Homogenized Yarn-Level Cloth

Supplementary Document

GEORG SPERL, IST Austria
RAHUL NARAIN, Indian Institute of Technology Delhi
CHRIS WOJTAN, IST Austria

Contents

S1 HOMOGENIZATION DETAILS

S1.1 Derivation of $\boldsymbol{R}$

In this section, we derive the analytic expression for the computation of the rotation matrix $\boldsymbol{R}$ from Section 4.2 of our main paper, and we summarize how to compute the micro-midsurface $\boldsymbol{\varphi}$ from the macroscale fundamental forms $\bar{\mathbf{I}}, \bar{\mathbf{II}}$.

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

$$\mathbf{II} = \bar{\mathbf{II}} \tag{S1a}$$
$$\boldsymbol{a}_\alpha \cdot \boldsymbol{n}_{,\beta} = \bar{\boldsymbol{a}}_\alpha \cdot \bar{\boldsymbol{n}}_{,\beta} \tag{S1b}$$
$$(\boldsymbol{R}\bar{\boldsymbol{a}}_\alpha)^\top \boldsymbol{n}_{,\beta} = \bar{\boldsymbol{a}}_\alpha{}^\top \bar{\boldsymbol{n}}_{,\beta} \tag{S1c}$$
$$\boldsymbol{R}^\top \boldsymbol{n}_{,\alpha} = \bar{\boldsymbol{n}}_{,\alpha}. \tag{S1d}$$

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

$$\boldsymbol{R}_{,\alpha}\bar{\boldsymbol{n}} = \boldsymbol{n}_{,\alpha}. \tag{S2}$$

Plugging (S2) into (S1d) we get

$$\boldsymbol{R}^\top \boldsymbol{R}_{,\alpha}\bar{\boldsymbol{n}} = \bar{\boldsymbol{n}}_{,\alpha}. \tag{S3}$$

Therefore, we want to find an expression for $\boldsymbol{R}$ that satisfies (S3). To this end, we parametrize $\boldsymbol{R}$ via the exponential map, $\boldsymbol{R} = \exp \boldsymbol{W}$, where $\boldsymbol{W}$ is a skew-symmetric matrix. The derivative of the exponential map is a fairly complicated expression [Rossmann 2006],

$$\boldsymbol{R}_{,\alpha} = \boldsymbol{R} \sum_{k=0}^{\infty} \frac{(-1)^k}{(k+1)!} (\operatorname{ad} \boldsymbol{W})^k \boldsymbol{W}_{,\alpha}, \tag{S4}$$

where $(\operatorname{ad} \boldsymbol{W})\boldsymbol{X} = \boldsymbol{W}\boldsymbol{X} - \boldsymbol{X}\boldsymbol{W}$. Note that when the surface is singly curved, $\boldsymbol{W}$ and $\boldsymbol{W}_{,\alpha}$ commute, so $\boldsymbol{R}_{,\alpha} = \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ holds *exactly* even for large $\boldsymbol{W}$. For the more general case of doubly curved surfaces, one can approximate $\boldsymbol{R}_{,\alpha} \approx \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ under the common assumption that the curvature is small relative to the microscale. Consequently, we approximate (S3) by

$$\boldsymbol{W}_{,\alpha}\bar{\boldsymbol{n}} = \bar{\boldsymbol{n}}_{,\alpha}. \tag{S5}$$

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

$$\boldsymbol{W}(\xi_1, \xi_2) = \begin{pmatrix} 0 & \bullet & \sum_\alpha \bar{n}_{1,\alpha}\xi_\alpha \\ & 0 & \sum_\alpha \bar{n}_{2,\alpha}\xi_\alpha \\ (\text{skew}) & & 0 \end{pmatrix}. \tag{S6}$$

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

Finally, we can recover $\boldsymbol{R}(\xi_1, \xi_2) = \exp \boldsymbol{W}(\xi_1, \xi_2)$. This is the matrix that rotates $\bar{\boldsymbol{n}} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top$ towards $\Delta\bar{\boldsymbol{n}} := \sum_\alpha \bar{\boldsymbol{n}}_{,\alpha}\xi_\alpha$ by an angle $\|\Delta\bar{\boldsymbol{n}}\|$. Importantly, this choice exactly satisfies the original constraint (S3) for singly curved surfaces. The exponential of $\boldsymbol{W}$ can be computed in multiple ways; however, many are not numerically robust for small bending strains. Here, we provide a closed form expression for $\boldsymbol{R}$ based on the (unnormalized) sinc function, which can be implemented to be numerically robust around 0 using Taylor expansions. Defining

$$a = \sum_\alpha \bar{n}_{1,\alpha}\xi_\alpha, \tag{S7}$$
$$b = \sum_\alpha \bar{n}_{2,\alpha}\xi_\alpha, \tag{S8}$$
$$r = \sqrt{a^2 + b^2}, \tag{S9}$$

we compute the rotation as

$$\boldsymbol{R} = \begin{pmatrix} 1 - \frac{1}{2}a^2 \operatorname{sinc}(r/2)^2 & -\frac{1}{2}ab \operatorname{sinc}(r/2)^2 & a \operatorname{sinc}(r) \\ -\frac{1}{2}ab \operatorname{sinc}(r/2)^2 & 1 - \frac{1}{2}b^2 \operatorname{sinc}(r/2)^2 & b \operatorname{sinc}(r) \\ -a \operatorname{sinc}(r) & -b \operatorname{sinc}(r) & \cos(r) \end{pmatrix}. \tag{S10}$$

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

To offer some insight, the proposed strategy for computing $\boldsymbol{R}$ can be seen as integrating curvature along straight lines. For singly curved surfaces where the order of integration of ξ_1 and ξ_2 does not matter, the expression is exact, whereas for doubly curved surfaces it is an approximation.

