DISS. ETH NO. 31052

EFFICIENT COMPUTATIONAL MODELS FOR FORWARD AND INVERSE ELASTICITY PROBLEMS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by

YUE LI

born 13.10.1995

accepted on the recommendation of

Prof. Dr. Stelian Coros, examiner
Dr. Bernhard Thomaszewski, co-examiner
Prof. Dr. Alec Jacobson, co-examiner
Prof. Dr. Julian Panetta, co-examiner

2025

ABSTRACT

Elasticity is at the core of many scientific and engineering applications, including the design of resilient structures and advanced materials, and the modeling of biological tissues. Simulating elastic systems poses significant computational challenges due to the inherent nonlinearity of the governing equations, which calls for efficient optimization methods to determine equilibrium states. Second-order methods are particularly attractive because of their superior convergence properties relative to first-order techniques. However, the effective use of second-order solvers requires that the underlying functions and their derivatives are sufficiently smooth and available in closed form. This smoothness can easily degrade when generalizing standard computational models to a broader set of design tasks. This thesis proposes efficient computational models that enable robust and effective simulations for physics-based modeling and the design of complex elastic systems. In chapter 2, we propose a novel fabric-like metamaterial that features persisting contacts between 3D-printed yarns. To avoid the complexities of explicit contact modeling, we adopt an Eulerian-on-Lagrangian simulation paradigm; however, current methods remain limited to straight rods. We leverage a C^2-continuous representation to allow for Newton-type minimization on naturally curved rods. Chapter 3 presents a computational paradigm for intrinsic minimization of distance-based objectives defined on triangle meshes. Although Euclidean distances meet the C^2-continuity requirement, geodesic distances on triangle meshes do not. To permit efficient second-order optimization of embedded elasticity problems, we provide analytical derivatives as well as suitable mollifiers to recover C^2-continuity. Finally, in chapter 4, we address non-smoothness issues that arise in nonlinear material design, where changes in geometry parameters can lead to discontinuous changes in simulation meshes. We employ neural networks with tailored nonlinearities as C^∞-continuous and differentiable representations to characterize the elastic properties of families of mechanical metamaterials. The resulting smooth representation enables gradient-based inverse design for various high-level design goals.

ZUSAMMENFASSUNG

Elastizität steht im Mittelpunkt vieler wissenschaftlicher und technischer Anwendungen, wie zum Beispiel des Entwurfes resilienter Strukturen und fortschrittlicher Materialien sowie der Modellierung biologischer Gewebe. Die Simulation elastischer Systeme stellt aufgrund der inhärenten Nichtlinearität der zugrunde liegenden Gleichungen erhebliche rechnerische Herausforderungen dar, was effiziente Optimierungsmethoden zur Bestimmung von Gleichgewichtszuständen erfordert. Methoden zweiter Ordnung sind aufgrund ihrer besseren Konvergenzeigenschaften im Verlgeich zu Methoden erster Ordnung besonders attraktiv. Der effektive Einsatz von Lösungsverfahren zweiter Ordnung setzt voraus, dass die zugrunde liegenden Funktionen und ihre Ableitungen hinreichend glatt und in geschlossener Form verfügbar sind. Diese Glattheit kann leicht verloren gehen wenn Standard-Rechenmodelle auf verschiedene Entwurfsaufgaben verallgemeinert werden. Diese Dissertation schlägt effiziente Rechenmodelle vor, die robuste und effektive Simulationen für physikbasierte Modellierung und den Entwurf komplexer elastischer Systeme ermöglichen. In Kapitel 2 schlagen wir ein neuartiges, textilähnliches Metamaterial vor, welches persistente Kontakte zwischen 3D-gedruckten Garnen aufweist. Um die Komplexität der expliziten Kontaktmodellierung zu vermeiden, greifen wir auf ein Eulerian-on-Lagrangian-Simulationsparadigma zurück; aktuelle Methoden sind jedoch auf gerade Stäbe beschränkt. Wir nutzen eine C^2-stetige Darstellung, um eine Newton-Typ-Minimierung auf natürlich gekrümmten Stäben zu ermöglichen. Kapitel 3 präsentiert eine Methode zur intrinsischen Minimierung von Zielfunktionen, die auf geodätischer Distanz basieren. Obwohl euklidische Distanzen die Voraussetzung der C^2-Stetigkeit erfüllen, trifft dies für die geodätische Distanz auf triangulierten Flächen nicht zu. Um eine effiziente Simulation von eingebetteter Elastizität mittels Methoden zweiter Ordnung zu ermöglichen, stellen wir analytische Ableitungen sowie geeignete Glättungskerne bereit, welche die C^2-Stetigkeit wiederherstellen. Schließlich befassen wir uns in Kapitel 4 mit nichtglatten Problemen, wie sie bei der Entwicklung nichtlinearer Materialien entstehen können, wenn Änderungen in den Geometrieparametern zu diskontinuierlichen Veränderungen der Simulationsgitter führen. Wir verwenden neuronale Netzwerke mit ausgewählten Nichtlinearitäten als C^∞-stetige und differenzierbare Repräsentationen, um die elastischen Eigenschaften einer Familie

mechanischer Metamaterialien zu charakterisieren. Die daraus resultierende glatte Darstellung ermöglicht ein gradientsbasiertes inverses Design für verschiedene übergeordnete Gestaltungsziele.

ACKNOWLEDGEMENTS

I consider the supervision of my advisors Stelian and Bernhard a genuine privilege. It's a privilege to receive this amount of guidance, supervision, and support. It's a privilege to be able to speak freely about my ideas, thoughts, and even hesitations. It's also a privilege to always be able to work on directions that interest me the most. This is a thesis that I'm proud of and absolutely enjoyed working on. It would not have been possible without the unwavering support of my advisors. Thank you Stelian and Bernhard, for equipping me with the skills to conduct research and for providing me with support and directions that are invaluable for my research career. Thank you, for all the late-night edits of overleaf documents. Thank you, Bernhard, for the amount of effort you spent on all of our projects. Finally, thank you for creating the most amazing research environment.

Thank you, Alec and Julian for joining my defense committee and for sharing your thoughts and suggestions. Thank you, Julian, for pointing out the confusion in the submitted manuscript.

My life as a PhD student wouldn't have been this enjoyable without all my peers at CRL, from master's students to postdocs and research scientists. I would like to express special thanks to Jonas, Simon, Juan, and Ronan for the insightful discussions and for generously sharing your knowledge without any hesitation. In particular, thank you, Juan, for patiently answering my endless questions throughout these five years. I would also like to thank my collaborators, Yinwei and Logan. It has been a great pleasure working with you on various projects, and I have learned so much from our collaboration. Thank you, Simon, Valentin, Logan, and Flavio, for proofreading this thesis. Thank you, Dongho, Simon, and Oliver, for providing me with emotional support when I needed it most. Thank you, Barbara and Bernadette, for the administrative support.

I am especially thankful to have my wife, Wenzhao, by my side throughout this journey. These five years have not been easy, and without your steadfast support, I would have been overwhelmed. Thank you for your companionship and, most importantly, for helping me make some of the most crucial decisions in my life. I also extend my heartfelt gratitude to my family for providing me with the opportunity to study abroad during both my master's and PhD studies.

This thesis was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 866480), and the Swiss National Science Foundation through SNF project grant 200021_200644.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

The natural world exhibits a profound level of complexity, evident in both the complex architectures of biological systems and the sophisticated innovations of human engineering. Computational models play a critical role in helping us unravel this complexity. These models provide deeper insights into the underlying mechanisms of the world and facilitate the systematic testing and refinement of new design concepts prior to their physical implementation. One crucial component of this complexity is elasticity, which both natural systems and engineered structures exploit to perform vital functions. For instance, biological tissues—such as tendons and muscles—leverage their elastic properties to store and release energy efficiently during movement, while engineered materials utilize elasticity to create flexible structures in applications such as soft robotics, adaptive structural components, and resilient architectural designs.

However, elasticity problems are often complicated by various types of nonlinearities. For instance, geometric nonlinearities—characterized by large deformations and contact—and material nonlinearities—stemming from complex material laws—can significantly influence system behavior. Therefore, obtaining the equilibrium states often leads to challenging optimization problems. Consequently, simulation and design in this context require dedicated computational models that are both robust and efficient. Furthermore, to facilitate design exploration (*forward* problem) and parameter optimization (*inverse* problem), numerical simulations must provide rapid feedback with sufficient accuracy. Second-order methods are particularly attractive due to significantly improved convergence compared to their first-order counterparts. However, albeit more powerful and efficient, second-order optimization methods demand at least C^2 continuity of the optimization objective w.r.t. its variables. Unfortunately, continuity can be compromised when generalizing standard computational models to a broader set of simulation and design tasks.

Consider simulating an interwoven rod network: piece-wise linear discretization is sufficient to predict the behavior of individual rods accurately [1, 2]. However, when simulating interactions such as multiple rods sliding on each other, as shown in Fig.1.1, *a*, discontinuities at vertices pre-

vent the use of second-order optimization methods. Consequently, existing work is limited to straight rods [3].

A similar challenge is encountered with triangle meshes, which are widely used for simulating shells and serve as fundamental building blocks in geometry processing. Nonetheless, when simulations are constrained to lie strictly on triangle meshes, discontinuities can occur. Whereas Euclidean distance is C^2-continuous in the ambient space, intrinsic distances, *e.g.* geodesic distances are not C^2-continuous at vertices of the embedded mesh (Fig. 1.1, *b-c*). Furthermore, the absence of closed-form second derivatives further prevents efficient second-order optimization.

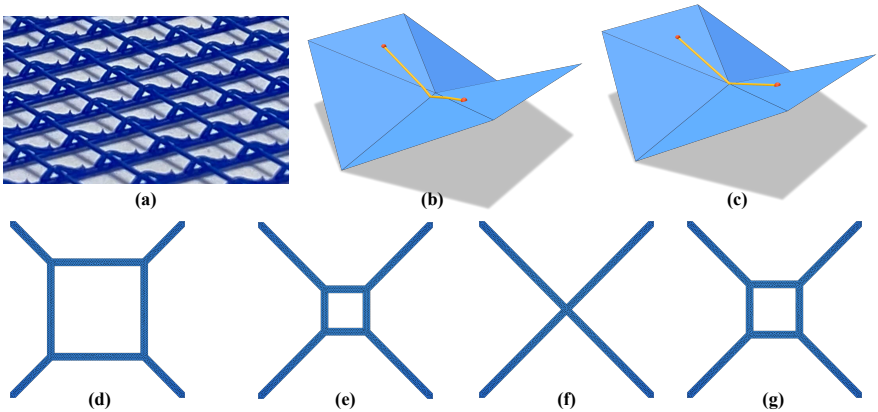


FIGURE 1.1: Non-smoothness appears with standard simulation models when, *e.g.*, rods slide on each other (*a*), a geodesic curve passes through embedded mesh vertices (*b-c*) or changing the geometry of a single-parameter metamaterial (*d-g*).

Undesired discontinuities for forward simulations are one problem, even in cases where triangle meshes perform adequately in forward simulations, they may still present challenges for inverse problems—tasks that involve determining the design parameters yielding optimal performance under specific criteria and constraints. Although the physical problem itself might not be discontinuous, the discretization can change with varying design parameters, introducing undesirable non-smoothness. In Fig. 1.1, *d-g*, we visualize the simulation meshes of a single-parameter metamaterial under different parameters. As the inner square vanishes, the mesh topology undergoes a discontinuous change. This topology singularity can lead to discontinuous gradients for inverse design.

To address these challenges, this thesis examines forward and inverse elasticity problems across three different contexts and develops efficient, dedicated computational models tailored to each. In Chapter 2, we propose a novel 3D-printable textile-like metamaterial that utilizes sliding connections (small-scale channels) to achieve tunable nonlinear and anisotropic stiffness profiles. The geometry of the channels restricts the movement of the strands, leading to persistent contacts. To avoid modeling these contacts explicitly, we adopt an Eulerian-on-Lagrangian framework and generalize it to naturally curved rods by parameterizing the rest shape with C^2-continuous, planar, interpolating splines. In Chapter 3, we generalize this EoL computational paradigm where one set of coordinate systems is embedded in another to a fully three-dimensional, Lagrangian-on-Lagrangian setting. These problems frequently occur when modeling natural phenomena *e.g.* leaf venation, the wing patterns of flies, giraffe coat markings, *etc.*, with Voronoi diagrams. Simulating these phenomena as well as solving for other minimization problems that call for intrinsic distances on discrete surfaces challenges the robustness of existing methods. To this end, we propose a novel framework that efficiently computes the analytical derivatives of geodesic distance and an intrinsic minimization framework that allows for higher-order methods for solving forward and inverse elasticity problems on triangle meshes. Finally, in Chapter 4, we leverage recent advances in neural representations to facilitate nonlinear material design. In this work, we propose Neural Metamaterial Networks—smooth neural representations that encode the nonlinear mechanics of families of metamaterials. Though trained on simulation data, the resulting neural representations do not inherit the discontinuities arising from topological changes in simulation meshes. They instead provide a smooth map from parameter to performance space that is fully differentiable and thus well-suited for gradient-based optimization.

All contributions presented in this thesis are made openly available to promote reproducibility and encourage further research in the field.

PUBLICATIONS The work presented in this thesis has previously been published as:

1. Li, Y., Montes, J., Thomaszewski, B. & Coros, S. Programmable digital weaves. *IEEE Robotics and Automation Letters* **7**, 2891 (2022).
2. Li, Y., Coros, S. & Thomaszewski, B. Neural Metamaterial Networks for Nonlinear Material Design. *ACM Transactions on Graphics (TOG)* **42**, 1 (2023).

3. Li, Y., Numerow, L., Thomaszewski, B. & Coros, S. Differentiable Geodesic Distance for Intrinsic Minimization on Triangle Meshes. *ACM Transactions on Graphics (TOG)* **43**, 1 (2024).

The following papers were also published during the course of the PhD:

1. Du, Y., Li, Y., Coros, S. & Thomaszewski, B. Robust and Artefact-Free Deformable Contact with Smooth Surface Representations. in *Computer graphics forum* **43** (2024).
2. Numerow, L., Li, Y., Coros, S. & Thomaszewski, B. Differentiable Voronoi Diagrams for Simulation of Cell-Based Mechanical Systems. *ACM Transactions on Graphics (TOG)* **43**, 1 (2024).
3. Zehnder, J., Li, Y., Coros, S. & Thomaszewski, B. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems* **34**, 10368 (2021).

2

PART I: PROGRAMMABLE DIGITAL WEAVES

Elastic lattice-like materials whose structures can be tuned to achieve desired mechanical properties hold great promise for many applications in robotics. However, existing methods rely on all-fused connections at lattice nodes, which limits the range of mechanical properties that can be achieved. In this work, we introduce Programmable Digital Weaves—3D-printable, textile-like materials with sliding connections that mimic the yarn structure of conventional woven fabrics. Our method allows for relative motion between crossing strands, thus unlocking a large space of programmable nonlinear and anisotropic material behavior. As we demonstrate through a set of virtual and physical experiments, this new concept extends to a wide variety of patterns, ranging from regular arrangements of orthogonal strands similar to conventional woven fabrics to curved yarns, interlacing closed loops, and dedicated strands for actuation.

2.1 INTRODUCTION

Structured sheet materials with programmable mechanical properties have many applications in soft robotics, wearable haptics, and medical devices [4]. These sheets are typically implemented as planar networks of elastic beams whose parameters—edge shape and orientation, network density and connectivity—can be controlled to create materials with tailored stiffness profiles [5]. While these custom materials can readily be manufactured using laser cutting or 3D printing, existing methods rely on all-fused connections, which prohibits relative motion between incident beams and, consequently, limits the range of deformation and mechanical properties that can be achieved.

In this work, we extend the concept of structured sheet materials to programmable digital weaves (PDWs): 3D-printable, textile-like materials with sliding connections that mimic the yarn structure of conventional woven fabrics. The internal degrees of freedom of programmable digital weaves allow for extreme contrast in directional tensile stiffness, and we show that a broad range of mechanical behavior—from highly anisotropic

to quasi-isotropic—can be obtained through strategic assignment of fused and slip connections.

On a technical level, we implement sliding connections using small-scale channels that allow for relative tangential motion between crossing strands while preventing separation; see Fig. 2.2. This approach lends itself to a one-shot fully-automated fabrication process that only requires a consumer-level fused filament 3D printer, which is controlled through custom G-code instructions.

As we show through our experiments, this methodology allows for a wide variety of patterns, ranging from regular arrangements of orthogonal strands similar to conventional woven fabrics to curved yarns, interlacing closed loops, and dedicated fibers for actuation. To predict the performance of such complex weaves before manufacturing, we further propose a dedicated computational model based on an existing yarn-level simulation code.

To explore the design space offered by programmable digital weaves, we perform simulation experiments for various pattern and connection parameters. We quantitatively analyse the impact of these design parameters on macromechanical performance using simulation-based homogenization. To test the feasibility of our designs, we fabricate a set of 3D-printed prototypes and observe good agreement between our simulation results and real-world behavior.

This chapter is based on our previous publication [6] and the supplementary video is linked in the footnote [1].

2.2 RELATED WORK

The design of materials with tailored mechanical properties is of great importance in the field of soft robotics. For example, custom materials can give robots increased flexibility, strength and protection, as well as enabling safer interaction with humans.

MECHANICAL METAMATERIALS Mechanical metamaterials achieve desired macro-mechanical properties by virtue of architected micro- or meso-structure. For conciseness, we only discuss work on planar metamaterials—or metasheets—and refer to Bertoldi *et al.* [7] for an extensive review of flexible mechanical metamaterials. Focusing on isohedral tilings, a particular class of patterns, Schumacher *et al.* [5] characterized the macro-

1 https://www.youtube.com/watch?v=OQfsXoXeAwg

mechanical behaviour of 3D-printed structured sheets using data-driven homogenization. To generate metasheets with locally varying material properties, Martinez *et al*. [8] introduce generalized Voronoi tilings based on star-shaped metrics that allow for continuous transitions between different patterns. With a similar goal in mind, Djourachkovitch *et al*. [9] address the problem of decoupling cell geometry from mechanical properties. In comparison to our channeled connections, these structures are more limited in the range of deformation they can achieve for the same applied force.

3D-PRINTED WEAVES Whereas 3D-printed mechanical metasheets have been studied intensively, only few works have attempted to emulate the yarn structure of woven fabrics. Takahashi and Kim [10] describe a process for creating digital textiles by routing 3D-printed thread through a jig, i.e., a set of vertical pillars. The resulting undulation in the strands leads to stretchable sheet materials with controllable thickness and interesting visual properties. Compared to our planar fabrication process, however, their vertical weaving approach limits flexibility in terms of yarn shapes and layouts. Similar in spirit, Forman *et al*. [11] exploit an under-extrusion artifact of fused filament fabrication printers to create thin, tulle-like sheets with high flexibility and structural detail. However, whereas their sheets derive macro-mechanical compliance from elastic deformations at the native scale, our digital weaves achieve flexibility through internal yarn sliding.

ROBOTIC MATERIALS Generalizing materials with tailored mechanical properties, robotic materials extend passive substrates with actuation, sensing and communication structures [12]. In the direction of extending fabrics, Buckner *et al*. [13] augmented passive textiles with active fibers for sensing and actuation to achieve desired locomotion targets. Using shape memory fibers, Chenal *et al*. [14] proposed a stability brace for human finger joints that allows for variable stiffness. Hiramitsu *et al*. [15] developed an active textile-like garment by knitting artificial muscles in the warp direction with strings in the weft direction. Shi *et al*. [16] explore the fabrication and integration of microelectronic systems into textile fabrics. Whereas most research on robotic materials has so far followed the dichotomy of *passive* substrates and *active* secondary structure, Sanchez *et al*. [17] advocate an *active programmable textile* paradigm which puts fabric, actuation, and sensing structures on an equal footing. Our approach follows this line of thought by integrating active and passive strands in a fully automated fabrication process.

MODELING WEAVES Physics-based modeling of elastic rods is a problem that has received much attention from the graphics community [18], [1]. Originally developed for the purpose of computer animation, the discrete elastic rod model by Bergou *et al*. [2] has been used for a variety of real-world applications, including robotic wire cutting [19], elastic grid-shells *et al*. [20] and structured sheet materials [5]. This model has also been extended to incorporate the physical behaviour of fused rod connections [21], connections with free rotations [22], and sliding between yarns in woven cloth [3]. Our computational model builds on the work by Cirio *et al*. [3], with extensions that allow sliding connections on initially curved yarns. Complementing these native-scale simulation models, Schumacher *et al*. [5] and Sperl *et al*. [23] developed homogenized, macro-mechanical descriptions and material models, respectively. We draw on these ideas to analyze the macro-mechanical behavior of our programmable digital weaves in a quantitative way.

2.3 DIGITAL WEAVES

Our approach for programmable digital weaves (PDWs) is based on a simple idea: we augment planar elastic grids, in which all crossings between strands are fused, with sliding connections that allow for relative tangential motion between strands. We start the technical description of our method with its practical implementation and corresponding manufacturing process. We then describe our computational model for PDWs that serves as a basis for virtual prototyping and design exploration. Finally, we briefly summarize how to compute macro-mechanical descriptions of programmable digital weaves.

2.3.1 *Manufacturing*

To create sliding connections in practice, we proceed in three steps. We first print one of the crossing strands in the usual way, i.e., by following a planar path. The second strand is printed by lifting the print head slightly before crossing the bottom-layer strand. Using appropriate settings for velocity, extrusion rate, and nozzle temperature, this motion leads to well separated strands with little or no adhesion. Finally, we print an arc-shaped channel that is fused to the bottom-layer strand, enclosing the crossing strand without connecting to it. A snapshot of the printing process can be seen in Fig 2.1. We implement the above strategy using custom G-code

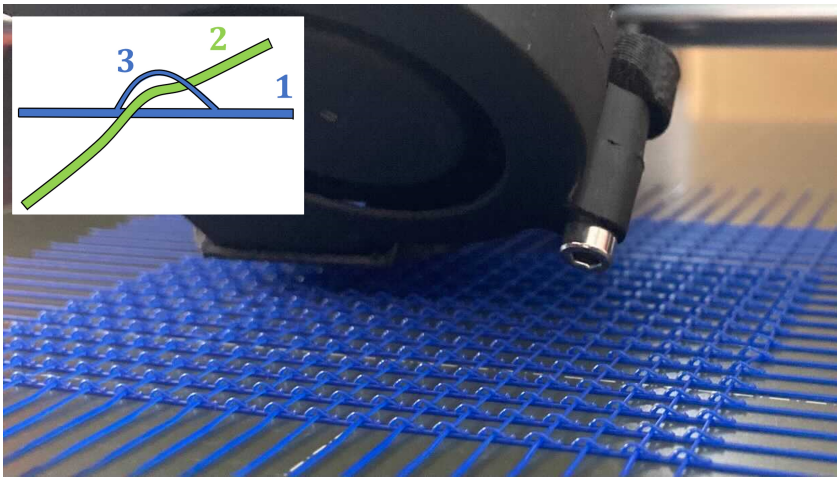


FIGURE 2.1: A snapshot of the 3D printing process for a regular digital weave with all-sliding connections. All channels have the same orientation, leaving the orthogonal direction free for sliding motions. With 20×20 strands on an 8cm × 8cm patch, the printing process takes about 40 minutes. The printing sequence is shown in the top-left corner.

that can be run on any conventional 3D printer based on fused filament fabrication. We use a Prusa I3 MK3S[2] with PLA filament for all prototypes and results reported in this work. The nozzle temperature is set to 180 and we pre-heat the print bed to 60 degrees. We set the extrusion rate based on the nozzle diameter (0.4mm), filament radius (1.75mm), desired strand width (0.5mm) and print head velocity. The latter is set to 600 mm/min by default and 300 mm/min when printing channels. Arc-shaped channels are created by moving the print head along a triangle path with the bottom edge centered at the crossing node. For better print quality, we decrease velocity when printing channels. We furthermore extend the second half of the channel by 20% compared to the length of the first half in order to avoid spurious connections with the sliding strand due to sagging. The channel height is set to twice the strand width (1.0mm).

2 https://www.prusa3d.com/original-prusa-i3-mk3/

2.3.2 *Simulation*

Programmable Digital Weaves can exhibit complex mechanical behavior characterized by strong nonlinearity and anisotropy. Predicting the performance of a given weave, or design, can be difficult and time consuming when relying solely on intuition. To allow for virtual prototyping of digital weaves and rapid design exploration, we propose a dedicated computational model that extends existing work for yarn-level cloth simulation [1, 3] to support sliding connections on initially curved yarns. As can be seen from our physical prototypes (Fig.2.2), the geometry of the channels restricts the movement of the strands, leading to persistent contacts. Based on this intrinsic property of our PDWs, we build on the Eulerian-On-Lagrangian (EoL) approach from Cirio *et al*. [3]. More concretely, we model each location where strands come into contact as a single nodal point $\mathbf{x}_i \in \mathbb{R}^3$, where sliding between crossing strands is achieved by introducing one Eulerian DoF $u_i^j \in \mathbb{R}$ for each strand j crossing a contact point i. Since persistent contacts are built into the discretization, we avoid the need for computationally expensive contact detection and handling.

Our models are printed as a 2D rod network with (potentially) curved strands. However, the original EoL implementation by Cirio *et al*. [3] assumes *naturally straight* rods, where the Eulerian DoFs are parameterized by arc-length. For our purpose, we extend this approach to *naturally curved* rods by modeling their rest configurations as a collection of C^2-continuous, planar, interpolating splines [24]. These splines offer a smooth representation of the Eulerian DoFs, allowing the use of efficient continuous optimization methods for simulation. Whereas under the sliding yarns model the undeformed configuration was fully contained in 1D, our solution maps Eulerian DoFs in 1D to points in 2D through the parameterization $\mathbf{X}(u_i) : \mathbb{R} \rightarrow \mathbb{R}^2$, allowing the use of naturally curved rods. Rods can stretch, bend, and twist. While stretch is trivial to compute from our current discretization, modeling bending and twisting deformations requires additional information. To this end, we add a rotational DoF θ_i for each two consecutive rod nodes $\{\mathbf{x}_i, \mathbf{x}_{i+1}\}$, representing the twist of an adapted frame with respect to the center-line of the edge connecting the two nodes. Bending and twisting deformation between edges can then be computed from the relative rotations between two adapted frames [2]. The bending and twisting behavior at crossing points requires extra care for our application. While sliding strands can freely rotate and twist with respect to each other, strands crossing in fused connections exhibit stiffer behavior. Following Zehnder *et al*. [21], we

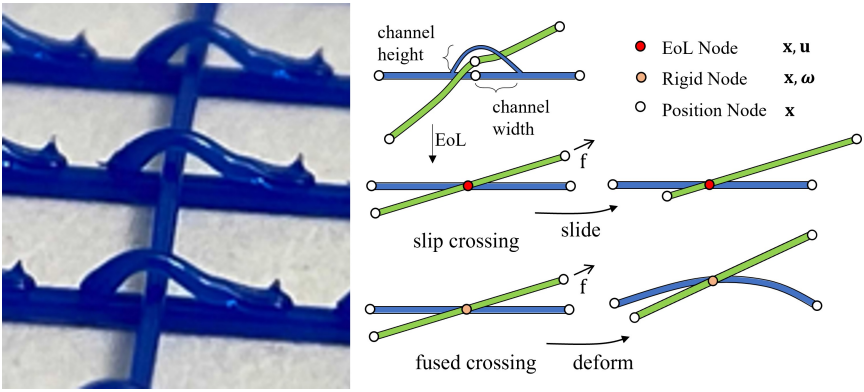


FIGURE 2.2: Close-up of 3D-printed channels at sliding connetions (*left*) and schematic view of the corresponding simulation model (*middle*). To represent crossing strands in persistent contact, Our Eulerian-on-Lagrangian (EoL) discretization uses a single node $\mathbf{x}$ augmented by Eulerian DoFs $\mathbf{u}$ to model sliding. Fused crossings are modeled as rigid bodies with additional rotational DoFs $\boldsymbol{\omega}$. The right column shows an example with an axial force applied along the green strand. The sliding connection leads to simple translation of the green strand, whereas the fused connection induces planar bending deformations in the blue strand.

restrain the relative motion of the strands crossing in a fused connection $\mathbf{x}_k$ with additional rotational DoFs $\boldsymbol{\omega}_k \in \mathbb{R}^3$. These rotational DoFs provide a common rigid frame of reference shared by the fused strands with the objective of capturing their bending and twisting deformation relative to each other. For our stretching, bending and twisting energies, we adopt the discrete energy model of Bergou *et al*. [2]. This model has been thoroughly studied and its predictive power confirmed in the context of various applications. As we show through our examples, We likewise obtained excellent agreement between simulation results and physical prototypes.

