SpringTime: Learning Simulatable Models of Cloth with Spatially-varying Constitutive Properties

Guanxiong Chen
gxchen@cs.ubc.ca
University of British Columbia
Vancouver, Canada

Shashwat Suri
suris@student.ubc.ca
University of British Columbia
Vancouver, Canada

Yuhao Wu
wuyuhao@cs.ubc.ca
University of British Columbia
Vancouver, Canada

Yixian Cheng
chengyx@student.ubc.ca
University of British Columbia
Vancouver, Canada

Ganidhu Abeysirigoonawardena
ganidhu.abey@gmail.com
University of British Columbia
Vancouver, Canada

Etienne Vouga
evouga@cs.utexas.edu
University of Texas at Austin
Austin, USA

David I.W. Levin
diwl.levin@utoronto.ca
University of Toronto
Toronto, Canada

Dinesh K. Pai
pai@cs.ubc.ca
University of British Columbia
Vancouver, Canada

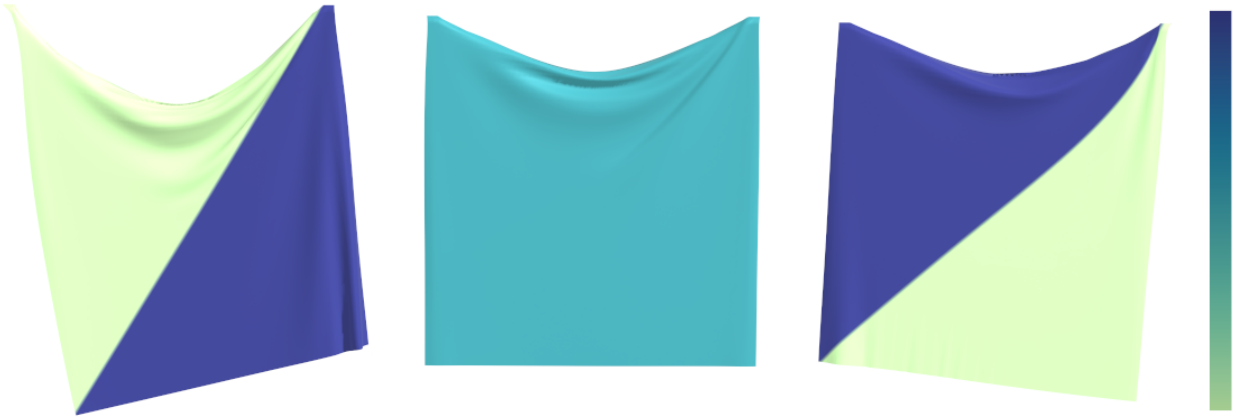

Figure 1: Equilibrium configurations of a square piece of cloth with spatially heterogeneous (left, right) vs. using homogeneous material (middle). Purple regions are stiffer than yellow ones. For the homogeneous cloth all triangles share the same stiffness, as in previous work, which is taken to be the average stiffness of triangles in the heterogeneous cloth. Despite identical initial and boundary conditions, stiffness variation leads to distinct behaviors. Our method can capture such variation.

Abstract

Materials used in real clothing exhibit remarkable complexity and spatial variation due to common processes such as stitching, hemming, dyeing, printing, padding, and bonding. Simulating these materials, for instance using finite element methods, is often computationally demanding and slow. Worse, such methods can suffer from numerical artifacts called "membrane locking" that makes cloth appear artificially stiff. Here we propose a general framework, called SpringTime, for learning a simple yet efficient surrogate model that captures the effects of these complex materials using only motion observations. The cloth is discretized into a mass-spring network with unknown material parameters that are learned directly from the motion data, using a novel force-and-impulse loss function. Our approach demonstrates the ability to accurately model spatially varying material properties from a variety of data sources, and resistance to membrane locking which plagues FEM-based simulations. Compared to graph-based networks and neural ODE-based architectures, our method achieves significantly faster training times, higher reconstruction accuracy, and improved generalization to novel dynamic scenarios. Codebase for the paper can be found at https://github.com/ericchen321/springtime.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GI '26, Kitchener-Waterloo, Ontario, Canada

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.

CCS Concepts

• **Computing methodologies** → **Modeling and simulation**; *Machine learning*; Massively parallel and high-performance simulations; Computer graphics.

Keywords

Simulation, Cloth, Deformable, Dynamics, Neural Networks, Differentiable, Surrogate, Material Properties

ACM Reference Format:

Guanxiong Chen, Shashwat Suri, Yuhao Wu, Yixian Cheng, Ganidhu Abeysirigoonawardena, Etienne Vouga, David I.W. Levin, and Dinesh K. Pai. 2026. SpringTime: Learning Simulatable Models of Cloth with Spatially-varying Constitutive Properties. In *Proceedings of Graphics Interface 2026 (GI '26)*. ACM, New York, NY, USA, 14 pages.

1 Introduction

Simulation of physical systems is a cornerstone of modern science and engineering, as well as physically based animation in computer graphics, games, and VR. The Finite Element Method (FEM) is widely used in computer graphics, to predict the behavior of everything from cars to characters. However, modeling and simulation of real-world objects remains challenging for several reasons. We will focus on the cloth simulation in this paper, but the issues are more general.

First, real objects may have complex, spatially varying, constitutive properties that significantly affect behavior. A simple example is shown in Fig. 1. A striking example is the Issey Miyake A-POC clothing design (generated from "A Piece of Cloth") demonstrated at SIGGRAPH Asia 2024[1]. Heat-shrinking different parts of cloth creates spatially varying constitutive properties that gives the cloth the desired 3D shape. Even common processes such as stitching, printing, and bonding introduce spatially varying material properties that are challenging to model using standard methods and by previous data-driven techniques.

Second, FEM cloth simulations can be complex to implement and slow. As a consequence, FEM is rarely used in real-time applications, and it is common to resort to using coarse meshes and low order (linear) elements.

Third, and most insidiously, FEM meshes typically used in computer graphics can suffer greatly from numerical artifacts such as *membrane locking*, which lead to unrealistic bending behavior. Fig. 2 (b) and (c) illustrates this phenomenon. In engineering, locking is ameliorated by using higher-order elements, reduced integration, or with special elements designed specially for shells, which add significantly to the complexity of FEM simulation.

In this work, we introduce a learning-based framework, called *SpringTime*, that addresses all these problems. By using a mass-spring network as the surrogate model and involving parallelization in the training curriculum, SpringTime can be trained efficiently and effectively to learn spatially-varying constitutive properties of cloth from point cloud data quite quickly. In the future, we hope to further improve the inference time of SpringTime to make it more suitable for real-time applications.

Even though there is a popular perception that mass-spring networks are unsophisticated and less appropriate for modeling continuum mechanics, that is not necessarily the case. As observed by Breen et al. [10], Hearle and colleagues at Manchester's renowned Department of Textiles had investigated applications of shell and plate models in the 1980s, and concluded: "But, in dealing with the three-dimensional buckling of textile fabrics, neither the terminology nor the methodology of the established mechanical theory of the bending of plates and shells is of much help because of the limiting assumptions that are made" [2]. Mass-spring networks could better capture the anisotropic behavior of cloth, and avoid numerical artifacts of FEM discretization of thin shells. We demonstrate, with a series of experiments, that SpringTime is resistant to the membrane locking problem that plagues FEM simulations, while avoiding the complexity of special solutions. Remarkably, we observe that SpringTime behaves better under bending than even the FEM model that it was trained with, thereby achieving superresolution in inference. We illustrate these phenomena and offer some potential explanations.

The main contribution of this paper is the mass-spring-based framework consisting of a modern stochastic learning pipeline with a physically based mass-spring (rather than neural) network. Specific features include:

- A novel force-and-impulse-based loss function;
- The ability to learn heterogeneous models with spatially-varying constitutive properties;
- Resistance to membrane locking problems that plague FEM-based simulations;
- Minimal requirements - it needs only point cloud data and total mass of the cloth for training, ground-truth cloth mesh not required;
- Flexibility in defining the resolution of the surrogate model.

The framework produces results that match or exceed state-of-the-art neural approaches in terms of accuracy and generalizability (see Fig. 3), while requiring significantly less training time.