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

$$\bar{\boldsymbol{S}} = \begin{pmatrix} \sqrt{\bar{\mathbf{I}}} \\ 0 \quad 0 \end{pmatrix}. \tag{S11}$$

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

$$\begin{pmatrix} \bar{n}_{1,1} & \bar{n}_{1,2} \\ \bar{n}_{2,1} & \bar{n}_{2,2} \end{pmatrix} = -\left(\sqrt{\bar{\mathbf{I}}}\,\right)^{-\top} \overline{\mathbf{II}}. \tag{S12}$$

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

We solve for $\boldsymbol{\varphi}$ numerically, by discretizing it on a regular grid large enough to enclose the entire yarn patch. Using standard finite differencing, we can discretize the Laplacian ∇^2 on the left-hand side as well as the right-hand side $\nabla \cdot \boldsymbol{R}\bar{\boldsymbol{S}}$ with the pure Neumann boundary conditions $\boldsymbol{N} \cdot \nabla\boldsymbol{\varphi} = \boldsymbol{N} \cdot \boldsymbol{R}\bar{\boldsymbol{S}}$. With $\bar{\boldsymbol{S}}$ and (S10), we can compute the required values at the grid nodes.

Note that, in the case of a doubly-curved least-squares surface, the rotation given by (S10) does in general not match the computed surface but is required for co-rotated boundary conditions. In this case, one can re-estimate $\boldsymbol{R}$ as the rotation from the polar decomposition of a finite-differenced gradient $\nabla\boldsymbol{\varphi}$.

S1.2 Derivation of Co-rotated Periodicity

In this section, we provide a brief derivation of our co-rotated periodicity boundary conditions (see Section 4.2 in our main paper). For comparison, in the case of solid homogenization explained in the paper, the constraint $\int_\Omega \nabla\tilde{\boldsymbol{u}}\, d\boldsymbol{X} = 0$ can be derived from the expansion $\boldsymbol{x}$ and the deformation average of $\boldsymbol{F}$. Here, we apply an analogous tactic, but instead of averaging the deformation $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}$ directly, we again leverage the rotation $\boldsymbol{R}$ from the polar decomposition $\nabla\boldsymbol{\varphi} = \boldsymbol{R}\bar{\boldsymbol{S}}$.

The proposed average is thus

$$\frac{1}{|\Omega|}\int_\Omega \boldsymbol{R}^\top \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\, d\Omega = \bar{\boldsymbol{S}}, \tag{S13}$$

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

From (S13) we now derive the periodic boundary conditions. Plugging the expansion $\boldsymbol{x} = \boldsymbol{\varphi} + h\boldsymbol{n} + \tilde{\boldsymbol{u}}$ into (S13),

$$\frac{1}{|\Omega|}\int_\Omega \boldsymbol{R}^\top \left(\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\xi}} + h\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\xi}} + \frac{\partial \tilde{\boldsymbol{u}}}{\partial \boldsymbol{\xi}}\right) d\Omega = \bar{\boldsymbol{S}}, \tag{S14}$$

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

$$\int_\Omega \left(h\boldsymbol{R}^\top \boldsymbol{n}_{,\alpha} + \boldsymbol{R}^\top \tilde{\boldsymbol{u}}_{,\alpha}\right) d\Omega = \boldsymbol{0}. \tag{S15}$$

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

$$\int_\Omega h\boldsymbol{R}^\top \boldsymbol{n}_{,\alpha}\, d\Omega = \int_\Gamma \Big(\int_H h\, dh\Big)\boldsymbol{R}^\top \boldsymbol{n}_{,\alpha}\, d\boldsymbol{\xi} = \boldsymbol{0}, \tag{S16}$$

where H denotes the thickness domain and Γ the midsurface domain. Therefore, (S15) simplifies to

$$\int_\Omega \boldsymbol{R}^\top \tilde{\boldsymbol{u}}_{,\alpha}\, d\Omega = \boldsymbol{0}. \tag{S17}$$

Splitting this integral into in-plane and out-of-plane parts, applying Leibniz's rule and using the divergence theorem, we can rewrite (S17) as

$$\iint_{H\partial\Gamma} \boldsymbol{R}^\top \tilde{\boldsymbol{u}} \otimes \boldsymbol{N}_\alpha\, d\partial\Gamma\, dh = \int_\Omega \boldsymbol{R}_{,\alpha}{}^\top \tilde{\boldsymbol{u}}\, d\Omega, \tag{S18}$$

where $\partial\Gamma$ is the in-plane boundary of the midsurface and $\boldsymbol{N}_\alpha$ are the normals to that boundary in the corresponding directions. As discussed in the previous section, $\boldsymbol{R}_{,\alpha} = \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ for singly-curved surfaces and $\boldsymbol{R}_{,\alpha} \approx \boldsymbol{R}\boldsymbol{W}_{,\alpha}$ for small curvatures, where $\boldsymbol{W}_{,\alpha}$ is constant. With our fitting strategy, we only apply cylindrical deformation, and as such it holds exactly. Therefore, the right hand side of (S18) vanishes:

$$\begin{aligned} \int_\Omega \boldsymbol{R}_\alpha^\top \tilde{\boldsymbol{u}}\, d\Omega &= \int_\Omega \boldsymbol{R}_\alpha^\top \boldsymbol{R}\boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega && \text{(S19a)} \\ &\simeq \int_\Omega \boldsymbol{W}_{,\alpha}^\top \boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega && \text{(S19b)} \\ &\simeq \boldsymbol{W}_{,\alpha}^\top \int_\Omega \boldsymbol{R}^\top \tilde{\boldsymbol{u}}\, d\Omega && \text{(S19c)} \\ &\simeq \boldsymbol{0}, && \text{(S19d)} \end{aligned}$$

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

Therefore and analogously to solid homogenization, (S18) can be satisfied by splitting Γ into opposite parts $\partial\Gamma^+$ and $\partial\Gamma^-$ and prescribing periodic boundary conditions

$$(\boldsymbol{R}^\top \tilde{\boldsymbol{u}})^+ = (\boldsymbol{R}^\top \tilde{\boldsymbol{u}})^-. \tag{S20}$$

Alternatively, one could satisfy (S18) more simply by setting $\tilde{\boldsymbol{u}} = \boldsymbol{0}$ at the boundary, effectively gluing micro-structures to the (deformed) boundary. However, this would be unnecessarily stiff as it does not allow, for example, yarns to resolve collision at the boundary during compression.

