. Muggeo, and Nadia Magnenat-Thalmann. 2007. Advanced Topics in Virtual Garment Simulation. In *Eurographics 2007 - Tutorials*, Karol Myszkowski and Vlastimil Havran (Eds.). The Eurographics Association.

Dezhong Tong, Andrew Choi, Jungseock Joo, and M Khalid Jawed. 2022. A Fully Implicit Method for Robust Frictional Contact Handling in Elastic Rods. *arXiv preprint arXiv:2205.10309* (2022).

Nobuyuki Umetani, Ryan Schmidt, and Jos Stam. 2014. Position-based elastic rods. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association.

O Van der Sluis, PJG Schreurs, WAM Brekelmans, and HEH Meijer. 2000. Overall behaviour of heterogeneous elastoviscoplastic materials: effect of microstructural modelling. *Mechanics of Materials* 32, 8 (2000), 449–462.

Wim M van Rees, Etienne Vouga, and Lakshminarayanan Mahadevan. 2017. Growth patterns for shape-shifting elastic bilayers. *Proceedings of the National Academy of Sciences* 114, 44 (2017), 11597–11602.

Raquel Vidaurre, Igor Santesteban, Elena Garces, and Dan Casas. 2020. Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library.

Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. 2009. A Simple Approach to Nonlinear Tensile Stiffness for Accurate Cloth Simulation. *ACM Transactions on Graphics (TOG)* 28, 4, Article 105 (2009).

Paras Wadekar, Vignesh Perumal, Genevieve Dion, Antonios Kontsos, and David Breen. 2020. An optimized yarn-level geometric model for Finite Element Analysis of weft-knitted fabrics. *Computer Aided Geometric Design* 80 (2020), 101883.

Kevin Wampler. 2016. Fast and reliable example-based mesh IK for stylized deformations. *ACM Transactions on Graphics (TOG)* 35, 6 (2016).

Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F O'Brien. 2010. Example-based wrinkle synthesis for clothing animation. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM.

Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. 2011. Data-driven elastic models for cloth: modeling and measurement. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM.

X. Wang, X. Liu, and C.Hurren Deakin. 2008. Physical and mechanical testing of textiles. In *Fabric Testing*, Jinlian Hu (Ed.). Woodhead Publishing, 90–124.

Zhendong Wang, Longhua Wu, Marco Fratarcangeli, Min Tang, and Huamin Wang. 2018. Parallel multigrid for nonlinear cloth simulation. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library.

Max Wardetzky, Miklós Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. 2007. Discrete Quadratic Curvature Energies. *Comput. Aided Geom. Des.* 24, 8–9 (2007), 499–518.

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).

Chris Wojtan, Nils Thürey, Markus Gross, and Greg Turk. 2009. Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 papers*.

Chris Wojtan and Greg Turk. 2008. Fast viscoelastic behavior with thin features. In *ACM SIGGRAPH 2008 papers*.

Wolfram Research, Inc. 2019. Mathematica, Version 12.0. `https://www.wolfram.com/mathematica` Champaign, IL.

Botao Wu, Zhendong Wang, and Huamin Wang. 2022. A GPU-Based Multilevel Additive Schwarz Preconditioner for Cloth and Deformable Body Simulation. *ACM Transactions on Graphics (TOG)* 41, 4, Article 63 (2022).

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

Kui Wu and Cem Yuksel. 2017. Real-time fiber-level cloth rendering. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

Rundong Wu, Claire Harvey, Joy Xiaoji Zhang, Sean Kroszner, Brooks Hagan, and Steve Marschner. 2020a. Automatic structure synthesis for 3D woven relief. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 102–1.

Rundong Wu, Joy Xiaoji Zhang, Jonathan Leaf, Xinru Hua, Ante Qu, Claire Harvey, Emily Holtzman, Joy Ko, Brooks Hagan, Doug James, François Guimbretière, and Steve Marschner. 2020b. Weavecraft: An Interactive Design and Simulation Tool for 3D Weaving. *ACM Transactions on Graphics (TOG)* 39, 6, Article 210 (2020).

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).

Shan Yang, Junbang Liang, and Ming C. Lin. 2017. Learning-Based Cloth Material Recovery From Video. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

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).

Julien Yvonnet, Eric Monteiro, and Qi-Chang He. 2013. Computational homogenization method and reduced database model for hyperelastic heterogeneous structures. *International Journal for Multiscale Computational Engineering* 11, 3 (2013).

Shuang Zhao, Fujun Luan, and Kavita Bala. 2016. Fitting procedural yarn models for realistic cloth rendering. *ACM Transactions on Graphics (TOG)* 35, 4 (2016).

Hua Zhong, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. 2001. Realistic and efficient rendering of free-form knitwear. *The Journal of Visualization and Computer Animation* 12, 1 (2001), 13–22.

Evgeny Zuenko and Matthias Harders. 2019. Wrinkles, Folds, Creases, Buckles: Small-Scale Surface Deformations as Periodic Functions on 3D Meshes. *IEEE Transactions on Visualization and Computer Graphics* (2019).

Javier S Zurdo, Juan P Brito, and Miguel A Otaduy. 2012. Animating wrinkles by example on non-skinned cloth. *IEEE Transactions on Visualization and Computer Graphics* 19, 1 (2012).

APPENDIX A

Thin-Shell Homogenization Details

This chapter provides details for thin-shell homogenization and periodic yarn simulation from Chapter 3, and for the fitting algorithm from Chapter 4. Specifically, we will discuss derivations of the rotation matrix and co-rotated boundary conditions, yarn optimization details (periodicity, Newton solver, rest shapes, …), and fitting details (algorithm, sampling, …), as well as robust eigenvalues of curvature and some implementation details for thin-shell cloth simulation. For further implementation details please refer to the code at `https://git.ist.ac.at/gsperl/HYLC` (yarn simulation and fitting) and `https://git.ist.ac.at/gsperl/ARCSim-HYLC` (cloth simulation).

A.1 Homogenization Details

A.1.1 Derivation of $\boldsymbol{R}$

In this section, we derive the analytic expression for the computation of the rotation matrix $\boldsymbol{R}$ from Section 3.2.2, and we summarize how to compute the micro-midsurface φ from the macroscale fundamental forms $\overline{\mathbf{I}}, \overline{\mathbf{II}}$.

We start with the goal $\overline{\mathbf{II}} = \mathbf{II}$. Using the definitions from Chapter 3 and with a slight abuse of index notation, this equals

$$\mathbf{II} = \overline{\mathbf{II}} \tag{A.1a}$$
$$\boldsymbol{a}_\alpha \cdot \boldsymbol{n}_{,\beta} = \overline{\boldsymbol{a}}_\alpha \cdot \overline{\boldsymbol{n}}_{,\beta} \tag{A.1b}$$
$$(\boldsymbol{R}\overline{\boldsymbol{a}}_\alpha)^\top \boldsymbol{n}_{,\beta} = \overline{\boldsymbol{a}}_\alpha{}^\top \overline{\boldsymbol{n}}_{,\beta} \tag{A.1c}$$
$$\boldsymbol{R}^\top \boldsymbol{n}_{,\alpha} = \overline{\boldsymbol{n}}_{,\alpha}. \tag{A.1d}$$

By definition, $\boldsymbol{R}\,\overline{\boldsymbol{n}} = \boldsymbol{n}$; thus, deriving both sides we have

$$\boldsymbol{R}_{,\alpha}\overline{\boldsymbol{n}} = \boldsymbol{n}_{,\alpha}. \tag{A.2}$$

Plugging (A.2) into (A.1d) we get

$$\boldsymbol{R}^\top \boldsymbol{R}_{,\alpha}\overline{\boldsymbol{n}} = \overline{\boldsymbol{n}}_{,\alpha}. \tag{A.3}$$

Therefore, we want to find an expression for $\boldsymbol{R}$ that satisfies (A.3). To this end, we parametrize $\boldsymbol{R}$ via the exponential map, $\boldsymbol{R} = \exp \boldsymbol{W}$, where $\boldsymbol{W}$ is a skew-symmetric matrix. The derivative

of the exponential map is a fairly complicated expression [Rossmann 2006],

$$\boldsymbol{R}_{,\alpha} = \boldsymbol{R} \sum_{k=0}^{\infty} \frac{(-1)^k}{(k+1)!} (\operatorname{ad} \boldsymbol{W})^k \boldsymbol{W}_{,\alpha}, \tag{A.4}$$

where $(\operatorname{ad} \boldsymbol{W})\boldsymbol{X} = \boldsymbol{W}\boldsymbol{X} - \boldsymbol{X}\boldsymbol{W}$. Note that when the surface is singly curved, $\boldsymbol{W}$ and $\boldsymbol{W}_{,\alpha}$ commute, so $\boldsymbol{R}_{,\alpha} = \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ holds *exactly* even for large $\boldsymbol{W}$. For the more general case of doubly curved surfaces, one can approximate $\boldsymbol{R}_{,\alpha} \approx \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ under the common assumption that the curvature is small relative to the microscale. Consequently, we approximate (A.3) by

$$\boldsymbol{W}_{,\alpha} \bar{\boldsymbol{n}} = \bar{\boldsymbol{n}}_{,\alpha}. \tag{A.5}$$

This only determines two of the three degrees of freedom (DOFs) of $\boldsymbol{W}$,

$$\boldsymbol{W}(X_1, X_2) = \begin{pmatrix} 0 & \bullet & \sum_\alpha \bar{n}_{1,\alpha} X_\alpha \\ & 0 & \sum_\alpha \bar{n}_{2,\alpha} X_\alpha \\ (\text{skew}) & & 0 \end{pmatrix}. \tag{A.6}$$

To fix the solution we take the minimum-norm choice, $\bullet = 0$.

Finally, we can recover $\boldsymbol{R}(X_1, X_2) = \exp \boldsymbol{W}(X_1, X_2)$. This is the matrix that rotates $\bar{\boldsymbol{n}} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top$ towards $\Delta\bar{\boldsymbol{n}} := \sum_\alpha \bar{\boldsymbol{n}}_{,\alpha} X_\alpha$ by an angle $\|\Delta\bar{\boldsymbol{n}}\|$. Importantly, this choice exactly satisfies the original constraint (A.3) for singly curved surfaces. The exponential of $\boldsymbol{W}$ can be computed in multiple ways; however, many are not numerically robust for small bending strains. Here, we provide a closed form expression for $\boldsymbol{R}$ based on the (unnormalized) sinc function, which can be implemented to be numerically robust around 0 using Taylor expansions. Defining

$$a = \sum_\alpha \overline{n}_{1,\alpha} X_\alpha, \tag{A.7}$$

$$b = \sum_\alpha \overline{n}_{2,\alpha} X_\alpha, \tag{A.8}$$

$$r = \sqrt{a^2 + b^2}, \tag{A.9}$$

we compute the rotation as

$$\boldsymbol{R} = \begin{pmatrix} 1 - \frac{1}{2}a^2 \operatorname{sinc}(r/2)^2 & -\frac{1}{2}ab \operatorname{sinc}(r/2)^2 & a \operatorname{sinc}(r) \\ -\frac{1}{2}ab \operatorname{sinc}(r/2)^2 & 1 - \frac{1}{2}b^2 \operatorname{sinc}(r/2)^2 & b \operatorname{sinc}(r) \\ -a \operatorname{sinc}(r) & -b \operatorname{sinc}(r) & \cos(r) \end{pmatrix}. \tag{A.10}$$

This expression robustly converges to the identity for $r \to 0$.

To offer some insight, the proposed strategy for computing $\boldsymbol{R}$ can be seen as integrating curvature along straight lines. For singly curved surfaces where the order of integration of X_1 and X_2 does not matter, the expression is exact, whereas for doubly curved surfaces it is an approximation.

Computing the Micro-Midsurface We now summarize how to compute the midsurface φ from $\bar{\mathrm{I}}$ and $\overline{\mathrm{II}}$. To solve the Poisson system $\nabla^2\varphi = \nabla \cdot \boldsymbol{R}\overline{\boldsymbol{S}}$, we need both the in-plane deformation $\overline{\boldsymbol{S}}$ and the rotation $\boldsymbol{R}$. We compute $\overline{\boldsymbol{S}}$ with

$$\overline{\boldsymbol{S}} = \begin{pmatrix} \sqrt{\bar{\mathrm{I}}} \\ 0 \quad 0 \end{pmatrix}. \tag{A.11}$$

To compute $\boldsymbol{R}$, we need the normal derivatives $\overline{\boldsymbol{n}}_{,\alpha}$, which can be computed using

$$\begin{pmatrix} \overline{n}_{1,1} & \overline{n}_{1,2} \\ \overline{n}_{2,1} & \overline{n}_{2,2} \end{pmatrix} = -\left(\sqrt{\overline{\mathbf{I}}}\right)^{-\top} \overline{\mathbf{II}}. \tag{A.12}$$

Note that $\overline{n}_{3,\alpha} = 0$ since $\overline{\boldsymbol{n}} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top$ by assumption.