2 Related Work

Neural constitutive models. A continuing challenge with deformable object simulation is defining an appropriate constitutive (material) model. Off-the-shelf models and parameters can be applied, but for more complicated objects, optimizing the constitutive model to agree with real-world measurements or specialized numerical data is common [9, 35, 38, 49]. More recently, deep learning has been deployed for this task. Graph Neural Networks [40, 41] or U-Nets [6] replace the entire simulation stack, spatial discretization and time integration, with learned update rules. These methods have been applied to cloth simulation, rigid body simulation with contact and granular material simulation, showing impressive results replicating scenarios similar to their training data. Rather than replacing both the spatial and temporal components of a simulation, Neural ODEs [14] choose to retain a classical, numerical time integrator while learning the integrand of the system (in our case the coupled first order dynamics). Neural constitutive models which learn offsets [11, 13, 48] or entire neural replacements [31] to existing continuum mechanics-based material models have been applied to volumetric objects (not thin shells as we do here), while Neural Jacobian [1] fields can learn static deformations (not dynamic like our work). Rather than making any neural substitutions, our work retains the stochastic training machinery of deep learning approaches but applies it to a physical model, in our case a mass-spring network.

[1] SIGGRAPH Asia 2024 keynote

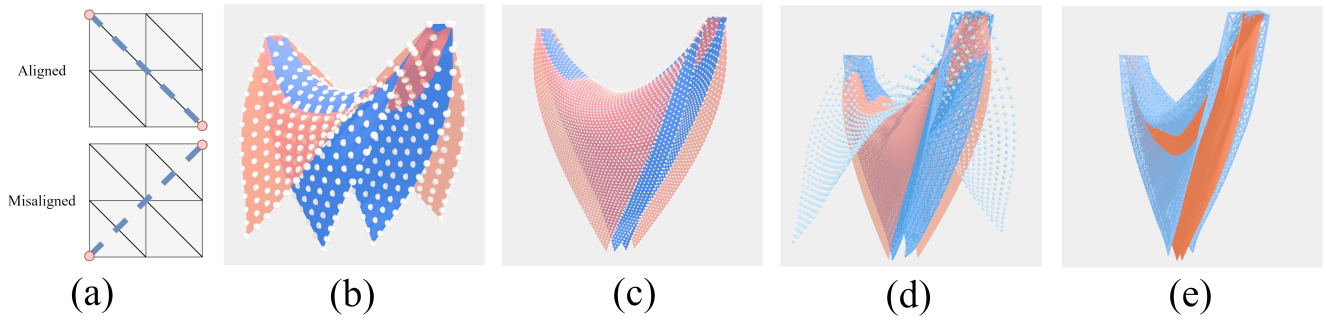

Figure 2: The folding experiment. A square piece of cloth is pinned at diagonally opposite corners, in two ways. (a) Illustration with 2x2 mesh, with pinned corners (red) and the fold line (dashed blue). In the "aligned" case, the fold is aligned with mesh edges; membrane locking occurs when it is "misaligned." (b) Aligned (blue) vs misaligned (red) low-resolution (16×16) mesh at an equilibrium configuration. The misaligned case appears stiffer due to locking. (c) Aligned (blue) vs misaligned (red) high-resolution (61×61) mesh at an equilibrium configuration. The differences due to locking are reduced. (d) SpringTime (blue mass-spring sheet) vs low-resolution mesh landmarks used for training (blue dots) vs high-resolution mesh (red) in misaligned rest configuration. Even though SpringTime was trained on the low-resolution mesh, its behavior is closer the high resolution mesh, with less locking. (e) SpringTime trained on mesh with bending stiffness (blue) vs high-resolution mesh without bending stiffness (red), in the aligned configuration. This shows that SpringTime can learn real bending stiffness while resisting locking. White dots are mesh vertices; all meshes have the same mass and physical dimensions, and only differ in bending stiffness or resolution.

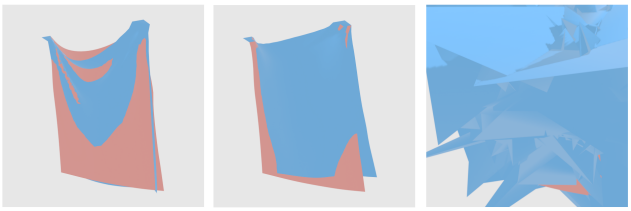

Figure 3: Results from different surrogate models (blue) after 8 seconds of simulation, compared with the ground truth (red). (Left) Our surrogate model; (middle), (right) are popular alternatives. Notice that SpringTime yields comparable steady-state reconstruction error as MeshGraphNet (MGN, middle), while Neural ODE (right) completely fails to reconstruct long rollouts.

Mass-spring systems are often regarded as outdated, but they offer important advantages over their more widely used discrete counterparts. Most notably, mass-spring networks can represent volumetric, thin-shell, and rod-like structures within a unified discretization, depending on the network topology and the assigned per-spring material parameters [16, 19, 24, 30, 50]. Despite their efficiency and expressiveness, deriving parameter values from classical constitutive models or estimating these parameters from data [7, 8, 16] has been historically challenging, especially due to the complexity of constitutive properties in heterogeneous materials [22, 54]. Closest to our method is Spring-Gaus [54], which fits a mass-spring-like network to volumetric Gaussian Splats. However, the employed node-based storage of material parameters allows for the same spring to produce forces that are not equal and opposite on its end-points, thus violating Newton's third law.

Cloth Simulation and constitutive property estimation. Simulations pertaining to cloth have always been a prime subfield in graphics and simulation research due to their varied industrial applications, from yarn-level simulators by Cirio et al. [15], Sperl et al. [45] which specialize in woven cloths, to continuum-level or meshed-based simulators [5, 17, 20, 22, 23, 27–29, 40, 42, 43, 49, 52]. Many of these works, including Feng et al. [17], Ju et al. [22], Wang et al. [49] focus on accurate constitutive parameter estimates by *real-to-sim* adaptations using specialized measurement devices; others [20, 40, 43] focused more on learning constitutive properties from synthetic data, and demonstrated great generalizability to real-world data [27–29, 53], or more specifically, learning garment deformation models jointly with human motion kinematics [5, 20, 26, 42, 47, 52]. However, cloth or garments chosen as the subject to estimate have been almost exclusively homogeneous, composed of a single fabric without complex fine-scale patterns and structures; the only exceptions are Stuyck and Chen [47] and Grigorev et al. [20], yet both explore fitting specifically of garments on virtual try-ons. The focus of our work is to address the broader problem of estimating any piece of cloth's *heterogeneous* constitutive properties, such as stiffness that varies across space using synthetic data. We distinguish from work like [28, 40, 47] that our estimation pipeline does not require a reference mesh as input, and from [5, 20, 47, 53] that we focus on the broader domain of modelling constitutive properties of a standalone cloth, rather than the specific domain of high-quality reconstruction of garment kinematics on moving human avatars. We show that by fitting a physically correct mass-spring network using our novel loss and training curriculum, we achieve comparable or better accuracy and generalizability than

previous approaches but at a significant reduction in training time and model complexity.

Membrane locking. For simulating dynamic elastica, the general consensus is that finite element spatial discretizations [44] coupled with implicit time integration [25] are state-of-the-art, with specialized discretization schemes available for thin sheets [37] or rods [4]. Finite element methods are rooted in continuum mechanics [34] and so allow for a wide-range of mathematically or experimentally derived material models which have been extensively validated across many domains. Yet a continuing challenge pertaining to FEM-based simulations is membrane locking [46]: when a finite-element mesh is not finely discretized, under certain configurations the mesh would appear to experience larger bending resistance due to some phantom stiffness. Previous work on FEM-based cloth simulation have attempted to address this issue by either simulating up to fine scales [36, 51] or applying constraints to solving dynamics [3, 12, 21]. While adaptive remeshing techniques used in recent garment simulators [20, 40] can address membrane locking to some extent, they incur higher computational cost in training and inference; Mass-spring models which, under appropriate stiffness and resolution definition are less prone to suffer from membrane locking, on the other hand, can be a natural choice for combating the problem. This motivates us to build a mass-spring based system, and we show that our trained surrogate resists membrane locking better than other FEM-based simulations.