S2 YARN-LEVEL OPTIMIZATION DETAILS

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

S2.1 Periodicity

Fractional Counting. For the simulation of periodic yarn patterns, we need periodic stretching, bending, twisting, and collision energies. As discussed in our main paper, we model periodic energies

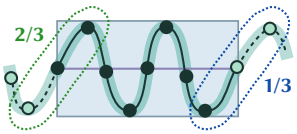

Fig. S1. We use fractional weights to ensure energies are integrated only over the periodic tile. Periodic vertices are plotted as empty circles. In this example, the contributions of periodic bending elements on the left and right are multiplied by their respective fraction of original (non-ghost) vertices.

by extending the pattern with ghost segments. These ghost segments serve only to compute energies over the periodic boundaries and should not introduce any additional energy into the system. To avoid double counting of energies and forces, we introduce *fractional counting* as illustrated in Figure S1.

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

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

Edge Orientation. In our main paper, the periodic twist constraint has the form

$$(\boldsymbol{R}^\top \underline{\boldsymbol{d}}_\alpha)^+ = (\boldsymbol{R}^\top \underline{\boldsymbol{d}}_\alpha)^-, \qquad \theta^+ = \theta^-. \tag{S21}$$

In our implementation, vertex order defines the direction of the reference frame directors $\underline{\boldsymbol{d}}_\alpha$. For (S21) to make sense, ghost segments have to have the same orientation and thus vertex order.

To extend a pattern with ghost yarns, we take input pattern geometry of connected vertices, copy and translate parts as new ghost segments, and then stitch the original and copied parts together. During this stitching process, we enforce the condition on edge orientation by reversing the vertex order of yarns as necessary. During optimization, updates to the vertices are subject to co-rotated periodicity and as such should not invalidate periodicity. For each edge, we use the value of $\boldsymbol{R}$ computed at the first vertex, so this is true only approximately and subject to discretization. In our experiments, we haven't noticed any significant drift, so we do not reproject the directors.

S2.2 Optimization

Here, we discuss the Newton step from Section 5.4 of our main paper in more detail.

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

$$\left(\boldsymbol{M}\begin{pmatrix}\tilde{\boldsymbol{C}}^\top \boldsymbol{H}\tilde{\boldsymbol{C}} & \tilde{\boldsymbol{C}}^\top \boldsymbol{C}_L{}^\top \\ \boldsymbol{C}_L\tilde{\boldsymbol{C}} & \boldsymbol{0}\end{pmatrix}\boldsymbol{M} + \begin{pmatrix}\alpha\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\right)\boldsymbol{w} = -\boldsymbol{M}\begin{pmatrix}\tilde{\boldsymbol{C}}^\top \nabla E \\ \boldsymbol{C}_L\boldsymbol{q} - \boldsymbol{d}_L\end{pmatrix} \tag{S22a}$$

$$\begin{pmatrix}\delta\boldsymbol{y} \\ \boldsymbol{\lambda}\end{pmatrix} = \boldsymbol{M}\boldsymbol{w} \tag{S22b}$$

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

Regularization. Similarly, the different stiffnesses of stretching and collision compared to bending and twisting energies can negatively affect convergence. As shown in our main paper and in (S22), we add a regularizer α to the system matrix to improve convergence. At the beginning of our simulations, the barrier-like collision forces typically dominate. In these cases, α improves convergence by shifting focus to larger gradients, which helps resolving and balancing out collisions first compared to elastic rod forces. In our experiments, we use $\alpha = \hat{\alpha}\,|\boldsymbol{M}\nabla E|_\infty$ and exponentially decay $\hat{\alpha}$ from $\hat{\alpha}_0 = 5000$ to $\hat{\alpha}_N = 5$ over $N = 400$ iterations, i.e. $\hat{\alpha}_i = \hat{\alpha}_0(\hat{\alpha}_N/\hat{\alpha}_0)^{\min(i,N)/N}$.

Step Limit. Increments $\delta\boldsymbol{y}$ computed from the linear system may be arbitrarily large. To ensure that collisions are not ignored, we rescale $\delta\boldsymbol{y}$ such that the maximum displacement of any vertex is smaller than a fraction p of its radius. The value of p generally depends on the macroscopic deformation; in compressive regimes where yarns initially overlap strongly, a lower p may be required. We set $p = 0.05 + 0.15\ \min(\max(0, \lambda_{\min}), 1)$, where $\lambda_{\min}$ is the smallest eigenvalue of $\bar{\mathbf{I}}$. Afterwards, in each optimization step we perform a simple decreasing linesearch on p, iteratively multiplying it by 0.1 until the objective is improved.

Initial Guess. As an initial guess to Newton's method, we set $\tilde{\boldsymbol{u}} = \boldsymbol{0}$ everywhere. This choice is natural in that it deforms the pattern rigidly according to the macroscopic strains. Additionally, the vertex periodicity and translation constraints are naturally satisfied.

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

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

$$\min_{\boldsymbol{z}} \frac{1}{2} \left| \boldsymbol{z} - \hat{\boldsymbol{z}} \right| \quad \text{s.t.} \quad \boldsymbol{C}_L \tilde{\boldsymbol{C}} \boldsymbol{z} = \boldsymbol{0}, \tag{S23}$$

the solution of which is

$$\boldsymbol{z} = \hat{\boldsymbol{z}} - \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{A}^\top)^{-1} \boldsymbol{A} \hat{\boldsymbol{z}}, \tag{S24}$$

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

$$|\boldsymbol{z}| < \varepsilon_z \left| \boldsymbol{z}^0 \right|, \tag{S25}$$

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

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

S2.3 Pattern Reference Configuration

Here, we discuss our heuristic to find a rest pattern that is in equilibrium with respect to in-plane stretching, which we mention in Section 5.1 of our main paper.