The rod network is then governed by the following potential energy,

$$\begin{aligned} E_{\text{rods}}(\mathbf{x}, \mathbf{X}(\mathbf{u}), \mathbf{u}, \theta, \boldsymbol{\omega}) = {} & E_{\text{stretch}}(\mathbf{x}, \mathbf{X}(\mathbf{u})) \\ & + E_{\text{bend}}(\mathbf{x}, \mathbf{X}(\mathbf{u}), \theta) \\ & + E_{\text{twist}}(\mathbf{x}, \mathbf{X}(\mathbf{u}), \theta) + E_{\text{rigid}}(\mathbf{x}, \boldsymbol{\omega}) \\ & + E_{\text{contact}}(\mathbf{u}) + E_{\text{reg}}(\mathbf{u}), \end{aligned}$$

where $\mathbf{x}, \mathbf{X}(\mathbf{u})$ denotes the deformed and undeformed nodal positions, $\mathbf{u}$ are the Eulerian DoFs, $\boldsymbol{\omega}$ are the Euler angles for the rotational DoFs of the fused crossings, and θ represent the twists of the adapted frames. The E_{contact} term enforces a minimal distance between two crossings [3] and E_{reg} is a small penalty term on the l^2-norm of the Eulerian variables as a regularizer. We refer to the original papers for the definition of each individual term. Computing equilibrium configurations for our weaves amounts to solving the following optimization problem,

$$\min_{\mathbf{q}} \; E_{\text{rod}}(\mathbf{q}) \quad \text{s.t.} \quad \mathbf{D}\mathbf{q} = \mathbf{q}^*, \tag{2.1}$$

where $\mathbf{q}$ stacks all of our system DoFs $(\mathbf{x}, \mathbf{u}, \boldsymbol{\omega}, \theta)$ in to a vector, $\mathbf{q}^*$ denotes the target prescription, and the matrix $\mathbf{D}$ imposes Dirichlet boundary conditions via variable substitution. We solve this unconstrained minimization problem using Newton's method with line search and dynamic regularization. To obtain a sparse Hessian, at each Newton iteration we perform time parallel transport [2], and accumulate rotation angles [21] to avoid singular configurations.

2.3.3 *Homogenization*

To characterize the macro-mechanical behavior of our digital weaves in a concise way, we resort to a numerical homogenization method that leverages simulation data from virtual tensile tests [5]. Concretely, we perform

simulations on tileable unit cells subjected to uniaxial strain in a given direction $\mathbf{d}$ and periodic boundary conditions, which are enforced via penalty terms. For each of these virtual tensile test, we compute the corresponding directional Young's modulus E and Poisson ration ν as

$$E = \frac{\mathbf{d}^T \boldsymbol{\sigma}_{\text{macro}} \mathbf{d}}{\mathbf{d}^T \boldsymbol{\epsilon}_{\text{macro}} \mathbf{d}}, \quad \nu = \frac{\mathbf{n}^T \boldsymbol{\epsilon}_{\text{macro}} \mathbf{n}}{\mathbf{d}^T \boldsymbol{\epsilon}_{\text{macro}} \mathbf{d}} \tag{2.2}$$

where $\mathbf{n}$ is orthogonal to $\mathbf{d}$ and $\boldsymbol{\sigma}_{\text{macro}}, \boldsymbol{\epsilon}_{\text{macro}}$ are the homogenized Cauchy stress and Cauchy strain computed from the boundary of the unit cell [5]. For all homogenization results shown in this work, we sample uniaxial strain directions ranging from 0 to 180 degrees at 200 uniformly distributed locations.

2.4 RESULTS

In this section, we explore the possibilities enabled by our programmable digital weaves. Starting with a regular weave, we then explore interlacing structures with straight and curved loops. For each case, we show that, by combining different choices for fused and sliding connections, we obtain vast range of nonlinear mechanical behavior ranging from quasi-isotropic to highly anisotropic. Finally we demonstrate a planar-to-parabolic deployment example, indicating the potential of our approach for integration of embedded actuation systems.

2.4.1 *Design Explorations on Conventional Weave Patterns*

We first examine the qualitative changes in mechanical behavior induced by introducing sliding connections to a regular weave with all-fused connections. As can be seen from Fig. 2.3, when subjected to shearing forces applied diagonally to the principal strand directions, the structure with fused connections deforms out of plane in order to reduce the stretching energy. When replacing fused with sliding connections, however, we instead observe large planar deformations characterized by rotation and sliding motion at strand crossings. Both scenarios are well predicted by our simulation model. Strain in the direction of strands leads to the same stiffness response regardless of the choice of fused/slip connections. This is due to the fact that strands on the orthogonal direction to the imposed strain exhibit no deformation at all. However, when shearing stress is applied, slip connections can freely rotate among themselves whereas fused connections

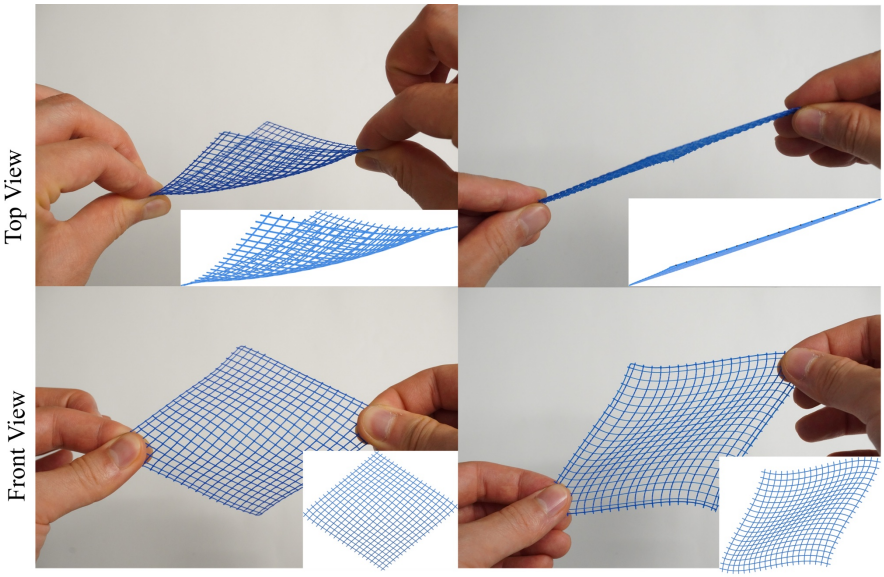


FIGURE 2.3: Qualitative comparisons of simulation results and corresponding physical counterparts. As can be seen from these images, the simulation model provides good predictive capacity for both types of connections, i.e., out-of-plane bending (all-fused connections, left) and in-plane shearing (all-slip connections, right).

resist the imposed force through bending. As can be observed from these experiments, exploring the combinatorial space of a 2×2 unit cell can already lead to structures with very different mechanical responses. The simulation results and the directional stiffness values at 45 degrees can be seen in Fig. 2.4.

2.4.2 *Meta-Material Designs with Interlacing Loops*

Our method generalizes beyond regular lattices structures and can be applied to curved rods and interlacing loops. In the first example, we examine the impact of fused and sliding connections on samples made from interlaced squares. At each corner where two squares intersect, the two crossings can either be all-fused or have a channel in one of the two directions of the square. We design and fabricate four structures corresponding

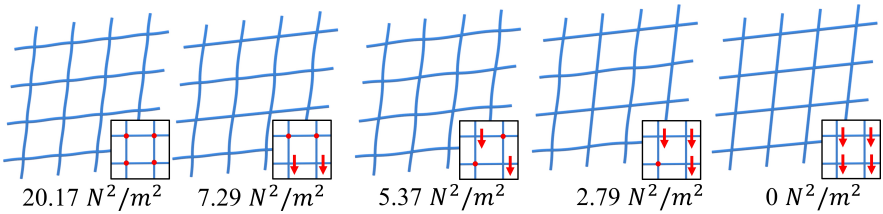


FIGURE 2.4: Shear stiffness response of a regular weave pattern with different combinations of fused-slip connections. Simulation results with corresponding fused-slip assignments shown at the bottom-right corner of each structure, with red circles indicating fused connections and red arrows indicating sliding directions, respectively. The stiffness for diagonal loading is listed at the bottom of each structure.

to representative examples of a high-dimensional design space. The two structures exhibiting the most prominent difference in stretching along the horizontal direction of the image plane are shown in Fig. 2.5. As a first experiment, we impose a uniaxial strain of 10% for all combinations and compute their directional Young's moduli and Poisson's ratios (Fig 2.6). When using all-fused connections (Type 1), the material responses appear to be quasi-isotropic and are relatively stiff in all directions. When substituting all fused connections with channels that allow sliding in the horizontal direction of the image plane (Type 2), we observe a significant change in the stiffness profile, with zero stiffness in the horizontal direction. This is explained by the fact that the imposed deformation is entirely absorbed by internal sliding. We then change the direction of one out of two channels to allow for vertical sliding motion (Type 3). The corresponding structure exhibits an orthotropic material response. For the last example, we fuse two crossings (at the bottom-left corner of the unit cell) for the Type 3 pattern, which leads to a highly anisotropic material response (Type 4). Since the homogenization plots indicate major difference for these four structures at 45 degrees, we further study the non-linear deformation space by imposing the same force in this direction to all structures. Fig. 2.6 shows significantly larger deformations for structure Type 1 and 2, indicating a lower stiffness. The same force, however, leads to visually imperceptible deformation for those involving fused connections or, equivalently, higher stiffness. We refer to the supplemental video for manipulation sequences of the printed structures.

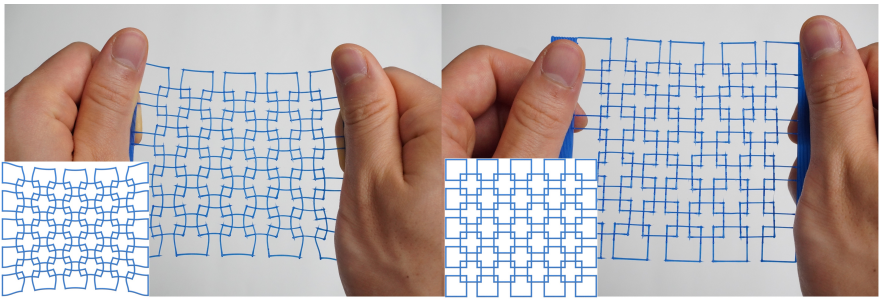


FIGURE 2.5: Qualitative comparison of simulation results and corresponding physical counterparts for the interlacing squares example. *Left*: all-fused connections. *Right*: connections enabling free sliding in the horizontal direction. Note that for the structure on the right, all imposed deformation is absorbed by internal sliding, leading to zero stiffness in the horizontal direction.

In the second experiment, we select the Type 2 structure and conduct homogenization with small and large strains. As can be seen from Fig. 2.7, until the squares reach their sliding limits, the structure exhibits zero stiffness. However once the imposed strain is larger than the absorption limit of the internal sliding mechanism, finite stiffness is observed. We visualize the simulation results at selected strain directions to further demonstrate that predicting the nonlinear deformation behavior solely on intuition is extremely challenging.

Programmable digital weaves can be readily applied to curved geometries. In the example shown in Fig. 2.8, we replace the squares in the previous example with circles. Each circle overlaps with its four neighbors by 10% of its radius (0.8 cm), leading to eight crossing points per circle. We explore the two extreme options of all-fused and all-sliding connections for these eight crossing points. Fig. 2.8 indicates that the all-fused variant results in an orthotropic stiffness response while its all-sliding counterpart leads to a highly anisotropic behavior. On the right of this figure, compare the resulting deformations for the same imposed for (along the vertical direction of the image plane). Qualitatively, it can be noted that the individual circles of the all-sliding structure have been distorted into ellipses, whereas the ones with the fused connections remain largely circular.

2.4.3 *Actuation*

Leveraging the possibility to combine fused and slip connections in a single design, we further explore how strands can be used to implement embedded actuation. In this example, we design a structure with the goal of achieving a planar-to-parabolic deployment motion by virtue of a single actuation strand. As can be seen from Fig. 2.9, the tip of the vertical actuation strand is fused to the outer circle but can freely slide on all the other crossings (red arrows). All other connections are fused. We impose sufficient boundary conditions by fixing the position of the bottom-most crossing of the actuation strand and the outer ring. Actuation is achieved by applying a compressive axial force at the bottom of the actuation strand. As can be seen in Fig. 2.9, our simulation again shows good agreement with the physical experiment. We emphasize that manufacturing of these actuation strands is fully automatic without any manual assembly required. We refer to the supplemental video for a real-world actuation sequence for this design.

2.5 CONCLUSIONS

We presented programmable digital weaves—a novel material concept that augments planar elastic grids with sliding connections. Although the experimental results confirm the feasibility of our approach, there are a number of limitations that should be addressed in future work. We do not consider cases where more than two strands cross in one connection. Furthermore, friction between crossing strands is not accurately modeled, which leads to certain amount of discrepancy between our simulation results and the real-world behaviors. We also observed a stiffness bias induced by the extra material required for channels. While beyond the scope of this work, we plan to systematically study the impact of sliding connections on bending stiffness. In particular, sliding connections enable the deployment of a flat patch into a surface with nonzero Gaussian curvature with little resistance, whereas all-fused connections strongly oppose such deformations. Contact beyond EoL nodes and multi-axis response will also be studied in the future. Finally, we would also like to explore the integration of strain sensors using, e.g., Piezo-resist filament [25] to allow for fully-integrated state estimation.

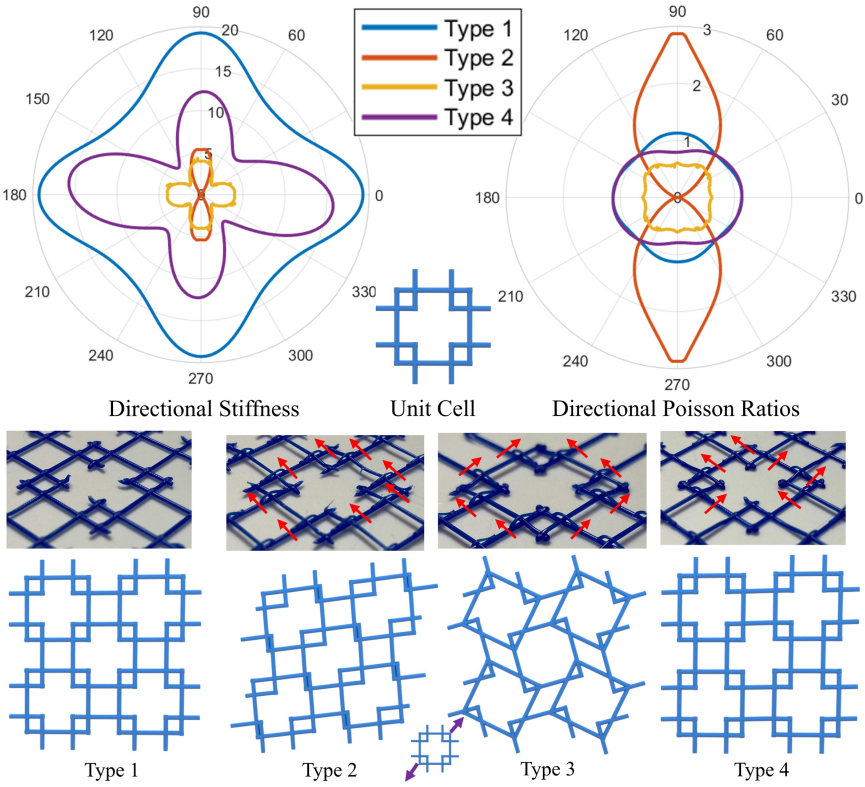


FIGURE 2.6: Interlacing squares. The different fused-slip connections for the four structures considered are shown on the physical prototypes with sliding directions indicated with red arrows (middle row). As shown in the stiffness profiles (top row), these structures exhibit widely different mechanical behaviors, ranging from quasi-isotropic (Type 1) to highly anisotropic (Type 4). Simulation results of these structures (bottom row) are obtained for the same applied load (diagonal direction indicated with purple arrows). It can be seen that the sliding mechanisms of the Type 2 and 3 structures are fully triggered while the ones involving fused connections remains largely undeformed.

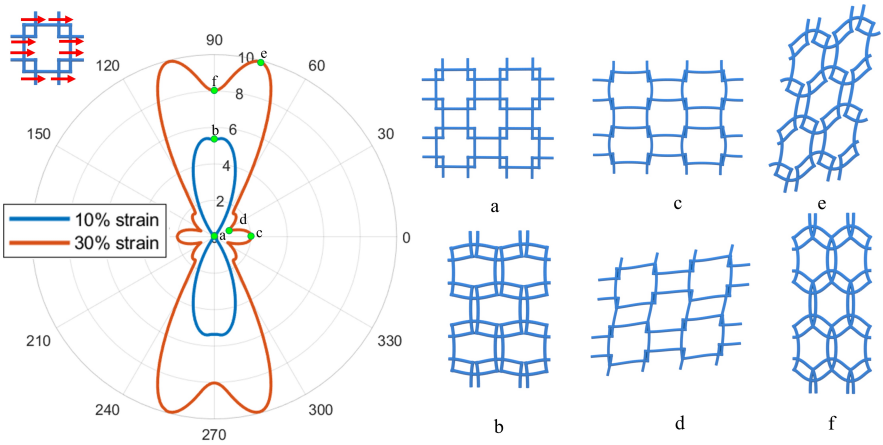


FIGURE 2.7: Nonlinear mechanical behavior. Here we show the deformations of our Type 2 structure when imposing uni-axial strain of 10 and 30 percent, respectively. The zero stiffness response shown initially by this structure for small imposed strains is in sharp contrast to the response for larger strains, indicating highly nonlinear behavior. The sliding directions of the channels on the unit cell is shown in the top-left corner.

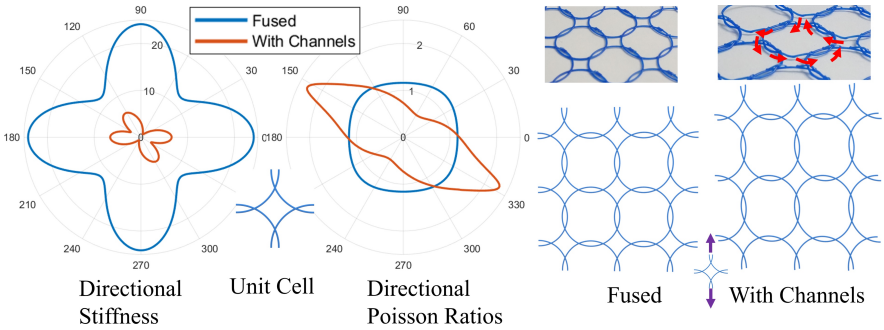


FIGURE 2.8: Interlacing circles. We explore the option of curved strands using circular loops. As indicated by the plots for directional stiffness and Poisson ratios, replacing all-fused with all-slip connections significantly modifies the mechanical properties. Due to the asymmetric orientation of the channels, the mechanical response is thus asymmetric along the diagonal direction. As an example, we show the simulation results for both structures induced by the same force along the vertical direction of the image plane (purple arrows). The sliding directions at the crossings are shown with red arrows.

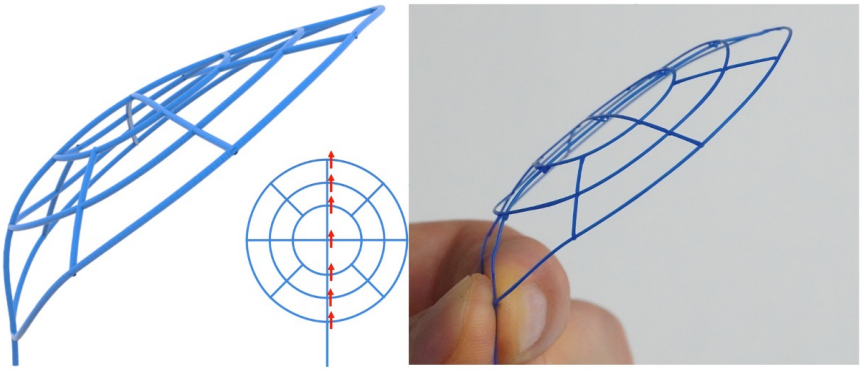


FIGURE 2.9: Actuation example. In this experiment, a single actuation strand is embedded in the structure to induce a planar-to-parabolic deployment upon pushing (red arrow). The simulation result of the actuated structure is shown on the left and the deformation of a physical prototype is shown on the right.

3

PART II: EMBEDDED ELASTICITY

Computing intrinsic distances on discrete surfaces is at the heart of many minimization problems in geometry processing and beyond. Solving these problems is extremely challenging as it demands the computation of on-surface distances along with their derivatives. We present a novel approach for intrinsic minimization of distance-based objectives defined on triangle meshes. Using a variational formulation of shortest-path geodesics, we compute first and second-order distance derivatives based on the implicit function theorem, thus opening the door to efficient Newton-type minimization solvers. We demonstrate our differentiable geodesic distance framework on a wide range of examples, including geodesic networks and membranes on surfaces of arbitrary genus, two-way coupling between hosting surface and embedded system, differentiable geodesic Voronoi diagrams, and efficient computation of Karcher means on complex shapes. Our analysis shows that second-order descent methods based on our differentiable geodesics outperform existing first-order and quasi-Newton methods by large margins.

3.1 INTRODUCTION

Computing intrinsic distances on triangle meshes is a fundamental task in geometry processing. From biological films on surfaces, to the fascia enveloping our muscles, and to tight-fitting clothing—there are countless examples of variational problems where the task is to minimize lengths on discrete manifolds. Solving such problems on triangle meshes requires the computation of geodesic distances on surfaces along with their derivatives. While gradients are readily computed, the convergence of first-order methods is generally poor, in particularly for embedded elasticity problems with stiff connections. Quasi-Newton methods such as L-BFGS may offer some acceleration, but as we show in our analysis, they still converge slowly, especially for stiff problems.

In this work, we present a novel approach for intrinsic minimization of arbitrary objectives based on geodesic lengths on piece-wise linear surfaces. Using a variational form of shortest-path geodesics, we obtain distance

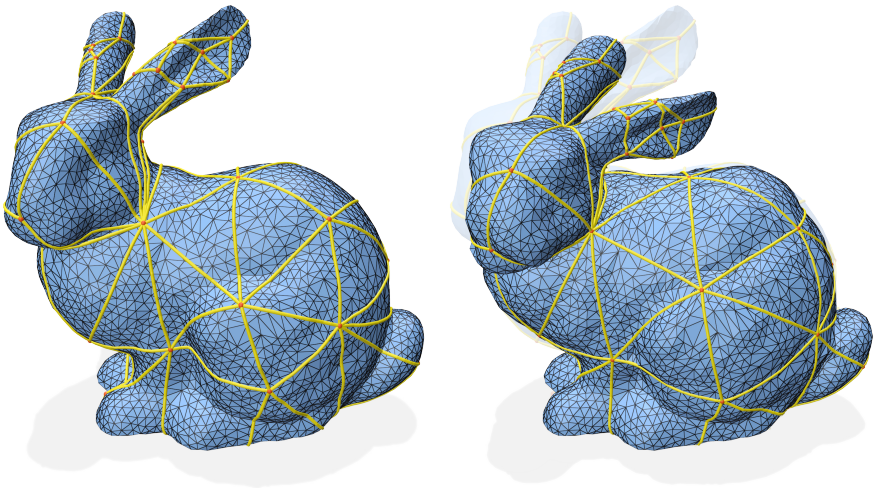


FIGURE 3.1: Embedded two-way coupling. We simulate a geodesic elastic network embedded in the surface of a deformable bunny, modeled with solid finite elements. Our analytical geodesic distance derivatives allow us to use Newton's method for simulating the deformations induced by tightening the network.

derivatives in closed form using the implicit function theorem. We show that first and second derivatives can be simplified to enable efficient computation, allowing us to leverage powerful Newton-type methods for simulation. Our approach is underpinned by the key observation that, although geodesic paths on triangle meshes are generally not smooth functions of their endpoints, geodesic distance is at least C^0-continuous everywhere—and infinitely smooth if geodesics are unique. Our analysis reveals two types of gradient discontinuities, one of which can be resolved through mollification while the other one does not occur close to minima and can therefore be ignored.

We demonstrate our differentiable geodesic distance framework on a large and diverse set of minimization problems. Specifically, we construct embedded elastic networks and membranes based on geodesic springs and triangle finite elements, respectively. We present simulation examples on shapes with varying topologies, including two-way coupling with the hosting surface. We furthermore show how our formulation enables differentiable on-surface Voronoi diagrams, whose site locations can be optimized for various objectives. Finally, we demonstrate that our simulation

framework can be used to compute Karcher means on arbitrary surfaces in efficient and highly accurate ways.

This chapter is based on our previous publication [26].

3.2 RELATED WORK

DISCRETE GEODESIC DISTANCE Geodesic distance has been extensively studied in the geometry processing community [27–29]. Many methods have been developed for calculating exact geodesic distances [30–34] as well as fast approximations [35–43]. While approximations through the solution of smooth energy functions offer advantages in terms of differentiability, accurately tracing the geodesic path from the distance field is a challenging task in itself [27]. We use an exact geodesic distance computation from which geodesic paths can be constructed. Although the numerical properties of discrete geodesics have been studied intensively, we are not aware of any second-order method for minimizing geodesics-based objectives. To construct such a method, we propose a differentiable geodesic distance formulation that is almost everywhere C^2-continuous and exhibits robust convergence when integrated into Newton-type minimization algorithms.

INTRINSIC GEOMETRY PROCESSING The computer graphics community has made great strides in intrinsic geometry processing [44–51]. Intrinsic triangulations facilitate tasks such as Delaunay refinement [48], mesh simplification [51], and construction of differential operators on non-manifold meshes [49]. Similar to intrinsic triangulations [48], our method uses endpoints as only degrees of freedom—connecting geodesic edges are defined implicitly and reconstructed on demand. Unlike existing work, our approach enables the computation of first and second derivatives of geodesic distances in closed form, thus opening the door to efficient second-order minimization algorithms.

STRUCTURAL CURVE NETWORKS Structural curve networks have gained increasing attention in the fields of computer graphics and robotics due to their aesthetic appeal and practical utility. For instance, Schumacher *et al*. [5] characterize the mechanical behavior of different families of tiling patterns, and Li *et al*. [6] explore the direction-dependent stiffness of 3D-printed weave structures. Another line of research focuses on the structural stability of curve networks [21, 52–56]. Our work is similar in the sense that we operate on curves embedded within surfaces. Instead of representing

these curves explicitly, however, they are represented implicitly as shortest geodesic paths. Our differentiable geodesic distance formulation allows us to solve intrinsic minimization problems defined on these implicit curve networks.

EMBEDDED SIMULATION Many physical phenomena can be described through partial differential equations on Riemannian manifolds. Examples include swirl dynamics on soap bubbles [57, 58], skin sliding [59], tight-fitting clothing [60], and elastic curve networks embedded in curved surfaces [21]. One approach to this embedded elasticity problem is to use 3D spline curves whose control vertices are constrained to lie on the surface [61–63]. Eulerian-on-Lagrangian methods are an alternative representation for simulating constrained motion of deformable bodies [64], rods [6, 65], and cloth [3, 66, 67]. Within this context, Li *et al*. [59] simulate skin sliding using texture-like material coordinates as Eulerian degrees of freedom. In order for this approach to work, however, a texture atlas of the underlying surface with quasi-isometric charts is required. Montes *et al*. [60] propose a Lagrangian-on-Lagrangian approach that combines subdivision surfaces with embedded triangle meshes to allow for smooth sliding of skin-tight clothing across the underlying body. However, they approximate on-surface lengths using Euclidean distances. Consequently, surface and cloth discretizations must have adequate resolutions to limit approximation error. Our simulation examples with embedded elastic networks likewise use a Lagrangian-on-Lagrangian representation. However, we avoid resolution dependence by computing exact geodesic distances on triangle meshes.

3.3 BACKGROUND AND CHALLENGES

Our goal is to minimize distance-based functions on triangle meshes with second-order optimizers. Naturally, this requires the distance metric to be sufficiently smooth and differentiable. While Euclidean distance satisfies these criteria, depending on the resolution of the embedded mesh, it can deviate from the actual distance between two surface points, *i.e.*, the geodesic distance, to different degrees. This deviation can lead to undesired local minima, hindering the optimization process (see Sec. 3.5.7). We therefore opt to use geodesic distances for simulation. Before describing our method in detail, we first briefly discuss the smoothness of geodesic distance.