3 Method

Learning task definition. Fig. 4 shows the neural surrogate modeling pipeline and the architecture of our neural constitutive model. Our method takes as input a *source system*: T frames of motion of a piece of cloth, with N landmark points on the cloth tracked from frame to frame. In addition to the positions $\mathbf{y}_i \in \mathbb{R}^{3N}$ of the landmarks at each frame i, we assume the source system also provides the external force $\mathbf{f}_{\text{land},i}$ acting on each landmark at each frame (e.g. gravity), an estimate of the cloth's area density ρ, and a binary classification of each landmark as either free or pinned in place. This source system data might come from a high-fidelity finite element simulation or from video observations of a real-world sample. We train a surrogate model to capture the behavior and constitutive properties of the source system while obeying the laws of physics. Once trained on the T input frames, the model can be used to efficiently and accurately predict the motion of the cloth given novel initial conditions, external forces, and boundary conditions. The surrogate system is a mass-spring network consisting of a user-specified number of particles P connected by S springs. The task is to learn constitutive parameters (stiffness and damping) of each spring from the source system's landmark trajectories. As a preprocessing step, we map the given landmark trajectories to target positions $\hat{\mathbf{x}}_i \in \mathbb{R}^{3P}$ of each particle, and the external forces to forces $\mathbf{f}_{\text{ext},i}$ on each particle (Section 3.1). These quantities are used to supervise training of the neural constitutive model at the heart of our surrogate system, which predicts the nonlinear restoring and damping forces $\mathbf{f}_{\text{springs}}$ exerted by the springs given the surrogate system's current position and velocity (Section 3.2):

$$h_\theta\left(\begin{bmatrix}\mathbf{x}^T\\ \dot{\mathbf{x}}^T\end{bmatrix}\right) : \mathbb{R}^{6P} \to \mathbb{R}^{3S}. \tag{1}$$

We map the predicted spring forces to internal forces on each surrogate particle [30],

$$\mathbf{f}_{\text{int}} = (\mathbf{A} \otimes I_{3\times 3})^T \mathbf{f}_{\text{springs}}, \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{S\times P}$ is the sparse signed incidence matrix of the mass-spring network (with $A_{sa} = -1, A_{sb} = 1$ if spring s connects particle a to particle b).

Finally, given these internal forces and the prescribed external forces and boundary conditions, we use semi-implicit Euler time integration to advance the state of the surrogate model from frame to frame. We learn the neural constitutive model parameters θ by comparing this simulated trajectory to the ground-truth motion observations $\hat{\mathbf{x}}_{i=1:T}$ (Section 3.3).

3.1 Spatial Discretization and Resampling

Given a desired number of surrogate particles P, we build our surrogate mass-spring network as follows: we isometrically unroll the source system's cloth specimen into a rectangle in the plane, then discretize this rectangle with a P-vertex regular square grid, as shown in Fig. 5 (left). We place a surrogate spring at each edge of the grid, with the spring rest length l_0 determined by the edge length in this rest configuration.

As shown in Fig. 5 (right), the source system landmarks do not necessarily correspond to the surrogate particles (and the resolution of the surrogate particle grid might be much coarser or finer than the density of landmark points). Therefore we must resample the given landmark positions $\mathbf{y}_i$ at each frame i to positions of the surrogate particles.

Let $\bar{\mathbf{y}}$ and $\bar{\mathbf{x}}$ be the rest positions of the unrolled landmarks and surrogate particles within the 2D rectangle, respectively. For each surrogate particle p, we identify its three nearest neighbors $v_{\{1,2,3\}}$ among the $\bar{\mathbf{y}}_j$. Let $b_{\{1,2,3\}}$ be the barycentric coordinates of $\bar{\mathbf{x}}_p$ within triangle $\left\{\bar{\mathbf{y}}_{v_1}, \bar{\mathbf{y}}_{v_2}, \bar{\mathbf{y}}_{v_3}\right\}$. We set the target position $\hat{\mathbf{x}}_{p,i}$ of particle p at frame i to

$$\hat{\mathbf{x}}_{p,i} = b_0 \mathbf{y}_{v_0,i} + b_1 \mathbf{y}_{v_1,i} + b_2 \mathbf{y}_{v_2,i}. \tag{3}$$

We map external forces to the surrogate particles using barycentric interpolation in an analogous way.

We also need to assign a mass m_p to each surrogate particle. We simply set $m_p = \rho A/P$, where A is the rectangle area.

3.2 Neural Constitutive Modeling

As shown in Fig. 4 (d), we use the incidence matrix $\mathbf{A}\otimes I_{3\times 3}$ to extract the displacement $\mathbf{d}$ and relative velocity $\mathbf{v}$ between each spring's two endpoints from the system position and velocity vectors $\mathbf{x}$ and $\dot{\mathbf{x}}$, as described in Liu et al. [30]'s method of fast simulation of mass-spring systems. The total internal force applied by each neural spring is computed as the sum of an elastic restoring force and viscous damping force (see Fig. 4 (e)):

$$\mathbf{f}_{\text{spring}} = \frac{k(\|\mathbf{d}\| - l_0)}{\|\mathbf{d}\|}\mathbf{d} + \frac{b(\mathbf{v}\cdot\mathbf{d})}{\|\mathbf{d}\|^2}\mathbf{d}, \tag{4}$$

where the elastic and damping coefficient $k_{s=1:S}, b_{s=1:S}$ are learnable parameters.

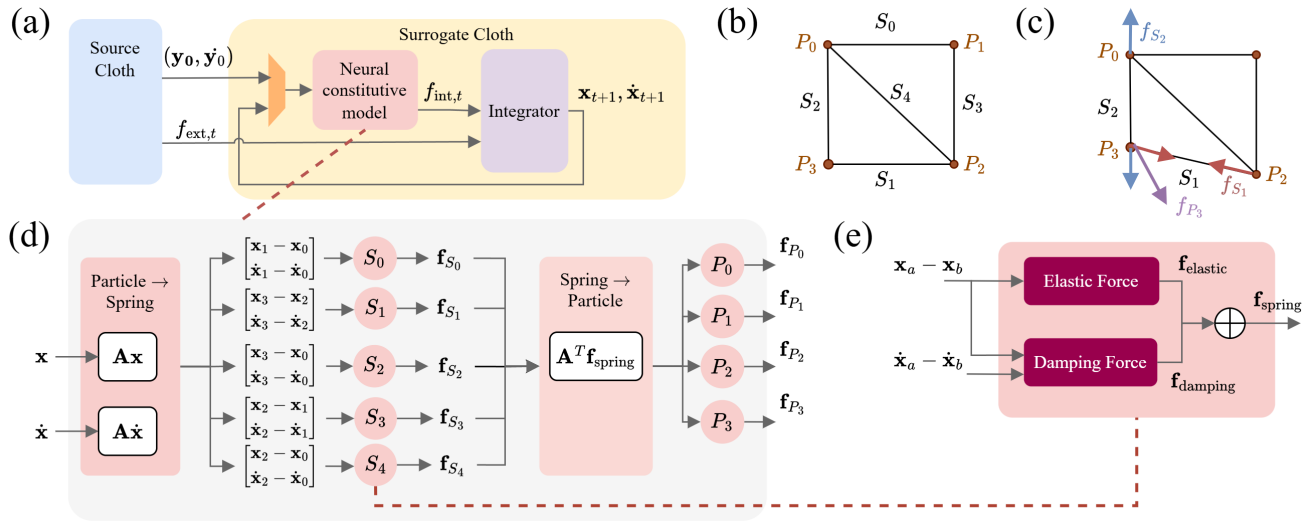

Figure 4: Method overview. (a) The material modeling pipeline. We sample system state $(\mathbf{y}_0, \dot{\mathbf{y}}_0)$ from the source cloth at the first time step of a rollout, resample following the scheme in Sec. 3.1 to get target particle positions, feed it to the neural constitutive model h_θ to predict neural particle force $\mathbf{f}_t$, then integrate. To further evolve the surrogate we use previously predicted states. (b)-(e) illustrates the architecture of our SpringTime which models a simple rectangular cloth, with rest configuration shown in (b) and a deformed configuration in (c). (d): Each neural spring (circle) takes the relative position and velocity of particles at its two ends, and predicts the spring's internal force. Then each particle accumulates forces applied by its surrounding springs. (e) Neural spring. At every time step, the module computes elastic and damping force and sums them up. Parts with learnable parameters are in dark red.