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

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

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

After recentering the pattern, the final reference configuration defines the initial coordinates $\begin{pmatrix} \xi_1 & \xi_2 & h \end{pmatrix}^\top$, the periodic lengths define the patch area, and the extents of the pattern along h define its thickness. We show in our results that this procedure is able to generate homogenized models that exhibit curling without in-plane deformation at rest.

S3 FITTING DETAILS

Here, we discuss the fitting procedure outlined in Section 6 of our main paper in more depth. We provide pseudocode for sampling deformations and fitting the model in a separate supplementary document. Additionally, we provide a python implementation of the fitting procedure along with the data with this paper.

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

$$\overline{\Psi} = f_0 + \sum_i f_i\Big(\frac{s_i}{v_i}\Big) + f_B + \sum_{ij} f_{ij}\Big(\frac{s_i}{v_i}, \frac{s_j}{v_j}\Big) + \sum_i f_{iB}, \tag{S26a}$$

$$\begin{aligned} f_B = {} & c^2 \left(f_x\Big(\frac{\lambda_1}{v_x}\Big) + f_y\Big(\frac{\lambda_2}{v_y}\Big)\right) \\ & + (1 - c^2) \left(f_x\Big(\frac{\lambda_2}{v_x}\Big) + f_y\Big(\frac{\lambda_1}{v_y}\Big)\right), \end{aligned} \tag{S26b}$$

$$\begin{aligned} f_{iB} = {} & \sum_i c^2 \left(f_{ix}\Big(\frac{s_i}{v_i}, \frac{\lambda_1}{v_x}\Big) + f_{iy}\Big(\frac{s_i}{v_i}, \frac{\lambda_2}{v_y}\Big)\right) \\ & + (1 - c^2) \left(f_{ix}\Big(\frac{s_i}{v_i}, \frac{\lambda_2}{v_x}\Big) + f_{iy}\Big(\frac{s_i}{v_i}, \frac{\lambda_1}{v_y}\Big)\right), \end{aligned} \tag{S26c}$$

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

For each term, we can sample data for its respective range of deformations and then fit cubic Hermite splines. To avoid overshoot in cubic interpolation, we use piecewise monotonic interpolation (see [Fritsch and Carlson 1980] and Section S5).

Our energy data is prone to noise, especially in compressive regimes (see Figure S4). What is more, piecewise monotonic functions can still have multiple local minima. To address these issues, we regularize our fits by smoothing out the data using Moving Least-Squares (MLS) and applying heuristic marching algorithms to achieve quasiconvexity in individual terms. Note that we want to regularize the total function $\overline{\Psi}$ in this way, but we fit individual terms f_i and f_{ij}. Therefore, our strategy is to fit each term to match the data and convert it to a residual afterwards.

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

S3.1 Sampling

To fit the univariate terms f_i, f_x, f_y we use 150 samples each. For the bivariate terms f_{ij}, f_{ix}, f_{iy} we sample a 50×50 grid. We sample symmetric ranges for $s_a, \lambda_x, \lambda_y$ more densely around the origin, and ranges for s_x, s_y more densely towards compression. We choose the

limits of deformation ranges empirically, based on self-intersection of the pattern under bending, and stability of collisions in general. Figure S2 demonstrates this non-uniform sampling.

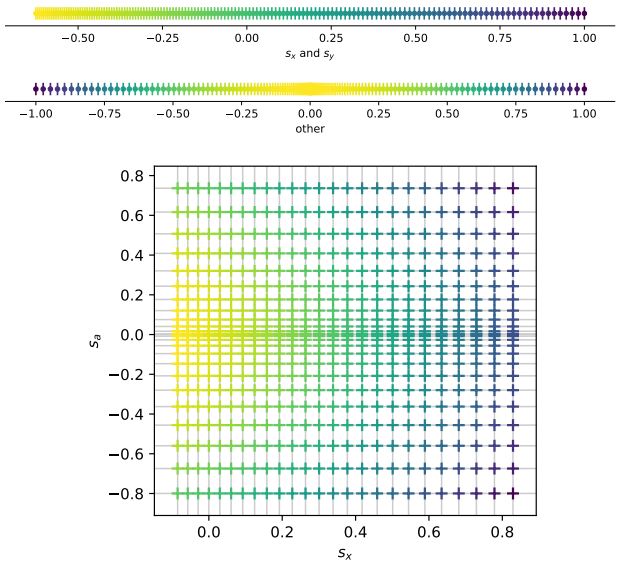

Fig. S2. Top: Our non-uniform sampling for nonsymmetric ranges (s_x and s_y) and symmetric ranges (other). Bottom: An example of a non-uniform grid created from one-dimensional sampling per dimension. We additionally indicate sample density through color.

With five 1D terms, and nine 2D terms, we therefore sample a total of 750 + 22500 = 23250 deformations. Table S1 lists the timings for the sampling stage of our method. Sampling is fast, especially when compared to full yarn-level simulations with simulation times on the order of hours and days. Furthermore, 1D terms already describe the rest shapes of draped fabric relatively well, but only require a few minutes to sample. We did not investigate if the number of samples for bivariate terms can be reduced without negatively affecting the fits to save computation time.

Table S1. Total computation time for all sampled microscale simulations for univariate and bivariate terms per pattern. The time is given in the format hrs:min:sec. We ran the simulations in parallel on 128 threads.

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

S3.2 Control Point Spacing

We want to space control points equidistantly. For our fitting scheme, we want to set e.g. $f_i(0) = 0$ and $f_{ij}(s_i, 0) = f_{ij}(0, s_j) = 0$, such that fitting another term does not influence previous fits. To this end, we require that control point spacing includes the origin for 1D terms and the axes for 2D terms. For each term and coordinate, we therefore combine two linearly spaced ranges to the left and right of 0 as illustrated in Figure S3. As discussed in our main paper, we found it necessary to modify the control point spacing for the stockinette pattern, where we concentrate them closer to the origin.