We solve for $\boldsymbol{\varphi}$ numerically, by discretizing it on a regular grid large enough to enclose the entire yarn patch. Using standard finite differencing, we can discretize the Laplacian ∇^2 on the left-hand side as well as the right-hand side $\nabla \cdot \boldsymbol{R}\overline{\boldsymbol{S}}$ with the pure Neumann boundary conditions $\boldsymbol{N} \cdot \nabla\boldsymbol{\varphi} = \boldsymbol{N} \cdot \boldsymbol{R}\overline{\boldsymbol{S}}$. With $\overline{\boldsymbol{S}}$ and (A.10), we can compute the required values at the grid nodes.

Note that, in the case of a doubly-curved least-squares surface, the rotation given by (A.10) does in general not match the computed surface but is required for co-rotated boundary conditions. In this case, one can re-estimate $\boldsymbol{R}$ as the rotation from the polar decomposition of a finite-differenced gradient $\nabla\boldsymbol{\varphi}$.

Analytic Solution We want to emphasize that we designed our solution to compute $\boldsymbol{R}$ and $\boldsymbol{\varphi}$ for general deformations. In case one only needs to consider singly-curved deformations, it is possible to instead use an *analytic* expression for $\boldsymbol{\varphi}$ (and thus $\boldsymbol{a}_\alpha$, $\boldsymbol{n}$, $\boldsymbol{R}$ etc.). In fact, the recent work of Rodriguez et al. [2022] uses such an analytic mapping.

A.1.2 Derivation of Co-rotated Periodicity

In this section, we provide a brief derivation of our co-rotated periodicity boundary conditions (see Section 3.2.2). For comparison, in the case of solid homogenization explained in Section 3.2.1, the constraint $\int_\Omega \nabla\tilde{\boldsymbol{u}}\, d\boldsymbol{X} = 0$ can be derived from the expansion $\boldsymbol{x}$ and the deformation average of $\boldsymbol{F}$. Here, we apply an analogous tactic, but instead of averaging the deformation $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$ directly, we again leverage the rotation $\boldsymbol{R}$ from the polar decomposition $\nabla\boldsymbol{\varphi} = \boldsymbol{R}\overline{\boldsymbol{S}}$.

The proposed average is thus

$$\frac{1}{|\Omega|}\int_\Omega \boldsymbol{R}^\top \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}\, d\Omega = \overline{\boldsymbol{S}}, \tag{A.13}$$

Notably, we average only the *in-plane* derivatives $\frac{\partial}{\partial \boldsymbol{X}}$ and omit $\frac{\partial}{\partial h}$. This directly relates to the fact that we do not want to impose constraints on the thickness and on out-of-plane shearing at the microscale, since these deformations are also not modeled on the macroscale.

From (A.13) we now derive the periodic boundary conditions. Plugging the expansion $\boldsymbol{x} = \boldsymbol{\varphi} + h\boldsymbol{n} + \tilde{\boldsymbol{u}}$ into (A.13),

$$\frac{1}{|\Omega|}\int_\Omega \boldsymbol{R}^\top \left(\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{X}} + h\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{X}} + \frac{\partial \tilde{\boldsymbol{u}}}{\partial \boldsymbol{X}}\right) d\Omega = \overline{\boldsymbol{S}}, \tag{A.14}$$

and using a rearranged polar decomposition $\boldsymbol{R}^\top \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{X}} = \overline{\boldsymbol{S}}$, we get

$$\int_\Omega \left(h\boldsymbol{R}^\top \boldsymbol{n}_{,\alpha} + \boldsymbol{R}^\top \tilde{\boldsymbol{u}}_{,\alpha}\right) d\Omega = \boldsymbol{0}. \tag{A.15}$$

Assuming that the RVE is centered with $\int_H h\, dh = 0$ and noticing that $\boldsymbol{R}$ and $\boldsymbol{n}_{,\alpha}$ are constant in h, we can simplify

$$\int_\Omega h\boldsymbol{R}^\top \boldsymbol{n}_{,\alpha}\, d\Omega = \int_\Gamma \Big(\int_H h\, dh \Big) \boldsymbol{R}^\top \boldsymbol{n}_{,\alpha}\, d\boldsymbol{X} = \boldsymbol{0}, \tag{A.16}$$

where H denotes the thickness domain and Γ the midsurface domain. Therefore, (A.15) simplifies to

$$\int_\Omega \boldsymbol{R}^\top \tilde{\boldsymbol{u}}_{,\alpha}\, d\Omega = \boldsymbol{0}. \tag{A.17}$$

Splitting this integral into in-plane and out-of-plane parts, applying Leibniz's rule and using the divergence theorem, we can rewrite (A.17) as

$$\int_H \int_{\partial\Gamma} \boldsymbol{R}^\top \tilde{\boldsymbol{u}} \otimes \boldsymbol{N}_\alpha\, d\partial\Gamma\, dh = \int_\Omega \boldsymbol{R}_{,\alpha}{}^\top \tilde{\boldsymbol{u}}\, d\Omega, \tag{A.18}$$

where $\partial\Gamma$ is the in-plane boundary of the midsurface and $\boldsymbol{N}_\alpha$ are the normals to that boundary in the corresponding directions. As discussed in the previous section, $\boldsymbol{R}_{,\alpha} = \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ for singly-curved surfaces and $\boldsymbol{R}_{,\alpha} \approx \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ for small curvatures, where $\boldsymbol{W}_{,\alpha}$ is constant. With our fitting strategy, we only apply cylindrical deformation, and as such it holds exactly. Therefore, the right-hand side of (A.18) vanishes:

$$\begin{aligned}
\int_\Omega \boldsymbol{R}_\alpha^\top \tilde{\boldsymbol{u}}\, d\Omega &= \int_\Omega \boldsymbol{R}_\alpha^\top \boldsymbol{R}\boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega && \text{(A.19a)}\\
&\simeq \int_\Omega \boldsymbol{W}_{,\alpha}^\top \boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega && \text{(A.19b)}\\
&\simeq \boldsymbol{W}_{,\alpha}^\top \int_\Omega \boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega && \text{(A.19c)}\\
&\simeq \boldsymbol{0}, && \text{(A.19d)}
\end{aligned}$$

since we have the translation constraint $\int_\Omega \boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega = \boldsymbol{0}$.

Therefore and analogously to solid homogenization, (A.18) can be satisfied by splitting Γ into opposite parts $\partial\Gamma^+$ and $\partial\Gamma^-$ and prescribing periodic boundary conditions

$$(\boldsymbol{R}^\top \tilde{\boldsymbol{u}})^+ = (\boldsymbol{R}^\top \tilde{\boldsymbol{u}})^-. \tag{A.20}$$

Alternatively, one could satisfy (A.18) more simply by setting $\tilde{\boldsymbol{u}} = \boldsymbol{0}$ at the boundary, effectively gluing microstructures to the (deformed) boundary. However, this would be unnecessarily stiff as it does not allow, for example, yarns to resolve collision at the boundary during compression.

A.2 Yarn-level Optimization Details

In this section, we provide more detail on periodic microscale energies; on initialization, regularization, scaling, and the stopping criterion in our Newton solver; and on how we generate a flat reference configuration for our yarn patterns.

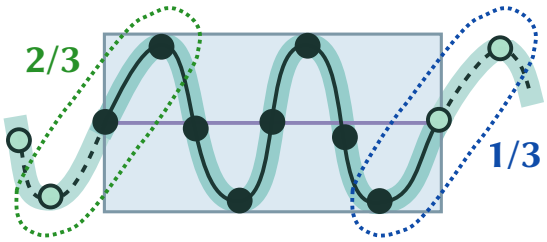


Figure A.1: We use fractional weights to ensure energies are integrated only over the periodic tile. Periodic vertices are plotted as empty circles. In this example, the contributions of periodic bending elements on the left and right are multiplied by their respective fraction of original (non-ghost) vertices.

A.2.1 Periodicity

Fractional Counting For the simulation of periodic yarn patterns, we need periodic stretching, bending, twisting, and collision energies. As discussed in Section 3.3.2, we model periodic energies by extending the pattern with ghost segments. These ghost segments serve only to compute energies over the periodic boundaries and should not introduce any additional energy into the system. To avoid double counting of energies and forces, we introduce *fractional counting* as illustrated in Figure A.1.

In our implementation, energies are computed from local elements: a stretching element consists of two connected vertices, bending and twisting both use three connected vertices and the two corresponding edges, and collisions are computed from pairs of two-vertex segments. For each element, we can then compute the fraction of non-ghost vertices to the total number of vertices. Multiplying the element's energy by this fraction ensures that in total, and including all (partial and full) periodic copies, the contribution is counted exactly once. Note that in the Newton step, the elimination of variables generates forces $\boldsymbol{f} = -\tilde{\boldsymbol{C}}^\top \nabla E$, where multiplication with $\tilde{\boldsymbol{C}}^\top$ sums up the partial contributions to each non-ghost DOF from its periodic copies. We also emphasize that the translation constraint only includes non-ghost vertices.

We can avoid computing purely periodic collisions of ghost segments. As suggested by Kaldor et al. [2008], we use an AABB tree with a static hierarchy as the collision broadphase. We can immediately prune subtrees with only ghost vertices for extra performance since they have a fractional count of 0 and will not add to the energy.

Edge Orientation In Section 3.3.2, the periodic twist constraint has the form

$$(\boldsymbol{R}^\top \underline{\boldsymbol{d}}_\alpha)^+ = (\boldsymbol{R}^\top \underline{\boldsymbol{d}}_\alpha)^-, \qquad \theta^+ = \theta^-. \tag{A.21}$$

In our implementation, vertex order defines the direction of the reference frame directors $\underline{\boldsymbol{d}}_\alpha$. For (A.21) to be correct, ghost segments need to have the same orientation and thus vertex order. To extend a pattern with ghost yarns, we take input pattern geometry of connected vertices, copy and translate parts as new ghost segments, and then stitch the original and copied parts together. During this stitching process, we enforce the condition on edge orientation by reversing the vertex order of yarns as necessary. During optimization, updates to the vertices are subject to co-rotated periodicity and as such should not invalidate periodicity. In our implementation, we approximate (A.21) using the value of $\boldsymbol{R}$ computed at the first vertex of each edge. This works well given that we also use short edge lengths on a scale similar to the yarn radius. In our experiments, we haven't noticed any significant drift, so we do not reproject the directors.

A.2.2 Elimination of Periodic Variables

The constraints on periodic vertex positions (3.21) and edge twists (3.27) can be concatenated into one constraint

$$\boldsymbol{C}\boldsymbol{q} = \boldsymbol{d}. \tag{A.22}$$

We eliminate periodic variables by parametrizing the nullspace of (A.22). We found that simply using a QR factorization for this is not numerically robust and produced dense matrices. However, we can exploit that each periodically copied degree of freedom is used in exactly one (sub-)constraint. As such, we can write $\boldsymbol{C}$ as

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A} \end{pmatrix} \boldsymbol{P}, \tag{A.23}$$

where $\boldsymbol{P}$ is a permutation matrix that permutes the columns of $\boldsymbol{C}$, splitting it into a left identity block $\boldsymbol{I}$ and a sparse right block $\boldsymbol{A}$; i.e., we bring the constraint matrix into reduced row echelon form, where the columns of $\boldsymbol{A}$ span the kernel of $\boldsymbol{C}$. In our case, the kernel represents exactly the original degrees of freedom that are copied. For our periodicity constraints, the matrix is of the form

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{A} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{3\times3} & & & -(\boldsymbol{R}^{+})(\boldsymbol{R}^{-})^{\top} & & \\ & 1 & & & -1 & \\ & & \ddots & & & \ddots \end{pmatrix}. \tag{A.24}$$

Note that $\boldsymbol{P}$ splits $\boldsymbol{q}$ into free variables $\boldsymbol{y}$ and copies $\hat{\boldsymbol{q}}_{\text{copy}}$:

$$\boldsymbol{P}\boldsymbol{q} = \begin{pmatrix} \hat{\boldsymbol{q}}_{\text{copy}} \\ \boldsymbol{y} \end{pmatrix}. \tag{A.25}$$

Then, we have

$$\boldsymbol{C}\boldsymbol{q} = \boldsymbol{d} \tag{A.26a}$$

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{A} \end{pmatrix} \boldsymbol{P}\boldsymbol{q} = \boldsymbol{d} \tag{A.26b}$$

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{A} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{q}}_{\text{copy}} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{d} \tag{A.26c}$$

$$\hat{\boldsymbol{q}}_{\text{copy}} = -\boldsymbol{A}\boldsymbol{y} + \boldsymbol{d}, \tag{A.26d}$$

and

$$\boldsymbol{q} = \boldsymbol{P}^{\top} \begin{pmatrix} \hat{\boldsymbol{q}}_{\text{copy}} \\ \boldsymbol{y} \end{pmatrix} \tag{A.27a}$$

$$= \boldsymbol{P}^{\top} \begin{pmatrix} -\boldsymbol{A}\boldsymbol{y} + \boldsymbol{d} \\ \boldsymbol{y} \end{pmatrix} \tag{A.27b}$$

$$= \boldsymbol{P}^{\top} \begin{pmatrix} -\boldsymbol{A} \\ \boldsymbol{I} \end{pmatrix} \boldsymbol{y} + \boldsymbol{P}^{\top} \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{0} \end{pmatrix} \tag{A.27c}$$

$$= \quad \tilde{\boldsymbol{C}} \qquad \boldsymbol{y} + \quad \tilde{\boldsymbol{d}}. \tag{A.27d}$$

Finally, we need to compute the initial $\boldsymbol{y}$ from $\boldsymbol{q}$. For generality, we consider the case when $\boldsymbol{C}_L\boldsymbol{q} \neq \boldsymbol{d}_L$ and find $\boldsymbol{y}$ as

$$\min_{\boldsymbol{y}} \left| \tilde{\boldsymbol{C}}\boldsymbol{y} + \tilde{\boldsymbol{d}} - \boldsymbol{q} \right|^2 \quad \text{s.t.} \quad \boldsymbol{C}_L(\tilde{\boldsymbol{C}}\boldsymbol{y} + \tilde{\boldsymbol{d}}) - \boldsymbol{d}_L = 0, \tag{A.28}$$

the solution of which is given by

$$\begin{pmatrix} \tilde{C}^\top \tilde{C} & \tilde{C}^\top {C_L}^\top \\ C_L \tilde{C} & 0 \end{pmatrix} \begin{pmatrix} y \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{C}^\top (q - \tilde{d}) \\ d_L - C_L \tilde{d} \end{pmatrix}. \tag{A.29}$$

A.2.3 Optimization

Here, we discuss the Newton step from Section 3.3.4 in more detail.