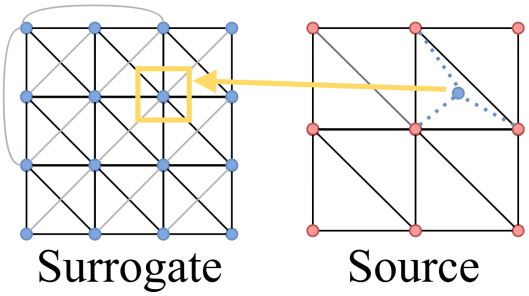

Figure 5: Spatial discretization of the surrogate system (left) and the source system (right). Bending springs and diagonal springs from top-left to bottom-right can be optionally excluded on construction. For simplicity, we only show two bending springs; the full surrogate contains bending springs connected at stride = 1. The dotted lines show how we compute the target position of each surrogate particle from the positions of nearby source landmarks via barycentric interpolation. Note that in training only landmarks from the source are needed, and topolgical information are not necessary.

3.3 Simulation and Training

Forward dynamics. We set the surrogate system's initial conditions based on the target positions computed in Section 3.1, i.e.

$$\mathbf{x}_0 = \hat{\mathbf{x}}_0, \quad \dot{\mathbf{x}}_0 = \frac{\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0}{\Delta t},$$

where Δt is the source system time step. We step these initial conditions forward in time with semi-implicit Euler integration. In particular, we update velocity using

$$\mathbf{M}(\dot{\mathbf{x}}_{j+1} - \dot{\mathbf{x}}_j) = \Delta t \left[\mathbf{f}_{\text{ext},j} + \mathbf{f}_{\text{springs}}(\mathbf{x}_j)\right], \tag{5}$$

where $\mathbf{M}$ is the $\mathbb{R}^{3P\times 3P}$ diagonal mass matrix, and position using $\mathbf{x}_{j+1} = \mathbf{x}_j + \Delta t\, \dot{\mathbf{x}}_{j+1}$.

Loss formulation. We iterate the above time integration process to simulate an entire trajectory $\mathbf{x}_{j=1:T}$. We compare this trajectory to the target trajectory $\hat{\mathbf{x}}_{j=1:T}$ using the loss

$$L = \lambda_{\text{f}} L_{\text{f}} + \lambda_{\text{J}} L_{\text{J}} + \lambda_{\text{k neg.}} L_{\text{k neg.}} + \lambda_{\text{b neg.}} L_{\text{b neg.}}. \tag{6}$$

In Eq. 6, L_{f} denotes the force loss, which penalizes difference between target and simulated net force applied on each particle in the system at each time step. The target net force is estimated from the target positions and Newton's Second Law:

$$L_{\text{f}} = \frac{1}{P(T-2)} \sum_{i=2}^{T-1} ||\mathbf{f}_i - \hat{\mathbf{f}}_i||_2^2 \tag{7}$$

$$\mathbf{f}_i = \mathbf{M}\left(\frac{\mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i+1}}{(\Delta t)^2}\right), \tag{8}$$

where $\mathbf{f}_i = \mathbf{f}_{\text{ext},i} + \mathbf{f}_{\text{springs}}(\mathbf{x}_i)$ denotes the net force (including gravity, environmental damping, and the spring forces) applied to each particle at time step i of forward dynamics. L_{J} denotes the impulse loss, which penalizes difference between target and predicted system impulse:

$$L_{\text{J}} = \frac{1}{P} \left\| \mathbf{J} - \hat{\mathbf{J}} \right\|_2^2, \tag{9}$$

where $\mathbf{J}$ is the accumulated impulse on each particle from the start to the end of the trajectory, integrated using the Trapezoid Rule:

$$\mathbf{J} = \frac{1}{2} \sum_{j=1}^{T-1} (\mathbf{f}_j + \mathbf{f}_{j+1}) \Delta t, \tag{10}$$

and $\hat{\mathbf{J}}$ is computed analogously from the $\hat{\mathbf{f}}$. Intuitively, the force and impulse loss terms act analogously to the P and I terms in a PID controller, where the former penalizes discrepancy between predicted and target instantaneous forces applied to each particle and the latter penalizes error accumulation over time. We notice through experiments that the optimal weights λ_J, λ_f vary across different sources of data (i.e. source systems) and need to be fine-tuned, just as the weights of a PID controller must be tuned when a mechanical system's parameters change.

The final two loss terms of Equation (6) are regularization terms that penalizes non-physical, negative spring stiffness or damping constants. Specifically,

$$L_{\text{k neg.}} = \sum_{s=1}^{S} \text{ReLU}(-k_s) \tag{11}$$

$$L_{\text{b neg.}} = \sum_{s=1}^{S} \text{ReLU}(-b_s), \tag{12}$$

where the k and b are the learned per-spring parameters of our neural constitutive model (see Sec. 3.2). The term stays at zero as long as the estimated parameter stays positive.

Many alternative loss functions for training the surrogate could also be used, such as position loss [31] or acceleration loss [41], but in our ablation experiments (Sec. C.2) we found that the force-and-impulse loss works best in our setting.

Training from multiple trajectories. For clarity of exposition, we've assumed above that the source system consists of only one trajectory with T frames of motion. Our approach extends in a straightforward way to training from multiple source system trajectory samples, where the same surrogate mass-spring network is used to fit all trajectories.

Dual-pass training. While the elastic force is only position-dependent, damping force depends on both particle position and velocity. To disentangle the effects of each force on particle dynamics, we devise a training curriculum that isolates learning stiffness from learning damping by splitting the training into two passes. In the first pass we freeze damping weights and optimize only the stiffness weights, and in the second pass we freeze stiffness weights and optimize only damping weights. Furthermore, we use low-velocity motion data sampled from the source system in the first pass of training, so non-gravitational force applied on particles will be dominated by the elastic force. We discuss more on how low-velocity data is sampled in Sec. A.

4 Experiments

4.1 Synthetic Data Generation

Constructing source systems. We build two source systems of cloths modeled under different simulation frameworks: (i) A square-shaped cloth made up of particles and springs. Forces that act upon particles come from springs only; (ii) A square-shaped cloth constructed from a triangular mesh. Internal forces are evaluated based on FEM principles. We use NVIDIA WARP's FEM-based cloth simulation publicly available on Github [32]. We discretize each cloth into a $P \times 3P$ particle grid. The choice of P varies across experiments; for the experiment in which we compare SpringTime with baselines on generalizability to novel boundary and initial conditions, we use $P = 31$, which is close to the discretization scale picked by recent work such as Shao et al. [43]. We assign different stiffness values to either springs in the mass-spring cloth or triangles in the mesh-based cloth so they have spatially-varying constitutive properties: see Fig. 1.

Generating rollouts with different initial and boundary conditions. To collect ground-truth simulation data, for each cloth we anchor a set of its vertices so they cannot move, and we simulate the deformation of the rest of the cloth over time. The set of vertices being fixed in space, formally defined as anchor neighborhood Ω defines the boundary condition of a rollout; the initial position and velocity of all particles $\{\mathbf{y}, \dot{\mathbf{y}}\}$ defines its initial condition. For each rollout n, we randomly pick two anchor neighborhoods as boundary condition Ω_n, and we apply a random rotation to the sheet to assign the cloth with initial condition $\{\mathbf{y}_n, \dot{\mathbf{y}}_n\}$: see Fig. 6. For training SpringTime and the Neural ODE baseline we use $N = 512$ rollouts; for testing we use $N = 16$ rollouts. Each rollout lasts 8 seconds, during which the cloth falls under the influence of gravity and Rayleigh damping force $\mathbf{f}_{\text{Rayleigh}} = -b_{\text{Rayleigh}}\dot{\mathbf{x}}$, where Rayleigh damping constant $b_{\text{Rayleigh}} = 0.1$. We simulate with timestep size of 1 ms, and this time discretization scheme gives us 8 k time steps of data from each rollout.