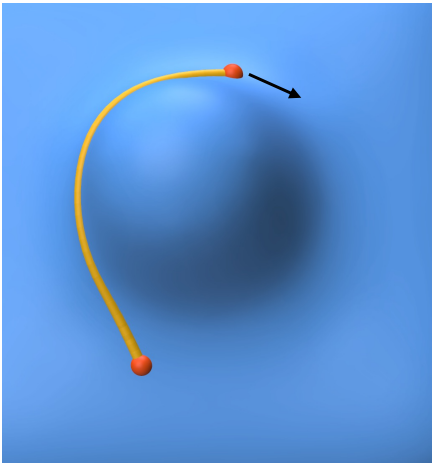
3.3.1 *Geodesic Distance*

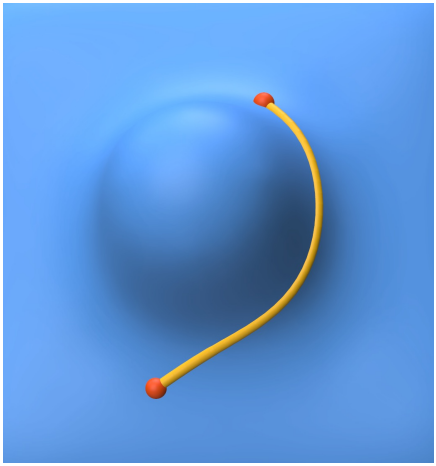
In the continuous setting, geodesics are locally shortest paths between two points on a surface. These paths are also straightest, *i.e.*, they exhibit zero geodesic curvature. However, in the discrete setting with non-smooth surfaces, these two definitions—straightest and shortest path—are not equivalent and will generally lead to different geodesic paths. The shortest path definition is a natural choice for boundary value problems with known endpoints, while the straightest-path definition is best suited for initial value problems with a given starting point and tangent direction. We refer to Crane *et al*. [27] for an in-depth discussion of this subject. In this work, we use shortest-path geodesics, which can be computed by minimizing a convex quadratic potential. To use this distance metric in a minimization algorithm, we must first understand its smoothness properties.

3.3.2 *Smoothness of Geodesic Distance*

The geodesic distance between two points is a continuous function of the points everywhere on the surface. The geodesic path, however, is not always unique and can change abruptly for smooth motion of the endpoints. The distance derivative is necessarily discontinuous in these singular configurations. It is worth mentioning that this discontinuity exists even for smooth surfaces and does not originate from discretization.

CONTINUOUS SETTING. To understand this discontinuity, consider a surface patch containing a bump and a geodesic whose endpoints lie on opposite sides of the bump (see inset figure). As we translate the upper endpoint to the right, the geodesic path reaches a singular point where it suddenly *flips* to the right. At the singularity, there exist two paths on different sides of the bump that have the same geodesic distance. A local perturbation of the endpoint can easily lead to flipping of the shortest geodesic. The collection of points for which geodesics are not unique is referred to as the *cut locus*. The shortest geodesic path from point a to point b is discontinuous as b passes through the cut locus of a, and along the cut locus, the gradient of the path is undefined. Although this discontinuity in the distance gradient exists even in the continuous setting, it does not prevent gradient-based minimization since points on the cut locus locally *maximize* the geodesic distance from the source point.





DISCRETE SETTING. Geodesic distance remains continuous in the discrete setting, but geodesic paths behave somewhat differently compared to the smooth setting. Special care is required when geodesics pass close to mesh vertices, which we classify into three categories depending on their discrete Gaussian curvature: spherical vertices (positive curvature), hyperbolic vertices (negative curvature), and planar vertices (zero curvature). In the vicinity of planar vertices, the geodesic is C^2-continuous. For spherical vertices, however, a geodesic cannot pass through the vertex as there always exists a shorter path going around it (see Fig. 3.2). Since geodesic paths vary discontinuously around spherical vertices, geodesic distance is only C^0-continuous. Fortunately, paths through spherical vertices are length-maximizing and are thus avoided during distance minimization. Lastly, geodesic paths can continuously pass through hyperbolic vertices, but the change in local tangent direction leads to only C^1-continuity in these cases. In Sec. 3.4.1 we show that C^2-continuity can be restored by adding a suitable mollifier.

3.3.3 *Challenges*

Our approach builds on the observation that, while geodesic paths are not generally continuous functions of their endpoints, their lengths vary continuously and can thus be used for gradient-based minimization. Computing the required derivatives, however, is no trivial task. Geodesics on

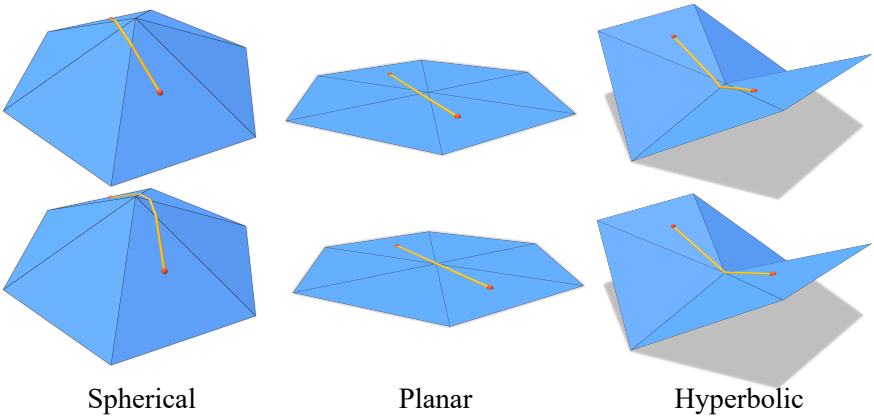


FIGURE 3.2: Behavior of geodesic paths passing over different types of vertices. Fixing one endpoint of a geodesic, we translate the other such that the path moves across the center vertex. Whereas the geodesic passes through the center vertex in the planar and hyperbolic case, it jumps over the spherical vertex to avoid the local distance maximum.

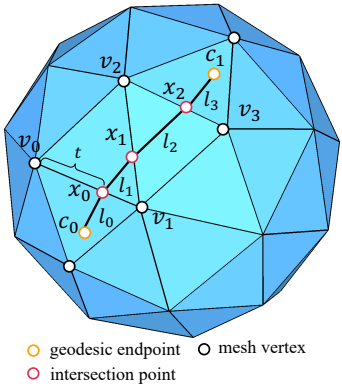
triangle meshes generally span multiple faces and intersect with the edges of the hosting surface. Explicitly tracking a varying number of intersection variables is cumbersome. Using endpoints as degrees of freedom circumvents this issue, but intersection points are then functions of the endpoints, adding another layer of complexity to the computation of first and second derivatives. In fact, we are not aware of any existing work that computes the analytical derivatives of geodesic distance on triangle meshes. In the following, we develop a differentiable geodesic distance formulation whose first and second derivatives can be computed efficiently. Since our formulation requires only intrinsic quantities, we refer to it as *intrinsic minimization*.

3.4 INTRINSIC MINIMIZATION

We develop a formulation for differentiable geodesic distance on triangle meshes that uses geodesic endpoints as the only explicit variables. The intersection points on the geodesic path are implicitly defined through equilibrium conditions of a shortest-path energy functional (Sec. 3.4.1). We show that an intrinsic simulation framework driven by geodesic distance can be formulated on this basis (Sec. 3.4.2) and the required derivatives

can be obtained in simple forms using sensitivity analysis and geometric insights. We extend our distance-driven formulation from geodesic edges to triangles such as to construct directionally-continuous deformation energies (Sec. 3.4.3). We furthermore elaborate on two-way coupling effects between geodesic networks and their embedding surfaces (Sec. 3.4.4). Finally, beyond simulating embedded elasticity, we generalize the intrinsic minimization paradigm to differentiable geodesic Voronoi diagrams (Sec. 3.4.5).

3.4.1 *Differentiable Geodesic Distance*



Our approach builds on the robust MMP algorithm [30] to compute exact geodesic paths on triangle meshes. The endpoints of geodesic paths are represented using barycentric coordinates $\mathbf{w}$ of the hosting triangle mesh. These endpoints are the only simulation degrees of freedom of a given geodesic path. The spatial coordinates of the endpoints $\mathbf{c}$, and the intersections $\mathbf{x}$ between the path and edges of the hosting mesh, are computed when either endpoint is moved (see inset). We use a scalar parameter t_i to denote the position of an intersection point $\mathbf{x}_i$ along a corresponding mesh edge e_{jk},

$$\mathbf{x}_i = \mathbf{v}_j + t_i(\mathbf{v}_k - \mathbf{v}_j) \,, \tag{3.1}$$

where $\mathbf{v}$ are mesh vertices. We compute the geodesic distance g between two points, $\mathbf{c}_0$ and $\mathbf{c}_1$, on the surface mesh as the sum of the lengths of the line segments l_i comprising the geodesic path,

$$g(\mathbf{c}_0, \mathbf{c}_1) = \sum_i l_i(\mathbf{c}_0, \mathbf{c}_1, \mathbf{x}(\mathbf{c}_0, \mathbf{c}_1)) \,. \tag{3.2}$$

The relationship of the simulation variables $\mathbf{w}$ and geodesic endpoints $\mathbf{c}$ are given explicitly by interpolating barycentric coordinates. The relationship of the intersection variables $\mathbf{t}$ are given implicitly by the algorithm for computing exact geodesics. All involved quantities ultimately depend only on the simulation variables $\mathbf{w}$ and we therefore write

$$g(\mathbf{c}, \mathbf{x}) = g(\mathbf{c}(\mathbf{w}), \mathbf{x}(\mathbf{t}(\mathbf{c}(\mathbf{w})))) \,. \tag{3.3}$$

To enable Newton-type solvers for intrinsic minimization, we must compute the first and second derivatives of the above expression. The first derivative is given by

$$\frac{dg}{d\mathbf{w}} = \frac{\partial g}{\partial \mathbf{c}}^T \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial g}{\partial \mathbf{x}}^T \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}}, \tag{3.4}$$

and the second derivative follows as

$$\begin{aligned}
\frac{d^2 g}{d\mathbf{w}^2} = {} & \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^T \frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^T \frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \sum_i \frac{\partial g}{\partial c_i} \frac{\partial^2 c_i}{\partial \mathbf{w}^2} \\
& + \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^T \left(\frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right) \\
& + \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^T \sum_i \frac{\partial g}{\partial x_i} \frac{\partial^2 x_i}{\partial \mathbf{t}^2} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right) \\
& + \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^T \left(\sum_i \frac{\partial g}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t_i} \frac{\partial^2 t_i}{\partial \mathbf{c}^2} \right) \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \sum_i \frac{\partial g}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial c_i} \frac{\partial^2 c_i}{\partial \mathbf{w}^2}.
\end{aligned} \tag{3.5}$$

While computing all of these terms is feasible, we show that both expressions can be significantly simplified using geometric insight.

The shortest geodesic is a local minimizer of distance, giving rise to the first-order optimality condition

$$\frac{dg}{d\mathbf{t}} = \frac{\partial g}{\partial \mathbf{x}}^T \frac{\partial \mathbf{x}}{\partial \mathbf{t}} = \mathbf{o}. \tag{3.6}$$

If the interior vertices $\mathbf{x}$ satisfy this optimality condition, perturbations $\delta\mathbf{t}$ to the intersection variables will not change length to first order: moving *along* the path changes adjacent segment lengths, but these changes sum to zero; moving *orthogonal* to the path simply rotates the segments, which does not change their lengths to first order. We can therefore simplify the total derivative to

$$\frac{dg}{d\mathbf{w}} = \frac{\partial g}{\partial \mathbf{c}}^T \frac{\partial \mathbf{c}}{\partial \mathbf{w}}. \tag{3.7}$$

This expression is readily evaluated since it only requires the gradient of length for the first and last segments and the (constant) Jacobian $\frac{\partial \mathbf{c}}{\partial \mathbf{w}}$ of the barycentric interpolation function.

Similarly, the Hessian can also be greatly simplified using implicit differentiation. Note that all terms can be evaluated algebraically except for $\frac{\partial \mathbf{t}}{\partial \mathbf{c}}$, *i.e.*, the derivatives of the edge intersection points with respect to the

endpoints of the geodesic. Computing this derivative explicitly requires differentiation through the entire pipeline of the algorithm used for computing the exact geodesics. Apart from the complexity of differentiating such a code, this strategy must fail for cases in which geodesics are not unique and, hence, path derivatives do not exist. Our key insight is that this problem can be entirely avoided by implicit differentiation. We begin by differentiating both sides of (3.6) w.r.t. $\mathbf{c}$,

$$\frac{\partial \mathbf{x}}{\partial \mathbf{t}}^{\mathsf{T}} \left(\frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} + \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \right) = \mathbf{o} \,. \tag{3.8}$$

To obtain $\frac{\partial \mathbf{t}}{\partial \mathbf{c}}$, we rearrange and solve the small linear system

$$\left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \right) \frac{\partial \mathbf{t}}{\partial \mathbf{c}} = - \frac{\partial \mathbf{x}}{\partial \mathbf{t}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \,. \tag{3.9}$$

The effective size of this $m \times m$ system depends on the number m of intersections between a given geodesic and the hosting mesh. Multiplying (3.8) by $(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}})^{\mathsf{T}}$ from the left and by $\frac{\partial \mathbf{c}}{\partial \mathbf{w}}$ from the right gives

$$\left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^{\mathsf{T}} \left(\frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right) = \mathbf{o} \,. \tag{3.10}$$

Substituting (3.6) and (3.10) into (3.5) and removing second derivatives of linear terms, we obtain the substantially simplified expression

$$\frac{\partial^2 g}{\partial \mathbf{w}^2} = \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \,. \tag{3.11}$$

DISCONTINUITIES & MOLLIFICATION With the expression for second derivatives in hand, we must still address potential discontinuities when intersection points $\mathbf{x}(\mathbf{t})$ approach mesh vertices. Depending on to which side of a mesh vertex the geodesic passes, it will intersect different mesh edges. The function $\mathbf{x}(\mathbf{t})$ is therefore ill-defined when the geodesic path coincides with a mesh vertex, *i.e.*, when $t = 0$ or $t = 1$ for individual edges, and the optimality condition (Eq. 3.6) does not hold. This discontinuity hinders convergence, as shortest geodesics tend to pass through hyperbolic vertices (Fig. 3.3*a*), and the total distance energy is not C^2-continuous. We resolve this issue by adding a mollifier to $\mathbf{x}(t)$ such that $\frac{d\mathbf{x}(\mathbf{t})}{dt}$ smoothly vanishes at vertex intersections. Here, we hold $\mathbf{t}$ fix, and the mollification is applied to $\mathbf{x}$.

A similar discontinuity appears when endpoints of geodesics coincide with vertices of the hosting mesh. We therefore apply the same mollifier to the barycentric coordinates of the endpoints.

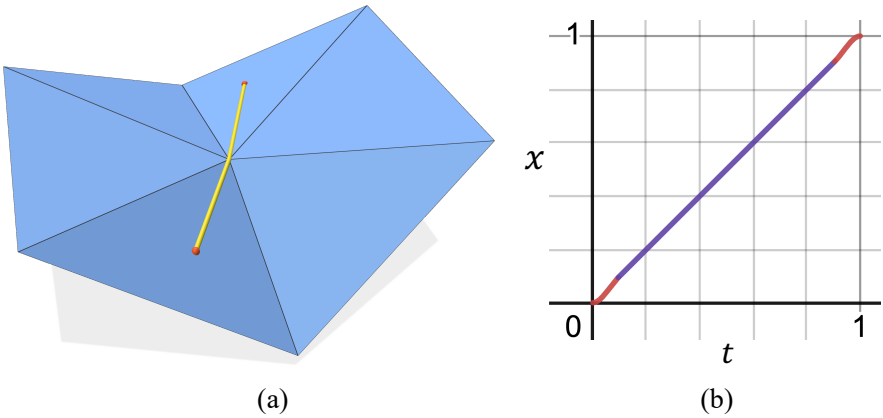


FIGURE 3.3: Mollification for edge intersection points. We smoothly blend linear edge intersection trajectories $\mathbf{x}(t)$ with cubic functions (b) to achieve C^2-continuity when geodesics pass through mesh vertices (a).

SMOOTH MOLLIFIER We smoothly blend the linear intersection point parameterization (3.1) with two cubic functions (see Fig. 3.3*b*) such that the derivatives vanish when the geodesic curve passes through mesh vertices, *i.e.*, when $t = 0$ or $t = 1$. The mollifier function is defined as

$$\hat{t}(t) = \begin{cases} -\frac{t^3}{\epsilon^2} + \frac{2t^2}{\epsilon} & 0 \leqslant t < \epsilon, \\ t & \epsilon \leqslant t \leqslant 1-\epsilon \\ -\frac{(t-1)^3}{\epsilon^2} + \frac{2(t-1)^2}{\epsilon} + 1 & 1-\epsilon < t \leqslant 1\,, \end{cases} \tag{3.12}$$

where $\epsilon = 10^{-6}$ defines the smoothing length. This mollifier is C^1-continuous along an edge, and as shown in Sec. 3.5.7, significantly accelerates the convergence of our Newton solver.

3.4.2 *Elastic Geodesic Networks*

With our differentiable geodesic distance formulation established, we now turn to energy minimization. As a first example, we consider elastic curve networks comprised of geodesic springs. The total energy of this system is

$$E_{\text{network}}(\mathbf{w}) = \sum_i (g_i(\mathbf{w}) - \bar{g}_i)^2 \,, \tag{3.13}$$

where i runs over all geodesic springs, whose current and rest lengths we denote as g_i and $\bar{g}_i$, respectively. We minimize this energy using Newton's method with a standard backtracking line search. The search direction per iteration is computed by solving the linear system

$$\mathbf{H}(\mathbf{w})\Delta\mathbf{w} = -\mathbf{y}(\mathbf{w}) \,, \tag{3.14}$$

where $\mathbf{y}(\mathbf{w})$ and $\mathbf{H}(\mathbf{w})$ are the gradient and Hessian of the objective function. We use adaptive diagonal regularization [68] to ensure that the Hessian is positive definite. Since all simulation variables are barycentric coordinates, the resulting search direction yields endpoint updates expressed in the barycentric coordinates of the corresponding mesh triangles. Inevitably, endpoints will move across edges and vertices into neighboring triangles, which requires updating the search direction to the new coordinate system. To this end, we use the approach by Sharp *et al.* [48] for tracing straightest geodesics [69] and update local coordinates as

$$\mathbf{w}^{j+1} = \mathbf{w}^j + \alpha \cdot \text{tracing}(\mathbf{w}^j, \Delta\mathbf{w}) \,, \tag{3.15}$$

where α is a step size determined by line search to ensure monotonic decrease in energy.

KARCHER MEANS Using a slightly different formulation, our method can be readily extended to compute so-called *Karcher means* on triangle meshes. Consider a special case of a geodesic spring network, where a point $\mathbf{p}$ is connected to several anchored points $\mathbf{x}$ on a surface. The energy in this case amounts to the variational formulation for the Karcher mean,

$$E_{\text{Karcher}}(\mathbf{w}) = \frac{1}{2N} \sum_{i=1}^{N} g(\mathbf{p}(\mathbf{w}), \mathbf{x}_i)^2 \,. \tag{3.16}$$

Sharp *et al.* [70] showed that the gradient of this energy can be approximated efficiently by solving a sparse linear system. While their approach

is robust and general, using only first-order derivatives precludes second-order convergence. Our analytical second-order derivatives of geodesic distance, in contrast, enable Newton-type minimization, which leads to significantly improved convergence (see Sec. 3.5.2).

3.4.3 *Elastic Geodesic Triangles*

In the previous examples, we considered potentials defined on geodesic networks. We now extend our formulation to model elastic membranes made of geodesic triangles, *i.e.*, finite element-like triangles with geodesic edges.

For linear triangle finite elements in Euclidean geometry—also known as constant strain triangles—the deformation gradient maps edge vectors from the undeformed configuration to corresponding deformed edges as

$$\mathbf{F}\bar{\mathbf{e}}_{ij} = \mathbf{e}_{ij} \,, \tag{3.17}$$

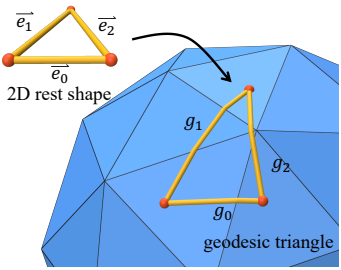
where $\bar{\mathbf{e}}_{ij} = (\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)$ and $\mathbf{e}_{ij} = (\mathbf{x}_j - \mathbf{x}_i)$ are undeformed and deformed edges, respectively. Squaring both sides, we obtain

$$\bar{\mathbf{e}}_{ij}^T \mathbf{C} \bar{\mathbf{e}}_{ij} = |\mathbf{e}_{ij}|^2 \,, \quad \mathbf{C} = \mathbf{F}^T\mathbf{F} \,, \tag{3.18}$$

where $\mathbf{C}$ is the Cauchy-Green tensor. A direct translation of this deformation measure to geodesic triangles would require geodesic edges, expressed in a common coordinate system. This representation, however, is not available since the vertices of a geodesic triangle generally fall into different triangles of the hosting surface. Fortunately, we can construct the Cauchy-Green tensor using only edge lengths (see inset figure). To this end, we first note that

$$\bar{\mathbf{e}}_{ij}^T \mathbf{C} \bar{\mathbf{e}}_{ij} = g_{ij}^2 \,, \tag{3.19}$$

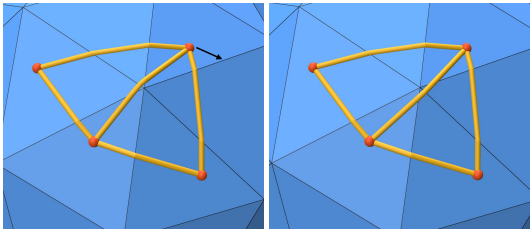
for all three edges of a geodesic triangle. Using the fact that $\mathbf{C}$ is a symmetric 2×2 tensor, it is evident that its three unknown entries can be computed by solving the three equations (3.19). With this nonlinear deformation measurement at hand, we can then use a standard Neo-Hookean constitutive model for membrane elasticity. The corresponding energy density function is given as



$$\Psi(\mathbf{C}) = \frac{\mu}{2}(\mathrm{tr}(\mathbf{C}) - 2 - 2\log(J)) + \frac{\lambda}{2}\log(J)^2 \,, \tag{3.20}$$

where $J = \sqrt{\det(\mathbf{C})}$. Since it only involves the lengths of geodesic edges, this energy density is a continuous function of the simulation variables $\mathbf{w}$.

DISCUSSION We assume that triangles are intrinsically flat when computing deformation gradients from edge lengths. While this is an approximation, it allows us to measure and penalize deformations in any direction, not just along edges. As another point, although it is possible to compute the area enclosed by a geodesic triangle, this area can be discontinuous



as the geodesic path *flips* in the vicinity of spherical vertices. An example is shown in the inset figure, where a small perturbation of one vertex changes the area of the two adjacent triangles abruptly. Instead of directly computing the area of geodesic triangles, our formulation captures changes in area through the determinant of the Cauchy-Green tensor, $\det(\mathbf{C})$. Since computing $\mathbf{C}$ requires only the lengths of geodesic edges, not their paths, we avoid discontinuities in area.

3.4.4 *Two-way Coupling*

The previous examples focused on geodesic elastic systems embedded in a rigid hosting surface. However, our approach can also be extended to full two-way coupling with deformable hosting objects. For these cases, the geodesic distance function takes the form

$$g(\mathbf{c}(\mathbf{w},\mathbf{v}),\mathbf{x}(\mathbf{t}(\mathbf{c}(\mathbf{w},\mathbf{v}),\mathbf{v}),\mathbf{v}) \,, \tag{3.21}$$

where $\mathbf{v}$ are the vertices of the hosting surface. The first and second derivatives of this expression can again be obtained using sensitivity analysis; see App. A.1. We describe the coupled system through its potential energy,

$$E_{\text{coupling}}(\mathbf{q}) = E_{\text{network}}(\mathbf{q}) + E_{\text{host}}(\mathbf{q}) \,, \tag{3.22}$$

where $\mathbf{q} = (\mathbf{w},\mathbf{v})^T$ concatenates the barycentric coordinates of the embedded system and the vertices on the hosting surface. The first term $E_{\text{network}}(\mathbf{q})$

models the energy of the embedded elastic system, whereas $E_{\text{host}}(\mathbf{q})$ is the elastic potential of the hosting object. We consider two types of models for the hosting object: a discrete shell model [71] and a standard volumetric finite element model [72].

3.4.5 *Differentiable Geodesic Voronoi Diagrams*

Up to this point, we have focused on elastic geodesic systems embedded in rigid and deformable hosting objects. We now extend our formulation to another application of intrinsic distances, geodesic Voronoi diagrams.

GEODESIC VORONOI DIAGRAMS For a given set of sample points (sites) on a surface, a surface Voronoi diagram partitions the surface mesh $\mathcal{M}$ into cells. A cell C_i is the set of all points $\mathbf{x}^*$ satisfying

$$C_i = \{\mathbf{x}^* \in \mathcal{M} \mid g(\mathbf{x}^*, \mathbf{s}_i) < g(\mathbf{x}^*, \mathbf{s}_j),\ \forall j \neq i\}\,, \tag{3.23}$$

where $\mathbf{s}_i$ and $\mathbf{s}_j$ are the barycentric coordinates of two sites. The corresponding surface Voronoi diagram, using geodesic distance as the metric, is referred to as a geodesic Voronoi diagram (GVD). To represent the Voronoi diagram, we use only the site locations $\mathbf{s}$ as explicit degrees of freedom. The cell boundary vertices $\tilde{\mathbf{x}}$, which are either Voronoi vertices (with 3 or more adjacent cells) or intersections between Voronoi edges and mesh edges (with 2 adjacent cells), are given implicitly as the unique solution to the equidistance constraint

$$g(\tilde{\mathbf{x}}_i, \mathbf{s}_0) - g(\tilde{\mathbf{x}}_i, \mathbf{s}_j) = 0, j = 1 \ldots N_i - 1 \tag{3.24}$$

where N_i is the number of adjacent cells to cell boundary vertex $\tilde{\mathbf{x}}_i$ and $\mathbf{s}_0, \ldots, \mathbf{s}_{N_i-1}$ are the sites of adjacent cells. We compute the geodesic Voronoi diagram by solving

$$\tilde{\mathbf{x}}_i = \arg\min_{\tilde{\mathbf{x}}^*} \sum_{j=1}^{N_i-1} (g(\tilde{\mathbf{x}}^*, \mathbf{s}_0) - g(\tilde{\mathbf{x}}^*, \mathbf{s}_j))^2 \tag{3.25}$$

for each boundary vertex. Note that this requires prior knowledge of cell connectivity, which we obtain using a fast GVD approximation [73]; see also Algorithm 1.

BI-LEVEL OPTIMIZATION We now turn to optimizing geodesic Voronoi diagrams for higher-level objective functions. Consider a constrained optimization problem of the form

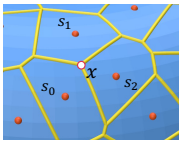
$$\min_{\tilde{\mathbf{x}},\mathbf{s}} O(\tilde{\mathbf{x}},\mathbf{s}) \quad \text{s.t.} \quad \mathbf{f}(\tilde{\mathbf{x}},\ \mathbf{s}) = \mathbf{0}\ , \tag{3.26}$$

where O is an *outer* design objective function and $\mathbf{f}$ is the vector-valued equidistance constraint function given by the gradient of the *inner* objective (3.25). The states $\tilde{\mathbf{x}}$ are coupled with the variables $\mathbf{s}$ through the constraint and we write $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}(\mathbf{s})$ to eliminate $\tilde{\mathbf{x}}$ as explicit degrees of freedom. Optimizing this objective with gradient-based methods requires the derivative $\frac{d\tilde{\mathbf{x}}}{d\mathbf{s}}$, which we obtain through sensitivity analysis. We refer to App. A.3 for derivations.