S3.3 MLS-Smoothing

We filter and resample the data for each term using MLS at the spline control points. With this, we can estimate the values and derivatives required for cubic Hermite splines in a way that is robust in the presence of noise.

Since hyperelastic energies often span multiple orders of magnitude, we found it beneficial to smooth the data in symlog-space; i.e., we convert data using $\widetilde{\log}(x) = \operatorname{sgn}(x)\log(|x|+1)$, estimate a smooth function using MLS, and then exponentiate the result back using $\widetilde{\exp}(x) = \operatorname{sgn}(x)(\exp(|x|) - 1)$. For example for a two-dimensional range parametrized by u, v, we define the smoothed function

$$g(u, v) = \widetilde{\exp}(\mathrm{MLS}(u, v \mid U_{\mathrm{data}}, V_{\mathrm{data}}, \widetilde{\log}(\Psi_{\mathrm{data}}))). \tag{S27}$$

We then estimate the first and second derivatives that make up the remaining Hermite spline coefficients by finite differencing (S27). At this point, we also enforce symmetry of our functions in s_a by symmetrizing (S27). This fits our data well and aids in preserving the symmetric rest shapes observed in yarn-level reference simulations also in our macroscale simulations. Figure S4 shows the result of applying MLS to noisy two-dimensional data.

S3.4 Marching

Piecewise monotone interpolation does not ensure quasiconvexity for any term. We therefore propose a marching heuristic to project the values and derivatives estimated with MLS to be quasiconvex. For each term, we define a minimum location $x_{\min}$, a minimum tangent magnitude $p^x_{\min}$, and then march outward while projecting

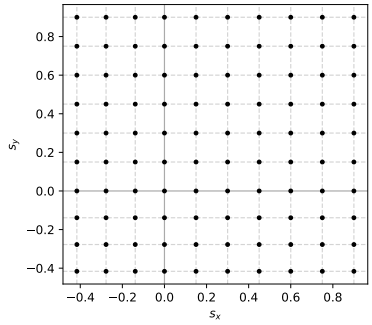

Fig. S3. Representative control point spacing for the 2D term $f_{13}(s_x, s_y)$. We use equidistant spacing to the left and right of each axis and include the axes $s_x = 0$ and $s_y = 0$.

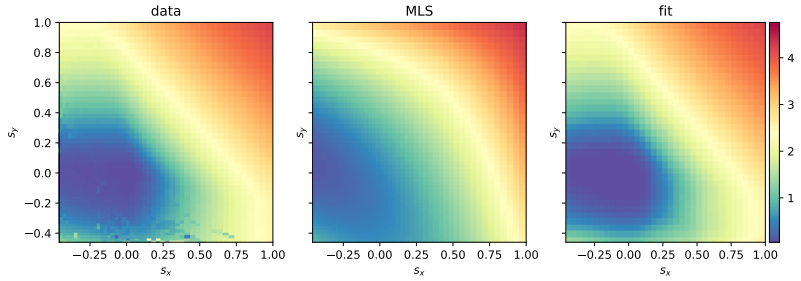

Fig. S4. Data in compressive regions can be noisy as seen on the left. In the middle, we show MLS applied to the data. During fitting we use MLS only at the spline control points to mitigate the data in the noise. Our final regularized fit on the right is similar to the data without noise.

values and derivatives. For 1D, we have

$$p_{i+1} \geq p_i + (x_{i+1} - x_i)\, p^x_{\min}\, \mathrm{sgn}(x_{i+1} - x_{\min}) \tag{S28a}$$
$$p^x_i \geq p^x_{\min}\, \mathrm{sgn}(x_i - x_{\min}). \tag{S28b}$$

We define the minimum as $s_i = 0$ for in-plane deformation, and compute the bending minimum as $\mathrm{argmin}_x\, \overline{\Psi}$ of the respective range; this ensures that the reference state of the cloth corresponds to the in-plane rest state, but allows bent rest shapes to induce curling.

For 2D, we analogously define a minimum and march outward. For the terms f_{ix}, f_{iy} we find the bending minimum subject to $s_i = 0$. However, this algorithm does not allow decreasing values away from the axes, which is crucial for modeling Poisson's ratio accurately. Therefore, we do not apply this regularization on the term $f_{13}(s_x, s_y)$, which is responsible for controlling the response to simultaneous deformation along warp and weft direction and as such for Poisson's ratio. Nevertheless, for the remainder of the terms we found that this is crucial to ensure stable simulations with smooth rest shapes. We choose $p^x_{\min} = 0.001$ for all patterns except the stockinette, where we use 0.01 in an effort to regularize the bad restshape illustrated in our main paper. It is after this projection step, that we apply the algorithms for monotone interpolation.

S3.5 Extrapolation

Simulations can in general exhibit strains outside of the sampled ranges due to discrete timesteps or constraints. Therefore, controlled extrapolation is crucial. We extrapolate splines linearly using their derivatives at the boundary. We enforce that extrapolation must increase energy such as to ensure that the simulation stays near the fitted deformation range. The marching algorithms already enforce that tangents on the boundary are not decreasing away from the minimum (for f_{13}, we additionally clamp the boundary tangents to be increasing outward). For 2D splines, we set the mixed derivatives p^{xy} to 0 at the boundary, such that linear extrapolation is consistent with interpolation.

Finally, we note that in the piecewise monotone bicubic interpolation scheme (Section S5), we treat extrapolation as a cell with an edge at infinity. This modifies (S40) and (S41) to have the extrapolated side of the inequality become 0.

S3.6 Residualization

To summarize the fitting procedure: we first smooth and resample the data using MLS, and use it to compute spline coefficients; next, we apply our quasiconvex marching algorithm to enforce a single minimum per term; then we apply the algorithms for piecewise monotone cubic and bicubic spline interpolation.