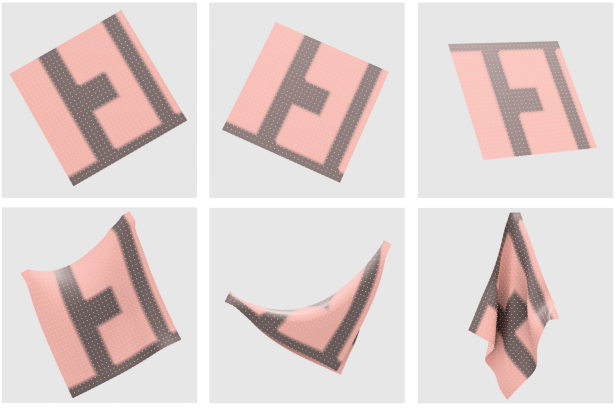

Figure 6: Generating $N = 3$ rollouts of equal lengths but with different initial conditions (top) and boundary conditions. Equilibrium states reached by the last step (bottom). Stiffer regions are colored darker.

Constructing motion clip dataset. We follow the approach in prior work to break down rollouts into shorter clips [18, 31], then train on a batch of clips in each iterations. We refer readers to Sec. A for more details.

4.2 Metrics

In some of our experiments, we evaluate SpringTime by applying it to a source system that is itself a simulation of a mass-spring network. In this setting, we can easily compare the surrogate's estimated spring stiffness and damping parameters to those of the source system. We quantify the error in the reconstructed parameters by computing the root-mean-squared error (RMSE) between their estimated and ground-truth values:

$$\text{RMSE}_k = \sqrt{\frac{1}{S}\sum_{s=1}^{S}\left(k_s - \hat{k}_s\right)^2}. \tag{13}$$

where $\hat{k}_s$ is the ground-truth stiffness constant in the source system. For the damping constants we compute RMSE_b analogously. These two metrics offer us an objective measure of how well the surrogate model captures the source system's constitutive properties.

When the source system is a finite element simulation or a video of real-world cloth, ground-truth spring parameters are not available. We instead assess the quality of our surrogate model by comparing its simulated motion to the source system trajectory. Specifically, we compute the RMSE between the estimated and target positions of each particle at trajectory frame j:

$$\text{Motion RMSE}_j = \sqrt{\frac{1}{P}\sum_{i=1}^{P}\left(\mathbf{x}_{j,i} - \hat{\mathbf{x}}_{j,i}\right)^2}, \tag{14}$$

where $\mathbf{x}_{j,i}$ denotes the position of the i-th particle at frame j and $\hat{\mathbf{x}}_{j,i}$ is the corresponding target position (computed via interpolation of the landmarks of the source system as described in Sec. 3.1). For an entire trajectory or ensemble of trajectories, we compute the mean and standard deviation of the Motion RMSE across all time steps of all trajectories.

4.3 Reconstructing Spatially Varying Materials

The goal of this experiment is to show that SpringTime can accurately estimate the stiffness and damping properties of a source cloth with spatially varying stiffness. We use the RMSE_k and RMSE_b metric established in Eq. 13 to assess the quality of our constitutive property estimates, and to use them, we build the source cloth with masses and springs, and we construct a surrogate system with the same resolution as the source cloth. Bending and the complete set of shear springs are included in both the source and the surrogate. This allows us to establish a one-to-one mapping from springs in the source cloth to springs in the surrogate cloth. We also assess the quality of motion reconstruction using the Motion RMSE metric from Eq. 14.

Fig. 7 shows the accuracy of stiffness and damping estimates, and the reconstruction error obtained from a test set which contains 16 eight-second-long rollouts. The source cloth we used to generate training and test rollouts is a square-shaped, 15.5×15.5 m cloth discretized into a 32×32 particle grid connected by springs. The source cloth contains springs with three stiffnesses: 25, 50 and 100 N/m, and all springs share the same damping constant of 6 N/m/s. Fig. 7 (left) tells that most of our estimated stiffnesses are close to the ground-truth. The damping estimates are also close to the ground-truth on average, despite some variations: see Fig. 7 (middle).

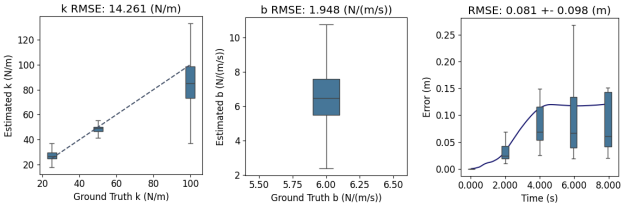

Figure 7: Predicted vs ground-truth (left) spring stiffness constants; (middle) damping constants; (right) per-frame motion reconstruction RMSE. The curve shows mean RMSE across 16 test rollouts on each time step; we subsample five steps to visualize the distribution across rollouts. Each cloth contains 5, 826 springs.

4.4 Robustness against Membrane Locking

Finite element methods are susceptible to membrane locking, a phenomenon that produces artificial bending stiffness that is more significant at coarse resolutions [46]. An interesting consequence of our approach, which is learning material properties from force and impulse losses and not the shape of the draped cloth, is that we are able to model material properties even when the training data generated from a low-resolution FEM-based simulation is significantly affected by the numerical limitations of membrane locking. Thus, our simulation results match higher-resolution FEM simulations more closely than the training data– see Fig. 8.

Figure 8: Our 32x32 surrogate (center) was trained on a data from a 31x31 FEM mesh (left) that exhibits significant membrane locking. This is apparent in the drape test of the cloth on a square table. Remarkably, our results more closely match the results of a higher resolution FEM cloth simulation (right), indicating that our model captures material properties well.

To further validate the super-resolution effect of the mass-spring-based surrogate, as shown in Fig. 2 (b), (c), we pin two corners of a square-shaped FEM-simulated mesh along a diagonal, make the cloth fold under the influence of gravity and Rayleigh damping, and sample the cloth's equilibrium configurations. We compare

the equilibrium configurations under two cases: (i) "Aligned", i.e. folding along the major diagonal that coincides with diagonal edges in the mesh; (ii) "Misaligned", i.e. folding along the minor diagonal. Under the misaligned case, as we expect, the cloth sags less than the aligned case, and the difference in equilibrium configurations is more apparent on the low-resolution mesh. As shown in Fig. 2 (d), our mass-spring-based surrogate which includes the full set of diagonal and bending springs, and trained on landmark trajectories sampled from the low-resolution mesh in (b), however is robust against this effect, because we can see that at equilibrium configuration it is closer to the high-resolution mesh that suffers less from membrane locking. Further, by comparing Fig. 2 (d), (e) we can see that the surrogate is robust against the pseudo-stiffness caused by membrane locking not by chance, but because it can properly fit to the actual bending resistance of the mesh— the surrogate at equilibrium configuration is closer to the mesh with bending stiffness (d) than the mesh with zero bending stiffness (e).

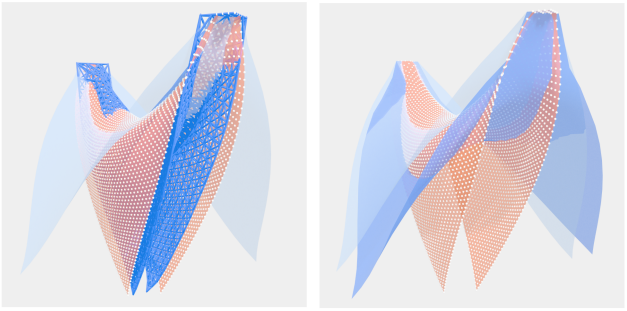

Figure 9: SpringTime (left, dark blue) vs MGN (right, dark blue) trained on kinematic data sampled from low-resolution (16×16) mesh (light blue), settled to the misaligned configuration in evaluation. A high-resolution (61×61) mesh (red) is placed in each scene for reference.

Moreover, we offer a qualitative comparison with MGN [40] in terms of the two surrogate's behavior in the misaligned equilibrium configuration. Notice that in Fig. 9 MGN matches the low-resolution mesh's behavior more than the high-resolution mesh's, which makes sense, because the surrogate is trained to model the behavior of the low-resolution mesh when boundary conditions and trajectories are provided as inputs.

While both SpringTime and MGN are data-driven methods that encode cloth's constitutive properties and reconstruct reference motion in inference, what makes SpringTime stands out is its robustness to numerical artifacts that exist in the training data. We believe SpringTime attains the robustness by more accurately estimating the cloth's constitutive properties, after seeing trajectories sampled in a multitude (512) rollouts, many of which would not lead to a "locked" equilibrium configuration as in the misaligned case. MGN on the other hand is more vulnerable to such artifacts in the training data, since it tends to memorize the locked equilibrium configuration corresponding to the input boundary condition. We refer readers to Sec. B.2 for hyperparameters used to train MGN.