The equidistance constraints derive from the definition of Voronoi diagrams, *i.e.*, a point on a Voronoi edge must have the same distance to all adjacent sites. This includes Voronoi vertices as well as intersections between Voronoi edges and mesh edges, for which $\mathbf{f}$ takes different forms. For intersections between Voronoi edges and mesh edges, $\tilde{\mathbf{x}}_i$ is equidistant to two sites and $\mathbf{f}$ takes the form

$$\mathbf{f}_i(\tilde{\mathbf{x}}_i, \mathbf{s}_0, \mathbf{s}_1) = g(\tilde{\mathbf{x}}_i(\mathbf{s}_0,\mathbf{s}_1),\mathbf{s}_0) - g(\tilde{\mathbf{x}}_i(\mathbf{s}_0,\mathbf{s}_1),\mathbf{s}_1)\ . \tag{3.27}$$

In this case, only one constraint equation is necessary to define a unique point along the intersected mesh edge, parameterized by a scalar as in Eq. (3.1). For Voronoi vertices that are equidistant to three or more sites (see inset), we use two constraint equations



$$\begin{aligned} \mathbf{f}_{i,0}(\tilde{\mathbf{x}}_i, \mathbf{s}_0, \mathbf{s}_1) &= g(\tilde{\mathbf{x}}_i(\mathbf{s}),\mathbf{s}_0) - g(\tilde{\mathbf{x}}_i(\mathbf{s}),\mathbf{s}_1)\ , \\ \mathbf{f}_{i,1}(\tilde{\mathbf{x}}_i, \mathbf{s}_0, \mathbf{s}_2) &= g(\tilde{\mathbf{x}}_i(\mathbf{s}),\mathbf{s}_0) - g(\tilde{\mathbf{x}}_i(\mathbf{s}),\mathbf{s}_2)\ . \end{aligned} \tag{3.28}$$

We minimize the inner objective (3.25) using Newton's method and the outer design objective (3.26) using quasi-Newton methods. A standard back-tracking line search is applied to both optimization processes.

3.4.6 *Implementation Details*

Our code is implemented in C++ using Eigen [74] for primary data structure and Intel TBB for parallelization. Linear systems are solved using CHOLMOD [75]. We use the MMP algorithm [30] implemented in the Geometry Central Library [76] for computing exact geodesics. Finally, we credit

Polyscope [77] for producing figures. Our code is available through the repository `https://github.com/liyuesolo/DifferentiableGeodesics`.

3.5 RESULTS

3.5.1 *Energy-minimizing Geodesic Networks*

In the first set of examples (Fig. 3.4), we simulate elastic curve networks comprised of zero-length geodesic springs. We showcase our approach on two challenging setups where the endpoints of the geodesic springs are positioned close to mesh vertices or edges. Our approach converges robustly in both scenarios. As can be seen in the close-up views, we arrive at energy minima that differ significantly from the initial network.

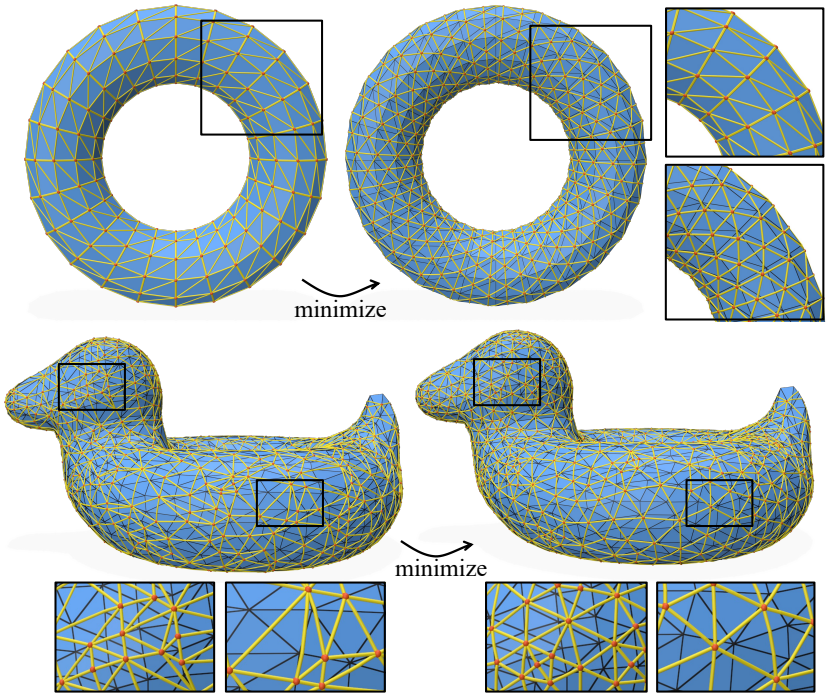


FIGURE 3.4: Elastic geodesic spring networks. We initialize the nodes (shown in red) in close proximity to either the vertices of the hosting mesh or its edge midpoints. Our method converges robustly in both scenarios.

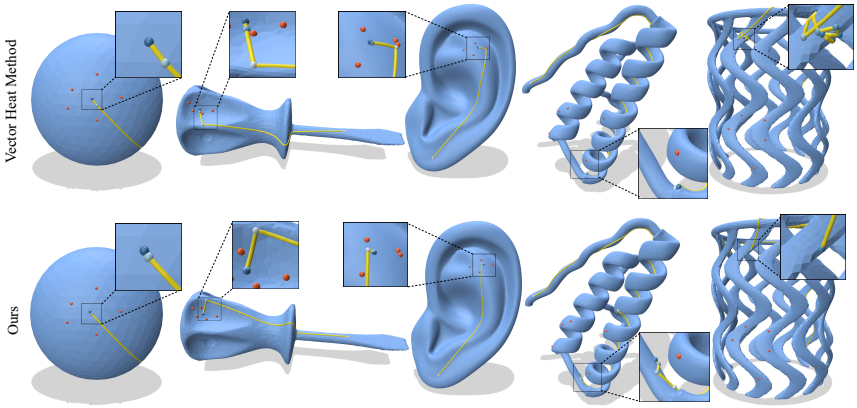


FIGURE 3.5: Qualitative comparisons with the Vector Heat Method [70] for computing Karcher Means on a selection of meshes. The yellow curves show the optimization trajectories toward the Karcher mean of a given set of points (shown in red). The initial guesses (chosen randomly) are shown in white and the Karcher mean in blue. The color gradient indicates steps toward the solution. As can be seen from these examples, our second-order solver allows for larger steps toward the minima (*col 1-3*) and significantly fewer iterations to convergence (*col 4-5*).

3.5.2 *Karcher Means*

Our approach enables second-order optimization for computing Karcher means on triangle meshes. We compare our approach to the state-of-the-art Vector Heat Method [70] on a set of examples with different geometries and resolutions. While Sharp *et al*. [70] propose an efficient method to compute the energy gradient by solving sparse linear systems, their approach is limited to linear convergence. As can be seen from the qualitative comparisons in Fig. 3.5, our second-order solver enables larger steps and converges to the solution in significantly fewer iterations. We use the open-source reference implementation from the Geometry Central Library [76], which adopts a relative convergence criterion based on the current triangle area and step size. For our experiments, we impose an absolute convergence criterion based on the gradient norm. Details on the convergence criteria are given in App. A.2. We report statistics for both convergence criteria in Table 3.1, showing that our approach converges to much tighter tolerances

Examples	Triangles	Time[s] (Heat*/Heat/**Ours**)	Iterations (Heat*/Heat/**Ours**)	GradientNorm	(Heat*/Heat/**Ours**)	Objective (Heat*/Heat/**Ours**)
sphere	1,280	0.0162 / 0.916 / **0.0284**	2 / 200 / **2**	4.35×10^{-4} / 3.04×10^{-5} / $\mathbf{2.31 \times 10^{-11}}$		0.0206 / 0.0204 / **0.00744**
screwdriver	6,786	0.119 / 17.5 / **0.291**	3 / 200 / **3**	3.04×10^{-4} / 2.67×10^{-4} / $\mathbf{1.57 \times 10^{-11}}$		0.00387 / 0.00383 / **0.00121**
ear	45,312	1.15 / 120 / **4.99**	3 / 200 / **3**	1.13×10^{-5} / 1.85×10^{-6} / $\mathbf{3.20 \times 10^{-12}}$		0.00974 / 0.00972 / **0.00225**
protein	60,820	17.0 / 120 / **12.8**	25 / 200 / **8**	1.81×10^{-1} / 9.64×10^{-2} / $\mathbf{3.05 \times 10^{-7}}$		0.970 / 0.991 / **0.208**
spiral cup	34,874	0.576 / 19.3 / **0.291**	13 / 200 / **10**	3.04×10^{-4} / 1.05×10^{-5} / $\mathbf{1.99 \times 10^{-11}}$		1.34 / 1.58 / **0.280**

TABLE 3.1: Comparison to the Vector Heat Method [70] for computing Karcher Means. We report the runtimes and optimization statistics for the examples shown in Fig. 3.5. *Heat** indicates using the convergence criterion in the official implementation [76], whereas *Heat* denotes using the same criterion as ours. While the Vector Heat Method converges to visually acceptable solutions in similar numbers of iterations, their gradient-based update is limited to linear convergence. Leveraging a Newton solver with analytical second derivatives, our approach converges quadratically.

with comparable performance. We set a maximum iteration of 200 for all approaches.

Another baseline for computing Karcher means is provided by Mancinelli and Puppo [78], which leverages approximate first and second derivatives of geodesic distance. While these approximations enable fast computations, the underlying convexity assumption for geodesic distance does not generally hold for arbitrary triangle meshes. Unfortunately, the publicly available algorithm[1] did not yield successful outcomes for any of the problem instances shown in Fig. 3.5. We therefore conduct a simpler test, computing the Karcher mean of five randomly placed points on a spherical mesh (Fig. 3.5, column 1). We repeat this experiment 100 times and report the average runtime, residual norm, and success rate in Table 3.2. Despite being an order of magnitude slower, our approach ensures robust convergence across all test cases.

Method	Avg Time [s]	Avg $\lvert \text{Grad} \rvert$	Success Rate
MancinelliPuppo	9.54×10^{-3}	1.40×10^{-2}	17%
Ours	$\mathbf{9.23 \times 10^{-2}}$	$\mathbf{6.52 \times 10^{-12}}$	**100%**

TABLE 3.2: Quantitative comparison with Mancinelli and Puppo [78] using 100 randomized problem instances on a spherical mesh. Averaged runtimes and gradient norms exclude failure cases.

1 https://github.com/Claudiomancinelli90/RCM_on_meshes

3.5.3 *Energy-minimizing Geodesic Triangles*

We simulate membranes defined by elastic geodesic triangles. Inspired by previous work from Li *et al*. [59], we simulate an elastic membrane made of geodesic triangles embedded in a rigid torus mesh (see Fig. 3.6). The torus mesh is stretched along the horizontal axis of the image plane in a non-uniform way. Simply keeping the barycentric coordinates of the embedded membrane unchanged leads to large isolated distortions (Fig. 3.6, top row). Optimizing these coordinates such as to minimize the membrane's elastic energy yields smoothly distributed deformations (Fig. 3.6, bottom row). Whereas simulating this effect in a 2D parametric space requires UV unwrapping [59], our approach operates directly on the 3D surface mesh. Our Newton solver converges in fewer than 5 iterations on average for this sequence.

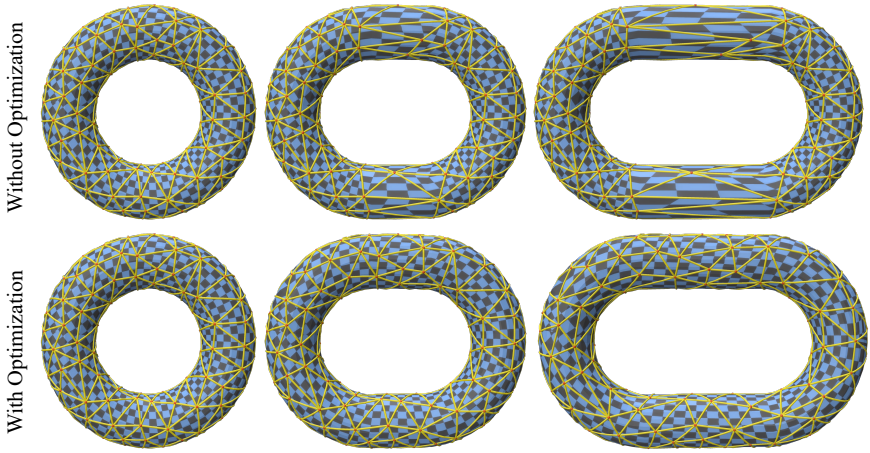


FIGURE 3.6: Simulation of an elastic membrane using geodesic triangles. We stretch the hosting torus mesh anisotropically to induce deformation of the embedded triangles. Keeping the (barycentric) membrane coordinates unchanged leads to large isolated distortions (*top row*). Optimizing coordinates such as to minimize the membrane's energy leads to smoothly distributed deformations (*bottom row*).

3.5.4 *Two-way Coupling*

We now consider two-way coupling between a geodesic elastic network and its hosting surface. In the first example, we simulate the tightening of an elastic geodesic network embedded in an inflated spherical shell (see Fig. 3.7). As the minimization proceeds, many regions of the shell bulge out in response to the compression forces induce by the embedded network. The hosting surface is simulated using a discrete shell model [71] augmented with a volume preservation term. To avoid factorizing a dense Hessian matrix resulting from the volume preservation term, we use the Sherman–Morrison formula [79] to compute the Newton step. In Fig. 3.1, we show that our intrinsic minimization pipeline can be coupled with a volumetric mesh discretized using tetrahedron finite elements. The hosting elastic object is simulated using a Neo-Hookean constitutive model. The ears of the bunny bend down and the back bulges in response to the tightening of the geodesic spring network.

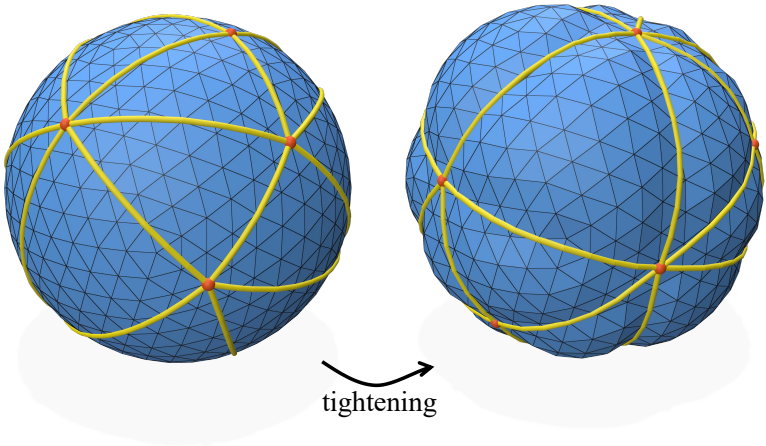


FIGURE 3.7: Two-way coupling. Simulation of a geodesic spring network on an inflated spherical shell. Realistic bulging effects emerge upon tightening the embedded curve network.

3.5.5 *Optimization of Geodesic Voronoi Diagrams*

We consider three examples with different design objectives defined on geodesic Voronoi diagrams. Each example is formulated as a bi-level optimization problem using sensitivity analysis as described in section 3.4.5.

LENGTH SIMILARITY In the first example, we optimize for uniform edge lengths such as to facilitate potential manufacturing with equal-length rods. To this end, we minimize the design objective

$$\min_{\mathbf{s}} O(\tilde{\mathbf{x}}(\mathbf{s}), \mathbf{s}) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} (g(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) - L)^2 \,, \tag{3.29}$$

where $\mathcal{E}$ is the index set of all Voronoi edges and L is the target edge length, which we set to the average Voronoi edge length in the initial configuration. As can be seen in Fig. 3.8, the optimized structures exhibit much less variation in edge length compared to the initial configurations.

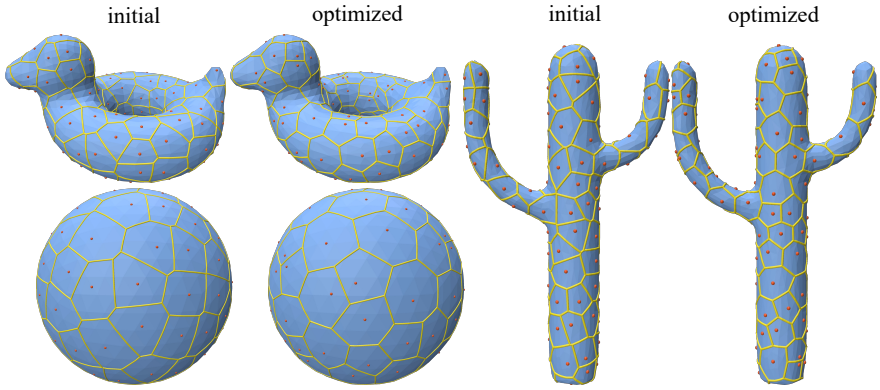


FIGURE 3.8: Optimizing for uniform edge length. By minimizing the deviation of all Voronoi edges from the target length, the optimized structure significantly reduces the length variation compared to the initial configuration.

CELL PLANARITY We consider the task of designing Voronoi diagrams with *as-planar-as-possible* cells such that the 3D structure can be assembled from flat panels. We define a corresponding objective as

$$\min_{\mathbf{s}} O(\tilde{\mathbf{x}}(\mathbf{s}), \mathbf{s}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N_i} \frac{1}{N_i} d(\tilde{\mathbf{x}}_j, \mathcal{P}_i(\tilde{\mathbf{x}}))^2 \,, \tag{3.30}$$

where N denotes the total number of Voronoi vertices and N_i is the number of vertices for Voronoi cell i. For each cell, we compute the least squares-fitting plane $\mathcal{P}_i$ and penalize the corresponding point-to-plane distance d for each cell vertex. As can be seen from Fig. 3.9, minimizing this planarity objective with our method leads to an order of magnitude reduction in average and maximum vertex-to-plane distance.

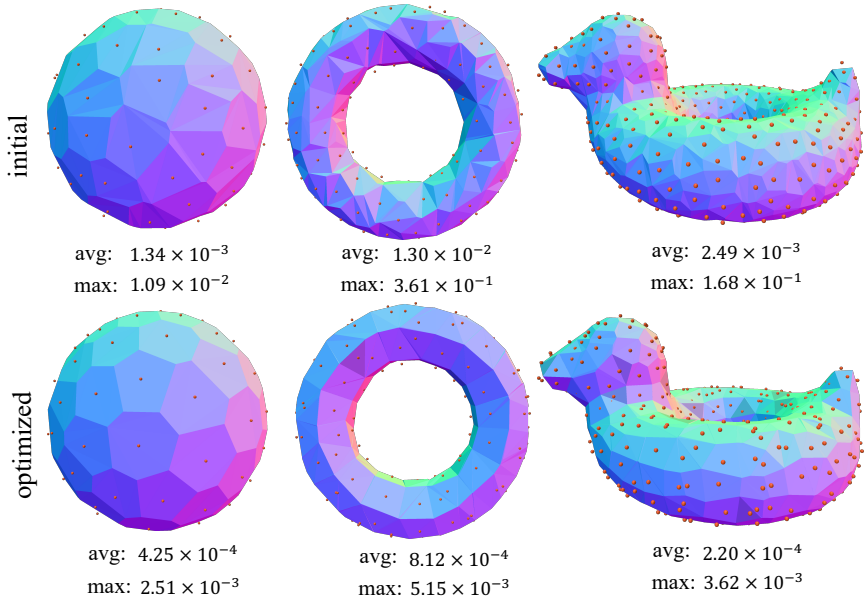


FIGURE 3.9: Optimizing for planarity. We optimize site locations such that the vertices of each Voronoi cell are as planar as possible. Our approach successfully reduces the average and maximum vertex-to-plane distance by an order of magnitude or more.

CELL REGULARITY Finally, we consider an objective to regularize cell shapes by minimizing the squared distances between each site and its surrounding vertices,

$$\min_{\mathbf{s}} O(\tilde{\mathbf{x}}(\mathbf{s}), \mathbf{s}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N_i} g(\mathbf{s}_i, \tilde{\mathbf{x}}_j(\mathbf{s}))^2 \,. \tag{3.31}$$

Fig. 3.10 illustrates the effect of this objective on a set of Voronoi diagrams with irregular initial cell shapes. Despite the complexity of the hosting surfaces, the resulting cell shape distributions show significant improvements.

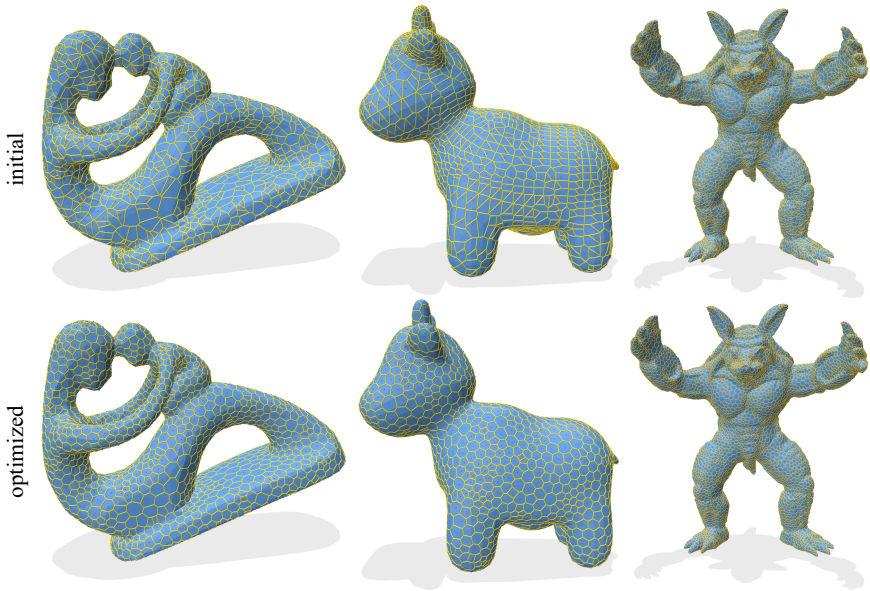


FIGURE 3.10: Optimizing cell regularity. Minimizing our regularity objective leads to quasi-isotropic cells when starting from a poorly shaped initial diagram.

3.5.6 *Timings*

We report detailed simulation statistics and averaged timings for all examples. The timings listed in Tables 3.1—3.4 are measured on a workstation with an *AMD Ryzen Threadripper PRO 5995WX* CPU.

Example	Triangles	DoFs	Two-way	Exact Geodesics	Constructing Hessian	Solving Linear System	Newton Step
Fig. 3.1	9,396	20,838	yes	221.55 ms	2.38 s	690.8 ms	3.64 s
Fig. 3.4 torus	576	1,152	no	73.43 ms	48.47 ms	42.58 ms	172.31 ms
Fig. 3.4 duck	2,000	4,000	no	791.46 ms	188.47 ms	114.28 ms	1.218 s
Fig. 3.5 sphere	1,280	2	no	8.19 ms	112.5 µs	63.5 µs	12.52 ms
Fig. 3.5 screwdriver	6,786	2	no	63.20 ms	12.24 ms	43.67 µs	94.47 ms
Fig. 3.5 ear	45,312	2	no	1.18 s	45.86 ms	59.6 µs	1.65 s
Fig. 3.5 protein	60,820	2	no	710.72 ms	948 ms	40.86 µs	1.668 s
Fig. 3.5 spiral cup	34,874	2	no	214.67 ms	285.3 ms	40.4 µs	555.91 ms
Fig. 3.6	2,304	282	no	89.36 ms	5.91 ms	596 µs	96.75 ms
Fig. 3.7	1,280	30	yes	8.28 ms	42.80 ms	65.64 ms	129.63 ms

TABLE 3.3: Simulation statistics and average timings.

Examples	Construct GVD[ms]	Time/Iter [s]	#Tri	#Sites
Fig. 3.8 sphere	5.87	1.90	1,280	98
Fig. 3.8 duck	9.97	4.39	2,000	125
Fig. 3.8 cactus	9.71	4.03	2,000	107
Fig. 3.9 sphere	14.37	0.474	1,280	396
Fig. 3.9 torus	6.41	0.0733	576	155
Fig. 3.9 duck	21.67	1.17	2,000	487
Fig. 3.10 fertility	66.46	7.09	2,250	2,250
Fig. 3.10 cow	93.12	39.3	5,856	2,928
Fig. 3.10 armadillo	262.6	180	20,000	10,000

TABLE 3.4: Statistics and average timing for energy-minimizing GVDs.

3.5.7 *Ablation Study*

COMPARISON WITH L-BFGS We show that our analytical Hessian significantly improves convergence compared to first-order or quasi-Newton methods without sacrificing performance. We compare the convergence and performance of different methods on two energy minimization problems involving geodesic spring networks. As can be seen from Fig. 3.11 (left), while L-BFGS offers better convergence than gradient descent (GD), Newton's method outperforms both by a large margin. In Fig. 3.11 (right), we show that our second-order approach also offers significant performance gains. Thanks to the compact analytical expression of the second derivative, computing the analytical Hessian of the geodesic distance only takes a fraction of the overall computation time. The initial and optimized configurations can be found in the inset figures. Dashed lines in the figures indicate residual tolerances that lead to qualitatively on-par results. We consider the results obtained from Newton and L-BFGS to be visually indistinguishable

when the maximum difference in nodal position is smaller than 1% of the scene bounding box diagonal.

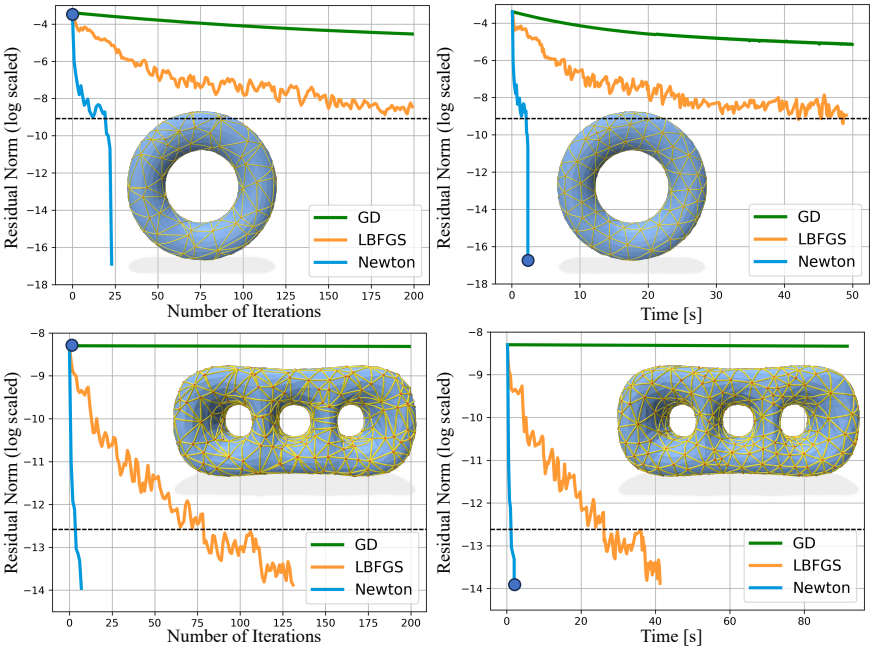


FIGURE 3.11: Convergence comparison. We use our second-order approach, gradient descent (GD), and L-BFGS to find equilibrium states of geodesic networks. Our approach exhibits quadratic convergence and demonstrates significantly better performance. Dashed lines indicate residual tolerances below which results become visually indistinguishable. Inset figures show the system states corresponding to the blue dots on the curves.

EUCLIDEAN DISTANCE Using Euclidean distance instead of geodesic distance greatly simplifies computation but leads to additional and undesirable local minima in intrinsic simulation. We illustrate this problem on an embedded curve consisting of three vertices and two connecting segments modeled as zero-length springs (shown in Fig. 3.12). The endpoints are anchored to the hosting surface while the central vertex is free to move. We compare simulations using Euclidean and geodesic distance metrics. In the left example, the hosting surface exhibits a right angle, creating a

local energy minimum for Euclidean-distance springs. In the right example, multiple local minima exist for Euclidean springs and the eventual equilibrium configuration depends on the initial position of the central vertex. Using geodesic distance, however, the simulation converges to the global optimum in both cases.

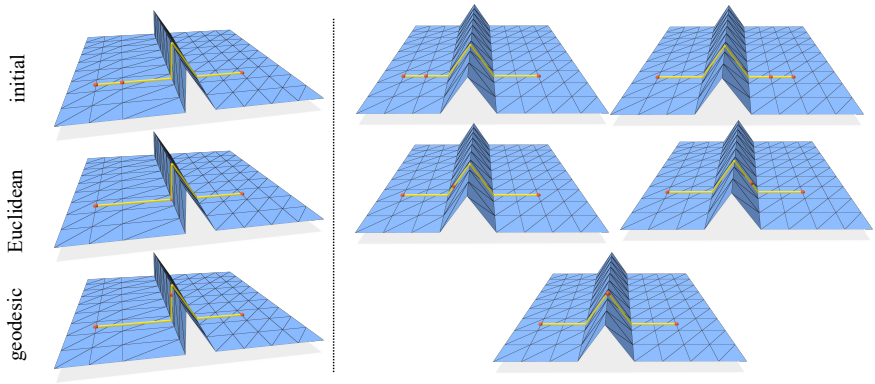


FIGURE 3.12: Spurious local minima when using Euclidean distance. In these two examples, we consider two zero-length springs connected by a free vertex and minimize the total elastic energy. Using Euclidean distance for the spring energy leads to undesired local minima while using geodesic distance resolves this problem. Geodesic paths are shown in all cases for visualization.