Newton System Scaling Vertex positions and edge twists have different units. Depending on the scale of yarns and the choice of length units, this may unfavorably affect the conditioning of the linear system and thus optimization performance. We therefore rescale the linear system for the microscale optimization with a diagonal scaling matrix M:

$$\left(M \begin{pmatrix} \tilde{C}^\top H \tilde{C} & \tilde{C}^\top {C_L}^\top \\ C_L \tilde{C} & 0 \end{pmatrix} M + \begin{pmatrix} \alpha I & 0 \\ 0 & 0 \end{pmatrix} \right) w = -M \begin{pmatrix} \tilde{C}^\top \nabla E \\ C_L q - d_L \end{pmatrix} \tag{A.30a}$$

$$\begin{pmatrix} \delta y \\ \lambda \end{pmatrix} = M w \tag{A.30b}$$

We use M with entries 1 for positional DOFs and $10^6\ r$ for twist DOFs, where r is the yarn radius.

Regularization Similarly, the different stiffnesses of stretching and collision compared to bending and twisting energies can negatively affect convergence. In (A.30), we add a regularizer α to the system matrix to improve convergence. At the beginning of our simulations, the barrier-like collision forces typically dominate. In these cases, α improves convergence by shifting focus to larger gradients, which helps resolving and balancing out collisions first compared to elastic rod forces. In our experiments, we use $\alpha = \hat{\alpha}\,|M \nabla E|_\infty$ and exponentially decay $\hat{\alpha}$ from $\hat{\alpha}_0 = 5000$ to $\hat{\alpha}_N = 5$ over $N = 400$ iterations, i.e. $\hat{\alpha}_i = \hat{\alpha}_0 (\hat{\alpha}_N / \hat{\alpha}_0)^{\min(i,N)/N}$.

Step Limit Increments δy computed from the linear system may be arbitrarily large. To ensure that collisions are not ignored, we rescale δy such that the maximum displacement of any vertex is smaller than a fraction p of its radius. The value of p generally depends on the macroscopic deformation; in compressive regimes where yarns initially overlap strongly, a lower p may be required. We set $p = 0.05 + 0.15\ \min(\max(0, \lambda_{\min}), 1)$, where $\lambda_{\min}$ is the smallest eigenvalue of $\bar{\mathbf{I}}$. Afterwards, in each optimization step we perform a simple decreasing linesearch on p, iteratively multiplying it by 0.1 until the objective is improved. More recently, for the project discussed in Chapter 6, we have found that the additive continuous collision detection from [Li et al. 2021] is a good replacement to the above strategy and less dependent on choosing good parameters. However, note that their algorithm works for discrete edge segments and this may in general not work well together with the spline-based collisions of Kaldor et al. [2008]. Also, if collision barriers are too weak, then the continuous collision detection may require step sizes going to 0 and thus stall in similar scenarios where the above step limit would instead result in tunneling.

Initial Guess As an initial guess to Newton's method, we set $\tilde{\boldsymbol{u}} = \boldsymbol{0}$ everywhere. This choice is natural in that it deforms the pattern rigidly according to the macroscopic strains. Additionally, the vertex periodicity and translation constraints are naturally satisfied. While warmstarting with solutions $\tilde{\boldsymbol{u}}$ from nearby macroscale deformations is an option, we have instead opted for naive parallelization using the naive initial guess.

Stopping Criterion The usual stopping criterion in Newton's method is the norm of the gradient of the objective. However, our system includes constraints $\boldsymbol{C}_L\boldsymbol{q} = \boldsymbol{d}_L$ with Lagrange multipliers and elimination of variables with $\tilde{\boldsymbol{C}}\boldsymbol{y} + \tilde{\boldsymbol{d}} = \boldsymbol{q}$ for periodicity. For robustness, we consider the case that the system can be in a state where $\boldsymbol{C}_L\boldsymbol{q} \neq \boldsymbol{d}_L$. Although using our initial state $\tilde{\boldsymbol{u}} = \boldsymbol{0}$ implies that this is satisfied, and the Newton step and scaling of $\delta\boldsymbol{y}$ should not affect it, there could be numerical drift. Therefore, as our stopping criterion we use the norm of the gradient with respect to the *free* variables $\boldsymbol{y}$ projected onto the tangent space of $\boldsymbol{C}_L\boldsymbol{q} = \boldsymbol{d}_L$.

The projection of the gradient $\hat{\boldsymbol{z}} = \nabla_{\boldsymbol{y}} E = \tilde{\boldsymbol{C}}^\top \nabla E$ is found by solving

$$\min_{\boldsymbol{z}} \frac{1}{2} |\boldsymbol{z} - \hat{\boldsymbol{z}}| \quad \text{s.t.} \quad \boldsymbol{C}_L\tilde{\boldsymbol{C}}\boldsymbol{z} = \boldsymbol{0}, \tag{A.31}$$

the solution of which is

$$\boldsymbol{z} = \hat{\boldsymbol{z}} - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}\hat{\boldsymbol{z}}, \tag{A.32}$$

where $\boldsymbol{A} = \boldsymbol{C}_L\tilde{\boldsymbol{C}}$. Note that computing $(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}$ only requires computing a 4×4 inverse. In an effort to nondimensionalize the stopping criterion, we finally require

$$|\boldsymbol{z}| < \varepsilon_z \left|\boldsymbol{z}^0\right|, \tag{A.33}$$

with $\boldsymbol{z}^0$ being the value of $\boldsymbol{z}$ in the first optimization step and $\varepsilon_z = 1 \times 10^{-5}$.

For completeness, we also check if the constraints modeled by the Lagrange multipliers are satisfied $\|\boldsymbol{C}_L\boldsymbol{q} - \boldsymbol{d}_L\| < \varepsilon_L\, r$ with r being the yarn radius and $\varepsilon_L = 1 \times 10^{-4}$ being a relative error threshold, althrough we found that this is always satisifed, starting from a valid configuration $\tilde{\boldsymbol{u}} = \boldsymbol{0}$.

A.2.4 Pattern Reference Configuration

Here, we discuss our heuristic to find a rest pattern that is in equilibrium with respect to in-plane stretching, which we mention in Section 3.3.1. Note that for Chapter 6 we introduce a similar but more automatic algorithm explained in that chapter.

The yarns of a fabric may exhibit residual tension from the fabrication process. This tension is crucial in achieving both the visual appeal of many knit patterns as well as emergent physical properties; for example, the tension causes the edges of the stockinette pattern to curl (as illustrated in Chapter 1).

We do not know the tension a priori. For animation purposes, we want it to not induce notable in-plane shrinking; i.e., the reference configuration of the pattern should be the minimum with respect to in-plane deformation. This way, the homogenized model may show its tendency to curl but doesn't shrink. To achieve this, we generate a stress-free state from input yarn geometry, apply tension, and then reduce the periodic lengths to find the in-plane minimum. We summarize the heuristic below and in Algorithm 2.

Algorithm 2 Compute reference yarn geometry under tension and at rest wrt. stretching.

Input: periodic yarn geometry not at rest
periodic lengths p_{X_1}, p_{X_2}
material parameters
Output: reference-state geometry X_1, X_2, h
rest lengths, rest curvatures, rest twists, ...

procedure ComputeReferenceState
  generate stress-free state:
    repeat
      reset rest lengths, rest curvatures and rest twists
        ▷ setting rest values to current values
      relax yarns ▷ $\min_{\boldsymbol{X}} E$
    until $E < 10^{-10}$
  apply tension:
    if knitted **then**
      rest lengths $\leftarrow$ 0.9 rest lengths
      rest curvatures $\leftarrow$ 0.8 rest curvatures
    else
      rest curvatures $\leftarrow$ 0.9 rest curvatures
    end if
  find in-plane minimum: $\min_{p_{X_1}, p_{X_2}} E$
    modify p_{X_1}, p_{X_2} and compute total energy at equilibrium
  translate center to $\int_\Omega (X_1, X_2, h) = \mathbf{0}$
end procedure

Initial yarn geometry can be overlapping; to generate the stress-free state, we iteratively reset rest lengths, curvatures, and twists, and then allow collisions to be resolved. This converges to a collision-free state, where discrete elastic rod forces are at rest. Next, we apply tension to the yarns by shortening rest lengths and flattening rest curvature. For knitted patterns, we multiply rest lengths by 0.9 and rest curvatures by 0.8; for woven patters, we only multiply rest curvature by 0.9. We found these values to work well as an approximation under our expectation of how real knits and woven materials behave. Finally, we reduce the periods in the warp and weft directions to find the energy minimum with respect to periodic lengths. The resulting final state can therefore only induce shearing or bending. In the results of Chapter 4, we chose patterns where shearing was negligible as observed in the yarn-level reference simulations.

After recentering the pattern, the final reference configuration defines the initial coordinates $\begin{pmatrix} X_1 & X_2 & h \end{pmatrix}^\top$, the periodic lengths define the patch area, and the extents of the pattern along h define its thickness. Our results demonstrate that this procedure is able to generate homogenized models that exhibit curling without in-plane deformation at rest.

A.3 Fitting Details

Here, we discuss the fitting procedure outlined in Section 4.1 in more depth.

Algorithm 3 Fit material model splines from data

Input: data $(s_x, s_a, s_y, \mathrm{II}_{00}, \mathrm{II}_{11}; \overline{\Psi})$ for 1D and 2D ranges
Output: $f_0, f_i, f_x, f_y, f_{ij}, f_{ix}, f_{iy}$

procedure FitSplines
  $\boldsymbol{X} \leftarrow$ normalize(strains)
    ▷ divide each coord. by max. abs. value in data
  $f_0 \leftarrow \overline{\Psi}(\mathbf{0})$
  for $i \in \{1, 2, 3, x, y\}$ **do**
    gather X^i, $\overline{\Psi}^i$ ▷ strain and energy for data range
    $f_i \leftarrow$ Fit1D(X^i,$\overline{\Psi}^i$) ▷ Algorithm 4
  end for
  for $ij \in \{12, 13, 23, 1x, 2x, 3x, 1y, 2y, 3y\}$ **do**
    gather X^{ij}, Y^{ij}, $\overline{\Psi}^{ij}$ ▷ strains and energy for data range
    $f_{ij} \leftarrow$ Fit2D(X^{ij},Y^{ij},$\overline{\Psi}^{ij}$) ▷ Algorithm 5
  end for
end procedure

Our goal is to fit elastic energy densities $\overline{\Psi}$ from data in the form of strain-energy pairs. We only sample subspaces of deformations with either bending along x or y directions respectively, and interpolate between those as discussed in Section 4.1. Additionally, we fit each subspace as a sum of constant, univariate, and bivariate terms. We can write out the total energy more verbosely as

$$\overline{\Psi} = f_0 + \sum_i f_i\left(\frac{s_i}{v_i}\right) + f_B + \sum_{ij} f_{ij}\left(\frac{s_i}{v_i}, \frac{s_j}{v_j}\right) + \sum_i f_{iB}, \tag{A.34a}$$

$$\begin{aligned} f_B = {} & c^2 \left(f_x\left(\frac{\lambda_1}{v_x}\right) + f_y\left(\frac{\lambda_2}{v_y}\right)\right) \\ & + (1 - c^2) \left(f_x\left(\frac{\lambda_2}{v_x}\right) + f_y\left(\frac{\lambda_1}{v_y}\right)\right), \end{aligned} \tag{A.34b}$$

$$\begin{aligned} f_{iB} = \sum_i {} & c^2 \left(f_{ix}\left(\frac{s_i}{v_i}, \frac{\lambda_1}{v_x}\right) + f_{iy}\left(\frac{s_i}{v_i}, \frac{\lambda_2}{v_y}\right)\right) \\ & + (1 - c^2) \left(f_{ix}\left(\frac{s_i}{v_i}, \frac{\lambda_2}{v_x}\right) + f_{iy}\left(\frac{s_i}{v_i}, \frac{\lambda_1}{v_y}\right)\right), \end{aligned} \tag{A.34c}$$

with $1 \leq i \leq 3$ and $(i + 1) \leq j \leq 3$. Therefore, we have to fit the constant f_0, the univariate in-plane terms f_i and bending terms f_x, f_y, and the bivariate terms f_{ij}, f_{ix}, f_{iy}. Here, v_i, v_x, and v_y denote normalization constants for the in-plane and bending strains; each parameter of each term is divided by the maximum absolute value in the data. See also Algorithm 3 along with Algorithm 4 and Algorithm 5.