This procedure gives as regularized fits f^* of the data, which we have to convert to residuals f for the cumulative function (S26a), with $f_i(0) = 0$ and $f_{ij}(\theta_i, 0) = f_{ij}(0, \theta_j) = 0$. We do this for 1D and 2D terms respectively with

$$f_i(\theta_i) = f^*_i(\theta_i) - f^*_i(0) \tag{S29}$$
$$f_{ij}(\theta_i, \theta_j) = f^*_{ij}(\theta_i, \theta_j) - f^*_{ij}(\theta_i, 0) - f^*_{ij}(0, \theta_j) + f^*_{ij}(0, 0) \tag{S30}$$

Finally, we had to enforce that the 2D residuals are 0 for compression in all terms except f_{13}. Compressive data seems to be affected the most from noise, likely due to buckling inducing varying microscale equilibria. Note that this only means that we do not model the *combined* response of e.g. compression and bending. The separable elastic response encoded in the 1D terms still captures the material behavior well, as our results show.

S4 SINGLE CURVATURE EIGENVALUES

Here we provide expressions to robustly compute the eigenvalues and squared cosine used in the bending energy curvature splitting (see our main paper Section 6.2) based on the formulas proposed by Blinn [1996]. With

$$A = \frac{\mathrm{II}_{11} + \mathrm{II}_{22}}{2}, \tag{S31a}$$
$$B = \frac{\mathrm{II}_{11} - \mathrm{II}_{22}}{2}, \tag{S31b}$$
$$S = \sqrt{\left(\frac{\mathrm{II}_{11} - \mathrm{II}_{22}}{2}\right)^2 + {\mathrm{II}_{12}}^2 + \varepsilon}, \tag{S31c}$$
$$k = \mathrm{sgn}(\mathrm{II}_{11} - \mathrm{II}_{22}), \tag{S31d}$$

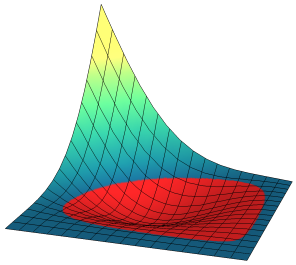

Fig. S5. A bicubic patch with cubic monotonicity applied to each direction can still produce non-monotone regions as highlighted in red here.

where ε is a small number guarding against division by zero, we have

$$\lambda_1 = A + S, \tag{S32a}$$

$$\lambda_2 = A - S, \tag{S32b}$$

$$c^2 = \frac{1}{2} + k\left(\frac{1}{2} - \frac{{\mathrm{II}_{12}}^2}{(B + k\,S)^2 + {\mathrm{II}_{12}}^2}\right). \tag{S32c}$$

Notably, our use of the sign k in (S32c) ensures that expressions of the form

$$c^2\big(f(\lambda_1) + g(\lambda_2)\big) + (1 - c^2)\big(f(\lambda_2) + g(\lambda_1)\big) \tag{S33}$$

are robust with respect to the order of eigenvalues even when they are similar, e.g.

$$\lambda_1 = \lambda \pm \varepsilon_1 \qquad\qquad \lambda_2 = \lambda \pm \varepsilon_2. \tag{S34}$$

S5 PIECEWISE MONOTONE BICUBIC INTERPOLATION

Here, we present the algorithm for piecewise monotone bicubic interpolation mentioned in Section 6.2 of our main paper. We also provide a python implementation with our supplementary data. For well-behaved simulations, we require our fitted energy functions to not exhibit any new local minima other than the ones present in the data. For bivariate functions, it is not sufficient to apply monotone piecewise cubic interpolation in both directions as shown in Figure S5. Instead, we adopt the monotone piecewise bicubic interpolation scheme of Carlson and Fritsch [1989]. Since their method as presented assumes globally monotone data, we modify it to work with arbitrary data while preserving its behavior in monotone regions.

The input to the algorithm is a grid of nodes (x_i, y_j) on which Hermite data is specified, namely values $p_{i,j}$, first derivatives $p^x_{i,j}$ and $p^y_{i,j}$, and mixed derivatives $p^{xy}_{i,j}$. For brevity, define $h_i = x_{i+1} - x_i$ and $k_j = y_{j+1} - y_j$, and define the operators Δ^x and Δ^y such that

$$\Delta^x f_{i,j} = f_{i+1,j} - f_{i,j}, \tag{S35a}$$

$$\Delta^y f_{i,j} = f_{i,j+1} - f_{i,j} \tag{S35b}$$

for any nodal data f. Our goal is to modify the specified derivatives so that, in regions where the data $p_{i,j}$ is monotone, the resulting piecewise bicubic Hermite interpolation is also monotone with the same sense.

First, we must define local monotonicity of the data. We declare a horizontal edge $E^x_{i,j} = [x_i, x_{i+1}] \times y_j$ to be *increasing* if $\Delta^x p_{i,j} \geq 0$ and *decreasing* if $\Delta^x p_{i,j} \leq 0$, and similarly define monotonicity for vertical edges $E^y_{i,j} = x_i \times [y_j, y_{j+1}]$. Now we consider a cell $R_{i,j} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$. We declare the cell to be *increasing in x* if the adjacent horizontal edges $E^x_{i,j}$ and $E^x_{i,j+1}$ are both increasing, *decreasing in x* if they are both decreasing, and *nonmonotone in x* otherwise. Similarly we define monotonicity in y using vertical edges $E^y_{i,j}$ and $E^y_{i+1,j}$.