MOLLIFICATION To study the influence of our smooth mollifier, we set up a two-way coupling example, where the geodesic path passes through a hyperbolic vertex in the minimum-energy state (Fig. 3.13, *right*). In this example, a rectangle with an inner crossbar is embedded in a deformable shell. The four corner vertices of this rectangle are fixed to the surface and all edges are zero-length geodesic springs. In this case, energy minimization results in the inner edge passing through the hyperbolic vertex of the shell. As can be seen in the convergence plot to the left, without the mollifier, Newton's method struggles to converge even after 200 iterations. When the energy is mollified into a C^2-continuous function, Newton's method exhibits quadratic convergence.

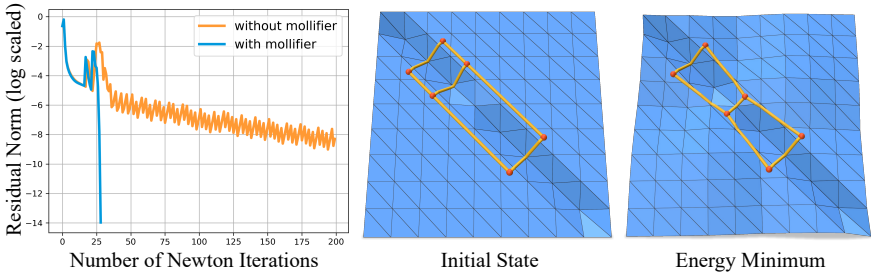


FIGURE 3.13: Mollification. In this two-way coupling example, the corners of the orange rectangle are fixed while the two inner vertices are free to slide on the surface. All embedded vertices are connected by geodesic springs with zero rest length. To arrive at the energy minimum, the inner spring must pass through the hyperbolic vertex of the hosting mesh. As can be seen from the convergence plot, without the mollifier (*orange*), Newton's method does not achieve quadratic convergence due to the lack of C^2-continuity at the minimum. With the mollifier, however, our approach converges quadratically (*blue*).

3.6 CONCLUSION & FUTURE WORK

We proposed a novel differentiable geodesic distance formulation for intrinsic minimization on triangle meshes. By applying the implicit function theorem to the variational formulation of shortest-path geodesics, we demonstrate that closed-form first and second derivatives can be obtained and significantly simplified, allowing for the use of Newton-type minimization solvers. We use our method to minimize a diverse set of objectives, including two-way coupling of embedded elastic structures with hosting geometries, Karcher means on arbitrary triangle meshes, and differentiable geodesic Voronoi diagrams.

LIMITATIONS While our intrinsic minimization paradigm enables robust second-order optimization on many instances of embedded elasticity, in almost all cases, the geodesics are under tension and the energy increases with geodesic length. However, when a geodesic curve is under compression and increasing length is energetically favorable, local distance maxima along the cut locus may lead to convergence failure.

We use the open-source implementation of the MMP algorithm for computing geodesic distance. While we perform this computation largely in

parallel, it remains the computational bottleneck of our method. While our triangle elements based on geodesic edge lengths are promising for simulating embedded elastic membranes, the approximation accuracy decreases as the curvature within each geodesic triangle increases. Using denser geodesic triangulations can alleviate this issue to some extent, but using internal quadrature points and additional degrees of freedom could be a promising direction. Finally, we only focused on triangle meshes in this work. Extending our intrinsic minimization paradigm to polygonal meshes is another avenue for future exploration.

4

PART III: NEURAL METAMATERIALS FOR NONLINEAR MATERIAL DESIGN

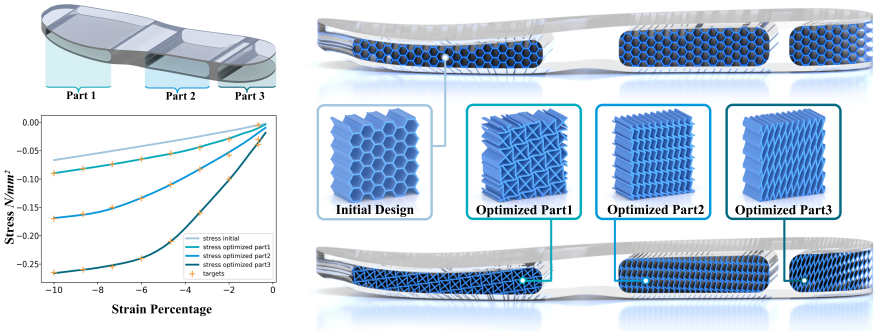


FIGURE 4.1: Inverse design with Neural Metamaterial Networks. For this custom shoe design, we aim to find metamaterials that best approximate given nonlinear stiffness targets in the forefoot, midfoot, and heel regions as shown on the *left*. Starting from a homogeneous hexagonal pattern, our method leverages a differentiable neural representation to optimize over entire metamaterial families, resulting in structures that closely track the desired strain-stress curves.

Nonlinear metamaterials with tailored mechanical properties have applications in engineering, medicine, robotics, and beyond. While modeling their macromechanical behavior is challenging in itself, finding structure parameters that lead to ideal approximation of high-level performance goals is a challenging task. In this work, we propose Neural Metamaterial Networks (NMN)—smooth neural representations that encode the nonlinear mechanics of entire metamaterial families. Given structure parameters as input, NMN return continuously differentiable strain energy density functions, thus guaranteeing conservative forces by construction. Though trained on simulation data, NMN do not inherit the discontinuities resulting from topological changes in finite element meshes. They instead provide a smooth map from parameter to performance space that is fully differentiable and thus well-suited for gradient-based optimization. On this basis, we formulate inverse material design as a nonlinear programming

problem that leverages neural networks for both objective functions and constraints. We use this approach to automatically design materials with desired strain-stress curves, prescribed directional stiffness and Poisson ratio profiles. We furthermore conduct ablation studies on network nonlinearities and show the advantages of our approach compared to native-scale optimization.

4.1 INTRODUCTION

Thanks to their programmable microstructures, mechanical metamaterials boast a wide range of macroscopic properties. There has recently been growing interest in flexible metamaterials that offer tailored mechanical properties in the finite-deformation regime [80]. Their ability to sustain large deformations makes these metamaterials interesting and useful for myriad applications including micro-electromechanical systems and precision instruments, functional sportswear and rehabilitation medicine, as well as robotics and architecture. A common trait shared by these examples is that they operate at very large deformations, which means that nonlinear effects in both geometry and material become predominant: beams in the microstructure rotate and straighten under stretching, whereas compression leads to buckling and self-contact.

To design in this nonlinear setting, we must be able to determine structure parameters that lead to an ideal approximation of given high-level performance goals. This problem is challenging for two main reasons. First, creating a nonlinear macromechanical model for a given metamaterial is a highly nontrivial task. While Finite Element simulations can be used to generate large amounts of stress-strain data, combining these data into an accurate representation that preserves first principles such as objectivity, convexity, and integrability is difficult unless limiting assumptions on material symmetries and nonlinearities are introduced. Second, changing a material's structure parameters will entail changes in the simulation meshes. While these changes are often smooth, there will inevitably be singular points at which mesh topology must change to accommodate changes in geometry. These topology changes, and the corresponding discontinuities in simulation derivatives, are challenging the robustness of gradient-based optimization methods.

In this work, we propose Neural Metamaterial Networks (NMN) to model the nonlinear macromechanical behavior of entire metamaterial families. Given structure parameters as input, NMN return a continuously differen-

tiable strain energy density function, which guarantees conservative forces by construction. Though trained on simulation data, NMN do not inherit the discontinuities resulting from topological changes in finite element meshes. They instead provide a smooth map from parameter to performance space that is fully differentiable and thus ideally suited for gradient-based optimization. Using neural metamaterial networks as a basis, we formulate inverse material design as a nonlinear programming problem that leverages neural networks for both the objective function and the constraints.

We demonstrate our approach on a set of inverse material design examples where we automatically compute structure parameters that best approximate desired strain-stress curves, prescribed directional stiffness, and Poisson ratio profiles. We furthermore conduct ablation studies on our choice of network nonlinearities, and show the advantage of computing gradient information from our neural representation compared to simple finite-differencing on native scale simulations.

This chapter is based on our previous publication [81].

4.2 RELATED WORK

STRUCTURED METAMATERIALS. Through careful tuning of their microscale geometry, metamaterials can cover a wide range of marcomechanical properties. Designing metamaterials with desired macromechanical properties is a problem that spans many areas explored in graphics, including geometry processing, physics-based modeling, and computational fabrication. Consequently, there has recently been a surge in works that explore various types of metamaterials [5, 6, 8, 82–86]. For example, Schumacher *et al*. [5] characterize the in- and out-of-plane behavior of structured sheet materials. Li *et al*. [6] proposed a methodology for creating 3D-printable, weave-like materials with desired mechanical properties. Based on star-shape metrics, Martínez *et al*. [8] and Efremov *et al*. [86] introduce spatially graded meta-materials that cover a wide range of elastic properties. Martínez *et al*. [84] leverage procedural Voronoi foams to efficiently create orthotropic microstructures. Moreover, Thibault *et al*. [85] generate freely-orientable microstructures with locally controllable anisotropy. While the above methods explore specific types of metamaterials, we address the fundamental problem of generating smooth representations of nonlinear macromechanical properties for entire metamaterial families—a problem that is at the core of inverse metamaterial design.

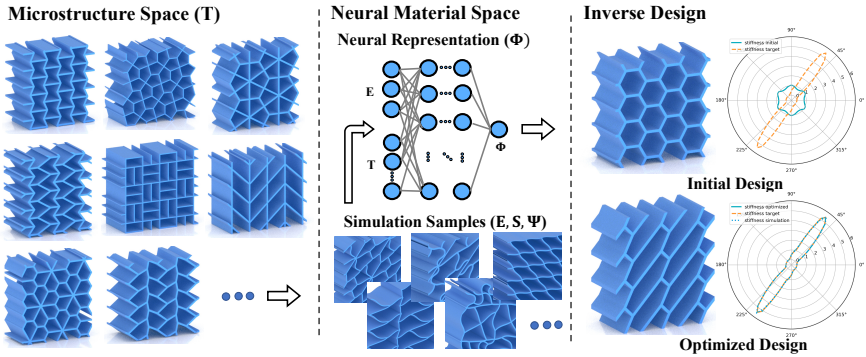


FIGURE 4.2: Pipeline. Our metamaterials leverage families of isohedral tiling patterns as geometries. For a given set of tiling parameters **T**, we sample the nonlinear response of the corresponding unit cell by computing equilibrium configurations for uni- and bi-axial states of strain with different directions and magnitudes. Using these simulation data, we perform homogenization on the unit patch to extract macroscale descriptions, *i.e.* Green strain **E**, Second Piola-Kirchhoff stress **S**, and the energy density Ψ. To capture the constitutive relationship for an entire metamaterial family, we train a deep neural network to learn the map from strain **E** and tiling parameters **T** to energy density functions Φ. The resulting Neural Metamaterial Networks are ideally suited for inverse material design with analytical derivatives.

NONLINEAR MATERIAL DESIGN. Vast parameter spaces for complex, nonlinear materials require inverse design approaches for efficient, performance driven navigation. With the goal of matching desired deformation behaviors, Bickel *et al*. [87] optimize for the best arrangements of layered microstructures. By varying beam density and radii, Martínez *et al*. [88] find 3D Voronoi foams for prescribed Young's modulus targets. Zehnder *et al*. [89] optimize for the sizes, numbers, and locations of stiff inclusions to generate silicone metamaterials with desired mechanical properties. Panetta *et al*. [90] introduce a library of elastic truss-like structures and a two-stage shape optimization process to find the best matching structure for a given target elasticity tensor. Building on this approach, Panetta *et al*. [91] further perform shape optimization to reduce peak stresses for worst-case loads. Schumacher *et al*. [92] find a set of microstructures with smoothly varying material properties, and Tozoni *et al*. [93] map material parameters to families of rhombic microstructures. Our method shares the same high-level

goal of finding metamaterials that best approximate macromechanical performance goals. However, instead of interpolating strain-stress samples in parameter space, we leverage deep neural networks to learn energy density functions for a large range of deformations across entire metamaterial families.

Previous work has successfully used sampling-based strategies for inverse metamaterial design [90, 92, 94] in the context of linear elasticity. Whereas linear materials are described by a single elasticity tensor, our work targets nonlinear materials, which require significantly more data (*i.e.*, multiple stress-strain samples). While interpolation between linear elasticity tensors can lead to reasonable behavior, it is unclear how to interpolate between nonlinear materials. Close to our work that also uses isohedral tilings, Schumacher *et al*. [5], constructs stiffness profiles in a pre-processing step and retrieves the closest matches for given targets at runtime. Whereas they only support isotropic strains with two pre-defined magnitudes, our method extends to arbitrary states of strain with a continuous range of magnitudes. More importantly, Schumacher *et al*. only provide a few snapshots of a material's performance whereas our method offers a complete and smooth material model that can be used for inverse design and forward simulation.

An alternative strategy for metamaterial design is to use topology optimization [94–102]. This approach can automatically discover optimal microstructures for given performance goals. However, whereas topology optimization returns a single microstructure for a given target behavior, our approach explores entire metamaterial families. While another stream of research explores triply periodic minimal surfaces as a design space for microstructures for two scale topology optimization [103–105], they target linear elasticity and small displacements, whereas we address nonlinear elasticity for large deformations. Moreover, we use machine learning for macromechanical characterization.

DIFFERENTIABLE SIMULATION A recent stream of graphics research focuses on differentiable simulation, which requires derivatives of simulation outputs w.r.t. input parameters. Whereas some works rely on auto-differentiation [106–108], others employ the adjoint method to compute the analytical derivative [90, 91, 109–113]. Panetta *et al*. [90, 91] optimize for structure parameters using analytical shape derivatives obtained from a level set formulation. Different from their work, which requires re-meshing and native-scale simulation per optimization step, we propose a novel learning-based approach that removes the need for native-scale simulation

and re-meshing during inverse design. Similar in spirit to previous work that replaced simulation with a learned model for optimization [114, 115], we leverage deep neural networks as analytically differentiable surrogates for nonlinear metamaterial design.

DATA-DRIVEN CONSTITUTIVE MODEL. Modeling material behavior is a problem that has received much attention from the graphics community in recent years. One stream of research focuses on fitting material parameters of existing models to real-world data [116–119]. Another line of works explores dedicated material models, *e.g.* for cloth [120, 121] and hyperelastic solids [122–125]. Most similar to our approach are methods based on neural networks [126–130]. While machine learning approaches for modeling constitutive relationships have been explored since the early 90s [131, 132], the vast majority do not guarantee conservative forces. For example, Wang *et al.* [126] learn corrective stresses for nominal models from simulation data. However, as stress corrections do not derive from a potential, they are generally not conservative. A notable exception is the work by Masi *et al.* [127], who propose a method for learning constitutive models using neural networks with guaranteed thermodynamical consistency. Whereas Le *et al.* [128] directly map linear strain tensors to energy density, Linka *et al.* [129] precompute generalized strain invariants and then learn an energy density function whose derivatives best approximates nonlinear stress-strain data. Beyond the purely elastic regime, Li *et al.* [130] learn energy representations for metal, sand, and snow plasticity. Similar to these works, our method also leverages deep neural networks to learn constitutive models from simulation data. To the best of our knowledge, however, our method is the first to model entire families of metamaterials using neural networks.

HOMOGENIZATION. Computational homogenization [133–135] aims to extract the macromechanical behavior from micro-scale geometry. Within the graphics community, homogenization theory has been explored for distilling coarse-scale material properties from fine-scale simulations [136–138] as a means to reduce computation times.

Using different computational models for micro- and macro-scale simulations, Sperl *et al.* [139] conduct homogenization on yarn-level simulation data to enable efficient knitwear simulation using a macro-scale shell solver. Most closely related to our work are methods that use homogenization to compute the macromechanical properties of metamaterials [5, 6,

90, 92]. While we also compute homogenized properties from simulation data distributed across strain space, we learn a continuous, analytically-differentiable energy density function whose derivatives best approximate the simulated strain-stress behavior.

4.3 METHOD

In this section, we describe the machinery required for nonlinear metamaterial design using Neural Metamaterial Networks. See also Fig. 4.2 for an overview. We use finite element simulation to model the nonlinear mechanics of metamaterials at their native scale (Sec. 4.3.2), and extract corresponding macromechanical properties using concepts from homogenization theory (Sec. 4.3.3). We use this data to train Neural Metamaterial Networks, *i.e.*, deep neural networks that smoothly map structure parameters to macromechanical descriptions in the form of energy density functions and corresponding nonlinear stress-strain relationships (Sec. 4.3.4). Finally, we demonstrate the potential of Neural Metamaterial Networks for a set of inverse design tasks, leveraging the analytical derivatives of the neural networks (Sec. 4.3.5).

4.3.1 *Isohedral Tilings*

Our method targets parametric metamaterial families that are described through a set of variables defining the material's microstructure. Here we focus on isohedral tilings, a particular class of parametric families that span a large and diverse space of two-dimensional patterns [5, 140][1]. An isohedral tiling family consists of patterns made by repeating a single base shape without gaps and overlaps. The geometry of the base shape is controlled by a number of continuous structure parameters $\mathbf{T}$ (see Fig. 4.3). There are no topology changes within families, but topologies generally vary across families. We extrude these patterns out of their plane to obtain three-dimensional materials that are simple to manufacture using, *e.g.*, 3D printers based on fused filament fabrication.

1 See also https://isohedral.ca/software/tactile/

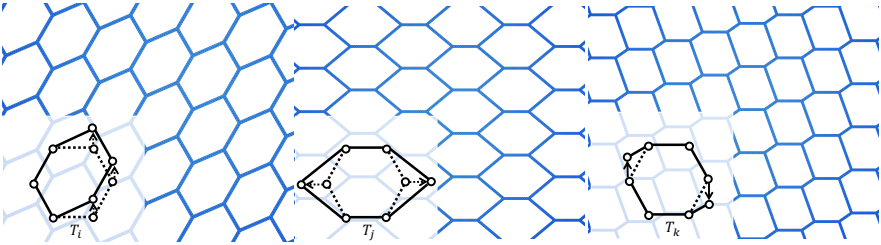


FIGURE 4.3: Tiling parameters of family IH01. Varying structure parameters leads to continuous changes in the basic tiling geometry (black hexagon). The effect of each parameter in this family is shown in the insets.

4.3.2 *Microscale Simulation*

Since we are only interested in their in-plane behavior, we assume that loads along the normal direction are negligible. The corresponding plane stress assumption allows for an entirely two-dimensional treatment. For a given set of tiling parameters $\mathbf{T}$, we first assign a width to all the tiling edges and then discretize them with quadratic triangle elements for simulation. We perform simulation on a periodic patch made of a 2×2 tiling of the minimal unit cell, see also Fig. 4.5.

We compute static equilibrium configurations for given macroscopic strains by minimizing the potential

$$\begin{aligned} E_{\text{total}} &= E_{\text{elastic}}(\mathbf{x}_r(\mathbf{T}), \mathbf{X}_r(\mathbf{T})) \\ &\quad + E_{\text{strain}}(\mathbf{x}_r(\mathbf{T}), \mathbf{X}_r(\mathbf{T})) + E_{\text{contact}}(\mathbf{x}_r(\mathbf{T}))\ , \end{aligned} \tag{4.1}$$

where $\mathbf{x}_r$ and $\mathbf{X}_r$ are the reduced simulation degrees of freedom (DoF) in the current and rest configurations, respectively. We define $\mathbf{x}_r := (\mathbf{x}, \mathbf{t}_{ij}, \mathbf{t}_{kl})$, where $\mathbf{t}_{ij}$ and $\mathbf{t}_{kl}$ are the translational DoF, with which the vertex positions on one side of the periodic patch is obtained from vertices on the other side. Therefore $\mathbf{x}$ excludes vertices on two adjacent boundaries of the unit patch. With this treatment, periodicity is imposed implicitly as a hard constraint. We use a standard incompressible Neo-Hookean material [141] for the elastic potential E_{elastic}, setting its Young's modulus and Poisson ratio according to the specifications of the printing filament. We impose uni- and bi-axial strain conditions through penalty function E_{strain}, asking

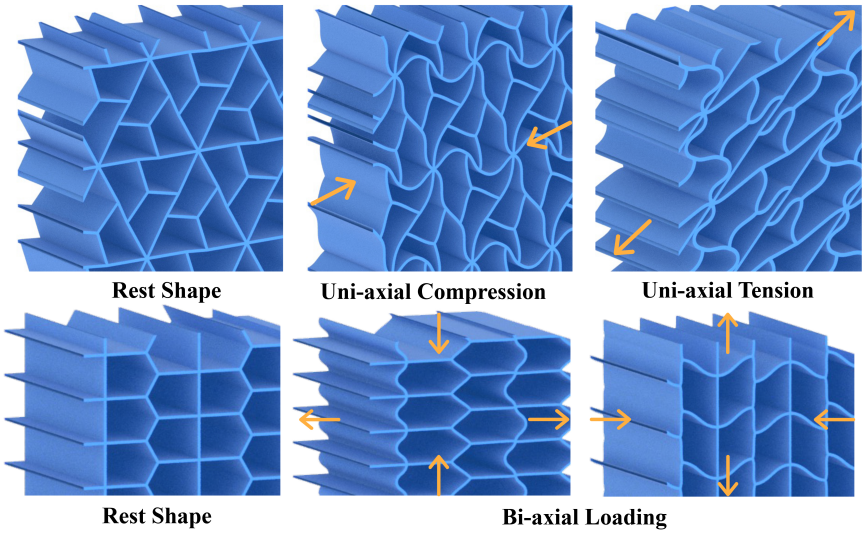


FIGURE 4.4: Simulation Samples. Here we show the static equilibrium configurations for uni- and bi-axial loading as indicated in orange. As can be seen from these images, our simulation framework reliably captures the substantial nonlinear buckling behaviors without any self-intersections.

that displacements of corresponding boundary vertices comply with target deformations. Specifically, we define E_{strain} as

$$E_{strain} = \omega_\epsilon \sum_i (\mathbf{b}_i^T \mathbf{d}_\alpha - E_\alpha \mathbf{B}_i^T \mathbf{d}_\alpha)^2 + (\mathbf{b}_i^T \mathbf{n}_\alpha - E_\alpha^\perp \mathbf{B}_i^T \mathbf{n}_\alpha)^2 , \tag{4.2}$$

where ω_ϵ is the penalty weight, $\mathbf{d}_\alpha$ is the direction vector

$$\mathbf{d}_\alpha = [\cos(\alpha), \sin(\alpha)]^T , \tag{4.3}$$

and $\mathbf{n}_\alpha$ is the vector orthogonal to it. Furthermore, E_α and $E_\alpha^\perp$ are the corresponding strain magnitude in the loading direction (for uni-axial loading) and the orthogonal direction (for bi-axial loading). The distance vectors for a given periodic boundary pair in its deformed and undeformed location are given by $\mathbf{b}_i$ and $\mathbf{B}_i$. We set $\omega_\epsilon = 10^6$ for all our experiments. The constraint violation is practically zero ($5.432 \times 10^{-5}\%$ on average) and we observe no failure cases for convergence. Finally, we use the logarithmic barrier potential $E_{contact}$ with an adaptive weighting strategy proposed by

Li *et al*. [142] to handle self-contacts. Note that contacts need to be resolved both within and across the simulation unit. For this purpose, we compute E_{contact} with a 2×2 tiling of the simulation unit, while retaining the DoF of a single unit. The corresponding derivatives are trivial to compute for this linear map. The total energy is minimized with Newton's method augmented with a back-tracking line search strategy that ensures inversion- and intersection-free steps [142] and adaptive diagonal regularization on the Hessian.

4.3.3 *Macroscale Homogenization*

HOMOGENIZATION We describe the macromechanical behavior of our metamaterials through the second Piola-Kirchhoff stress tensor $\mathbf{S}$ as a function of the Green-Lagrange strain tensor $\mathbf{E}$. These two quantities are work-conjugate [141], *i.e.*, they satisfy the differential relation

$$\mathbf{S}(\mathbf{E}) = \frac{\partial \Psi(\mathbf{E})}{\partial \mathbf{E}}\,, \tag{4.4}$$

where Ψ is the strain energy density function. While we use the Green-Lagrange strain and the second Piola-Kirchhoff stress, our formulation readily extends to other work-conjugate pairs such as deformation gradient and first Piola-Kirchhoff stress. We compute the deformation gradient $\mathbf{F}$

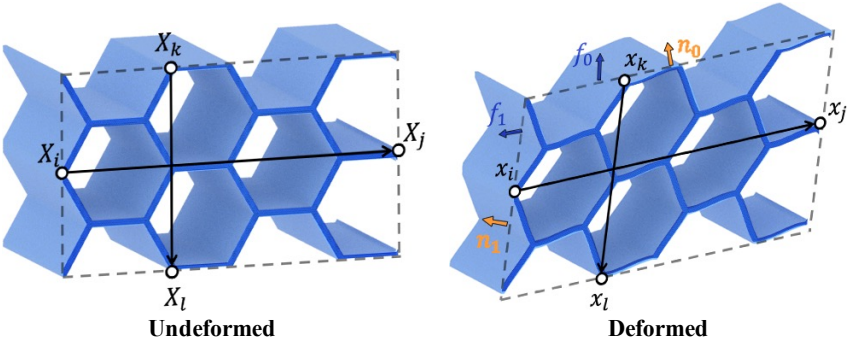


FIGURE 4.5: Homogenization. We compute the deformation gradient from corresponding vertex pairs on opposite boundaries of the simulation patch in its deformed (right) and rest (left) state. The averaged Cauchy stress is computed from the internal forces on the boundaries and their corresponding normals.

and the Cauchy stress $\boldsymbol{\sigma}$ from the boundary of the simulation patch as

$$\begin{aligned} \mathbf{F} &= \begin{bmatrix} \mathbf{x}_j - \mathbf{x}_i & \mathbf{x}_l - \mathbf{x}_k \end{bmatrix} \begin{bmatrix} \mathbf{X}_j - \mathbf{X}_i & \mathbf{X}_l - \mathbf{X}_k \end{bmatrix}^{-1} , \\ \boldsymbol{\sigma} &= \begin{bmatrix} \mathbf{f}_0 & \mathbf{f}_1 \end{bmatrix} \begin{bmatrix} \mathbf{n}_0 & \mathbf{n}_1 \end{bmatrix}^{-1} , \end{aligned} \tag{4.5}$$

where $\mathbf{x}$ and $\mathbf{X}$ denote the deformed and undeformed locations of corresponding vertex pairs i, j and k, l on opposite boundaries of the simulation patch. Furthermore, $\mathbf{f}_0, \mathbf{f}_1$ and $\mathbf{n}_0, \mathbf{n}_1$ are the forces and normal directions on adjacent boundaries. See also Fig. 4.5. Having computed these macroscale quantities, the corresponding Green-Lagrange strain and second Piola-Kirchhoff stress follow as

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) , \quad \mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}^T\mathbf{F}^{-T} , \tag{4.6}$$

where $J = \det \mathbf{F}$. Finally, the energy density of the unit patch is computed as

$$\Psi = \frac{E_{\text{total}}}{V} , \tag{4.7}$$

where V is the volume of the simulation patch.

DATA GENERATION To sample a given material's stress-strain response, we impose macroscopic strains corresponding to uni- and bi-axial loading with different directions and magnitudes. Although uni-axial loading could be considered a special case of bi-axial loading, it is arguably the scenario that occurs most frequently in applications. It is also worth noting that uni-axial loading generally leads to bi-axial strain: the *lateral* strain orthogonal to the *axial* load is the one that minimizes the elastic energy—and, depending on the material's Poisson ratio, the energetically-optimal lateral strain is generally nonzero.

Instead of uniformly sampling bi-axial load cases, we, therefore, create a dedicated set corresponding to uni-axial loading such as to properly capture this characteristic material response. We start by uniformly sampling directions ranging from 0 to π. For uni-axial loading, we create a 1D grid of samples along the given direction, ranging from 30% compression to 50% tension. For bi-axial loading, we generate a 2D grid of equidistant samples ranging from 10% compression to 20% tension along either axis. For a given set of tiling parameters, we thus run 2,250 simulations in total.