For each term, we can sample data for its respective range of deformations and then fit cubic Hermite splines. To avoid overshoot in cubic interpolation, we use piecewise monotonic interpolation (see [Fritsch and Carlson 1980] and Appendix B).

Our energy data is prone to noise, especially in compressive regimes (see Figure A.4). What is more, piecewise monotonic functions can still have multiple local minima. To address these

Algorithm 4 Fit 1D-splines

Input: strains X, energy densities $\overline{\Psi}$
Output: cubic hermite spline f with coefficients
$\boldsymbol{x}$ …control point locations
$\boldsymbol{p}$ …control point values
$\boldsymbol{p}^x$ …control point derivatives

function FIT1D(X, $\overline{\Psi}$)
  MLS:
    def. $g(u) = \text{symexp}(\text{MLS}(u \mid X, \text{symlog}(\overline{\Psi})))$
      ▷ for s_a replace $\text{MLS}(u) = 0.5\ (\text{MLS}(u) + \text{MLS}(-u))$
  INITIAL FIT:
    $\boldsymbol{x} \leftarrow$ linspace within $[0.95 \min X, 0.95 \max X]$
      ▷ include $x = 0$
    $\boldsymbol{p} \leftarrow g(\boldsymbol{x})$
    $\boldsymbol{p}^x \leftarrow$ finite-difference $g(u)$
  QUASICONVEXIFY:
    $p^x_{\min} \leftarrow 0.001$ ▷ 0.01 if stockinette
    if in-plane strain **then**
      $i_{\min} \leftarrow x_{i_{\min}} = 0$
    else
      $i_{\min} \leftarrow \arg\min_i (p_i)$ ▷ find bending min.
    end if
    march outward from $i_{\min}$ and enforce:
    $p_{i+1} \geq p_i + (x_{i+1} - x_i)\ p^x_{\min}$ ▷ $i - 1$ if $i < i_{\min}$
    $p^x_i \geq p^x_{\min}$ ▷ $p^x_i \leq -p^x_{\min}$ if $i < i_{\min}$
  apply monotone cubic algorithm ▷ [Fritsch1980]
  ensure increasing extrapolation ▷ clamp p^x on boundary
  $p_i \leftarrow p_i - p_{x=0}$ ▷ convert to residual
  return $f \leftarrow \{\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{p}^x\}$
end function

issues, we regularize our fits by smoothing out the data using Moving Least-Squares (MLS) and applying heuristic marching algorithms to achieve quasiconvexity in individual terms. Note that we want to regularize the total function $\overline{\Psi}$ in this way, but we fit individual terms f_i and f_{ij}. Therefore, our strategy is to fit each term to match the data and convert it to a residual afterwards.

The value of f_0 is simply the energy of the sample at zero strain, $(\boldsymbol{s}, \lambda_1, \lambda_2, c^2) = \boldsymbol{0}$. We discuss the details of sampling, control point spacing, MLS-smoothing, quasiconvex marching algorithms, and residualization in the following sections.

A.3.1 Sampling

To fit the univariate terms f_i, f_x, f_y we use 150 samples each. For the bivariate terms f_{ij}, f_{ix}, f_{iy} we sample a 50×50 grid. We sample symmetric ranges for s_a, λ_x, λ_y more densely around the origin, and ranges for s_x, s_y more densely towards compression. We choose the limits of deformation ranges empirically, based on self-intersection of the pattern under bending,

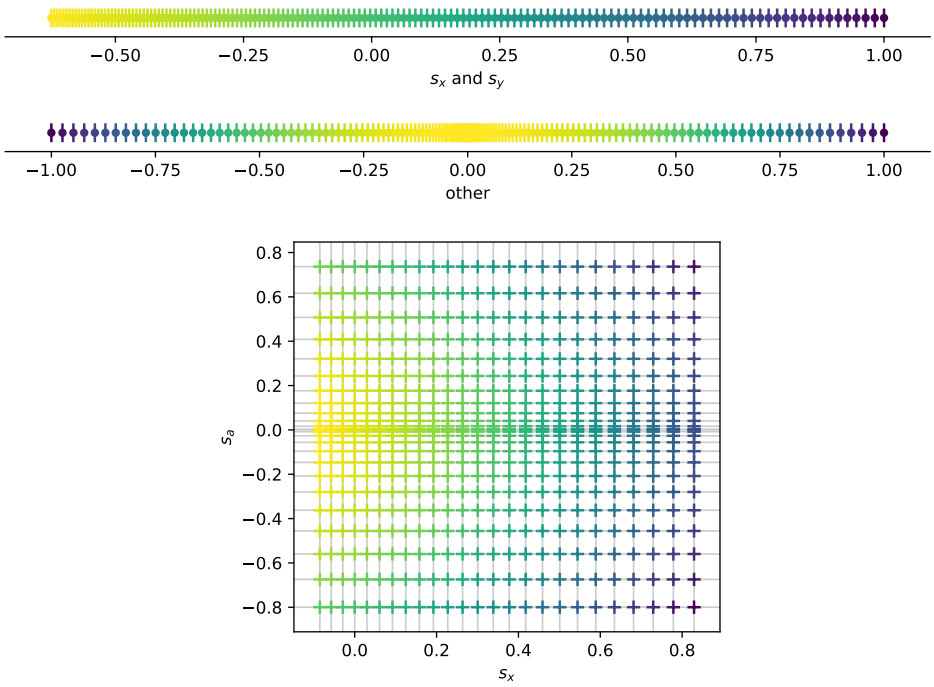


Figure A.2: Top: Our non-uniform sampling for nonsymmetric ranges (s_x and s_y) and symmetric ranges (other). Bottom: An example of a non-uniform grid created from one-dimensional sampling per dimension. We additionally indicate sample density through color.

Table A.1: Total computation time for all sampled microscale simulations for univariate and bivariate terms per pattern. The time is given in the format hrs:min:sec. We ran the simulations in parallel on 128 threads.

Pattern	Time 1D	Time 2D
Basket	00:01:05	00:18:36
Honey	00:01:15	00:56:37
Rib	00:02:46	01:12:57
Satin	00:00:50	00:26:56
Stock.	00:00:13	00:16:46
Satin small	00:00:46	00:25:15
Stock. small	00:00:28	00:14:02

and stability of collisions in general. See also Algorithm 6. Figure A.2 demonstrates this non-uniform sampling.

With five 1D terms, and nine 2D terms, we therefore sample a total of $750 + 22500 = 23250$ deformations. Table A.1 lists the timings for the sampling stage of our method. Sampling is fast, especially when compared to full yarn-level simulations with simulation times on the order of hours and days. Furthermore, 1D terms already describe the rest shapes of draped fabric relatively well, but only require a few minutes to sample. We did not investigate if the number of samples for bivariate terms can be reduced without negatively affecting the fits to save computation time.

A.3.2 Control Point Spacing

We want to space control points equidistantly. For our fitting scheme, we want to set e.g. $f_i(0) = 0$ and $f_{ij}(s_i, 0) = f_{ij}(0, s_j) = 0$, such that fitting another term does not influence previous fits. To this end, we require that control point spacing includes the origin for 1D terms and the axes for 2D terms. For each term and coordinate, we therefore combine two linearly spaced ranges to the left and right of 0 as illustrated in Figure A.3. As discussed in Section 4.3, we found it necessary to modify the control point spacing for the stockinette pattern, where we concentrate them closer to the origin.

A.3.3 MLS-Smoothing

We filter and resample the data for each term using MLS at the spline control points. With this, we can estimate the values and derivatives required for cubic Hermite splines in a way that is robust in the presence of noise.

Since hyperelastic energies often span multiple orders of magnitude, we found it beneficial to smooth the data in symlog-space; i.e., we convert data using $\widetilde{\log}(x) = \operatorname{sgn}(x)\log(|x| + 1)$, estimate a smooth function using MLS, and then exponentiate the result back using $\widetilde{\exp}(x) = \operatorname{sgn}(x)(\exp(|x|) - 1)$. For example for a two-dimensional range parametrized by u, v, we define the smoothed function

$$g(u, v) = \widetilde{\exp}(\mathrm{MLS}(u, v \mid U_{\mathrm{data}}, V_{\mathrm{data}}, \widetilde{\log}(\overline{\Psi}_{\mathrm{data}}))). \tag{A.35}$$

We then estimate the first and second derivatives that make up the remaining Hermite spline coefficients by finite differencing (A.35). At this point, we also enforce symmetry of our functions in s_a by symmetrizing (A.35). This fits our data well and aids in preserving the symmetric rest shapes observed in yarn-level reference simulations also in our macroscale simulations. Figure A.4 shows the result of applying MLS to noisy two-dimensional data.

A.3.4 Marching

Piecewise monotone interpolation does not ensure quasiconvexity for any term. We therefore propose a marching heuristic to project the values and derivatives estimated with MLS to be quasiconvex. For each term, we define a minimum location $x_{\min}$, a minimum tangent

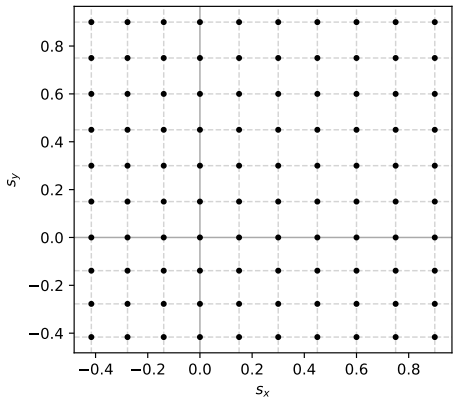


Figure A.3: Representative control point spacing for the 2D term $f_{13}(s_x, s_y)$. We use equidistant spacing to the left and right of each axis and include the axes $s_x = 0$ and $s_y = 0$.

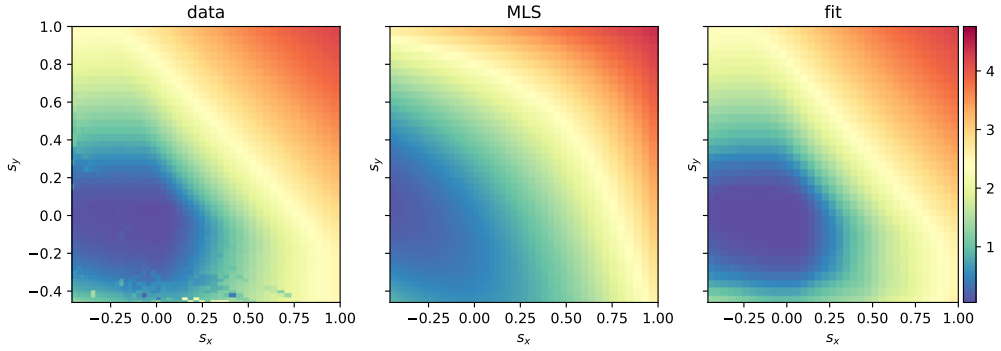


Figure A.4: Data in compressive regions can be noisy as seen on the left. In the middle, we show MLS applied to the data. During fitting we use MLS only at the spline control points to mitigate the data in the noise. Our final regularized fit on the right is similar to the data without noise.

magnitude $p^x_{\min}$, and then march outward while projecting values and derivatives. For 1D, we have

$$p_{i+1} \geq p_i + (x_{i+1} - x_i)\, p^x_{\min} \operatorname{sgn}(x_{i+1} - x_{\min}) \tag{A.36a}$$

$$p^x_i \geq p^x_{\min} \operatorname{sgn}(x_i - x_{\min}). \tag{A.36b}$$

We define the minimum as $s_i = 0$ for in-plane deformation, and compute the bending minimum as $\arg\min_x \overline{\Psi}$ of the respective range; this ensures that the reference state of the cloth corresponds to the in-plane rest state, but allows bent rest shapes to induce curling.

For 2D, we analogously define a minimum and march outward. For the terms f_{ix}, f_{iy} we find the bending minimum subject to $s_i = 0$. However, this algorithm does not allow decreasing values away from the axes, which is crucial for modeling Poisson's ratio accurately. Therefore, we do not apply this regularization on the term $f_{13}(s_x, s_y)$, which is responsible for controlling the response to simultaneous deformation along warp and weft direction and as such for Poisson's ratio. Nevertheless, for the remainder of the terms we found that this is crucial to ensure stable simulations with smooth rest shapes. We choose $p^x_{\min} = 0.001$ for all patterns except the stockinette, where we use 0.01 in an effort to regularize the bad restshape illustrated in Chapter 4 (Figure 4.6). It is after this projection step, that we apply the algorithms for monotone interpolation.