Table 1: Comparison of our method with prior differentiable cloth simulators with publicly available codebases across key features. × indicates a method not supporting the feature natively; ○ indicates native support provided but without demonstration; ✓ indicates demonstrated native support. Res-independence means support for simulating surrogate at finer or more coarse resolution than reference data; Long rollout means simulating rollouts longer than 2 seconds.

Method	Heterogeneity?	Res-indep.?	Ref-mesh free?	Long rollouts?
Ours	✓	✓	✓	✓
NCLaw [31]	○	×	×	○
DiffCloth [27]	×	×	×	✓
MGN [40]	○	○	✓	✓
GNN [41]	○	×	✓	○
Liang et al. [29]	×	×	×	✓
Neural ODE [14]	○	✓	✓	○

4.5 Generalization Experiments

Here we discuss two experiments that assess how well SpringTime generalizes to novel scenarios not included in training rollouts. Moreover, we demonstrate that despite our neural constitutive model is built from masses and springs, it can be trained to emulate the behavior of source cloths constructed from triangular meshes bearing a different resolution and simulated with FEM principles.

Baselines. Tab. 1 compares our method with several recent, open-sourced differentiable cloth or general deformable simulators across key features— if the simulator supports modelling heterogeneous cloths, can simulate surrogate bearing different spatial resolution from the source, if it needs a reference mesh to kick-start training and if it supports simulating long rollouts. We focus exclusively on methods that allow us to model material properties of a stand-alone piece of cloth from kinematic data and optionally, the total mass and physical dimension of the system. While state-of-the-art garment simulators as in [20, 26, 42, 43] has achieved stunning results in reconstructing garment kinematics on moving human avatars, we exclude them as they tackle the specific body-and-garment joint fitting problem. In Sec. 4.4 we have made a brief qualitative comparison with Pfaff et al. [40] in terms of robustness to membrane locking in simulation data. Here we compare, both quantitatively and qualitatively between our surrogate system and the baselines, in terms of ability to generalize to new initial and boundary conditions: (i) Neural ODE [14] which is an obvious choice for modelling constitutive properties, due to the simplicity and flexibility of its MLP-based architecture — see Fig. 12 in Sec. C for an illustration of the architecture; (ii) MeshGraphNet (MGN) [40] which can learn a surrogate model for complex thin-shell materials from point cloud data. We exclude state-of-the-art neural constitutive methods like NCLaw [31] from being compared as baselines, since they require premium knowledge such as internal stress in their training curricula to learn complex 3D deformables. Another criterion for our comparisons is the method's ability to model heterogeneity in cloth, so we have excluded methods like Li et al. [27], Liang et al. [29] which places greater emphasis on modelling homogeneous cloth. We demonstrate that our neural constitutive model is able to generalize better to scenes with novel initial or boundary conditions, and it adapts well to scenes that involve interaction of new objects over extended period of time. We also compare SpringTime with the two selected baselines in

Table 2: Motion reconstruction RMSE of our method vs the baseline across test rollouts of different lengths. We report mean and standard deviation of per-frame reconstruction RMSE across 200 frames, 16 rollouts for 8 s long simulations.

Time	SpringTime	MGN	Neural ODE
8 s	**0.64 ± 0.42**	0.68 ± 0.45	6.34 ± 6.30

terms of training time and model complexity, measured in number of learnable parameters.

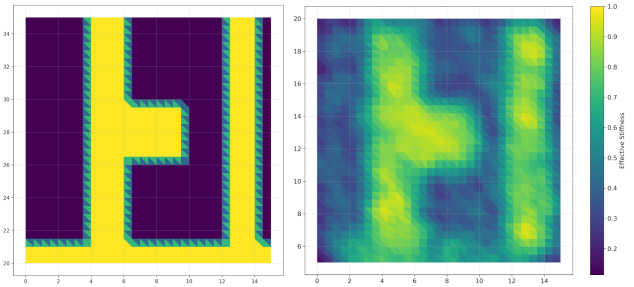

Figure 10: Distribution of material stiffness for the (left) The mesh-based source cloth; (right) learned mass-spring-based surrogate cloth. The stiffness of the two cloth are normalized to the same scale from 0 to 1. Darker regions are more stiff.

Generalization to novel initial and boundary conditions. We train SpringTime and the baselines on 512 rollouts generated from a triangular mesh-based cloth with spatially-varying stiffness: see Fig. 10 (left). The mesh-based cloth is discretized into a 31×31 particle grid. For running SpringTime and Neural ODE, we discretize the surrogate cloth into a 32×32 particle grid, which is relatively prime to the resolution of the source cloth. MeshGraphNet on the other hand requires the source cloth's default topological structure for training and testing, so we do not perform resampling. We then train SpringTime and the baselines, and use the trained models to reconstruct 16 test-set rollouts with new initial and boundary conditions, visualize the reconstructed rollouts and compute reconstruction error reported in Tab. 2. We see SpringTime outperforms both baselines in terms of motion reconstruction accuracy over long rollouts. Fig. 10 (right) shows the spatial variation in stiffness in the fitted surrogate cloth. Notice that despite the source cloth is built from meshes rather than springs and masses, SpringTime is still able to recover the two stiffer stripes that run from the top to the bottom and the central patch, although reconstruction of the bottom, horizontal stripe is sub-optimal. In terms of qualitative assessment, we show in Fig. 3 that after simulating for 8s SpringTime can attain a steady state closer to the ground-truth than the baselines. We refer readers to Fig. 13 for more qualitative results. We see from the motion RMSE in Tab. 2 and Fig. 3 that Neural ODE surrogate completely disintegrates after 8s of simulation. We hypothesize this behavior to be due to lack of topological information available to the model. As the surrogate evolves over time, lacking of topological information allows the particles to roam freely, thereby disintegrating the cloth. Methods that use MLPs to learn dynamics [31], do so along with other strong priors as mentioned in the previous section.

Turning to training time and model complexity: without leveraging domain-specific information, both baselines require more learnable parameters to learn the topology of a cloth: while SpringTime contains 11,652 learnable parameters to fit a surrogate system with resolution of 32×32, Neural ODE requires over 6M learnable parameters, and MGN contains 2.3M parameters. We also measured the training time of SpringTime and the baselines on the same platform, and found that training SpringTime for two passes requires only 120 minutes, while Neural ODE requires 100 minutes of training time which is slightly less than SpringTime. MGN on the other hand requires considerably more training time: training for $3.8M$ iterations requires 48 hours, and we expect training for $10M$ steps as used in [40] would require 126 hours. As explained in Sec. B.2, we had to reduce the number of training iterations for MGN due to our resource limitations, but we find that the loss in the experiment has mostly converged after $3.8M$ iterations, and the motion reconstruction error is comparable for cases involving coarse meshes.

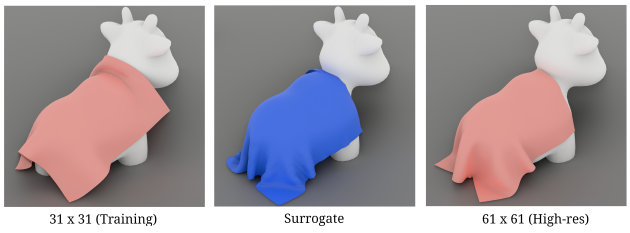

Figure 11: Generalization to novel dynamics. Left: after simulation starts, under gravity, the source cloth of low resolution (31×31) drapes onto a cow beneath it and eventually settles to a steady state. Middle: the surrogate cloth of comparable resolution (32×32) drapes onto a replica of the same cow. Right: the high-resolution (61×61) source cloth drapes onto the cow for reference.

Generalization to novel dynamics. We place the trained surrogate cloth models from the above experiment under a new scene, in which all particles on the surrogate cloth are free to move, and the surrogate cloth starts falling from time $t = 0s$ until it drops onto another object. We also instantiate the source cloth in the same scene and set it to have the same initial state (position and orientation of the cloth) as the surrogate, and place an identical object below it for the source cloth to drape onto. Figs. 8 and 11 show the steady state after each system settles. We use Warp [32]'s built-in SDF-based collision detector.