When sampling the tiling parameters, we first define a region of interest in parameter space such that no self-intersection occurs in the unit cell and

perform uniform sampling within this range. Depending on the number of structure parameters, we arrive at a total of 900,000 to 4,500,000 simulations per metamaterial family.

4.3.4 *Neural Metamaterials*

With the simulation data in hand, our goal is to find a smooth representation of the macromechanical behavior across entire metamaterial families. Interpolating stress-strain data with radial basis function is an option that has been explored in the past [117]. While we can expect reasonable behavior close to samples, stresses will generally not be conservative away from the data and it is unclear how to estimate, or even bound, integrability error a-priori. To avoid these problems by construction, we turn to machine learning and use deep neural networks to learn a differentiable energy density function whose derivatives best approximate the stress-strain data obtained from simulation. Training this network on simulation data from many different structures means that the resulting energy density function will describe the constitutive behavior across entire metamaterial families. This neural representation—which we term Neural Metamaterial Networks (NMN)—has two important advantages. First, stresses are obtained from the network through standard back-propagation. Since these stresses derive from a potential, they are guaranteed to be conservative everywhere—even away from data points. Second, the network is agnostic to parametric discontinuities that would arise due to mesh connectivity changes in native-scale simulations. Consequently, the derivatives of the network with respect to structure parameters are smooth and, as discussed in Sec. 4.3.5, can thus be used for inverse material design.

As illustrated in Fig. 4.6, our Neural Metamaterial Network uses a two-branch Multi-Layer Perceptron (MLP) architecture that receives tiling parameters $\mathbf{T}$ and strain $\mathbf{E}$ as separate inputs and returns the corresponding energy density Φ as output. Both inputs are fed into sub-blocks of 3 layers and then concatenated together to pass through the remaining layers. We use 256 neurons and Swish activation functions [143], *i.e.*, $x \rightarrow x \cdot \text{sigmoid}(x)$, for all layers except the final one, which uses a *softplus* activation function, $x \rightarrow \log(1 + \exp(x))$, to ensure positive energy. We refer to Sec. 4.4.3 for an ablation study on the influence of different activation functions.

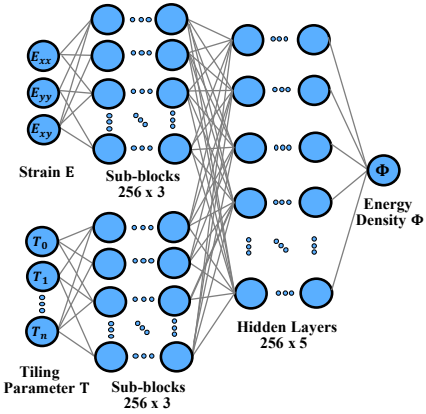


FIGURE 4.6: Network Architecture. We use a two-branch MLP that maps strain and tiling parameters to energy density.

For training, we use both energy densities $\Psi_i = \Psi(\mathbf{E}_i)$ and stress-strain pairs $(\mathbf{S}_i, \mathbf{E}_i)$ from the simulation data and minimize the loss function

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & L_{NN}(\boldsymbol{\theta}, \mathbf{E}, \mathbf{T}, \boldsymbol{\Psi}) \\ & = \frac{1}{N_{sp}} \sum_{i=0}^{N_{sp}} \left(\frac{\Phi(\mathbf{T_i}, \mathbf{E}_i, \boldsymbol{\theta}))}{\Psi_i} - 1 \right)^2 \\ & + \frac{1}{N_{sp}} \sum_{i=0}^{N_{sp}} \left\| \frac{1}{\|\mathbf{S}_i\|_2} \left(\frac{\partial \Phi(\mathbf{T}_i, \mathbf{E}_i, \boldsymbol{\theta})}{\partial \mathbf{E}_i} - \mathbf{S}_i \right) \right\|_2^2, \end{aligned} \tag{4.8}$$

with respect to the network parameters $\boldsymbol{\theta}$. The first and second terms in this expression penalize the differences in energy and gradient across all N_{sp} samples. We further normalize each sample's contribution with respect to its target values, thus ensuring proper relative weighting for different strain magnitudes.

4.3.5 *Optimization-based Inverse Design*

Neural Metamaterial Networks offer a smooth representation of the macromechanical stress-strain behavior of a given metamaterial family. This description directly supports interactive explorations of the parameter space with instant feedback on the nonlinear mechanics of any material inside the

corresponding family. What is more, derivatives of energy density and stress-strain behavior with respect to structure parameters can be obtained in closed form through simple back-propagation. As we describe below, this gradient information opens the door to optimization-based material design algorithms that automatically compute tiling parameters such as to best approximate macromechanical performance goals.

UNIAXIAL STRESS-STRAIN PROFILE. A basic inverse design task is to ask for a material whose strain-stress curve best interpolates a set of target pairs $(E_{\alpha,i}, S_{\alpha,i})$ under uniaxial loading in a given direction α. We formulate this task as a bi-level optimization problem

$$\min_{\mathbf{T}} \; O(\mathbf{T}, \mathbf{E}(\mathbf{T})) = \sum_i \left(\left(\mathbf{d}_\alpha^T \frac{\partial \Phi(\mathbf{T}, \mathbf{E}_{\alpha,i}(\mathbf{T}))}{\partial \mathbf{E}_{\alpha,i}} \mathbf{d}_\alpha \right) - S_{\alpha,i} \right)^2 \tag{4.9a}$$

$$\text{s.t.} \quad \mathbf{T}_{\min} \leqslant \mathbf{T} \leqslant \mathbf{T}_{\max}, \tag{4.9b}$$

$$\mathbf{E}_{\alpha,i} = \underset{\mathbf{E}^*_\alpha}{\operatorname{argmin}} \, \Phi(\mathbf{E}^*_\alpha, \mathbf{T}), \quad \forall i \qquad \text{s.t.} \quad \mathbf{d}_\alpha^T \mathbf{E}^*_\alpha \mathbf{d}_\alpha = E_{\alpha,i} \tag{4.9c}$$

where $\mathbf{T}_{\min}, \mathbf{T}_{\max}$ are the parameter bounds used to generate the training data. The outer objective penalizes deviations from the target stress $S_{\alpha,i}$, whereas the inner optimization ensures that the associated strains $\mathbf{E}_{\alpha,i}$ have the prescribed magnitude $E_{\alpha,i}$ in the direction of loading while minimizing the elastic energy orthogonal to it.

DIRECTIONAL STIFFNESS PROFILE. Instead of prescribing a particular strain-stress behavior in a given direction, we can also specify directional stiffness profiles. Working with these directional stiffness profiles also enables control over isotropic, orthotropic, or completely anisotropic behavior.

We cast directional stiffness design as another bi-level optimization problem

$$\begin{aligned}
&\min_{\mathbf{T}} \; O(\mathbf{T}, \mathbf{E}(\mathbf{T})) \\
&= \sum_{\alpha} \left(\left((\mathbf{d}_\alpha \mathbf{d}_\alpha^\mathsf{T}) : \mathbb{S}(\mathbf{T}, \mathbf{E}_\alpha(\mathbf{T}))_\alpha : (\mathbf{d}_\alpha \mathbf{d}_\alpha^\mathsf{T}) \right)^{-1} - k_\alpha \right)^2 , && \text{(4.10a)} \\
&\quad \text{s.t.} \quad \mathbf{T}_{\min} \leqslant \mathbf{T} \leqslant \mathbf{T}_{\max}, && \text{(4.10b)} \\
&\qquad \mathbf{E}_\alpha = \underset{\mathbf{E}^*_\alpha}{\operatorname{argmin}} \, \Phi(\mathbf{E}^*_\alpha, \mathbf{T}), \quad \forall \alpha \\
&\qquad\quad \text{s.t.} \quad \mathbf{d}_\alpha^\mathsf{T} \mathbf{E}^*_\alpha \mathbf{d}_\alpha = E_\alpha && \text{(4.10c)}
\end{aligned}$$

in which the outer objective aims to match the directional stiffness targets k_α and $\mathbb{S}_\alpha$ is the compliance tensor for direction α. The inner objective again ensures that strain corresponds to uni-axial loading. It should be noted that the compliance tensor, which is defined through

$$\mathbb{S}_\alpha : \mathbb{C}_\alpha = \mathbb{I} \quad , \quad \mathbb{C}_\alpha = \frac{\partial \Phi^2(\mathbf{T}, \mathbf{E}_\alpha)}{\partial \mathbf{E}_\alpha^2} , \tag{4.11}$$

involves second derivatives of the network. In practice, we compute $\mathbb{S}_\alpha$ by inverting $\mathbb{C}_\alpha$ in matrix form.

POISSON RATIO PROFILE. Optimizing for stiffness curves along given directions, or profiles spanning all directions, corresponds to designing for generalized Young's moduli. We can likewise define design objectives for generalized Poisson's ratios that control the contraction (or expansion) rate in the directions orthogonal to uni-axial loading. To this end, we pose another optimization problem,

$$\begin{aligned}
&\min_{\mathbf{T}} \; O(\mathbf{T}, \mathbf{E}(\mathbf{T})) \\
&= \sum_{\alpha} \left(- \frac{(\mathbf{d}_\alpha \mathbf{d}_\alpha^\mathsf{T}) : \mathbb{S}(\mathbf{T}, \mathbf{E}_\alpha(\mathbf{T}))_\alpha : (\mathbf{n}_\alpha \mathbf{n}_\alpha^\mathsf{T})}{(\mathbf{d}_\alpha \mathbf{d}_\alpha^\mathsf{T}) : \mathbb{S}(\mathbf{T}, \mathbf{E}_\alpha(\mathbf{T}))_\alpha : (\mathbf{d}_\alpha \mathbf{d}_\alpha^\mathsf{T})} - \nu_\alpha \right)^2 ,
\end{aligned} \tag{4.12}$$

where ν_α are target values for Poisson's ratio and $\mathbf{n}_\alpha$ is the direction orthogonal to $\mathbf{d}_\alpha$. It should be noted that this expression is subject to the constraints defined in Eq. 4.10a, which we omitted here for brevity.

OPTIMIZATION. The constrained optimization problems introduced above can be expressed in general form as

$$\begin{aligned} &\min_{\mathbf{T}} \; O(\mathbf{T}, \mathbf{q}(\mathbf{T})) \\ &\text{s.t.} \quad \mathbf{g}(\mathbf{T}, \mathbf{q}(\mathbf{T})) = \mathbf{o}, \end{aligned} \tag{4.13}$$

where $\mathbf{g}$ is a vector-valued constraint function, and $\mathbf{q} = [\mathbf{E}, \boldsymbol{\lambda}]^T$ is the concatenation of strain variables and Lagrange multipliers associated with the directional strain magnitude constraints. We solve these optimization problems using a combination of sensitivity analysis for the outer problem and sequential quadratic programming (SQP) for the inner problem. To this end, we express the total derivative of the objective function as

$$\frac{dO}{d\mathbf{T}} = \frac{\partial O}{\partial \mathbf{T}} + \frac{\partial O}{\partial \mathbf{q}} \frac{d\mathbf{q}}{d\mathbf{T}} , \tag{4.14}$$

where we have used the fact that equilibrium strain and Lagrange multipliers are completely determined by the choice of tiling parameters and we can therefore write $\mathbf{q} = \mathbf{q}(\mathbf{T})$. While the partial derivatives in (4.14) are straightforward to compute using auto-differentiation, the remaining total derivative requires more attention. The relation between equilibrium $\mathbf{q}$ and $\mathbf{T}$ is implicitly given through the inner optimization problem. To obtain an expression for this term, we start by writing out the Lagrangian of the inner optimization problem as

$$\mathcal{L}(\mathbf{q}) = \sum_{\alpha} \Phi(\mathbf{E}^*_\alpha) - \lambda_\alpha (\mathbf{d}_\alpha^T \mathbf{E}^*_\alpha \mathbf{d}_\alpha - E_\alpha) . \tag{4.15}$$

The first-order optimality conditions require that the gradient of the Lagrangian vanish at the optimum, *i.e.*,

$$\mathbf{g} := \frac{d\mathcal{L}}{d\mathbf{q}} = \mathbf{o} . \tag{4.16}$$

Differentiating both sides w.r.t. the tiling parameters $\mathbf{T}$, we obtain the closed-form expression

$$\frac{d\mathbf{g}}{d\mathbf{T}} = \frac{\partial \mathbf{g}}{\partial \mathbf{T}} + \frac{d\mathbf{g}}{d\mathbf{q}} \frac{d\mathbf{q}}{d\mathbf{T}} = 0 , \tag{4.17}$$

$$\frac{d\mathbf{q}}{d\mathbf{T}} = -\frac{d\mathbf{g}}{d\mathbf{q}}^{-1} \frac{\partial \mathbf{g}}{\partial \mathbf{T}} . \tag{4.18}$$

With the total derivative of the design objective in hand, we can leverage quasi-Newton methods for efficient, gradient-based minimization. We choose the L-BFGS-B method with a backtracking line search, which proved a robust and efficient choice for all our examples. Whenever we evaluate the objective function or its gradient, we must first solve the inner optimization problem. We use sequential quadratic programming (SQP) for this purpose, with analytical gradient and Hessian of the objective function obtained via network auto differentiation. For increased robustness, we additionally enforce the Hessian to be positive definite. Since the size of this matrix is relatively small, we perform eigen decomposition and shift eigenvalues such that the smallest one is equal to 10^{-7}.

4.4 RESULTS

We analyze our Neural Metamaterial Networks on a set of examples that highlight the advantages of this smooth neural representation for inverse material design. We first report the training statistics for multiple metamaterial families (Sec. 4.4.1). Leveraging these neural representations, we then perform inverse design for strain-stress profile, stiffness, and Poisson ratio optimization (Sec. 4.4.2). Finally, we conduct ablation studies on the benefits of the smooth neural representation and different choices of network activation functions (Sec. 4.4.3).

4.4.1 *Training*

We use Adam as our optimizer with a learning rate of 10^{-4} for all of our experiments. Training is performed using a *GeForce RTX 3080* with a batch size of 40,000, and we train for 8,000 epochs. We set aside 5% of the simulation data for testing, whereas the remainder is used for training the network. Test errors are summarized in Table 4.1, with energy and gradient error measuring the relative difference in energy densities and the second Piola-Kirchhoff stresses. With a maximum test error of 1.63% across all families, we conclude that our approach is well-suited to represent the macromechanical behavior of metamaterial spaces. Depending on the number of structure parameters, the training time can vary between 2 to 4 days.

In addition to validations on the test set, we further analyze the accuracy of the network predictions using uni-axial loading tests. We create a separate test set by randomly sampling 50 combinations of different structure

parameters and loading directions per family. We use strains in the range of 30% compression and 50% tension at 25 uniformly distributed locations, *i.e.* 1,250 testing points per family. We evaluate the relative error between the directional stress computed from optimizing over the network and native-scale simulation. We obtain average and maximum errors of 0.475% and 3.57% across all families, which we consider sufficiently accurate for inverse design. We refer to the supplementary material on the publication site of this paper for the directional strain-stress curves for all test cases.

Tiling Family	# Params	Rel. Gradient Err.	Rel. Energy Err.
IH29	1	1.38%	0.709%
IH21	2	1.61%	0.883%
IH50	2	1.14%	0.921%
IH67	2	1.63%	0.739%
IH28	2	1.05%	0.472%
IH22	3	1.57%	1.13%
IH01	4	1.20%	0.579%

TABLE 4.1: Error evaluated on the test set for different tiling families. As can be seen from the two rightmost columns, we consistently achieve low error for both energy and gradient loss.

4.4.2 *Inverse Design*

We analyze the ability of our network to perform inverse design tasks. To this end, we select three representative criteria that are intuitive as user inputs: uni-axial stress, directional stiffness, and directional Poisson ratio, as defined in Sec 4.3.5. All tasks can be formulated as bi-level constrained optimization problems, in which we leverage the analytical derivatives from auto differentiation over the network. In the first example, we optimize for the strain-stress behavior in a given direction for uni-axial loading, whereas in the second and third tasks, we optimize for the directional stiffness and Poisson ratio profile for a given strain magnitude. For all examples discussed in this section, we compute the ground-truth values for the optimized structures using native-scale simulation (shown as dotted curves). These comparisons indicate that our neural representation offers very good agreement with the simulation reference.

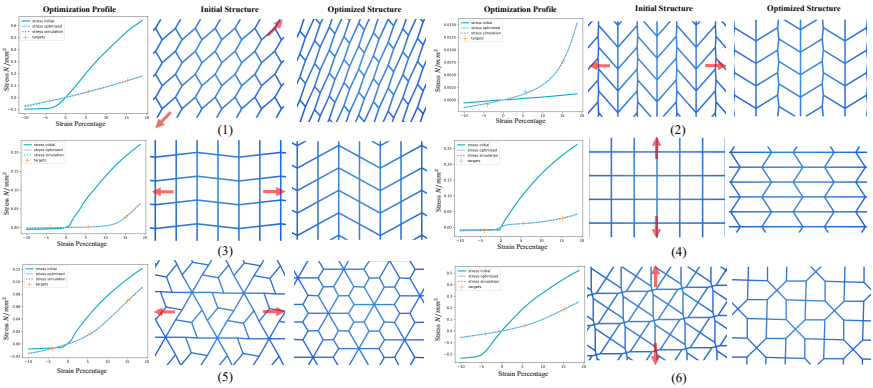


FIGURE 4.7: Optimization of Strain-Stress Curves. Here we perform inverse design on different tiling families to obtain structures whose strain-stress curve passes through prescribed targets. Orange crosses mark target points, and cyan curves indicate the initial profile. Solid sky-blue curves show the optimized structure, while the dotted blue curves plot the reference from native-scale simulation. We showcase examples of mapping between highly nonlinear and quasi-linear profiles (first row), significantly decreasing stiffness for tension (second row), and modifying the mechanical responses under both tensile and compressive loading (third row). The uni-axial loading directions are indicated with red arrows.

UNI-AXIAL STRESS. In the set of examples shown in Fig. 4.7, we apply uni-axial loading along a given direction and seek optimal tiling parameters such that the stresses for different strain magnitudes match prescribed target values. In the examples shown in the first row, we set directional stress targets such as to map between nonlinear and quasi-linear profiles. For the two examples on the second row of Fig. 4.7, we set targets that significantly decrease stress in the given directions. The optimized structures change stress by creating geometries that modify the deformation range compared to the initial structure. For the design target in the bottom row, we decrease stiffness both under compression and tension. As can be seen in all these examples, tiling geometries change significantly to minimize the design objectives. The strain used for these examples is within 10% compression and 20% tension.

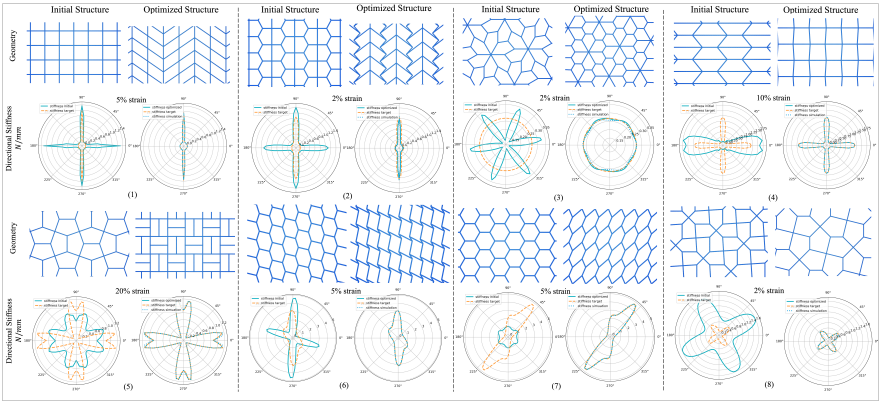


FIGURE 4.8: Directional Stiffness Optimization. We optimize for structure parameters such as to transform initial stiffness profiles (cyan) into given target profiles (orange). As can be seen that, while showing good agreement with their simulation counterparts (dotted blue curves), our approach performs robustly for various types of design tasks, *e.g.* significantly changing the stiffness for an orthotropic material (*1*, *2*), transforming an anisotropic material into nearly isotropic (*3*) and orthotropic (*4*) ones, and mapping between different anisotropic targets (*5*–*8*).

DIRECTIONAL STIFFNESS. In a second series of experiments, whose results are shown in Fig. 4.8, we investigate the ability of our method to optimize for directional stiffness profiles. For these tests, we impose tensile strains between 2% and 20% In the first and second examples, we start with orthotropic materials and ask for a significant decrease in stiffness in one of the principal directions. As can be seen from the resulting geometry, the horizontal beams fold to allow for a larger range of low-energy deformations, thus reducing stiffness in the corresponding direction. In the remaining examples, we transform an initially anisotropic material into an isotropic and orthotropic one, and map between different anisotropic targets.

DIRECTIONAL POISSON RATIO. Complementing the previous examples for stiffness design in the direction of loading, we now turn to lateral coupling in terms of Poisson's ratio.

As can be seen from Fig. 4.9, our method successfully turns a high-frequency profile into an isotropic one (first example), transforms Poisson

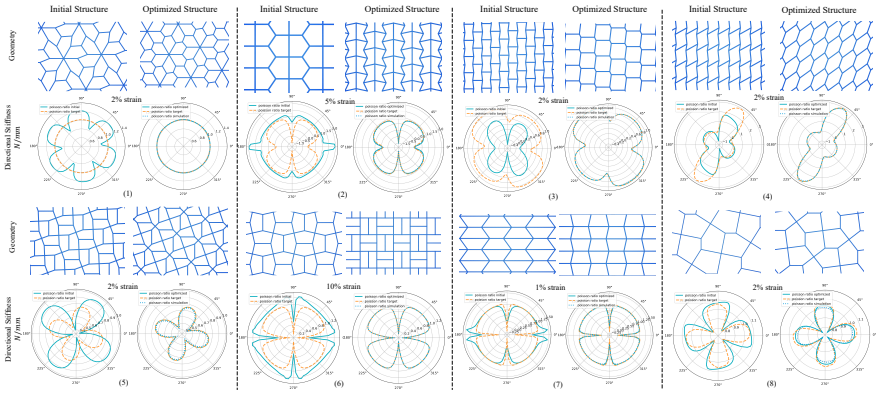


FIGURE 4.9: Poisson Ratio Optimization. We optimize for structure parameters such as to transform initial Poisson ratio profiles (cyan) into given target profiles (orange). It can be seen that optimized structures closely match the prescribed target and show good agreement with the simulation reference (dotted curves).

ratios in a given direction from positive to negative (second example), and vice versa (third example). The remaining examples further showcase our approach on a range of anisotropic targets. We impose tensile strains between 1% and 10% for these tests.

TARGET SPECIFICATION. In order to specify target profiles in an intuitive and convenient way, we built a simple interface that allows users to change the control points of an underlying spline representation. Fig. 4.10 demonstrates the process of creating the Poisson ratio target for the last example (*8*) shown in Fig. 4.9.

TIMINGS. Our network architecture allows for real-time inference. Specifically, the average time for evaluating the energy and its first and second derivatives are 3.26×10^{-4}s, 2.14×10^{-4}s and 2.77×10^{-4}s, respectively. The timings for the inverse design examples can vary depending on the objective and initial parameters provided by the user. For brevity, we only report timings for the first examples shown in Fig. 4.7, 4.8, and 4.9, which are 3.93s, 49.3s and 36.1s, respectively. These timings were obtained on a workstation with a *GeForce RTX 3080* GPU and an *AMD Ryzen 9 5950X* CPU. The simulations used to generate training data are performed in parallel on a cluster. Although the exact timing depends on the available computation

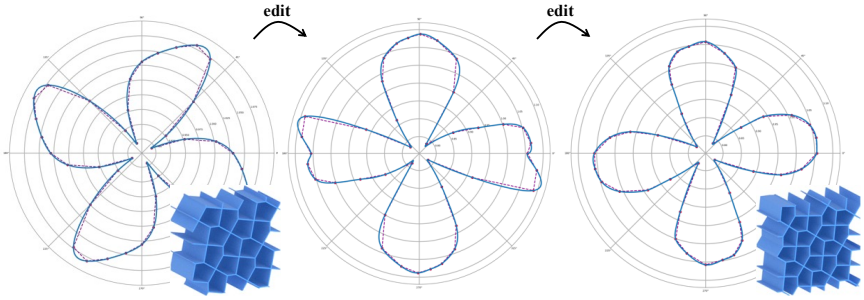


FIGURE 4.10: Interactive Target Profile Modifier. The user can edit target profiles by modifying the control points of an underlying spline representation. The initial and optimized structures are shown in the bottom right corners.

resources, the time for generating the training data used in this paper is roughly the same time required for training the corresponding networks.

IMPLEMENTATION DETAILS. Our customized simulator is implemented in C++ using Eigen [74] for linear algebra operations, Intel TBB for parallelization, and CHOLMOD [75] for solving linear systems. To create microstructure geometry from parameters, we use the Clipper2 library[2] to assign thickness to the wired edges, and Gmsh [144] for boolean operations as well as generating periodic meshes. These two procedures together take on average 1.2s.

4.4.3 *Ablation Study*

COMPARSION TO NATIVE-SCALE SIMULATION. All our inverse design examples involve an inner loop that ensures uni-axial loading. Unlike simple network evaluation, this process requires solving an optimization problem as in Eqs. (4.9c) and (4.10c). Nonetheless, our neural representation offers significant computational advantages compared to native-scale simulation. Since our approach targets large deformations, it is likely that the native-scale simulation will encounter numerically challenging cases, *e.g.* when beams are under compression or contact. These cases, on the other hand, do not pose any additional challenges for the learned model. We compare the

2 https://github.com/AngusJohnson/Clipper2

computation time for finding the equilibrium states under uni-axial loading using network prediction and native-scale simulation. As can be seen in Fig. 4.11, the time required for native-scale simulations varies substantially with imposed strain magnitude. The reason for these fluctuations is that the Newton solver can require many iterations for challenging load cases, and each iteration can require several regularization and line search steps. In contrast, our neural representation exhibits nearly constant computation time, regardless of the loading conditions, thus achieving performance gains of up to 2000%. For this example, the underlying mesh has 109,362 degrees of freedom.

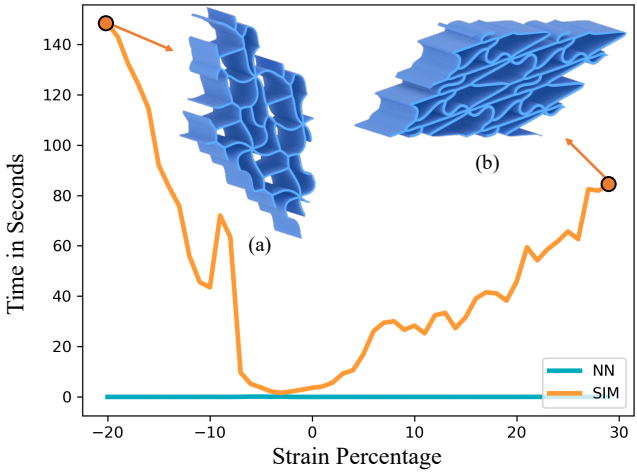


FIGURE 4.11: Timing comparison between network prediction and native-scale simulation. We compare the time required for finding equilibrium states under uni-axial loading using our neural representation (cyan curve) and native-scale simulation (orange curve). Optimizing over our neural representation yields consistently lower computation time (0.077s on average) for finding equilibrium states. This comes with no surprise as the native-scale simulation has to cope with compression (a) and contacts (b) under large deformation, The performance of native-scale simulation varies strongly with loading conditions and can take up to 148s.