A.3.5 Extrapolation

Simulations can in general exhibit strains outside of the sampled ranges due to discrete timesteps or constraints. Therefore, controlled extrapolation is crucial. We extrapolate splines linearly using their derivatives at the boundary. We enforce that extrapolation must increase energy such as to ensure that the simulation stays near the fitted deformation range. The marching algorithms already enforce that tangents on the boundary are not decreasing away from the minimum (for f_{13}, we additionally clamp the boundary tangents to be increasing outward). For 2D splines, we set the mixed derivatives p^{xy} to 0 at the boundary, such that linear extrapolation is consistent with interpolation.

Finally, we note that in the piecewise monotone bicubic interpolation scheme (Appendix B), we treat extrapolation as a cell with an edge at infinity. This modifies (B.6) and (B.7) to have

the extrapolated side of the inequality become 0.

A.3.6 Residualization

To summarize the fitting procedure: we first smooth and resample the data using MLS, and use it to compute spline coefficients; next, we apply our quasiconvex marching algorithm to enforce a single minimum per term; then we apply the algorithms for piecewise monotone cubic and bicubic spline interpolation.

This procedure gives as regularized fits f^* of the data, which we have to convert to residuals f for the cumulative function (A.34a), with $f_i(0) = 0$ and $f_{ij}(\theta_i, 0) = f_{ij}(0, \theta_j) = 0$. We do this for 1D and 2D terms respectively with

$$f_i(\theta_i) = f_i^*(\theta_i) - f_i^*(0) \tag{A.37}$$

$$f_{ij}(\theta_i, \theta_j) = f_{ij}^*(\theta_i, \theta_j) - f_{ij}^*(\theta_i, 0) - f_{ij}^*(0, \theta_j) + f_{ij}^*(0, 0) \tag{A.38}$$

Finally, we had to enforce that the 2D residuals are 0 for compression in all terms except f_{13}. Compressive data seems to be affected the most from noise, likely due to buckling inducing varying microscale equilibria. Note that this only means that we do not model the *combined* response of e.g. compression and bending. The separable elastic response encoded in the 1D terms still captures the material behavior well.

A.4 Single Curvature Eigenvalues

Here we provide expressions to robustly compute the eigenvalues and squared cosine used in the bending energy curvature splitting (see Section 4.1.2) based on the formulas proposed by Blinn [1996]. With

$$A = \frac{\mathrm{II}_{11} + \mathrm{II}_{22}}{2}, \tag{A.39a}$$

$$B = \frac{\mathrm{II}_{11} - \mathrm{II}_{22}}{2}, \tag{A.39b}$$

$$S = \sqrt{\left(\frac{\mathrm{II}_{11} - \mathrm{II}_{22}}{2}\right)^2 + {\mathrm{II}_{12}}^2 + \varepsilon}, \tag{A.39c}$$

$$k = \operatorname{sgn}(\mathrm{II}_{11} - \mathrm{II}_{22}), \tag{A.39d}$$

where ε is a small number guarding against division by zero, we have

$$\lambda_1 = A + S, \tag{A.40a}$$

$$\lambda_2 = A - S, \tag{A.40b}$$

$$c^2 = \frac{1}{2} + k\left(\frac{1}{2} - \frac{{\mathrm{II}_{12}}^2}{(B + k\,S)^2 + {\mathrm{II}_{12}}^2}\right). \tag{A.40c}$$

Notably, our use of the sign k in (A.40c) ensures that expressions of the form

$$c^2\big(f(\lambda_1) + g(\lambda_2)\big) + (1 - c^2)\big(f(\lambda_2) + g(\lambda_1)\big) \tag{A.41}$$

are robust with respect to the order of eigenvalues even when they are similar, e.g.

$$\lambda_1 = \lambda \pm \varepsilon_1 \qquad\qquad \lambda_2 = \lambda \pm \varepsilon_2. \tag{A.42}$$

A.5 Cloth Simulation Derivative Split

In this section, we briefly discuss how we compute the derivatives of the triangle energy $E_\triangle = A\,\overline{\Psi}(\boldsymbol{z}(\boldsymbol{q}_\triangle))$ from Section 4.2. We found it useful to keep the computation modular, using the chain rule to separate geometric derivatives from strain derivatives.

The energy depends on the positional degrees of freedom of the triangle's vertices and the additional vertices from its adjacent triangles, collectively denoted as $\boldsymbol{q}_\triangle$, from which we can compute the collected strains $\boldsymbol{z} = (s_x, s_a, s_y, \lambda_1, \lambda_2, c^2)$ as discussed in Chapter 4. Using the chain rule, we have

$$\frac{\partial E_\triangle}{\partial \boldsymbol{q}_\triangle} = A\,\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}^\top \frac{\partial \overline{\Psi}}{\partial \boldsymbol{z}}, \tag{A.43}$$

$$\frac{\partial^2 E_\triangle}{\partial \boldsymbol{q}_\triangle \partial \boldsymbol{q}_\triangle} = A\left(\frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle \partial \boldsymbol{q}_\triangle}^\top \frac{\partial \overline{\Psi}}{\partial \boldsymbol{z}} + \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}^\top \frac{\partial^2 \overline{\Psi}}{\partial \boldsymbol{z} \partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}\right). \tag{A.44}$$

This split allows us to easily swap out different energy models, while not affecting the strain part of the computation. To compute the derivatives $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}$ and $\frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle \partial \boldsymbol{q}_\triangle}$, we generate code using Mathematica [Wolfram Research, Inc. 2019]. Additionally, we concatenate both of these derivatives and compute them at the same time, which reduces redundant computation in the generated code by a significant amount.

Algorithm 5 Fit 2D-splines

Input: strains X and Y, energy densities $\overline{\Psi}$
Output: bicubic hermite spline f with coefficients
$\boldsymbol{x}$ …control point locations
$\boldsymbol{p}$ …control point values
$\boldsymbol{p}^x$, $\boldsymbol{p}^y$, $\boldsymbol{p}^{xy}$ …control point derivatives

function FIT2D(X, Y, $\overline{\Psi}$)
  MLS:
    def. $g(u,v) = \text{symexp}(\text{MLS}(u,v \mid X, Y, \text{symlog}(\overline{\Psi})))$
      ▷ for s_a replace $\text{MLS}(u,v) = 0.5\ (\text{MLS}(u,v) + \text{MLS}(-u,v))$
  INITIAL FIT:
    $\boldsymbol{x} \leftarrow$ linspace [0.9*min X, 0] and [0, 0.9*max X]
    $\boldsymbol{y} \leftarrow$ linspace [0.9*min Y, 0] and [0, 0.9*max Y]
      ▷ for stockinette replace 0.9 with 0.3
    $\boldsymbol{p} \leftarrow g(\boldsymbol{x}, \boldsymbol{y})$
    $\boldsymbol{p}^x, \boldsymbol{p}^y, \boldsymbol{p}^{xy} \leftarrow$ finite-difference $g(u,v)$
  QUASICONVEXIFY: ▷ if not (s_x, s_y)
    $p^x_{\min} \leftarrow 0.001$ ▷ 0.01 if stockinette
    $i_{\min} \leftarrow x_{i_{\min}} = 0$ ▷ i always in-plane
    if Y is in-plane strain **then**
      $j_{\min} \leftarrow y_{j_{\min}} = 0$
    else
      $j_{\min} \leftarrow \arg\min_j (p(x_{i_{\min}}, y_j))$ ▷ find bending min.
    end if
    march outward from $i_{\min}$, $j_{\min}$ and enforce:
    $p_{i+1,j} \geq p_{i,j} + (y_{i+1} - y_i)\ p^x_{\min}$ ▷ $i-1$ if $i < i_{\min}$
    $p_{i,j+1} \geq p_{i,j} + (x_{i+1} - x_i)\ p^x_{\min}$ ▷ $j-1$ if $j < j_{\min}$
    $p^y_{ij} \geq p^x_{\min}$ ▷ $p^y_{ij} \leq -p^x_{\min}$ if $i < i_{\min}$
    $p^x_{ij} \geq p^x_{\min}$ ▷ $p^x_{ij} \leq -p^x_{\min}$ if $j < j_{\min}$
  ENSURE INCREASING EXTRAPOLATION:
    clamp p^x and p^y on boundary
    $p^{xy} \leftarrow 0$ on boundary
  apply monotone bicubic algorithm ▷ Appendix B
  CONVERT TO RESIDUAL: ▷ r(x,y) = f(x,y) - f(x,0) - f(0,y) + f(0,0)
    $p_{i,j} \leftarrow p_{i,j} - p_{y=0,j} - p_{i,x=0} + p_{y=0,x=0}$
    $p^x_{i,j} \leftarrow p^x_{i,j} - p_{y=0,j}$
    $p^y_{i,j} \leftarrow p^x_{i,j} - p_{i,x=0}$
  0-COMPRESSION RESIDUAL: ▷ if not (s_x, s_y)
    for $\{i, j\}$ compressed **do**
      $p_{i,j} \leftarrow p^x_{i,j} \leftarrow p^y_{i,j} \leftarrow p^{xy}_{i,j} \leftarrow 0$
    end for
  return $f \leftarrow \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{p}, \boldsymbol{p}^x, \boldsymbol{p}^y, \boldsymbol{p}^{xy}\}$
end function

Algorithm 6 Sample deformation ranges for fitting. All individual simulations can be run in parallel.

Input: 1D-sampling ranges $[z_{\min}, z_{\max}]$ for each strain z
2D-sampling ranges $[z_{i\min}, z_{i\max}]$, $[z_{j\min}, z_{j\max}]$ for (z_i, z_j)

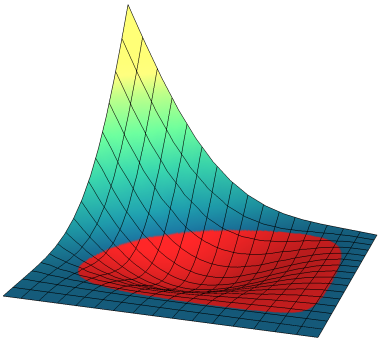
function SampleRange(z, $z_{\min}$, $z_{\max}$, N)
  if $z = s_x$ **or** $z = s_y$ **then**
    $Z \leftarrow \text{linspace}\big((z_{\min}+1)^{\frac{1}{10}}, (z_{\max}+1)^{\frac{1}{10}}, N\big)^{10} - 1$
  else
    $Z \leftarrow \text{linspace}\big(-\sqrt{\text{abs}(z_{\min})}, \sqrt{\text{abs}(z_{\max})}, N\big)$
    $Z \leftarrow \text{sgn}(Z)\, Z^2$
  end if
  return Z
end function

procedure Sample
  for z **in** $\{s_x, s_a, s_y, \text{II}_{00}, \text{II}_{11}\}$ **do**
    $Z \leftarrow$ SampleRange(z, $z_{\min}$, $z_{\max}$, 150)
    simulate each sample $\boldsymbol{z} = (0 \ldots, z_i = Z, \ldots 0)$
  end for
  for all $\{z_i, z_j\}$ **do**
    $Z_i \leftarrow$ SampleRange(z_i, $z_{i\min}$, $z_{i\max}$, 50)
    $Z_j \leftarrow$ SampleRange(z_j, $z_{j\min}$, $z_{j\max}$, 50)
    simulate each sample $\boldsymbol{z} = (0 \ldots, z_i = Z_i, z_j = Z_j, \ldots 0)$
  end for
end procedure

APPENDIX B

Piecewise Monotone Bicubic Interpolation

Here, we present the algorithm for piecewise monotone bicubic interpolation mentioned in Section 4.1.2. A python implementation is also available at `https://git.ist.ac.at/gsperl/HYLC`.



For well-behaved simulations, we require our fitted energy functions to not exhibit any new local minima other than the ones present in the data. For bivariate functions, it is not sufficient to apply monotone piecewise cubic interpolation in both directions since this can still produce non-monotone regions as shown in the inset in red. Instead, we adopt the monotone piecewise bicubic interpolation scheme of Carlson and Fritsch [1989]. Since their method as presented assumes globally monotone data, we modify it to work with arbitrary data while preserving its behavior in monotone regions.

The input to the algorithm is a grid of nodes (x_i, y_j) on which Hermite data is specified, namely values $p_{i,j}$, first derivatives $p^x_{i,j}$ and $p^y_{i,j}$, and mixed derivatives $p^{xy}_{i,j}$. For brevity, define $h_i = x_{i+1} - x_i$ and $k_j = y_{j+1} - y_j$, and define the operators Δ^x and Δ^y such that

$$\Delta^x f_{i,j} = f_{i+1,j} - f_{i,j}, \tag{B.1a}$$

$$\Delta^y f_{i,j} = f_{i,j+1} - f_{i,j} \tag{B.1b}$$

for any nodal data f. Our goal is to modify the specified derivatives so that, in regions where the data $p_{i,j}$ is monotone, the resulting piecewise bicubic Hermite interpolation is also monotone with the same sense.