SpringTime produces a satisfactory match to the static, draped position of the source cloth, with fine wrinkles that can be difficult to observe if the FEM-simulated source cloth has low resolution. The Neural ODE model fails to yield a stable result during generalization, and methods like MGN cannot be easily applied to scenarios with previously unseen objects or contacts in the scene, since such methods require an input window consisting of multiple frames of kinematic data, and they do not take external forces, contact information or actions as inputs to the model.

5 Conclusion

We presented SpringTime, a surrogate model trained with a novel force-impulse loss and training curriculum that can learn complex, spatially-varying constitutive properties of a piece of simulated cloth without relying on a reference mesh. We demonstrated that our method can reproduce the observed behavior of both homogeneous and heterogeneous cloth sheets better than previous neural network approaches. It generalizes more readily to new scenarios, including those with previously unobserved contacts. Further, we showed that SpringTime's mass-spring-based architecture is fairly robust against membrane locking which plagues FEM-based simulations.

Our method has some limitations. While mass-spring lattices can represent a wide-range of material properties, it is uncertain if they provide universal approximating power for constitutive behavior. The topology itself could be learned for better performance and better resolution of the spatial variation. Finally, to learn spatially varying material properties, all parts of the cloth should be "'sufficiently excited" in the training data. Better methods to achieve this than dropping the cloth in various ways could be designed, to improve the recovery of material properties.

Despite these limitations, we were pleasantly surprised by how well mass-spring systems work when trained with FEM simulation data, and their surprising ability to avoid membrane locking artifacts even when they are present in some of the training examples. Thus, when trained by our offline process, SpringTime could provide effective bridge between sophisticated and slow FEM simulators and the needs of real-time applications in games and VR.

References

[1] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. 2022. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *arXiv preprint arXiv:2205.02904* (2022).

[2] J Amirbayat and JWS Hearle. 1989. The anatomy of buckling of textile fabrics: Drape and conformability. *Journal of the Textile Institute* 80, 1 (1989), 51–70.

[3] Jan Bender, Raphael Diziol, and Daniel Bayer. 2011. Simulating Inextensible Cloth Using Locking-free Triangle Meshes.. In *VRIPHYS*. 11–17.

[4] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*. ACM, 1–12.

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

[6] Anindya Bhaduri, Ashwini Gupta, and Lori Graham-Brady. 2022. Stress field prediction in fiber-reinforced composite materials using a deep learning approach. *Composites Part B: Engineering* 238 (2022), 109879.

[7] Gérald Bianchi, Matthias Harders, and Gábor Székely. 2003. Mesh topology identification for mass-spring models. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003: 6th International Conference, Montréal, Canada, November 15-18, 2003. Proceedings 6*. 50–58.

[8] Gérald Bianchi, Barbara Solenthaler, Gábor Székely, and Matthias Harders. 2004. Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2004: 7th International Conference, Saint-Malo, France, September 26-29, 2004. Proceedings, Part II 7*. Springer, 293–301.

[9] Bernd Bickel, Moritz Bächer, Miguel A Otaduy, Wojciech Matusik, Hanspeter Pfister, and Markus Gross. 2009. Capture and modeling of non-linear heterogeneous soft tissue. *ACM transactions on graphics (TOG)* 28, 3 (2009), 1–9.

[10] David E Breen, Donald H House, and Michael J Wozny. [n. d.]. Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 365–372.

[11] Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M Chiaramonte, Kevin Carlberg, and Eitan Grinspun. 2023. LiCROM: Linear-subspace continuous reduced order modeling with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

[12] Hsiao-yu Chen, Paul Kry, and Etienne Vouga. 2019. Locking-free Simulation of Isometric Thin Plates. *arXiv preprint arXiv:1911.05204* (2019).

[13] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, GA Pershing, Henrique Teles Maia, Maurizio M Chiaramonte, Kevin Carlberg, and Eitan Grinspun. 2022. CROM: Continuous reduced-order modeling of PDEs using implicit neural representations. *arXiv preprint arXiv:2206.02607* (2022).

[14] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. *Advances in neural information processing systems* 31 (2018).

[15] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A Otaduy. 2014. Yarn-level simulation of woven cloth. *ACM Transactions on Graphics (TOG)* 33 (2014), 1–11. Issue 6.

[16] Josildo Pereira da Silva, Gilson A Giraldi, and Antônio L Apolinário Jr. 2015. A new optimization approach for mass-spring models parameterization. *Graphical Models* 81 (2015), 1–17.

[17] Xudong Feng, Wenchao Huang, Weiwei Xu, and Huamin Wang. 2022. Learning-based bending stiffness parameter estimation by a drape tester. *ACM Transactions on Graphics (TOG)* 41 (2022), 1–16. Issue 6.

[18] Erik Gärtner, Mykhaylo Andriluka, Erwin Coumans, and Cristian Sminchisescu. 2022. Differentiable dynamics for articulated 3d human motion reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 13190–13200.

[19] Karolina Golec, J-F Palierne, Florence Zara, Stéphane Nicolle, and Guillaume Damiand. 2020. Hybrid 3D mass-spring system for simulation of isotropic materials with any Poisson's ratio. *The Visual Computer* 36, 4 (2020), 809–825.

[20] Artur Grigorev, Michael J Black, and Otmar Hilliges. 2023. Hood: Hierarchical graphs for generalized modelling of clothing dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16965–16974.

[21] Ning Jin, Wenlong Lu, Zhenglin Geng, and Ronald P Fedkiw. 2017. Inequality cloth. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. 1–10.

[22] Eunjung Ju, Kwang yun Kim, Sungjin Yoon, Eungjune Shim, Gyoo-Chul Kang, Phil Sik Chang, and Myung Geol Choi. 2024. Estimating Cloth Simulation Parameters From Tag Information and Cusick Drape Test. In *Computer Graphics Forum*. e15027.

[23] Navami Kairanda, Marc Habermann, Christian Theobalt, and Vladislav Golyanik. 2023. Neuralclothsim: Neural deformation fields meet the thin shell theory. *arXiv preprint arXiv:2308.12970* (2023).

[24] Maciej Kot, Hiroshi Nagahashi, and Piotr Szymczak. 2015. Elastic moduli of simple mass spring models. *The visual computer* 31 (2015), 1339–1350.

[25] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy R Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4 (2020), 49.

[26] Yifei Li, Hsiao-yu Chen, Egor Larionov, Nikolaos Sarafianos, Wojciech Matusik, and Tuur Stuyck. 2024. Diffavatar: Simulation-ready garment optimization with differentiable simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4368–4378.

[27] Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2022. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics (TOG)* 42, 1 (2022), 1–20.

[28] Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2022. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics (TOG)* 42 (2022), 1–20. Issue 1.

[29] Junbang Liang, Ming Lin, and Vladlen Koltun. 2019. Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems* 32 (2019).

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

[31] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. 2023. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*. 23279–23300.

[32] Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/nvidia/warp. NVIDIA GPU Technology Conference (GTC).

[33] Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. NVIDIA GPU Technology Conference (GTC).

[34] G Thomas Mase, Ronald E Smelser, and George E Mase. 2009. *Continuum mechanics for engineers*. CRC press.