SMOOTH NEURAL REPRESENTATION. Our method removes the need for native-scale simulation and re-meshing at design time. As one particular ad-

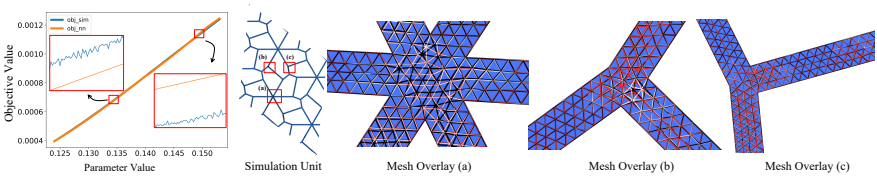


FIGURE 4.12: Non-smoothness due to meshing. *From left to right*: objective values obtained by sampling along a given direction in parameter space through both native scale simulations and optimization over the network (*1*). As can be seen from the insets in panel *1*, the objective values from the simulations show clear zig-zag patterns due to the meshing process, whereas our neural metamaterial networks lead to perfectly smooth behavior. To visualize the meshing inconsistency, in panels (*3*—*5*) we overlay the simulation meshes for three consecutive steps for three critical regions of the structure (*2*) with different colors. While parameter values only vary by 10^{-6}, the discretization changes significantly both in nodal positions and topology.

vantage, our neural representation provides smoother behavior of macromechanical properties across parameter space compared to simulation-based methods with meshing in the loop. To examine this smooth behavior, we sample a given objective function along a given direction in parameter space and compare results obtained through native-scale simulations and our neural representation, respectively. This inverse design example is taken from Fig. 4.7-(*5*). As can be seen from Fig. 4.12, when evaluating the design objective based on native-scale simulations, the plot exhibits high-frequency oscillations around a quasi-linear trend. To identify the source of this non-smoothness, we overlay the simulation rest state meshes for three consecutive steps in parameter space using a step size of 10^{-6}. The close-up views shown in Fig. 4.12 (3–5) reveal that the change in discretization is rather significant compared to the small magnitude of parameter perturbations. Although our neural representation is trained on mesh-based simulations, it does not require *meshing-in-the-loop* and thus produces smooth behavior across the entire parameter space.

While adaptive re-meshing and smoothing could alleviate this issue to some extent, they cannot remove non-smoothness for cases where the mesh topology must change to accommodate changes in geometry. To isolate this problem, we consider a metamaterial family whose parameter space contains a topological singularity. As illustrated in Fig. 4.13(a—d), the single parameter of this metamaterial controls the size of an inner

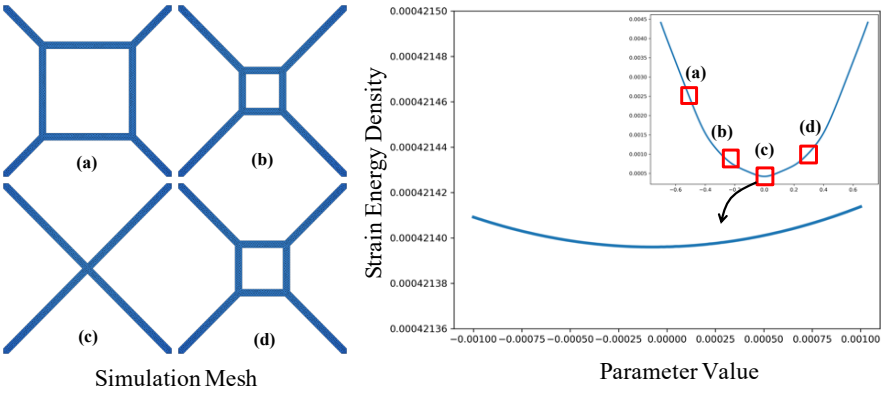


FIGURE 4.13: Topological Singularity. We consider a metamaterial family whose single parameter $p \in [-1, 1]$ controls the size of an inner square as shown in (a—d). After training, we probe the macromechanical behavior of this material under uniaxial loading by plotting the energy density from our neural network. We sample across the entire parameter space using a step size of 10^{-5}. While moving through parameter space, the geometry of the structure passes through a singular point (c) where mesh topology must change and native-scale simulation gradients are undefined. As can be seen from the plots on the right our neural representation is smooth even when stepping through the singularity located at $p = 0.0$.

square, which first contracts to a single point and then expands again. Shortly before and after the point of full contraction (c), there are two points that correspond to a change in genus. It is evident that there cannot be a mesh of constant connectivity that would be valid for the entire parameter space. As can be seen from the right plot, our neural representation, on the other hand, shows perfectly smooth behavior when passing through this singularity. Although these discontinuities do not necessarily imply failure for native-scale methods, our neural representation provides a simple and effective way to obtain smooth analytical derivatives that avoid meshing discontinuities from the outset.

NETWORK ACTIVATION FUNCTIONS. To study the influence of activation functions, we compare different choices on the simple task of fitting an isotropic Neo-Hookean model [141], whose analytical derivatives are readily available. Specifically, we compare *Swish* (our choice), *tanh*, and *sine*

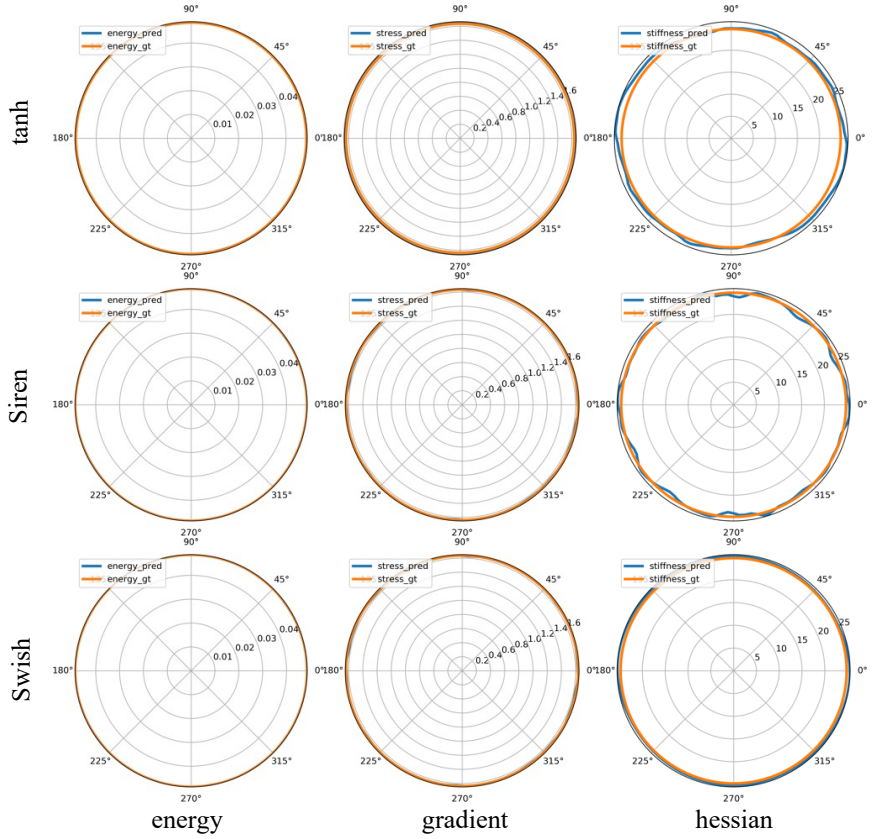


FIGURE 4.14: Impact of Activation Functions. We compare three activation functions on the task of fitting an isotropic Neo-Hookean material. While all choices approximate energy and first derivatives accurately, only the Swish activation function produces smooth second derivatives.

activation functions, all of which provide sufficient smoothness for our application. While *tanh* is a common choice, sinusoidal activation functions [145] recently showed promising results in physics-informed learning [146]. In Fig. 4.14, we show network predictions and ground truth values for the directional energy, gradient, and Hessian for the different activation functions. Since the underlying constitutive model is isotropic, the material response should be identical for all directions, *i.e.*, all plots should be perfect circles. While all three choices lead to a very good approximation of energy and

gradient values, only the Swish activation function produces smooth second derivatives.

4.5 CONCLUSION

We have presented Neural Metatmaterial Networks—a new approach for representing the nonlinear macromechanics of metamaterial families. Different from previous methods, we use deep neural networks to represent the macromechanical behavior through a continuous space of structure parameters. Using isohedral tilings as an example, we demonstrated the potential of our network to learn constitutive relationships for a wide range of materials. In particular, the smooth analytically-differentiable nature of our network enables gradient-based optimization in parameter space, which we demonstrated on various inverse material design tasks. We further showed that, by directly learning from macromechanical quantities, our network avoids gradient discontinuities due to meshing that arise when working with native-scale simulations.

4.5.1 *Limitations and Future Work*

Our current implementation relies on a comparatively simple sampling strategy. While regular sampling in strain space allows for efficient simulation warm starts, adaptive strategies might be able to improve sample efficiency. As the number of structure parameters grows, the number of samples required to probe regions of interest quickly explodes. A strategy to mitigate this curse of dimensionality could be to explore reparameterization and dimension-reduction in parameter space such as to focus sampling density where variation in structure is the largest.

We have used data obtained from native-scale simulations to train our networks. While we observed very good agreement between network predictions and simulation baselines, an interesting avenue for future research is to combine synthetic data with real-world measurements. Although energy density cannot be captured from experiments, training only on gradient data is a possibility, in principle. Another direction to explore is to conduct calibration of the native-scale model to experimental data. While we expect our Neo-Hookean material model to provide accurate predictions for actual rubber materials, 3D-printed materials such as thermoplastic polyurethane might require more advanced constitutive models. Extending our method to support continuously graded metamaterials is another exciting direction for

future work. The parametric space of Voronoi diagrams with star-shaped metrics would be a promising place to start [8]. Finally, although we focused on structures extruded from planar patterns in this work, extending our core formulation to 3D would only require a change in the number of strain inputs (from 3 to 6). We plan to apply our methodology to general 3D metamaterial families, *e.g.* those from Panetta *et al.* [90], in the future.

5

CONCLUSION & FUTURE WORK

In this thesis, we have proposed three novel computational models to address challenges occurring in forward and inverse elasticity problems for complex systems. In the first project, we introduced a novel 3D-printed, fabric-like metamaterial that utilizes small-scale tunnels to permit sliding motions. We endow Eulerian-on-Lagrangian simulation with a C^2 continuous rest shape representation that allows for smooth sliding motions for curved rods. The resulting simulation framework supports exploration of various nonlinear and anisotropic designs without resorting to physical trial-and-error experiments. In the second project, we shift our focus to discrete geodesic distance on triangle meshes. As the endpoints of a geodesic curve move, its path crosses vertices, edges, and faces of the embedded mesh. However, our analysis demonstrates that, with a suitable mollifier, the geodesic distance is almost always a C^2 continuous function—with exceptions that do not occur during distance minimization. We further provide a convenient method for computing the second derivative of the geodesic distance with respect to its endpoints. This framework enabled a wide range of applications, such as simulating energy-driven geodesic Voronoi diagrams on surfaces. Finally, we dive deeper into inverse design tasks and seek to optimize metamaterial parameters for high-level design goals. Changes in metamaterial geometry parameters lead to discrete changes in the simulation mesh, posing potential discontinuities for gradient-based inverse design. To mitigate this issue and to obtain a smooth representation of families of metamaterials, we turn to deep learning and utilize neural networks to encode their constitutive models as functions of their geometry parameters. We demonstrate that this neural framework is ideally suited for gradient-based inverse design.

FUTURE WORK Although this thesis explored the potential of leveraging neural representations as a smooth alternative to discrete simulation counterparts, machine learning was not its primary focus. Nonetheless, employing neural networks for efficient forward and inverse elasticity problems represents an exciting direction for future research.

One possibility is to eliminate discretization entirely by using neural networks to represent solutions for both forward and inverse problems. In

related work developed during this PhD [146]—though not included in this thesis—we introduced a mesh-free topology optimization framework in which one network represents equilibrium states for linear elasticity problems and another parameterizes feasible material distributions. Extending this approach to solve nonlinear partial differential equations and a broader range of inverse design tasks is a promising avenue for further study.

Similarly, implicit neural representations have been leveraged for discretization agnostic simulation of deformable objects [147], for example, through Gaussian splats and CT scans. This opens the door to enforcing physics-based supervision in other fields, such as computer vision, robotics, and beyond. An exciting opportunity is to adopt this paradigm for unstructured real-world data, where simulation, classification, and parameter identification—among other tasks—must be solved jointly. Developing simulation models that are both efficient and robust in these settings remains a significant challenge.

Throughout this thesis, we consistently employed second-order optimization methods to efficiently solve the forward problems. Although higher-order methods demonstrated significantly better convergence than their first-order counterparts, computing second-order derivatives and performing Newton-type optimization will most likely prohibit real-time applications. A promising alternative is to leverage neural networks—for example, by directly learning a dynamic or static integrator from data generated by a higher-order solver, either in a supervised or unsupervised manner [148].

Alternatively, integrating neural networks within an optimization-based framework is a viable strategy [149]. Recent work in subspace simulation has shown that latent-space, optimization-based time integration can effectively incorporate higher-order derivatives [150, 151]. Maintaining real-time performance while utilizing higher-order derivatives from network auto-differentiation, however, remains a non-trivial task.

A

APPENDIX

A.1 FIRST AND SECOND DERIVATIVE OF GEODESIC DISTANCE FOR TWO-WAY COUPLING

The geodesic distance in this case is given by (3.21), which we also include here for completeness

$$g(\mathbf{c}(\mathbf{w},\mathbf{v}),\mathbf{x}(\mathbf{t}(\mathbf{c}(\mathbf{w},\mathbf{v}),\mathbf{v})),\mathbf{v}). \tag{A.1}$$

The gradient of g w.r.t. $\mathbf{w}$ is

$$\frac{dg}{d\mathbf{w}} = \frac{\partial g}{\partial \mathbf{c}}^{\mathsf{T}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}}, \tag{A.2}$$

and gradient of g w.r.t. $\mathbf{v}$ is

$$\frac{dg}{d\mathbf{v}} = \frac{\partial g}{\partial \mathbf{c}}^{\mathsf{T}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}}\right) + \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{x}}{\partial \mathbf{v}}. \tag{A.3}$$

Following the same procedure as detailed in Sec. 3.4.1, we again use the first-order optimality condition,

$$\frac{dg}{d\mathbf{t}} := \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} = \mathbf{o} \tag{A.4}$$

to simplify the gradient expression to

$$\frac{dg}{d\mathbf{w}} = \frac{\partial g}{\partial \mathbf{c}}^{\mathsf{T}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}}, \quad \frac{dg}{d\mathbf{v}} = \frac{\partial g}{\partial \mathbf{c}}^{\mathsf{T}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{x}}{\partial \mathbf{v}}. \tag{A.5}$$

The hessian is more complex due to the coupling. The complete expression include these individual blocks,

$$\begin{aligned}
\frac{\partial^2 g}{\partial \mathbf{w}^2} &= \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \sum_i \frac{\partial g}{\partial c_i} \frac{\partial^2 c_i}{\partial \mathbf{w}^2} \\
&+ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^{\mathsf{T}} \left(\frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right) \\
&+ \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^{\mathsf{T}} \sum_i \frac{\partial g}{\partial x_i} \frac{\partial^2 x_i}{\partial \mathbf{t}^2} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right) \\
&+ \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \left(\sum_i \frac{\partial g}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t_i} \frac{\partial^2 t_i}{\partial \mathbf{c}^2} \right) \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \sum_i \frac{\partial g}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial c_i} \frac{\partial^2 c_i}{\partial \mathbf{w}^2}.
\end{aligned} \tag{A.6}$$

$$\begin{aligned}
\frac{\partial^2 g}{\partial \mathbf{v}^2} &= \frac{\partial \mathbf{c}}{\partial \mathbf{v}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{c}}{\partial \mathbf{v}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right) \\
&+ \sum_i \frac{\partial g}{\partial c_i} \frac{\partial^2 c_i}{\partial \mathbf{v}^2} + \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right)^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c} \partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} \\
&+ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right)^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{x}^2} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right) \\
&+ \sum_i (\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}})^{\mathsf{T}} \frac{\partial g}{\partial x_i} \left(\frac{\partial^2 x_i}{\partial \mathbf{t}^2} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial^2 x_i}{\partial \mathbf{t} \partial \mathbf{v}} \right) \\
&+ \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \sum_i \frac{\partial \mathbf{x}}{\partial t_i} \left(\frac{\partial \mathbf{c}}{\partial \mathbf{v}}^{\mathsf{T}} \left(\frac{\partial^2 t_i}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial^2 t_i}{\partial \mathbf{c} \partial \mathbf{v}} \right) + \sum_j \frac{\partial t_i}{\partial c_j} \frac{\partial^2 c_j}{\partial \mathbf{v}^2} + \frac{\partial^2 t_i}{\partial \mathbf{c} \partial \mathbf{v}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} \right) \\
&+ \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \sum_i \frac{\partial \mathbf{x}}{\partial t_i} \left(\frac{\partial^2 t_i}{\partial \mathbf{v}^2} \right) + \sum_i \frac{\partial g}{\partial x_i} \left(\frac{\partial^2 x_i}{\partial \mathbf{t} \partial \mathbf{v}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial^2 x_i}{\partial \mathbf{v}^2} \right).
\end{aligned} \tag{A.7}$$

$$
\begin{aligned}
\frac{\partial^2 g}{\partial \mathbf{v}\partial \mathbf{w}} &= \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \left(\frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right) \right) \\
&+ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^{\mathsf{T}} \left(\frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial^2 g}{\partial \mathbf{x}^2} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right) \right) \\
&+ \sum_i (\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}})^{\mathsf{T}} \frac{\partial g}{\partial x_i} \left(\frac{\partial^2 x_i}{\partial \mathbf{t}^2} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial^2 x_i}{\partial \mathbf{t}\partial \mathbf{v}} \right) \\
&+ \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \left(\frac{\partial g}{\partial \mathbf{x}} \sum_i \frac{\partial \mathbf{x}}{\partial t_i} \left(\frac{\partial^2 t_i}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial^2 t_i}{\partial \mathbf{c}\partial \mathbf{v}} \right) \right) + \sum_i \frac{\partial g}{\partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial c_i} \frac{\partial^2 c_i}{\partial \mathbf{v}\partial \mathbf{w}} \\
&+ \sum_i \frac{\partial g}{\partial c_i} \frac{\partial c_i^2}{\partial \mathbf{v}\partial \mathbf{w}}.
\end{aligned}
\tag{A.8}
$$

We begin by simplifying the hessian expression using (A.4), and after removing the second derivatives of linear functions we obtain:

$$
\begin{aligned}
\frac{\partial^2 g}{\partial \mathbf{w}^2} = \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \\
+ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^{\mathsf{T}} \left(\frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right).
\end{aligned}
\tag{A.9}
$$

$$
\begin{aligned}
\frac{\partial^2 g}{\partial \mathbf{v}^2} &= \frac{\partial \mathbf{c}}{\partial \mathbf{v}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{c}}{\partial \mathbf{v}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right) \\
&+ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right)^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}}^{\mathsf{T}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} \\
&+ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right)^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{x}^2} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right) \\
&+ \sum_i \frac{\partial g}{\partial x_i} \left(\frac{\partial^2 x_i}{\partial \mathbf{t}\partial \mathbf{v}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) \right) + \sum_i \frac{\partial g}{\partial x_i} \left(\frac{\partial^2 x_i}{\partial \mathbf{t}\partial \mathbf{v}} \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}} \right) \right)^{\mathsf{T}}.
\end{aligned}
\tag{A.10}
$$

The mixed second-order partial derivative of g w.r.t. $\mathbf{w}$ and $\mathbf{v}$ reads

$$\begin{aligned}\frac{\partial^2 g}{\partial \mathbf{v}\partial \mathbf{w}} &= \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^\mathsf{T}\left(\frac{\partial^2 g}{\partial \mathbf{c}^2}\frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}}\frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}}\right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}}\right)\right) \\ &+ \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\frac{\partial \mathbf{t}}{\partial \mathbf{c}}\frac{\partial \mathbf{c}}{\partial \mathbf{w}}\right)^\mathsf{T}\left(\frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}}\frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial^2 g}{\partial \mathbf{x}^2}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}}\frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}}\right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}}\right)\right) \\ &+ \sum_i \left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}}\frac{\partial \mathbf{c}}{\partial \mathbf{w}}\right)^\mathsf{T}\frac{\partial g}{\partial x_i}\frac{\partial^2 x_i}{\partial \mathbf{t}\partial \mathbf{v}} + \sum_i \frac{\partial g}{\partial c_i}\frac{\partial c_i^2}{\partial \mathbf{v}\partial \mathbf{w}}.\end{aligned} \tag{A.11}$$

Now that all terms can be evaluated in close form except for $\frac{\partial \mathbf{t}}{\partial \mathbf{c}}$ and $\frac{\partial \mathbf{t}}{\partial \mathbf{v}}$ which can be computed using sensitivity analysis. We begin by differentiate both sides of (3.6) w.r.t. $\mathbf{c}$,

$$\frac{\partial \mathbf{x}}{\partial \mathbf{t}}^\mathsf{T}\left(\frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} + \frac{\partial^2 g}{\partial \mathbf{x}^2}\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\frac{\partial \mathbf{t}}{\partial \mathbf{c}}\right) = \mathbf{0}. \tag{A.12}$$

To obtain $\frac{\partial \mathbf{t}}{\partial \mathbf{c}}$ we rearrange the terms and solve for

$$\mathbf{A}\frac{\partial \mathbf{t}}{\partial \mathbf{c}} = \mathbf{B}, \tag{A.13}$$

where

$$\begin{aligned}\mathbf{A} &= \frac{\partial \mathbf{x}}{\partial \mathbf{t}}^\mathsf{T}\frac{\partial^2 g}{\partial \mathbf{x}^2}\frac{\partial \mathbf{x}}{\partial \mathbf{t}}, \\ \mathbf{B} &= -\frac{\partial \mathbf{x}}{\partial \mathbf{t}}^\mathsf{T}\frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}}.\end{aligned} \tag{A.14}$$

Differentiating both sides of Eqn. A.4 w.r.t. $\mathbf{v}$, we arrive at

$$\begin{aligned}&\frac{\partial \mathbf{x}}{\partial \mathbf{t}}^\mathsf{T}\left(\frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}}\frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial^2 g}{\partial \mathbf{x}^2}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}}\frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}}\right) + \frac{\partial \mathbf{x}}{\partial \mathbf{v}}\right)\right) \\ &+ \sum_i \frac{\partial g}{\partial x_i}\frac{\partial^2 x_i}{\partial \mathbf{t}\partial \mathbf{v}}\left(\frac{\partial \mathbf{t}}{\partial \mathbf{c}}\frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{t}}{\partial \mathbf{v}}\right) = \mathbf{0}.\end{aligned} \tag{A.15}$$

To obtain $\frac{\partial \mathbf{t}}{\partial \mathbf{v}}$ we again rearrange the terms and solve for

$$\mathbf{C}\frac{\partial \mathbf{t}}{\partial \mathbf{v}} = \mathbf{D}, \tag{A.16}$$

where

$$\begin{aligned}\mathbf{C} &= \frac{\partial \mathbf{x}}{\partial \mathbf{t}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}},\\ \mathbf{D} &= -\frac{\partial \mathbf{x}}{\partial \mathbf{t}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} - \frac{\partial \mathbf{x}}{\partial \mathbf{t}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{x}^2} \left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{v}} + \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \right) - \sum_i \frac{\partial g}{\partial x_i} \frac{\partial^2 x_i}{\partial \mathbf{t}\partial \mathbf{v}}.\end{aligned} \tag{A.17}$$

Multiply (A.12) by $(\frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}})^{\mathsf{T}}$ on the left and $\frac{\partial \mathbf{c}}{\partial \mathbf{w}}$ on the right we have

$$\left(\frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right)^{\mathsf{T}} \left(\frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial^2 g}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} \right) = \mathbf{0}. \tag{A.18}$$

Therefore, as in the one-way coupling case, $\frac{\partial^2 g}{\partial \mathbf{w}^2}$ can be further simplified to

$$\frac{\partial^2 g}{\partial \mathbf{w}^2} = \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}^2} \frac{\partial \mathbf{c}}{\partial \mathbf{w}} + \frac{\partial \mathbf{c}}{\partial \mathbf{w}}^{\mathsf{T}} \frac{\partial^2 g}{\partial \mathbf{c}\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{w}}. \tag{A.19}$$

A.2 CONVERGENCE CRITERIA

Here we provide the details for computing the convergence criteria shown in Fig. 3.5. Let us use $\frac{\mathrm{d}O}{\mathrm{d}\mathbf{w}}^{i}$ to define the gradient of the energy shown in (3.16) w.r.t. the barycentric coordinate $\mathbf{w}$ of the Karcher mean point in iteration i, and the convergence criterion defined in Geometry Central is given by

$$\left| \alpha \cdot \frac{\mathrm{d}O}{\mathrm{d}\mathbf{w}} \right| < \frac{1}{3}\sqrt{\mathcal{A}^i}, \tag{A.20}$$

where $\mathcal{A}$ is the area of the triangle that the Karcher mean point lies on in iteration i, and α is the line search scaling factor to ensure energy decrease.

Our absolute convergence criterion is defined as

$$\left| \frac{\mathrm{d}O}{\mathrm{d}\mathbf{w}} \right| < 10^{-6}. \tag{A.21}$$

As a reference of scale, for a given input geometry, we perform a uniform scaling such that the diagonal of its bounding box is unit length. While we set the tolerance to be 10^{-6}, our approach often arrives at values orders-of-magnitude smaller thanks to the quadratic convergence.

A.3 GEODESIC VORONOI DIAGRAM

In this section, we demonstrate how to leverage sensitivity analysis to compute the gradient of the objective function defined on the GVDs, *i.e.*, (3.26) in Sec. 3.4.5. The gradient of this objective is given by

$$\frac{dO}{d\mathbf{s}} = \frac{\partial O}{\partial \mathbf{s}} + \frac{\partial O}{\partial \tilde{\mathbf{x}}}^{\mathsf{T}} \frac{d\tilde{\mathbf{x}}}{d\mathbf{s}}. \tag{A.22}$$

Whereas $\frac{\partial O}{\partial \mathbf{s}}$ and $\frac{\partial O}{\partial \tilde{\mathbf{x}}}$ can be computed algebraically, $\frac{d\tilde{\mathbf{x}}}{d\mathbf{s}}$ requires the following sensitivity analysis

$$\begin{aligned} \frac{d\mathbf{f}}{d\mathbf{s}} = \mathbf{0} &= \frac{\partial \mathbf{g}}{\partial \mathbf{s}} + \frac{\partial \mathbf{f}}{\partial \tilde{\mathbf{x}}}^{\mathsf{T}} \frac{d\tilde{\mathbf{x}}}{d\mathbf{s}}, \\ \frac{d\tilde{\mathbf{x}}}{d\mathbf{s}} &= -\frac{\partial \mathbf{f}}{\partial \tilde{\mathbf{x}}}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{s}}. \end{aligned} \tag{A.23}$$

PSEUDOCODE Here we provide the pseudocode for computing the exact geodesic Voronoi diagram and the general recipe we follow for minimizing objectives defined on GVDs.