Now, we review Carlson and Fritsch's sufficient conditions for monotonicity. Without loss of generality, we only consider monotonicity in x, and further that the sense of monotonicity is increasing; decreasingness only requires reversing all the inequalities. Increasingness along an edge $E^x_{i,j}$ is ensured if

$$0 \leq p^x_{i,j} \leq 3\frac{\Delta^x p_{i,j}}{h_i}, \tag{S36a}$$

$$0 \leq p^x_{i+1,j} \leq 3\frac{\Delta^x p_{i,j}}{h_i}. \tag{S36b}$$

Increasingness in x over a region $R_{i,j}$ is ensured by constraining differences in y-derivatives along adjacent horizontal edges,

$$\Delta^x p^y_{i,j} \geq -3\frac{\Delta^x p_{i,j}}{k_j}, \tag{S37a}$$

$$\Delta^x p^y_{i,j+1} \leq 3\frac{\Delta^x p_{i,j+1}}{k_j}, \tag{S37b}$$

and constraining mixed derivatives at adjacent nodes,

$$-3\frac{p^x_{i,j}}{k_j} \leq p^{xy}_{i,j} \leq 3\left(\frac{\Delta^x p^y_{i,j}}{h_i} + 3\frac{\Delta^x p_{i,j}}{h_i k_j} - \frac{p^x_{i,j}}{k_j}\right), \tag{S38a}$$

$$-3\frac{p^x_{i+1,j}}{k_j} \leq p^{xy}_{i+1,j} \leq 3\left(\frac{\Delta^x p^y_{i,j}}{h_i} + 3\frac{\Delta^x p_{i,j}}{h_i k_j} - \frac{p^x_{i+1,j}}{k_j}\right), \tag{S38b}$$

$$3\left(\frac{\Delta^x p^y_{i,j+1}}{h_i} - 3\frac{\Delta^x p_{i,j+1}}{h_i k_j} + \frac{p^x_{i,j+1}}{k_j}\right) \leq p^{xy}_{i,j+1} \leq 3\frac{p^x_{i,j+1}}{k_j}, \tag{S38c}$$

$$3\left(\frac{\Delta^x p^y_{i,j+1}}{h_i} - 3\frac{\Delta^x p_{i,j+1}}{h_i k_j} + \frac{p^x_{i+1,j+1}}{k_j}\right) \leq p^{xy}_{i+1,j+1} \leq 3\frac{p^x_{i+1,j+1}}{k_j}. \tag{S38d}$$

To guarantee that the above constraints always have a solution, we further require that 0 is a valid value for each mixed derivative p^{xy}. This leads to

$$\Delta^x p^y_{i,j} \geq -\frac{1}{k_j}\left(3\Delta^x p_{i,j} - h_i \max(p^x_{i,j}, p^x_{i+1,j})\right), \tag{S39a}$$

$$\Delta^x p^y_{i,j+1} \leq \frac{1}{k_j}\left(3\Delta^x p_{i,j+1} - h_i \max(p^x_{i,j+1}, p^x_{i+1,j+1})\right). \tag{S39b}$$

We now develop the algorithm for satisfying the constraints. The first step is to modify the first derivatives p^x, p^y to satisfy the inequalities (S36) arising from edges, and (S37) and (S39) arising from monotone cells. Then, we modify the mixed derivatives p^{xy} to satisfy the inequalities (S38) arising from monotone cells, which will always be possible thanks to (S39). The entire algorithm requires only implementing operations for enforcing monotonicity in x;

these can then be used for monotonicity in y as well by flipping the coordinates (i.e. transposing all grids and swapping p^x with p^y).

As in Carlson and Fritsch's original algorithm, we first satisfy (S36) by clamping p^x and p^y. As long as we only shrink them towards zero in subsequent operations, (S36) will remain satisfied.

Next, both sets of remaining constraints on first derivatives act on their "mixed differences" $\Delta^x p^y$ and $\Delta^y p_x$. Here, unlike the original algorithm, some more care is needed since adjacent regions may differ in monotonicity. Consider a horizontal edge $E^x_{i,j}$ and suppose it is increasing. Then the adjacent cells $R_{i,j-1}$ and $R_{i,j}$ cannot be decreasing in x; they are either increasing or nonmonotone in x. Then from (S37) we have

$$-3\frac{\Delta^x p_{i,j}}{k_j} \le \Delta^x p^y_{i,j} \le 3\frac{\Delta^x p_{i,j}}{k_{j-1}}, \tag{S40}$$

where the first inequality arises if $R_{i,j}$ is increasing in x, and the second if $R_{i,j-1}$ is increasing in x. Nevertheless, there is no harm in imposing both bounds even if the adjacent cells are nonmonotone, since 0 is always a feasible value due to the increasingness of $E^x_{i,j}$. Similarly, from (S39) we have

$$-\frac{b_{i,j}}{k_j} \le \Delta^x p^y_{i,j} \le \frac{b_{i,j}}{k_{j-1}} \tag{S41}$$

where $b_{i,j} = 3\Delta^x p_{i,j} - h_i \max(p^x_{i,j}, p^x_{i+1,j})$, which we again impose regardless of monotonicity of $R_{i,j-1}$ and $R_{i,j}$, because 0 is always a feasible value thanks to (S36).

We now discuss how to satisfy these constraints without violating (S36). Note that they are all of the form $l \le m_1 - m_0 \le u$ where $l \le 0 \le u$. For any constraint, given the current values (m_0, m_1), we project to the closest values (m_0^*, m_1^*) which satisfy the constraints as well as $|m_i^*| \le |m_i|$ and $\operatorname{sgn}(m_i^*) = \operatorname{sgn}(m_i)$. The solution is given in closed form by several cases, illustrated in Figure S6. Since this projection may cause the constraints on adjacent edges to be violated, we must make multiple sweeps across the grid. Following Carlson and Fritsch, we make two sweeps across the grid, first in increasing order in x and then in decreasing order, applying projections to enforce the $\Delta^x p^y$ constraints on each edge. We then do the same in y to enforce the $\Delta^y p^x$ constraints, albeit using the *old* values of p^y so that the results are order-independent.