First, we must define local monotonicity of the data. We declare a horizontal edge $E^x_{i,j} = [x_i, x_{i+1}] \times y_j$ to be *increasing* if $\Delta^x p_{i,j} \geq 0$ and *decreasing* if $\Delta^x p_{i,j} \leq 0$, and similarly define monotonicity for vertical edges $E^y_{i,j} = x_i \times [y_j, y_{j+1}]$. Now we consider a cell $R_{i,j} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$. We declare the cell to be *increasing in x* if the adjacent horizontal edges $E^x_{i,j}$ and $E^x_{i,j+1}$ are both increasing, *decreasing in x* if they are both decreasing, and *nonmonotone in x* otherwise. Similarly we define monotonicity in y using vertical edges $E^y_{i,j}$ and $E^y_{i+1,j}$.

Now, we review Carlson and Fritsch's sufficient conditions for monotonicity. Without loss of generality, we only consider monotonicity in x, and further that the sense of monotonicity is

increasing; decreasingness only requires reversing all the inequalities. Increasingness along an edge $E^x_{i,j}$ is ensured if

$$0 \le p^x_{i,j} \le 3\frac{\Delta^x p_{i,j}}{h_i}, \tag{B.2a}$$

$$0 \le p^x_{i+1,j} \le 3\frac{\Delta^x p_{i,j}}{h_i}. \tag{B.2b}$$

Increasingness in x over a region $R_{i,j}$ is ensured by constraining differences in y-derivatives along adjacent horizontal edges,

$$\Delta^x p^y_{i,j} \ge -3\frac{\Delta^x p_{i,j}}{k_j}, \tag{B.3a}$$

$$\Delta^x p^y_{i,j+1} \le 3\frac{\Delta^x p_{i,j+1}}{k_j}, \tag{B.3b}$$

and constraining mixed derivatives at adjacent nodes,

$$-3\frac{p^x_{i,j}}{k_j} \le p^{xy}_{i,j} \le 3\left(\frac{\Delta^x p^y_{i,j}}{h_i} + 3\frac{\Delta^x p_{i,j}}{h_i k_j} - \frac{p^x_{i,j}}{k_j}\right), \tag{B.4a}$$

$$-3\frac{p^x_{i+1,j}}{k_j} \le p^{xy}_{i+1,j} \le 3\left(\frac{\Delta^x p^y_{i,j}}{h_i} + 3\frac{\Delta^x p_{i,j}}{h_i k_j} - \frac{p^x_{i+1,j}}{k_j}\right), \tag{B.4b}$$

$$3\left(\frac{\Delta^x p^y_{i,j+1}}{h_i} - 3\frac{\Delta^x p_{i,j+1}}{h_i k_j} + \frac{p^x_{i,j+1}}{k_j}\right) \le p^{xy}_{i,j+1} \le 3\frac{p^x_{i,j+1}}{k_j}, \tag{B.4c}$$

$$3\left(\frac{\Delta^x p^y_{i,j+1}}{h_i} - 3\frac{\Delta^x p_{i,j+1}}{h_i k_j} + \frac{p^x_{i+1,j+1}}{k_j}\right) \le p^{xy}_{i+1,j+1} \le 3\frac{p^x_{i+1,j+1}}{k_j}. \tag{B.4d}$$

To guarantee that the above constraints always have a solution, we further require that 0 is a valid value for each mixed derivative p^{xy}. This leads to

$$\Delta^x p^y_{i,j} \ge -\frac{1}{k_j}\left(3\Delta^x p_{i,j} - h_i \max(p^x_{i,j}, p^x_{i+1,j})\right), \tag{B.5a}$$

$$\Delta^x p^y_{i,j+1} \le \frac{1}{k_j}\left(3\Delta^x p_{i,j+1} - h_i \max(p^x_{i,j+1}, p^x_{i+1,j+1})\right). \tag{B.5b}$$

We now develop the algorithm for satisfying the constraints. The first step is to modify the first derivatives p^x, p^y to satisfy the inequalities (B.2) arising from edges, and (B.3) and (B.5) arising from monotone cells. Then, we modify the mixed derivatives p^{xy} to satisfy the inequalities (B.4) arising from monotone cells, which will always be possible thanks to (B.5). The entire algorithm requires only implementing operations for enforcing monotonicity in x; these can then be used for monotonicity in y as well by flipping the coordinates (i.e. transposing all grids and swapping p^x with p^y).

As in Carlson and Fritsch's original algorithm, we first satisfy (B.2) by clamping p^x and p^y. As long as we only shrink them towards zero in subsequent operations, (B.2) will remain satisfied.

Next, both sets of remaining constraints on first derivatives act on their "mixed differences" $\Delta^x p^y$ and $\Delta^y p_x$. Here, unlike the original algorithm, some more care is needed since adjacent

regions may differ in monotonicity. Consider a horizontal edge $E^x_{i,j}$ and suppose it is increasing. Then the adjacent cells $R_{i,j-1}$ and $R_{i,j}$ cannot be decreasing in x; they are either increasing or nonmonotone in x. Then from (B.3) we have

$$-3\frac{\Delta^x p_{i,j}}{k_j} \le \Delta^x p^y_{i,j} \le 3\frac{\Delta^x p_{i,j}}{k_{j-1}}, \tag{B.6}$$

where the first inequality arises if $R_{i,j}$ is increasing in x, and the second if $R_{i,j-1}$ is increasing in x. Nevertheless, there is no harm in imposing both bounds even if the adjacent cells are nonmonotone, since 0 is always a feasible value due to the increasingness of $E^x_{i,j}$. Similarly, from (B.5) we have

$$-\frac{b_{i,j}}{k_j} \le \Delta^x p^y_{i,j} \le \frac{b_{i,j}}{k_{j-1}} \tag{B.7}$$

where $b_{i,j} = 3\Delta^x p_{i,j} - h_i \max(p^x_{i,j}, p^x_{i+1,j})$, which we again impose regardless of monotonicity of $R_{i,j-1}$ and $R_{i,j}$, because 0 is always a feasible value thanks to (B.2).

We now discuss how to satisfy these constraints without violating (B.2). Note that they are all of the form $l \le m_1 - m_0 \le u$ where $l \le 0 \le u$. For any constraint, given the current values (m_0, m_1), we project to the closest values (m_0^*, m_1^*) which satisfy the constraints as well as $|m_i^*| \le |m_i|$ and $\operatorname{sgn}(m_i^*) = \operatorname{sgn}(m_i)$. The solution is given in closed form by several cases, illustrated in Figure B.1. Since this projection may cause the constraints on adjacent edges to be violated, we must make multiple sweeps across the grid. Following Carlson and Fritsch, we make two sweeps across the grid, first in increasing order in x and then in decreasing order, applying projections to enforce the $\Delta^x p^y$ constraints on each edge. We then do the same in y to enforce the $\Delta^y p^x$ constraints, albeit using the *old* values of p^y so that the results are order-independent.

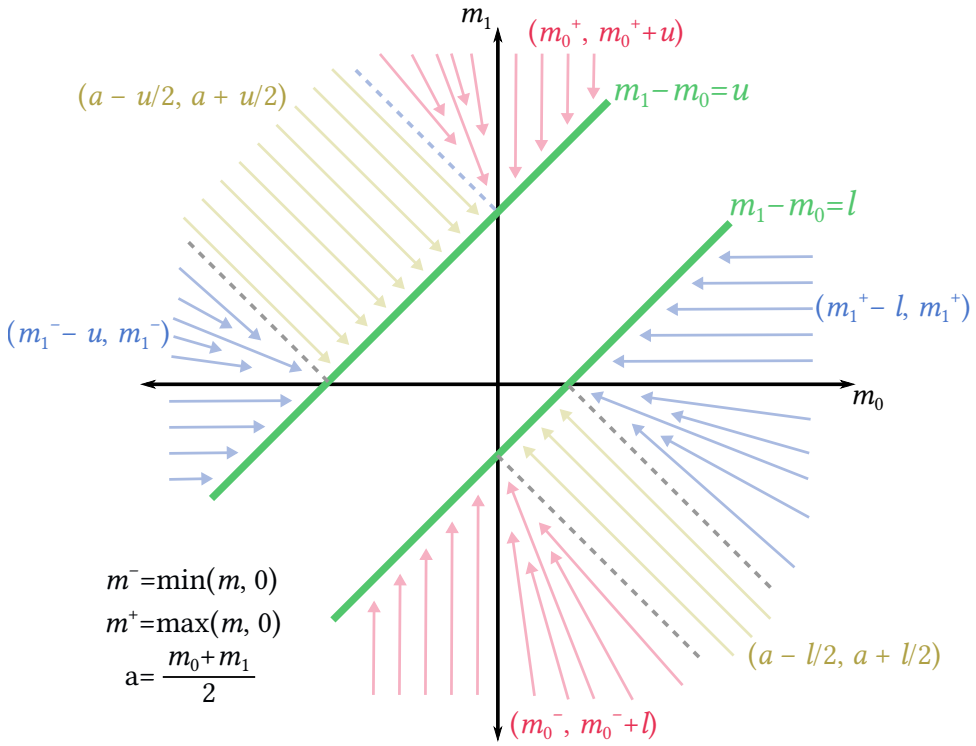


Figure B.1: We want to project (m_0^*, m_1^*) to the central diagonal slab $l \le m_1 - m_0 \le u$ (green) without increasing either entry's absolute value. There are six cases (not including if the initial value is already feasible), which we show as bundles of arrows alongside their projected values.

Finally, the mixed derivative p^{xy} at each node is subject to constraints (B.4) from the adjacent monotone cells. For monotonicity in x, the constraints acting on $p^{xy}_{i,j}$ are

$$-3\frac{p^x_{i,j}}{k_j} \le p^{xy}_{i,j} \le 3\left(\frac{\Delta^x p^y_{i,j}}{h_i} + 3\frac{\Delta^x p_{i,j}}{h_i k_j} - \frac{p^x_{i,j}}{k_j}\right), \tag{B.8a}$$

$$-3\frac{p^x_{i,j}}{k_j} \le p^{xy}_{i,j} \le 3\left(\frac{\Delta^x p^y_{i-1,j}}{h_{i-1}} + 3\frac{\Delta^x p_{i-1,j}}{h_{i-1} k_j} - \frac{p^x_{i,j}}{k_j}\right), \tag{B.8b}$$

$$3\left(\frac{\Delta^x p^y_{i,j}}{h_i} - 3\frac{\Delta^x p_{i,j}}{h_i k_{j-1}} + \frac{p^x_{i,j}}{k_{j-1}}\right) \le p^{xy}_{i,j} \le 3\frac{p^x_{i,j}}{k_{j-1}}, \tag{B.8c}$$

$$3\left(\frac{\Delta^x p^y_{i-1,j}}{h_{i-1}} - 3\frac{\Delta^x p_{i-1,j}}{h_{i-1} k_{j-1}} + \frac{p^x_{i,j}}{k_{j-1}}\right) \le p^{xy}_{i,j} \le 3\frac{p^x_{i,j}}{k_{j-1}} \tag{B.8d}$$

if the adjacent cells $R_{i,j}$, $R_{i-1,j}$, $R_{i,j-1}$, and $R_{i-1,j-1}$ respectively are increasing in x, and with the corresponding inequalities reversed if they are decreasing in x instead. Observe that each constraint above only involves quantities along a single edge, namely $E^x_{i,j}$, $E^x_{i-1,j}$, $E^x_{i,j}$, and $E^x_{i-1,j}$ respectively. Furthermore, if said edge is increasing, and we have satisfied all four constraints on its mixed difference $\Delta^x p^y$, then the constraint always has 0 as a feasible value. Therefore, it is safe to enforce each of the constraints on $p^{xy}_{i,j}$, regardless of the monotonicity of the cell it arises from, remembering only to reverse the inequalities if the relevant *edge* is decreasing instead of increasing. We do so simply by clamping $p^{xy}_{i,j}$ to lie between the two bounds of each constraint. Then we repeat the procedure on the flipped data to enforce monotonicity in y.

The algorithm presented here is equivalent to the original algorithm for globally monotone data [Carlson and Fritsch 1989]. It also guarantees monotonicity on each cell, as long as there are no cells with nonmonotone data. We have not yet conducted any analysis of what guarantees, if any, are available for cells that are nonmonotone in both directions, such as $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

APPENDIX C

Mechanics-Aware Deformation Details

This chapter provides details for Chapter 5. Here we discuss material-space pullback of yarn displacements, the experimental bending models, as well as several implementation details such as rendering. For further implementation details, please refer to the code in the public repository: `https://git.ist.ac.at/gsperl/MADYPG`.