ALGORITHM 1: Compute Exact Geodesic Voronoi Diagram

Data: triangle mesh $\mathcal{M}$, number of Voronoi sites n, Voronoi site location $\mathbf{s}$
Result: Voronoi boundary vertex position $\tilde{\mathbf{x}}$
$\tilde{\mathbf{x}}^* \leftarrow$ ComputeGVDApproximation($\mathcal{M}, n, \mathbf{s}$) // [73]
parallel for $i \leftarrow 1$ **to** *len* $(\tilde{\mathbf{x}}^*)$ **do**
  $\tilde{\mathbf{x}}_i \leftarrow$ ComputeExactGVD($\mathcal{M}, n, \mathbf{s}, \tilde{\mathbf{x}}^*$) // Eqn.(3.25)
end

ALGORITHM 2: Energy-minimizing Geodesic Voronoi Diagram

Data: triangle mesh $\mathcal{M}$, number of Voronoi sites n, Voronoi site location $\mathbf{s}$
Result: Voronoi site location $\mathbf{s}^*$
$\mathbf{s}^* \leftarrow \mathbf{s}$, $\epsilon \leftarrow 10^{-6}$ // initialize
while *True* **do**
  $\tilde{\mathbf{x}} \leftarrow$ ComputeExactGVD($\mathcal{M}, n, \mathbf{s}^*$) // Alg.(1)
  $\nabla O \leftarrow$ computeGrad($\mathcal{M}, n, \mathbf{s}^*, \tilde{\mathbf{x}}$) // Eqn.(A.22)
  if $|\nabla O| < \epsilon$ **then**
    return s^*;
  end
  $O_0 \leftarrow$ computeObj($\mathcal{M}, n, \mathbf{s}^*, \tilde{\mathbf{x}}$) // Eqn.(3.31)
  $\nabla^2 O \leftarrow$ computeHessApprox($\mathcal{M}, n, \mathbf{s}^*, \tilde{\mathbf{x}}$)
  $\triangle \mathbf{s} \leftarrow -(\nabla^2 O)^{-1} \nabla O$ // linear solve
  $\alpha \leftarrow 1$ // line search step size
  while *True* **do**
    $\mathbf{s}^* \leftarrow \mathbf{s}^* + \alpha \cdot \triangle \mathbf{s}$ // Eqn.(3.15)
    $\tilde{\mathbf{x}} \leftarrow$ ComputeExactGVD($\mathcal{M}, n, \mathbf{s}^*$) // Alg.(1)
    $O_1 \leftarrow$ computeObj($\mathcal{M}, n, \mathbf{s}^*, \tilde{\mathbf{x}}$) // Eqn.(3.31)
    if $O_1 < O_0$ **then**
      break;
    end
    $\alpha \leftarrow 0.5 \cdot \alpha$
  end
end

BIBLIOGRAPHY

1. Bergou, M., Wardetzky, M., Robinson, S., Audoly, B. & Grinspun, E. *Discrete Elastic Rods* in *ACM SIGGRAPH 2008 Papers* (Association for Computing Machinery, Los Angeles, California, 2008).
2. Bergou, M., Audoly, B., Vouga, E., Wardetzky, M. & Grinspun, E. Discrete viscous threads. *ACM Transactions on graphics (TOG)* **29**, 1 (2010).
3. Cirio, G., Lopez-Moreno, J., Miraut, D. & Otaduy, M. A. Yarn-level simulation of woven cloth. *ACM Transactions on Graphics (TOG)* **33**, 1 (2014).
4. Pattinson, S. W., Huber, M. E., Kim, S., Lee, J., Grunsfeld, S., Roberts, R., Dreifus, G., Meier, C., Liu, L., Hogan, N. & Hart, A. J. Additive Manufacturing of Biomechanically Tailored Meshes for Compliant Wearable and Implantable Devices. *Advanced Functional Materials* **29**, 1 (2019).
5. Schumacher, C., Marschner, S., Gross, M. & Thomaszewski, B. Mechanical characterization of structured sheet materials. *ACM Transactions on Graphics (TOG)* **37**, 1 (2018).
6. Li, Y., Montes, J., Thomaszewski, B. & Coros, S. Programmable digital weaves. *IEEE Robotics and Automation Letters* **7**, 2891 (2022).
7. Bertoldi, K., Vitelli, V., Christensen, J. & van Hecke, M. Flexible mechanical metamaterials. *Nature Reviews* **2**, 17066 (2017).
8. Martínez, J., Skouras, M., Schumacher, C., Hornus, S., Lefebvre, S. & Thomaszewski, B. Star-shaped metrics for mechanical metamaterial design. *ACM Transactions on Graphics (TOG)* **38**, 1 (2019).
9. Djourachkovitch, T., Blal, N., Hamila, N. & Gravouil, A. Multiscale Topology Optimization of 3D Structures: A Micro-Architectured Materials Database Assisted Strategy. *Comput. Struct.* **255** (2021).
10. Takahashi, H. & Kim, J. *3D Printed Fabric: Techniques for Design and 3D Weaving Programmable Textiles* in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (Association for Computing Machinery, New Orleans, LA, USA, 2019), 43.

11. Forman, J., Dogan, M. D., Forsythe, H. & Ishii, H. *DefeXtiles: 3D Printing Quasi-Woven Fabric via Under-Extrusion* in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Association for Computing Machinery, Virtual Event, USA, 2020), 1222.
12. McEvoy, M. A. & Correll, N. Materials that couple sensing, actuation, computation, and communication. *Science* **347**, 1261689 (2015).
13. Buckner, T. L., Bilodeau, R. A., Kim, S. Y. & Kramer-Bottiglio, R. Roboticizing fabric by integrating functional fibers. *Proceedings of the National Academy of Sciences* **117**, 25360 (2020).
14. Chenal, T. P., Case, J. C., Paik, J. & Kramer, R. K. *Variable stiffness fabrics with embedded shape memory materials for wearable applications* in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), 2827.
15. Hiramitsu, T., Suzumori, K., Nabae, H. & Endo, G. *Experimental evaluation of textile mechanisms made of artificial muscles* in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)* (2019), 1.
16. Shi, J., Liu, S., Zhang, L., Yang, B., Shu, L., Yang, Y., Ren, M., Wang, Y., Chen, J., Chen, W., Chai, Y. & Tao, X. Smart Textile-Integrated Microelectronic Systems for Wearable Applications. *Advanced Materials* **32**, 1901958 (2020).
17. Sanchez, V., Walsh, C. J. & Wood, R. J. Textile technology for soft robotic and autonomous garments. *Advanced Functional Materials* **31**, 2008278 (2021).
18. Bertails, F., Audoly, B., Cani, M.-P., Querleux, B., Leroy, F. & Lévêque, J.-L. Super-Helices for Predicting the Dynamics of Natural Hair. *ACM Trans. Graph.* **25**, 1180 (2006).
19. Duenser, S., Poranne, R., Thomaszewski, B. & Coros, S. RoboCut: hot-wire cutting with robot-controlled flexible rods. *ACM Transactions on Graphics (TOG)* **39**, 98 (2020).
20. Baek, C., Sageman-Furnas, A. O., Jawed, M. K. & Reis, P. M. Form finding in elastic gridshells. *Proceedings of the National Academy of Sciences* **115**, 75 (2018).
21. Zehnder, J., Coros, S. & Thomaszewski, B. Designing structurally-sound ornamental curve networks. *ACM Transactions on Graphics (TOG)* **35**, 1 (2016).

22. Panetta, J., Konaković-Luković, M., Isvoranu, F., Bouleau, E. & Pauly, M. X-Shells: A New Class of Deployable Beam Structures. *ACM Trans. Graph.* **38** (2019).

23. Sperl, G., Narain, R. & Wojtan, C. Homogenized Yarn-Level Cloth. *ACM Trans. Graph.* **39** (2020).

24. Yuksel, C. A Class of C 2 Interpolating Splines. *ACM Transactions on Graphics (TOG)* **39**, 1 (2020).

25. Bächer, M., Hepp, B., Pece, F., Kry, P. G., Bickel, B., Thomaszewski, B. & Hilliges, O. *Defsense: Computational design of customized deformable input devices* in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), 3806.

26. Li, Y., Numerow, L., Thomaszewski, B. & Coros, S. Differentiable Geodesic Distance for Intrinsic Minimization on Triangle Meshes. *ACM Transactions on Graphics (TOG)* **43**, 1 (2024).

27. Crane, K., Livesu, M., Puppo, E. & Qin, Y. A Survey of Algorithms for Geodesic Paths and Distances. *arXiv e-prints*, arXiv:2007.10430 (2020).

28. Peyré, G., Péchaud, M., Keriven, R., Cohen, L. D., *et al.* Geodesic methods in computer vision and graphics. *Foundations and Trends® in Computer Graphics and Vision* **5**, 197 (2010).

29. Bose, P., Maheshwari, A., Shu, C. & Wuhrer, S. A survey of geodesic paths on 3D surfaces. *Computational Geometry* **44**, 486 (2011).

30. Mitchell, J. S., Mount, D. M. & Papadimitriou, C. H. The discrete geodesic problem. *SIAM Journal on Computing* **16**, 647 (1987).

31. Chen, J. & Han, Y. *Shortest paths on a polyhedron* in *Proceedings of the sixth annual symposium on Computational geometry* (1990), 360.

32. Qin, Y., Han, X., Yu, H., Yu, Y. & Zhang, J. Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation. *ACM Transactions on Graphics (TOG)* **35**, 1 (2016).

33. Sharp, N. & Crane, K. You can find geodesic paths in triangle meshes by just flipping edges. *ACM Transactions on Graphics (TOG)* **39**, 1 (2020).

34. Liu, B., Chen, S., Xin, S.-Q., He, Y., Liu, Z. & Zhao, J. An optimization-driven approach for computing geodesic paths on triangle meshes. *Computer-Aided Design* **90**, 105 (2017).

35. Crane, K., Weischedel, C. & Wardetzky, M. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)* **32**, 1 (2013).

36. Belyaev, A. G. & Fayolle, P.-A. *On variational and PDE-based distance function approximations* in *Computer Graphics Forum* **34** (2015), 104.

37. Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S. J. & Hoppe, H. Fast exact and approximate geodesics on meshes. *ACM transactions on graphics (TOG)* **24**, 553 (2005).

38. Edelstein, M., Guillen, N., Solomon, J. & Ben-Chen, M. *A Convex Optimization Framework for Regularized Geodesic Distances* in *ACM SIGGRAPH 2023 Conference Proceedings* (Association for Computing Machinery, Los Angeles, CA, USA, 2023).

39. Belyaev, A. & Fayolle, P.-A. An ADMM-based scheme for distance function approximation. *Numerical Algorithms* **84**, 983 (2020).

40. Kimmel, R. & Sethian, J. A. Computing geodesic paths on manifolds. *Proceedings of the national academy of Sciences* **95**, 8431 (1998).

41. Ying, X., Wang, X. & He, Y. Saddle vertex graph (SVG) a novel solution to the discrete geodesic problem. *ACM Transactions on Graphics (TOG)* **32**, 1 (2013).

42. Trettner, P., Bommes, D. & Kobbelt, L. *Geodesic distance computation via virtual source propagation* in *Computer graphics forum* **40** (2021), 247.

43. Pang, B., Zheng, Z., Wang, G. & Wang, P.-S. Learning the geodesic embedding with graph neural networks. *ACM Transactions on Graphics (TOG)* **42**, 1 (2023).

44. Liu, Y.-J., Fan, D., Xu, C.-X. & He, Y. Constructing intrinsic Delaunay triangulations from the dual of geodesic Voronoi diagrams. *ACM Transactions on Graphics (TOG)* **36**, 1 (2017).

45. Bobenko, A. I. & Springborn, B. A. A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry* **38**, 740 (2007).

46. Xin, S.-Q., Chen, S.-M., He, Y., Wang, G.-J., Gu, X. & Qin, H. *Isotropic mesh simplification by evolving the geodesic delaunay triangulation* in *2011 Eighth International Symposium on Voronoi Diagrams in Science and Engineering* (2011), 39.

47. Fisher, M., Springborn, B., Bobenko, A. I. & Schroder, P. in *ACM SIGGRAPH 2006 Courses* 69 (2006).

48. Sharp, N., Soliman, Y. & Crane, K. Navigating Intrinsic Triangulations. *ACM Trans. Graph.* **38** (2019).

49. Sharp, N. & Crane, K. *A laplacian for nonmanifold triangle meshes* in *Computer Graphics Forum* **39** (2020), 69.

50. Gillespie, M., Sharp, N. & Crane, K. Integer coordinates for intrinsic geometry processing. *ACM Transactions on Graphics (TOG)* **40**, 1 (2021).

51. Liu, H.-T. D., Gillespie, M., Chislett, B., Sharp, N., Jacobson, A. & Crane, K. Surface Simplification using Intrinsic Error Metrics. *ACM Transactions on Graphics (TOG)* **42**, 1 (2023).

52. Pérez, J., Thomaszewski, B., Coros, S., Bickel, B., Canabal, J. A., Sumner, R. & Otaduy, M. A. Design and fabrication of flexible rod meshes. *ACM Transactions on Graphics (TOG)* **34**, 1 (2015).

53. Miguel, E., Lepoutre, M. & Bickel, B. Computational design of stable planar-rod structures. *ACM Transactions on Graphics (TOG)* **35**, 1 (2016).

54. Ren, Y., Panetta, J., Chen, T., Isvoranu, F., Poincloux, S., Brandt, C., Martin, A. & Pauly, M. 3D weaving with curved ribbons. *ACM Transactions on Graphics* **40**, 127 (2021).

55. Neveu, W., Puhachov, I., Thomaszewski, B. & Bessmeltsev, M. *Stability-aware simplification of curve networks* in *ACM SIGGRAPH 2022 Conference Proceedings* (2022), 1.

56. Liu, J., Xin, S., Gao, X., Gao, K., Xu, K., Chen, B. & Tu, C. Computational Object-Wrapping Rope Nets. *ACM Transactions on Graphics (TOG)* **41**, 1 (2021).

57. Yang, B., Corse, W., Lu, J., Wolper, J. & Jiang, C.-F. Real-time fluid simulation on the surface of a sphere. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* **2**, 1 (2019).

58. Huang, W., Iseringhausen, J., Kneiphof, T., Qu, Z., Jiang, C. & Hullin, M. B. Chemomechanical simulation of soap film flow on spherical bubbles. *ACM Transactions on Graphics (TOG)* **39**, 41 (2020).

59. Li, D., Sueda, S., Neog, D. R. & Pai, D. K. Thin skin elastodynamics. *ACM Transactions on Graphics (TOG)* **32**, 1 (2013).

60. Montes, J., Thomaszewski, B., Mudur, S. & Popa, T. Computational design of skintight clothing. *ACM Transactions on Graphics (TOG)* **39**, 105 (2020).

61. Lee, Y.-J. & Lee, S.-Y. Geometric Snakes for Triangular Meshes. *Journal of the Korea Computer Graphics Society* **7**, 9 (2001).

62. Hofer, M. & Pottmann, H. in *ACM SIGGRAPH 2004 Papers* 284 (2004).

63. Wallner, J., Pottmann, H. & Hofer, M. in *ACM SIGGRAPH 2005 Sketches* 32 (2005).

64. Fan, Y., Litven, J., Levin, D. I. & Pai, D. K. Eulerian-on-lagrangian simulation. *ACM Transactions on Graphics (TOG)* **32**, 1 (2013).

65. Sueda, S., Jones, G. L., Levin, D. I. & Pai, D. K. in *ACM SIGGRAPH 2011 papers* 1 (2011).

66. Cirio, G., Lopez-Moreno, J. & Otaduy, M. A. *Efficient simulation of knitted cloth using persistent contacts* in *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), 55.

67. Weidner, N. J., Piddington, K., Levin, D. I. & Sueda, S. Eulerian-on-lagrangian cloth simulation. *ACM Transactions on Graphics (TOG)* **37**, 1 (2018).

68. Nocedal, J. & Wright, S. J. *Numerical optimization* (Springer, 1999).

69. Polthier, K. & Schmies, M. in *ACM SIGGRAPH 2006 Courses* 30 (2006).

70. Sharp, N., Soliman, Y. & Crane, K. The vector heat method. *ACM Transactions on Graphics (TOG)* **38**, 1 (2019).

71. Grinspun, E., Hirani, A. N., Desbrun, M. & Schröder, P. *Discrete shells* in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), 62.

72. Kim, T. & Eberle, D. in *ACM SIGGRAPH 2020 Courses* 1 (2020).

73. Xin, S., Wang, P., Xu, R., Yan, D., Chen, S., Wang, W., Zhang, C. & Tu, C. SurfaceVoronoi: Efficiently Computing Voronoi Diagrams Over Mesh Surfaces with Arbitrary Distance Solvers. *ACM Transactions on Graphics (TOG)* **41**, 1 (2022).

74. Guennebaud, G., Jacob, B., *et al. Eigen v3* http://eigen.tuxfamily.org. 2010.

75. Chen, Y., Davis, T. A., Hager, W. W. & Rajamanickam, S. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)* **35**, 1 (2008).

76. Sharp, N., Crane, K., *et al.* GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing (2019).

77. Sharp, N. *et al. Polyscope* www.polyscope.run. 2019.

78. Mancinelli, C. & Puppo, E. Computing the Riemannian center of mass on meshes. *Computer Aided Geometric Design* **103**, 102203 (2023).

79. Sherman, J. & Morrison, W. J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics* **21**, 124 (1950).

80. Bertoldi, K., Vitelli, V., Christensen, J. & Van Hecke, M. Flexible mechanical metamaterials. *Nature Reviews Materials* **2**, 1 (2017).

81. Li, Y., Coros, S. & Thomaszewski, B. Neural Metamaterial Networks for Nonlinear Material Design. *ACM Transactions on Graphics (TOG)* **42**, 1 (2023).

82. Leimer, K. & Musialski, P. *Reduced-Order Simulation of Flexible Meta-Materials* in (Association for Computing Machinery, Inc, 2020).

83. Leimer, K. & Musialski, P. Analysis of a reduced-order model for the simulation of elastic geometric zigzag-spring meta-materials. *Computers & Graphics* **102**, 187 (2022).

84. Martínez, J., Song, H., Dumas, J. & Lefebvre, S. Orthotropic k-nearest foams for additive manufacturing. *ACM Transactions on Graphics (TOG)* **36**, 1 (2017).

85. Tricard, T., Tavernier, V., Zanni, C., Martínez, J., Hugron, P.-A., Neyret, F. & Lefebvre, S. Freely orientable microstructures for designing deformable 3D prints. *ACM Trans. Graph.* **39**, 211 (2020).

86. Efremov, S., Martínez, J. & Lefebvre, S. 3D periodic cellular materials with tailored symmetry and implicit grading. *Computer-Aided Design* **140**, 103086 (2021).

87. Bickel, B., Bächer, M., Otaduy, M. A., Lee, H. R., Pfister, H., Gross, M. & Matusik, W. Design and fabrication of materials with desired deformation behavior. *ACM Transactions on Graphics (TOG)* **29**, 1 (2010).

88. Martínez, J., Dumas, J. & Lefebvre, S. Procedural voronoi foams for additive manufacturing. *ACM Transactions on Graphics (TOG)* **35**, 1 (2016).

89. Zehnder, J., Knoop, E., Bächer, M. & Thomaszewski, B. Metasilicone: design and fabrication of composite silicone with desired mechanical properties. *ACM Transactions on Graphics (TOG)* **36**, 1 (2017).

90. Panetta, J., Zhou, Q., Malomo, L., Pietroni, N., Cignoni, P. & Zorin, D. Elastic textures for additive fabrication. *ACM Transactions on Graphics (TOG)* **34**, 1 (2015).

91. Panetta, J., Rahimian, A. & Zorin, D. Worst-case stress relief for microstructures. *ACM Transactions on Graphics (TOG)* **36**, 1 (2017).

92. Schumacher, C., Bickel, B., Rys, J., Marschner, S., Daraio, C. & Gross, M. Microstructures to control elasticity in 3D printing. *ACM Transactions on Graphics (Tog)* **34**, 1 (2015).

93. Tozoni, D. C., Dumas, J., Jiang, Z., Panetta, J., Panozzo, D. & Zorin, D. A low-parametric rhombic microstructure family for irregular lattices. *ACM Transactions on Graphics (TOG)* **39**, 101 (2020).

94. Chen, D., Skouras, M., Zhu, B. & Matusik, W. Computational discovery of extremal microstructure families. *Science advances* **4**, eaao7005 (2018).

95. Wang, F., Sigmund, O. & Jensen, J. S. Design of materials with prescribed nonlinear properties. *Journal of the Mechanics and Physics of Solids* **69**, 156 (2014).

96. Zhu, B., Skouras, M., Chen, D. & Matusik, W. Two-scale topology optimization with microstructures. *ACM Transactions on Graphics (TOG)* **36**, 1 (2017).

97. Andreassen, E., Lazarov, B. S. & Sigmund, O. Design of manufacturable 3D extremal elastic microstructure. *Mechanics of Materials* **69**, 1 (2014).

98. Clausen, A., Wang, F., Jensen, J. S., Sigmund, O. & Lewis, J. A. Topology optimized architectures with programmable Poisson's ratio over large deformations. *Adv. Mater* **27**, 5523 (2015).

99. Behrou, R., Abi Ghanem, M., Macnider, B. C., Verma, V., Alvey, R., Hong, J., Emery, A. F., Kim, H. A. & Boechler, N. Topology optimization of nonlinear periodically microstructured materials for tailored homogenized constitutive properties. *Composite Structures* **266**, 113729 (2021).

100. Nakshatrala, P. B., Tortorelli, D. & Nakshatrala, K. Nonlinear structural design using multiscale topology optimization. Part I: Static formulation. *Computer Methods in Applied Mechanics and Engineering* **261**, 167 (2013).

101. Allaire, G., Jouve, F. & Toader, A.-M. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics* **194**, 363 (2004).

102. Wang, Y., Luo, Z., Zhang, N. & Kang, Z. Topological shape optimization of microstructural metamaterials using a level set method. *Computational Materials Science* **87**, 178 (2014).

103. Feng, Y., Huang, T., Gong, Y. & Jia, P. Stiffness optimization design for TPMS architected cellular materials. *Materials & Design* **222**, 111078 (2022).

104. Zhang, S., Da, D. & Wang, Y. TPMS-infill MMC-based topology optimization considering overlapped component property. *International Journal of Mechanical Sciences* **235**, 107713 (2022).

105. Xu, W., Zhang, P., Yu, M., Yang, L., Wang, W. & Liu, L. Topology Optimization Via Spatially-Varying TPMS. *IEEE Transactions on Visualization and Computer Graphics* (2023).

106. Hu, Y., Liu, J., Spielberg, A., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D. & Matusik, W. *Chainqueen: A real-time differentiable physical simulator for soft robotics* in *2019 International conference on robotics and automation (ICRA)* (2019), 6265.

107. Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J. & Durand, F. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935* (2019).

108. Jatavallabhula, K. M., Macklin, M., Golemo, F., Voleti, V., Petrini, L., Weiss, M., Considine, B., Parent-Lévesque, J., Xie, K., Erleben, K., *et al.* gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646* (2021).

109. Geilinger, M., Hahn, D., Zehnder, J., Bächer, M., Thomaszewski, B. & Coros, S. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)* **39**, 1 (2020).

110. Panetta, J., Konaković-Luković, M., Isvoranu, F., Bouleau, E. & Pauly, M. X-shells: A new class of deployable beam structures. *ACM Transactions on Graphics (TOG)* **38**, 1 (2019).

111. Tozoni, D. C., Zhou, Y. & Zorin, D. Optimizing contact-based assemblies. *ACM Transactions on Graphics (TOG)* **40**, 1 (2021).

112. Liang, J., Lin, M. & Koltun, V. Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems* **32** (2019).

113. Chen, T., Panetta, J., Schnaubelt, M. & Pauly, M. Bistable auxetic surface structures. *ACM Transactions on Graphics (TOG)* **40**, 1 (2021).

114. Tymms, C., Wang, S. & Zorin, D. Appearance-preserving tactile optimization. *ACM Transactions on Graphics (TOG)* **39**, 1 (2020).

115. Wang, Y., Verheul, J., Yeo, S.-H., Kalantari, N. K. & Sueda, S. Differentiable simulation of inertial musculotendons. *ACM Transactions on Graphics (TOG)* **41**, 1 (2022).

116. Becker, M. & Teschner, M. *Robust and Efficient Estimation of Elasticity Parameters using the linear Finite Element Method.* in *SimVis* **15** (2007), 28.

117. Bickel, B., Bächer, M., Otaduy, M. A., Matusik, W., Pfister, H. & Gross, M. Capture and modeling of non-linear heterogeneous soft tissue. *ACM transactions on graphics (TOG)* **28**, 1 (2009).

118. Destrade, M., Saccomandi, G. & Sgura, I. Methodical fitting for mathematical models of rubber-like materials. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **473**, 20160811 (2017).

119. Hahn, D., Banzet, P., Bern, J. M. & Coros, S. Real2Sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)* **38**, 1 (2019).

120. Wang, H., O'Brien, J. F. & Ramamoorthi, R. Data-driven elastic models for cloth: modeling and measurement. *ACM transactions on graphics (TOG)* **30**, 1 (2011).

121. Miguel, E., Bradley, D., Thomaszewski, B., Bickel, B., Matusik, W., Otaduy, M. A. & Marschner, S. *Data-driven estimation of cloth simulation models* in *Computer Graphics Forum* **31** (2012), 519.

122. Martin, S., Thomaszewski, B., Grinspun, E. & Gross, M. Example-Based Elastic Materials. *ACM Trans. Graph.* **30** (2011).

123. Li, Y. & Barbič, J. Stable anisotropic materials. *IEEE transactions on visualization and computer graphics* **21**, 1129 (2015).

124. Xu, H., Sin, F., Zhu, Y. & Barbič, J. Nonlinear material design using principal stretches. *ACM Transactions on Graphics (TOG)* **34**, 1 (2015).

125. Miguel, E., Miraut, D. & Otaduy, M. A. Modeling and Estimation of Energy-Based Hyperelastic Objects. *Computer Graphics Forum* **35**, 385 (2016).

126. Wang, B., Deng, Y., Kry, P., Ascher, U., Huang, H. & Chen, B. Learning Elastic Constitutive Material and Damping Models. *Computer Graphics Forum* **39**, 81 (2020).

127. Masi, F., Stefanou, I., Vannucci, P. & Maffi-Berthier, V. Thermodynamics-based Artificial Neural Networks for constitutive modeling. *Journal of the Mechanics and Physics of Solids* **147**, 104277 (2021).

128. Le, B., Yvonnet, J. & He, Q.-C. Computational homogenization of nonlinear elastic materials using neural networks. *International Journal for Numerical Methods in Engineering* **104**, 1061 (2015).

129. Linka, K., Hillgärtner, M., Abdolazizi, K. P., Aydin, R. C., Itskov, M. & Cyron, C. J. Constitutive artificial neural networks: a fast and general approach to predictive data-driven constitutive modeling by deep learning. *Journal of Computational Physics* **429**, 110010 (2021).

130. Li, X., Cao, Y., Li, M., Yang, Y., Schroeder, C. & Jiang, C. *PlasticityNet: Learning to Simulate Metal, Sand, and Snow for Optimization Time Integration* in *Advances in Neural Information Processing Systems* (eds Oh, A. H., Agarwal, A., Belgrave, D. & Cho, K.) (2022).

131. Ghaboussi, J., Garrett Jr, J. & Wu, X. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics* **117**, 132 (1991).

132. Jung, S. & Ghaboussi, J. Neural network constitutive model for rate-dependent materials. *Computers & Structures* **84**, 955 (2006).

133. Bensoussan, A., Lions, J.-L. & Papanicolaou, G. *Asymptotic analysis for periodic structures* (American Mathematical Soc., 2011).

134. Guedes, J. & Kikuchi, N. Preprocessing and postprocessing for materials based on the homogenization method with adaptive finite element methods. *Computer methods in applied mechanics and engineering* **83**, 143 (1990).

135. Geers, M. G., Kouznetsova, V. G. & Brekelmans, W. Multi-scale computational homogenization: Trends and challenges. *Journal of computational and applied mathematics* **234**, 2175 (2010).

136. Kharevych, L., Mullen, P., Owhadi, H. & Desbrun, M. Numerical coarsening of inhomogeneous elastic materials. *ACM Transactions on graphics (TOG)* **28**, 1 (2009).

137. Nesme, M., Kry, P. G., Jeřábková, L. & Faure, F. in *ACM SIGGRAPH 2009 papers* 1 (2009).

138. Chen, D., Levin, D. I., Sueda, S. & Matusik, W. Data-driven finite elements for geometry and material design. *ACM Transactions on Graphics (TOG)* **34**, 1 (2015).

139. Sperl, G., Narain, R. & Wojtan, C. Homogenized yarn-level cloth. *ACM Trans. Graph.* **39**, 48 (2020).

140. Kaplan, C. S. & Salesin, D. H. *Escherization* in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), 499.

141. Bonet, J. & Wood, R. D. *Nonlinear continuum mechanics for finite element analysis* (Cambridge university press, 1997).

142. Li, M., Ferguson, Z., Schneider, T., Langlois, T. R., Zorin, D., Panozzo, D., Jiang, C. & Kaufman, D. M. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* **39**, 49 (2020).

143. Ramachandran, P., Zoph, B. & Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941* (2017).

144. Geuzaine, C. & Remacle, J.-F. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering* **79**, 1309 (2009).

145. Sitzmann, V., Martel, J., Bergman, A., Lindell, D. & Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems* **33** (2020).

146. Zehnder, J., Li, Y., Coros, S. & Thomaszewski, B. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems* **34**, 10368 (2021).

147. Modi, V., Sharp, N., Perel, O., Sueda, S. & Levin, D. I. Simplicits: Mesh-Free, Geometry-Agnostic Elastic Simulation. *ACM Transactions on Graphics (TOG)* **43**, 1 (2024).

148. Li, Y., Lin, G. W.-C., Larionov, E., Bozic, A., Roble, D., Kavan, L., Coros, S., Thomaszewski, B., Stuyck, T. & Chen, H.-y. *Self-supervised Learning of Latent Space Dynamics* Manuscript in preparation. 2025.

149. Fulton, L., Modi, V., Duvenaud, D., Levin, D. I. & Jacobson, A. *Latent-space dynamics for reduced deformable simulation* in *Computer graphics forum* **38** (2019), 379.

150. Shen, S., Yin, Y., Shao, T., Wang, H., Jiang, C., Lan, L. & Zhou, K. High-order differentiable autoencoder for nonlinear model reduction. *arXiv preprint arXiv:2102.11026* (2021).

151. Lyu, A., Zhao, S., Xian, C., Cen, Z., Cai, H. & Fang, G. Accelerate Neural Subspace-Based Reduced-Order Solver of Deformable Simulation by Lipschitz Optimization. *ACM Transactions on Graphics (TOG)* **43**, 1 (2024).