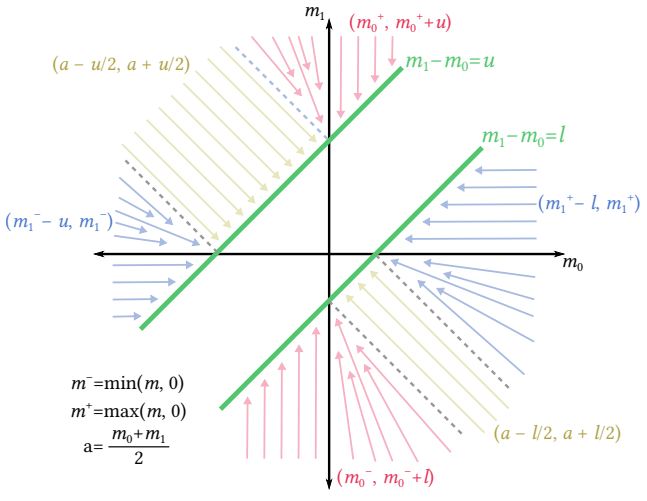

Fig. S6. We want to project (m_0^*, m_1^*) to the central diagonal slab $l \le m_1 - m_0 \le u$ (green) without increasing either entry's absolute value. There are six cases (not including if the initial value is already feasible), which we show as bundles of arrows alongside their projected values.

Finally, the mixed derivative p^{xy} at each node is subject to constraints (S38) from the adjacent monotone cells. For monotonicity in x, the constraints acting on $p^{xy}_{i,j}$ are

$$-3\frac{p^x_{i,j}}{k_j} \le p^{xy}_{i,j} \le 3\left(\frac{\Delta^x p^y_{i,j}}{h_i} + 3\frac{\Delta^x p_{i,j}}{h_i k_j} - \frac{p^x_{i,j}}{k_j}\right), \tag{S42a}$$

$$-3\frac{p^x_{i,j}}{k_j} \le p^{xy}_{i,j} \le 3\left(\frac{\Delta^x p^y_{i-1,j}}{h_{i-1}} + 3\frac{\Delta^x p_{i-1,j}}{h_{i-1} k_j} - \frac{p^x_{i,j}}{k_j}\right), \tag{S42b}$$

$$3\left(\frac{\Delta^x p^y_{i,j}}{h_i} - 3\frac{\Delta^x p_{i,j}}{h_i k_{j-1}} + \frac{p^x_{i,j}}{k_{j-1}}\right) \le p^{xy}_{i,j} \le 3\frac{p^x_{i,j}}{k_{j-1}}, \tag{S42c}$$

$$3\left(\frac{\Delta^x p^y_{i-1,j}}{h_{i-1}} - 3\frac{\Delta^x p_{i-1,j}}{h_{i-1} k_{j-1}} + \frac{p^x_{i,j}}{k_{j-1}}\right) \le p^{xy}_{i,j} \le 3\frac{p^x_{i,j}}{k_{j-1}} \tag{S42d}$$

if the adjacent cells $R_{i,j}$, $R_{i-1,j}$, $R_{i,j-1}$, and $R_{i-1,j-1}$ respectively are increasing in x, and with the corresponding inequalities reversed if they are decreasing in x instead. Observe that each constraint above only involves quantities along a single edge, namely $E^x_{i,j}$, $E^x_{i-1,j}$, $E^x_{i,j}$, and $E^x_{i-1,j}$ respectively. Furthermore, if said edge is increasing, and we have satisfied all four constraints on its mixed difference $\Delta^x p^y$, then the constraint always has 0 as a feasible value. Therefore, it is safe to enforce each of the constraints on $p^{xy}_{i,j}$, regardless of the monotonicity of the cell it arises from, remembering only to reverse the inequalities if the relevant *edge* is decreasing instead of increasing. We do so simply by clamping $p^{xy}_{i,j}$ to lie between the two bounds of each constraint. Then we repeat the procedure on the flipped data to enforce monotonicity in y.

The algorithm presented here is equivalent to the original algorithm for globally monotone data [Carlson and Fritsch 1989]. It also guarantees monotonicity on each cell, as long as there are no cells with nonmonotone data. We have not yet conducted any analysis of what guarantees, if any, are available for cells that are nonmonotone in both directions, such as $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

S6 CLOTH SIMULATION DERIVATIVE SPLIT

In this section, we briefly discuss how we compute the derivatives of the triangle energy $E_\triangle = A\,\overline{\Psi}(\boldsymbol{z}(\boldsymbol{q}_\triangle))$ from Section 7 of our main paper. We found it useful to keep the computation modular, using the chain rule to separate geometric derivatives from strain derivatives.

The energy depends on the positional degrees of freedom of the triangle's vertices and the additional vertices from its adjacent triangles, collectively denoted as $\boldsymbol{q}_\triangle$, from which we can compute the collected strains $\boldsymbol{z} = (s_x, s_a, s_y, \lambda_1, \lambda_2, c^2)$ as discussed in the

main paper. Using the chain rule, we have

$$\frac{\partial E_\triangle}{\partial \boldsymbol{q}_\triangle} = A \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}\right)^\top \frac{\partial \overline{\Psi}}{\partial \boldsymbol{z}}, \tag{S43}$$

$$\frac{\partial^2 E_\triangle}{\partial \boldsymbol{q}_\triangle \partial \boldsymbol{q}_\triangle} = A \left(\frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle \partial \boldsymbol{q}_\triangle}^\top \frac{\partial \overline{\Psi}}{\partial \boldsymbol{z}} + \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}^\top \frac{\partial^2 \overline{\Psi}}{\partial \boldsymbol{z} \partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}\right). \tag{S44}$$

This split allows us to easily swap out different energy models, while not affecting the strain part of the computation. To compute the derivatives $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle}$ and $\frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{q}_\triangle \partial \boldsymbol{q}_\triangle}$, we generate code using Mathematica [Wolfram Research, Inc. 2019]. Additionally, we concatenate both of these derivatives and compute them at the same time, which reduces redundant computation in the generated code by a significant amount.

REFERENCES

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

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

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

Jonathan M Kaldor, Doug L James, and Steve Marschner. 2008. Simulating knitted cloth at the yarn level. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 65.

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

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