C.1 Pullback Into Material Space

Here, we detail the Newton iteration to transform optimized yarn geometry from Section 5.1 back into material space. The goal is to find $\hat{\boldsymbol{X}}$ s.t. $\boldsymbol{x} = \overline{\boldsymbol{x}}(\hat{\boldsymbol{X}})$, which we do using Newton's method on $\boldsymbol{f}(\hat{\boldsymbol{X}}) = \boldsymbol{x} - \overline{\boldsymbol{x}}(\hat{\boldsymbol{X}})$, for which we need the gradient $\nabla \boldsymbol{f}$.

The mapping $\overline{\boldsymbol{x}}$ is defined as

$$\overline{\boldsymbol{x}}(\hat{\boldsymbol{X}}) = \boldsymbol{\varphi}(\hat{X}_1, \hat{X}_2) + \hat{X}_3\, \boldsymbol{n}(\hat{X}_1, \hat{X}_2), \tag{C.1}$$

where $\boldsymbol{\varphi}$ is the deformed midsurface and $\boldsymbol{n}$ its normal. Its gradient is

$$\nabla \overline{\boldsymbol{x}}(\hat{\boldsymbol{X}}) = \left[\nabla \boldsymbol{\varphi}(\hat{X}_1, \hat{X}_2) + \hat{X}_3\, \nabla \boldsymbol{n}(\hat{X}_1, \hat{X}_2), \quad \boldsymbol{n}(\hat{X}_1, \hat{X}_2)\right]. \tag{C.2}$$

By definition we have

$$\nabla \boldsymbol{\varphi} = \boldsymbol{R}\boldsymbol{S}, \tag{C.3}$$

and we compute $\nabla \boldsymbol{n} = -\left(\sqrt{\overline{\mathbf{I}}}\,\right)^{-\top} \overline{\mathbf{II}}$ (see also (3.14) and (A.12)). The Newton iterations are

$$\hat{\boldsymbol{X}}_{i+1} = \hat{\boldsymbol{X}}_i - \nabla \boldsymbol{f}(\hat{\boldsymbol{X}}_i)^{-1} \boldsymbol{f}(\hat{\boldsymbol{X}}_i) \tag{C.4}$$

starting from the rest configuration $\hat{\boldsymbol{X}}_0 = \boldsymbol{X}$.

In our experiments, the iterations converge to a fraction of the yarn radius within three iterations and the cost is negligible compared to the elastostatic optimization. For pure in-plane deformations, $\mathbf{II} = \mathbf{0}$, and thus $\boldsymbol{R} = \mathrm{Id}$ and $\nabla \boldsymbol{f} = -\begin{bmatrix} \sqrt{\mathbf{I}} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$, simplifying the pullback into the constant expression

$$\hat{\boldsymbol{X}} = \boldsymbol{X} + \begin{bmatrix} \left(\sqrt{\mathbf{I}}\right)^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\boldsymbol{u}}. \tag{C.5}$$

C.2 Bending Models

We briefly outline the two bending models we experimented with: a 4D combination of in-plane and bending deformations (C.6), and the linearization of bending as stretching (C.9).

Due to tileability constraints, in Chapter 4 we only generate data for singly curved deformations and split the contributions of general bending onto data sampled for bending along either the X_1 or X_2 directions. Here, we apply the same idea to define our "ground-truth" bending model. For brevity, we write $\boldsymbol{s} = (s_x, s_a, s_y)$. Then, we compute two sets of data for combined in-plane deformation with the two bending directions: $\Delta\boldsymbol{Q}_{X_1}(\boldsymbol{s}, \lambda)$ and $\Delta\boldsymbol{Q}_{X_2}(\boldsymbol{s}, \lambda)$. With the eigenvalues λ_1 and λ_2 of $\mathbf{II}$ and the cosine c of the angle between the X_1 axis and the eigenvector corresponding to λ_1, we distribute the bending contributions to get

$$\begin{aligned}\Delta\boldsymbol{Q}(\boldsymbol{s}, \mathbf{II}) = {} & c^2\,\Delta\boldsymbol{Q}_{X_1}(\boldsymbol{s}, \lambda_1) + (1-c^2)\,\Delta\boldsymbol{Q}_{X_1}(\boldsymbol{s}, \lambda_2) \\ & + c^2\,\Delta\boldsymbol{Q}_{X_2}(\boldsymbol{s}, \lambda_2) + (1-c^2)\,\Delta\boldsymbol{Q}_{X_2}(\boldsymbol{s}, \lambda_1) \\ & - \Delta\boldsymbol{Q}_{X_1}(\boldsymbol{s}, 0)\end{aligned} \tag{C.6}$$

Since both data sets include the displacements caused by pure in-plane deformations $\lambda_1 = \lambda_2 = 0$, we have to subtract these displacements once to avoid double counting. This model exactly reproduces the data samples for bending aligned with either X_1 or X_2. However, it requires five 4D-texture look-ups and is thus much more expensive than the linearized model requiring only a single 3D-texture look-up. For the speed comparison mentioned in Section 5.3, we used two sets of 9^4 samples for the 4D model and 9^3 samples for the linearized model. The linearized model was still 7× faster even when increasing its sample density to 15^3.

C.2.1 Linearized Bending

For a thin shell represented by a midsurface $\boldsymbol{\varphi}$ with normal $\boldsymbol{n}$, its full domain is

$$\boldsymbol{x} = \boldsymbol{\varphi} + h\,\boldsymbol{n}, \tag{C.7}$$

with the normal coordinate $h \in [-H/2, H/2]$ for shell thickness H. Its right Cauchy-Green deformation tensor is

$$\nabla\boldsymbol{x}^\top\nabla\boldsymbol{x} = \nabla\boldsymbol{\varphi}^\top\nabla\boldsymbol{\varphi} + h\,(\nabla\boldsymbol{\varphi}^\top\nabla\boldsymbol{n} + \nabla\boldsymbol{n}^\top\nabla\boldsymbol{\varphi}) + O(h^2), \tag{C.8}$$

which can be interpreted as a first fundamental form $\mathbf{I}(h)$ depending quadratically on h. The quadratic term is commonly neglected (e.g. [Kiendl et al. 2015]). Additionally, we see that $\nabla\boldsymbol{\varphi}^\top\nabla\boldsymbol{\varphi} = \mathbf{I}$ and similarly $\nabla\boldsymbol{\varphi}^\top\nabla\boldsymbol{n} = \nabla\boldsymbol{n}^\top\nabla\boldsymbol{\varphi} = -\mathbf{II}$ are the fundamental forms of $\boldsymbol{\varphi}$. As a result, we get the linearized expression

$$\mathbf{I}(h) \approx \mathbf{I} - 2\,h\,\mathbf{II}. \tag{C.9}$$

Then, with precomputed data $\Delta\boldsymbol{Q}_{\boldsymbol{s}}(\boldsymbol{s})$ for in-plane deformations $\boldsymbol{s}$, the linearized bending model is

$$\Delta\boldsymbol{Q}(\mathbf{I}, \mathbf{II}) = \Delta\boldsymbol{Q}_{\boldsymbol{s}}(\boldsymbol{s}(\mathbf{I} - 2\,h\,\mathbf{II})). \tag{C.10}$$

C.3 Removal of Short Yarn Fragments

After tiling the periodic yarn pattern over the mesh as in Section 5.2, yarns near mesh boundaries may be very short. This may generate tiny floating yarn fragments as illustrated in Figure C.1, which we want to remove.

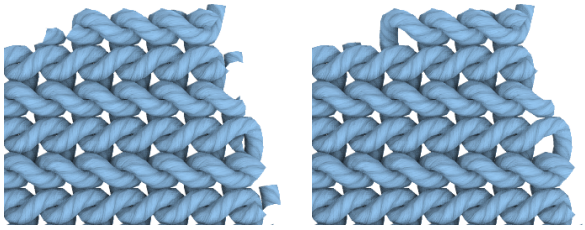


Figure C.1: Tiled yarn geometry before (left) and after (right) removing short yarn fragments.

The yarn patterns used in the optimization of Section 5.1 also store rest lengths $\bar{l}_i$ of edges i. During tiling, we copy these rest lengths along with the other pattern data. After tiling, we can now traverse connected yarns and accumulate these rest lengths to compute a parameter

$$a_0 = 0, \qquad a_i = \sum_{j=1}^{i} \bar{l}_{j-1}. \tag{C.11}$$

Here, we use the final value of a on a yarn to remove yarns that are smaller than some user-specified minimum length. We also use a during rendering of ply twists, since it provides a parametrization that is independent of local stretching (and so stretching a yarn segment would not change the amount of twisting). Note that in general the rest lengths are different from the edge lengths in the pattern geometry corresponding to $\mathbf{I} = \mathrm{Id}$, $\mathbf{II} = \mathbf{0}$, where the yarns may locally still exhibit strains, e.g. tension balancing out collisions.

C.4 Rendering Details

In this section, we provide detail on world-space mapping of edge normals, the tessellation of yarn centerlines into cylindrical geometry including yarn twists, our volume-preserving radius-scaling heuristic, and on the twistable ply- and fiber-level detail texture maps.

C.4.1 Transformation of Edge Normals

Yarn twists are defined with a twist variable θ and reference directors $\underline{\boldsymbol{d}}_1$ (edge normals) per edge. In Section 5.2, we displace quantities from a material-space configuration $\boldsymbol{Q}$ to a deformed material-space configuration $\hat{\boldsymbol{Q}}$ and then map that to world-space values $\boldsymbol{q}$ along with the mesh. The twist variables transform trivially (simply copying values), since the edge normals take care of the actual orientation of the twisting frame. In principle, this means that we would have to first transform $\underline{\boldsymbol{d}}_1$ along with the vertex-displacements ($\hat{\boldsymbol{X}} = \boldsymbol{X} + \Delta\boldsymbol{X}$) and then along with the world-space mapping $\boldsymbol{x}$ from (5.9), which may be nonlinear (due to Phong deformation [James 2020]). Similar to how normals transform with normal matrices, $\underline{\boldsymbol{d}}_1$ get transformed by the inverse transpose of the Jacobian $\boldsymbol{J}^{-\top}$ of the total mapping $\boldsymbol{X} \to \boldsymbol{x}$. We efficiently approximate the transformation with

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{F}, & \boldsymbol{n} \end{bmatrix}, \tag{C.12}$$

where $\boldsymbol{F}$ is the underlying triangle's 3×2 deformation gradient and $\boldsymbol{n}$ is the interpolated mesh-vertex normal. We apply this transformation to the initial values of $\underline{\boldsymbol{d}}_1$ from the pattern tiling. This approximation is cheap compared to the full Jacobian including Phong Deformation and local displacements. During rendering, we simply reorthonormalize the transformed edge normals with respect to yarn centerline tangents.

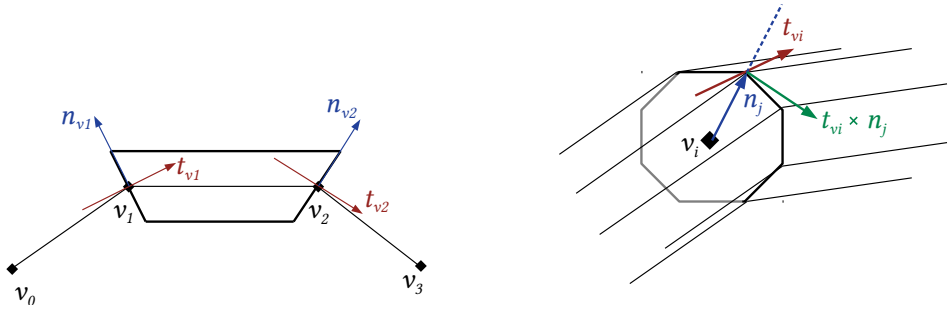


Figure C.2: Top: Four-vertex input primitive for the geometry shader. A yarn segment is tessellated for the middle edge, using averaged per-vertex tangents $\boldsymbol{t}_{v_i}$ and normals $\boldsymbol{n}_{v_i}$. Bottom: Cylinders are generated from tessellated cross-sections per centerline vertex vi, with tangential frames $\boldsymbol{M}_j = [\boldsymbol{t}_{v_i} \times \boldsymbol{n}_j, \boldsymbol{t}_{v_i}, \boldsymbol{n}_j]$.

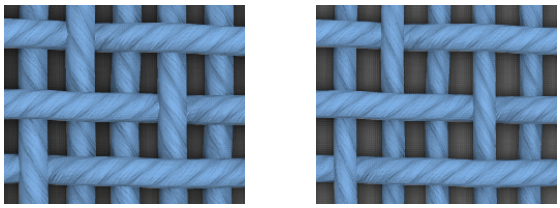


Figure C.3: Tessellated yarn geometry without (left) and with (right) volume-preserving radius scaling. In our experiments, we found the effect to be small such as shown here.

C.4.2 Geometry Tessellation

We use a geometry shader to tessellate cylindrical yarn meshes. The inputs are a global list of yarn vertices and index lists for each yarn. Each yarn vertex has the following data: a position $\boldsymbol{x}$, the twist θ of it's outgoing incident edge, a transformed normal $\boldsymbol{n} = \boldsymbol{J}^{-\top}\underline{\boldsymbol{d}}_1$ of the same edge using (C.12), the cumulative rest length parameter a up until that vertex, as well as a copy of the original undeformed material-space coordinates for cloth-level texturing. Since the material space of the mesh is encoded as uv-coordinates and the yarn pattern is tiled over that space, this is a natural way of texturing or coloring a garment.

The vertices are passed to the geometry shader stage as primitives of four consecutive yarn vertices, from which we consistently tessellate the middle edge such that the adjacent primitives seamlessly connect, as illustrated in Figure C.2. To do so, we push the edge-based quantities onto the vertices with rest-length-weighted averaging. The rest lengths can be computed from a or stored explicitly per edge. We average vertex tangents $\boldsymbol{t}_{v_i}$, vertex normals $\boldsymbol{n}_{v_i}$, and vertex twists θ_{v_i} for the inner two vertices $vi = v_1$ and $v_i = v_2$ of the four-vertex primitive. We orthogonalize the $\boldsymbol{n}_{v_i}$ with respect to $\boldsymbol{t}_{v_i}$, normalize both vectors and compute vertex binormals b_{v_i} as the cross-product

$$\boldsymbol{b}_{v_i} = \boldsymbol{t}_{v_i} \times \boldsymbol{n}_{v_i}. \tag{C.13}$$

Volume-Preserving Heuristic At this point, we also compute local yarn radii such that the deforming yarns preserve volume compared to their rest shape. This is a fast approximation that appears to work well as shown in Figure C.3. It would be possible to instead integrate the method of Montazeri et al. [2019] for a data-driven solution to adapting yarn cross-sections to stretching, or to use a different thickness heuristic.

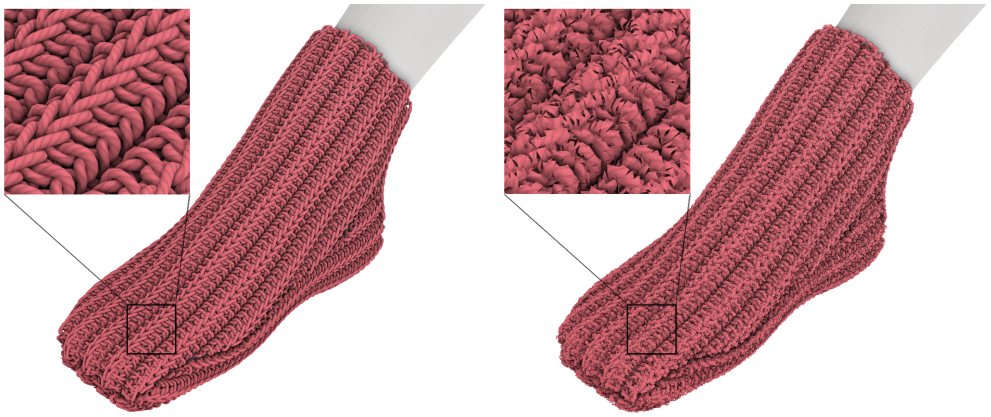


Figure C.4: Adding random displacements to tessellated yarn cylinder vertices is an efficient way of generating a fuzzy look.

For each of the three edges e_i in the geometry primitive, we compute volume-preserving radii r_{e_i} from their current length l_{e_i}, their rest lengths $\bar{l}_{e_i}$, and the base radius R:

$$r_{e_i} = R\sqrt{\bar{l}_{e_i}/l_{e_i}} \tag{C.14}$$

We additionally clamp the ratio of the modified vs. original radii such that their ratio to the original yarn radius is between 0.25 and 2.0, although this almost never takes effect for "reasonable" deformation within the realm of the strains sampled for our precomputed physical yarn data. We compute vertex radii r_{v_i} using the weighted average discussed above.

Tessellation Finally, with the rotational frames $(\boldsymbol{n}_{v_i}\ \boldsymbol{b}_{v_i})$ and the twists θ_{v_i} at the two vertices of the middle segment, we can generate the cylindrical mesh by generating a discretized circle at each vertex and connecting them with triangular faces.

We compute the position $\boldsymbol{x}_j$ of the j-th circle vertex at primitive vertex vi as

$$s = \sin(\frac{2\pi j}{N-1} + \theta_{v_i}), \tag{C.15a}$$

$$c = \sin(\frac{2\pi j}{N-1} + \theta_{v_i}), \tag{C.15b}$$

$$\boldsymbol{n}_j = c\ \boldsymbol{n}_{v_i} + s\ \boldsymbol{b}_{v_i}, \tag{C.15c}$$

$$\boldsymbol{x}_j = \boldsymbol{x}_{v_i} + r_{v_i}\ \boldsymbol{n}_j, \tag{C.15d}$$

where N is the number of vertices per circle. $\boldsymbol{n}_j$ is the normal to the cylinder at cylinder-vertex j and different from the edge normal $\boldsymbol{n}_{v_i}$ of the centerline vertex v_i. We implement the elastic rods yarn twist here as an offset to the arguments of s and c. Note that this basically corresponds to the combined computation and tessellation of "material frames" from [Bergou et al. 2010]. It is possible to perform a naive level-of-detail optimization at this stage by reducing N depending on the depth value of vertex v_i. For normal mapping, we also emit a tangent-frame matrix $\boldsymbol{M}_j$ per vertex

$$\boldsymbol{M}_j = \left[\boldsymbol{t}_{v_i} \times \boldsymbol{n}_j,\quad \boldsymbol{t}_{v_i},\quad \boldsymbol{n}_j\right], \tag{C.16}$$

We render the tessellated geometry with "matcaps" (spherical environment maps) and screen-space ambient occlusion. In Figure C.4, we briefly experimented adding fuzz detail by displacing vertices similar to [Zhong et al. 2001].

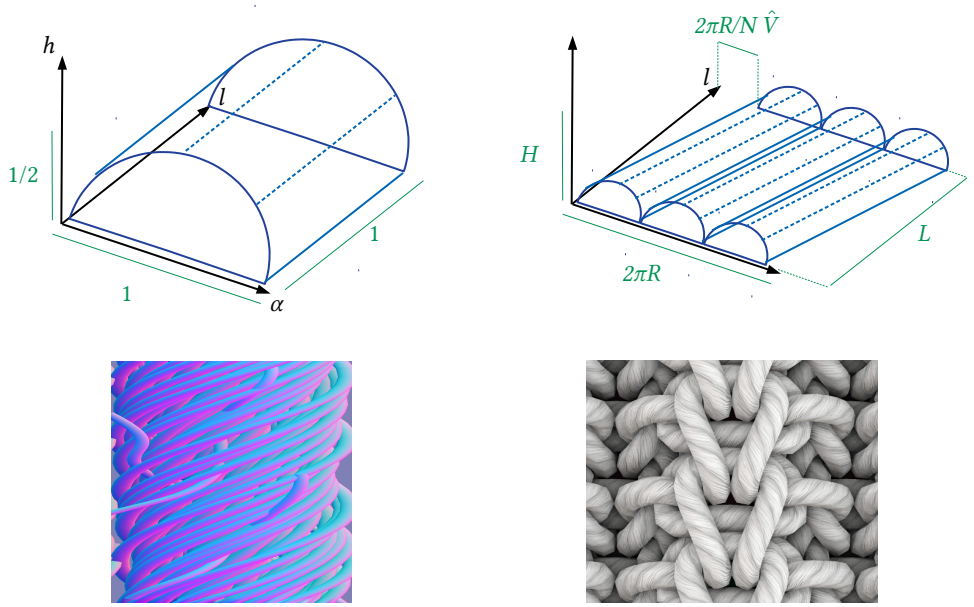


Figure C.5: Top left: the normalized single-ply space from which we bake geometry (indicated by the blue half-cylinder) into normal and ambient occlusion textures. Top right: transformation of single-ply into stretched/tiled/twisted ply space. Bottom left: our baked combined normal and ambient occlusion texture with fiber geometry for a single ply. Bottom right: yarn geometry rendered with our twistable texture maps.

C.4.3 Ply Texture Mapping

We add ply- and fiber-level detail onto the tessellated yarn geometry using *twistable* normal maps and ambient occlusion maps. The twists discussed here are separate from the discrete elastic rods twist θ, which we already incorporate into the cylinder tessellation. Instead, the twists in this section represent the helical geometry of yarn plies. We propose texture maps that can be tiled and twisted along the yarn in a parametric fashion. Notably, this twisting and tiling has an influence on the resulting normals.

The process with which we construct parametrized normals is as follows: we define ambient occlusion and normal textures for a single ply half-cylinder (Figure C.5 top left), we stretch, tile and shear these textures into an intermediate space (Figure C.5 top right), and lastly with "standard" normal mapping we map the normal into the tangent frame of our textured yarn geometry (Figure C.2).

The projection of geometry in single-ply space can be described using a height field h

$$\boldsymbol{p}_{\text{ply}}(\hat{\alpha}, \hat{l}) = \begin{pmatrix} \hat{\alpha} \\ \hat{l} \\ h(\hat{\alpha}, \hat{l}) \end{pmatrix} \tag{C.17}$$

with the "across-ply" parameter $\hat{\alpha}$ and the "along-ply" $\hat{l}$ in $[0, 1]$. This single-ply geometry is what we bake into a combined normal and occlusion texture (Figure C.5 bottom left). Specifically, we bake $\left(\frac{n_x+1}{2}, \frac{n_y+1}{2}, \text{AO}\right)$ into the RGB channels, where n_x and n_y are the components of the normals to h, and AO is the ambient occlusion value.

Next we stretch, tile, and shear the single-ply geometry into an intermediate space (corresponding to a flattening of the tessellated yarn cylinder)

$$\boldsymbol{p}_{\text{int}}(\hat{\alpha}, a) = \begin{pmatrix} 2\pi R\hat{\alpha} \\ a \\ 2H\ h(N\hat{\alpha} - a/L\ \hat{V}, a/L) \end{pmatrix} \tag{C.18}$$

with cumulative rest length a along the yarn, N the number of tiled plies, R the yarn radius (not the volume-preserving one), H the transformed height, L the transformed length, and $\hat{V}$ a twist factor.

We want a nice expression for the normal $\boldsymbol{n}_{\text{int}}$ of $\boldsymbol{p}_{\text{int}}$ using the baked single-ply texture values n_x and n_y. We redefine $L = \hat{L}R/N$ and $H = \hat{H}R$. Then with texture coordinates

$$\boldsymbol{w} = \begin{pmatrix} N\hat{\alpha} - a\ \hat{V}/(R\hat{L}) \\ a/(R\hat{L}) \end{pmatrix}, \tag{C.19}$$

the normal of p_{int} is proportional to

$$\boldsymbol{n}_{\text{int}}(\hat{\alpha}, a) \propto \begin{pmatrix} \frac{\hat{H}Nn_x}{\pi}(\boldsymbol{w}) \\ \frac{2\hat{H}N}{\hat{L}}\big(n_y(\boldsymbol{w}) - \hat{V}n_x(\boldsymbol{w})\big) \\ \sqrt{1 - n_x^2(\boldsymbol{w}) - n_y^2(\boldsymbol{w})} \end{pmatrix}. \tag{C.20}$$

To apply our transformed normal maps, during tessellation we compute $\hat{\alpha} = \frac{2\pi j}{N-1}$ from (C.15), and store $\boldsymbol{w}$ as well as $\boldsymbol{M}_j$ from (C.16) per cylinder vertex. These quantities are interpolated in the fragment shader, where we then compute $\boldsymbol{n}_{\text{int}}$ from (C.20) and map it into the tangent frame as $\boldsymbol{n}_{\text{final}} = \boldsymbol{M}\ \boldsymbol{n}_{\text{int}}$.

C.5 Other Implementation Details

Preloading Animation Data We experiment with cloth mesh animations stored as obj-file sequences. For meshes of moderate size, loading meshes each frame can become a bottleneck. Therefore, we preload mesh animations into memory before adding and animating yarn detail. Additionally, we convert and serialize several mesh animations into a binary format for faster loading.

GPU Buffers As stated in the Chapter 5, we perform all per-yarn-vertex computation on the GPU and cloth-mesh computations on the CPU. With our results based on non-remeshing cloth simulations, we can precompute material-space mesh quantities (such as shape matrices and topology-dependent interpolation weights) once on the CPU without becoming a bottleneck. For more performance and remeshing cloth simulations, we suggest to implement all mesh processing on the GPU.

For yarn vertices, we mainly use two GPU buffers: one for material-space data and one for world-space data. The material-space data basically corresponds to the initial flat tiling and triangle assignment, which we do once on the CPU. We then buffer this data onto the GPU, and allocate space for the world-space buffer. Afterwards, the yarn data remains on the GPU, getting passed directly into shaders for rendering, and thus avoiding the need for expensive data-passing between the CPU and GPU.

Mesh Interpolation Weights We use modified Shepard weights [James 2020] for strain interpolation from mesh faces to mesh vertices. We also use the same weights to compute vertex normals. To compute these weights, we use the mesh-topology in material space. Specifically, we have two sets of faces for material-space and world-space topology respectively. This way we can prohibit interpolation over uv-seams, i.e. faces that are neighbors in world space but not in material space do not count as neighbors for face-to-vertex interpolation of strains and normals.