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

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

[37] Rahul Narain, Armin Samii, Tobias Pfaff, and J O'Brien. 2014. ARCSim: Adaptive refining and coarsening simulator. *ACM Trans. Graph* 1 (2014), 2016.
[38] Dinesh K Pai, Kees van den Doel, Doug L James, Jochen Lang, John E Lloyd, Joshua L Richmond, and Som H Yau. 2001. Scanning physical interaction behavior of 3D objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 87–96.
[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
[40] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. 2020. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409* (2020).
[41] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. 2020. Learning to simulate complex physics with graph networks. In *International conference on machine learning*. 8459–8468.
[42] Igor Santesteban, Miguel A Otaduy, and Dan Casas. 2022. SNUG: Self-Supervised Neural Dynamic Garments. IEEE. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2. 9. Issue 5.
[43] Yidi Shao, Chen Change Loy, and Bo Dai. 2025. Learning 3D Garment Animation from Trajectories of A Piece of Cloth. *arXiv preprint arXiv:2501.01393* (2025).
[44] Eftychios Sifakis and Jernej Barbic. 2012. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*. 1–50.
[45] Georg Sperl, Rosa M Sánchez-Banderas, Manwen Li, Chris Wojtan, and Miguel A Otaduy. 2022. Estimation of yarn-level simulation models for production fabrics. *ACM Transactions on Graphics (TOG)* 41 (2022), 1–15. Issue 4.
[46] Henryk Stolarski and Ted Belytschko. 1982. Membrane locking and reduced integration for curved elements. (1982).
[47] Tuur Stuyck and Hsiao-yu Chen. 2023. Diffxpbd: Differentiable position-based simulation of compliant constraint dynamics. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (2023), 1–14.
[48] Bin Wang, Yuanmin Deng, Paul Kry, Uri Ascher, Hui Huang, and Baoquan Chen. 2020. Learning elastic constitutive material and damping models. In *Computer Graphics Forum*, Vol. 39. 81–91. Issue 7.
[49] Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. 2011. Data-driven elastic models for cloth: modeling and measurement. *ACM transactions on graphics (TOG)* 30, 4 (2011), 1–12.
[50] Wen Xu, Yong Wang, Weimin Huang, and Yuping Duan. 2022. An efficient nonlinear mass-spring model for anatomical virtual reality. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–10.
[51] Jiayi Eris Zhang, Jérémie Dumas, Yun Fei, Alec Jacobson, Doug L James, and Danny M Kaufman. 2022. Progressive simulation for cloth quasistatics. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–16.
[52] Fang Zhao, Zekun Li, Shaoli Huang, Junwu Weng, Tianfei Zhou, Guo-Sen Xie, Jue Wang, and Ying Shan. 2023. Learning anchor transformations for 3d garment animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 491–500.
[53] Yang Zheng, Qingqing Zhao, Guandao Yang, Wang Yifan, Donglai Xiang, Florian Dubost, Dmitry Lagun, Thabo Beeler, Federico Tombari, Leonidas Guibas, et al. 2024. Physavatar: Learning the physics of dressed 3d avatars from visual observations. In *European Conference on Computer Vision*. Springer, 262–284.
[54] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. 2024. Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians. *European Conference on Computer Vision (ECCV)* (2024).
[55] Zeshun Zong, Xuan Li, Minchen Li, Maurizio M Chiaramonte, Wojciech Matusik, Eitan Grinspun, Kevin Carlberg, Chenfanfu Jiang, and Peter Yichen Chen. 2023. Neural stress fields for reduced-order elastoplasticity and fracture. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

A Synthetic Motion Clip Set Construction

Other than particle positions and velocities over time, for each rollout we register (1) the particles fixed in space which is the boundary condition Ω discussed in Sec. 4.1; (2) the mass of the cloth. For a simulated, dynamic source cloth this can be accessed by querying the area density ρ and surface area A of the cloth from the simulator used to generate data; for real-world objects we expect users to have the means of measuring the object's weight and from there they can obtain the mass of the cloth. (3) physical dimension of the cloth, i.e. length and width. We assume the cloth always take a rectangular shape.

Although in Sec. 3 we mentioned that external force injected into the source cloth is required as an input to our training curriculum, in practice we do not need to track it on a per-time step basis: gravity can be easily simulated for the surrogate knowing that $-9.81\ \mathrm{m/s^2}$, the Rayleigh damping constant is know prior to simulation starts since this is a user-specified parameter. If we are to extend our method to source cloth that exist in the real world, we can simply ignore the damping force, e.g. air drag and assume gravity being the only external force.

Construction of the point cloud clip dataset. Following the approach adopted by prior work on system identification with differentiable simulators [18, 31] in training, we partition each rollout into non-overlapping "clips" with each clip containing T steps of simulation data, and make the surrogate system track the source system's motion throughout a clip on each training iteration. For training mass-spring net, we set $T = 500$, so each rollout gives $\frac{8k}{500} = 16$ clips, where 8s of rollout gives $8k$ steps of data at timestep size of 1ms. So from 512 rollouts we get $8k$ clips, or $4M$ time steps of kinematic data. To reduce memory footprint and disk space usage, we regularly downsample time steps– taking 1 in every 40 steps, and that leaves us $100k$ time steps of training data. We then take clips from the last $1s$ of each rollout to yield $50k$ time steps of low-velocity training motion data.

B Implementation Details and Hyperparameters

B.1 Implementation Details

As in Ma et al. [31] we implement our neural constitutive model in PyTorch [39], and we implement contact resolution and forward dynamics in WARP [33]. Both forward and backward propagation through our surrogate model are fast, since CUDA tensors from PyTorch can be directly used by WARP, and we have exploited WARP's parallelism by allowing simulation of multiple clips to happen in parallel. All experiments are run on NVIDIA V100's with 32GB of GPU memory.

For all experiments we train the neural constitutive model with the Adam optimizer. Learning rates, number of training iterations and the weight of different loss terms are task-specific.

B.2 Hyperparameters

Impulse loss vs. force loss. When training SpringTime on kinematic data sampled from mass-spring based cloth, we used a combination of force loss and impulse loss; specifically, we set $\lambda_{\mathrm{f}} = 1$ and $\lambda_{\mathrm{J}} = 10$ in both the stiffness and the damping pass. On the other hand, for training on data sampled from mesh-based cloths, in the stiffness pass we used the impulse loss exclusively with $\lambda_{\mathrm{J}} = 1$, and in the damping pass we used the force loss exclusively with $\lambda_{\mathrm{f}} = 1$. We believe the optimal combination of the force and impulse term depends on (i) the source of training data and (ii) whether we are estimating springs' elastic stiffness or damping properties. What is particularly interesting is that for FEM-mesh-based training data, using impulse loss is crucial for learning stiffness, yet both impulse and force loss are crucial for obtaining good damping estimates. We hypothesize that impulse loss which tracks error in force accumulation over time is more resistant to noises induced by taking the

second derivative with respect to particle positions, therefore more suitable for learning from low-velocity data in the stiffness pass. On the other hand, force loss is sensitive to minor perturbations induced by taking derivatives. While this is the case for learning from meshes that bear a different resolution from the surrogate, we see that in the case of learning from mass-spring sheet with the same resolution as the surrogate, both impulse and force loss enable effective learning in either the stiffness or the damping pass.

Hyperparameter choices for the reconstruction accuracy experiment. We train SpringTime for 192 iterations, including 128 iterations in the first (stiffness) pass and 64 in the second (damping) pass. We treat each 1000-step-long motion clip as a sample, and we simulate with batch size of 32 clips in forward and backpropagation. Given that the training set contains 512 clips, training for 256 iterations in the stiffness pass is equivalent to training for 16 epochs. For the loss function, we use $\lambda_f = 1$, $\lambda_J = 10$. We notice that including the force loss is critical for narrowing the distribution of stiffness and damping estimates near ground-truth values.

Hyperparameter choices for the membrane locking experiment. To model actual bending and show that our bending springs can model the proper bending in a source FEM mesh, we included bending springs in the surrogate. Also, we included both diagonal springs to eliminate the possibility of introducing pseudo-stiffness in the misaligned case due to asymmetry. We trained MGN [40] for 3.8M steps over 48 hours. The number of training iterations we used is much lower than the 10M iterations Pfaff et al. [40] used to train MGN on the `flag_simple` example reported in their paper. We trained for fewer steps because (i) the low-resolution mesh is more coarse than the `flag_simple` example, as our mesh contains roughly only 0.5× vertices as the mesh they used; (ii) time is the constraining factor– training for 10M steps would require more than 126 hours on a server with NVIDIA V100 which is prohibitively long for our purpose. The excessively long training time can be partially attributed to lacking of parallelism in the training curriculum, e.g. simulations can only be run in batches of 1 rollout; but also can be attributed to not leveraging sufficient physical priors for inferring constitutive properties. Those being said, for a qualitative assessment to demonstrate that MGN can be vulnerable to membrane locking in training data, the results from training after 830k iterations should be sufficient, because we expect that after training for more iterations the results will be more visually close to the low-resolution source system's equilibrium configuration, and we do not see evidence of MGN learning to resist membrane locking throughout the course of training.

Hyperparameter choices for the generalization experiment in which we subject SpringTime and the baselines under novel initial and boundary conditions. The Neural ODE baseline contains 6 hidden layers, with 512 neurons in each hidden layer, and we use the exponential ReLU activation after each hidden layer. We train SpringTime for 256 iterations in the stiffness pass, and for 64 iterations in the damping pass, until loss converges. we train Neural ODE for 256 iterations, with the same loss function as used for SpringTime, on a single pass, and we have made sure that training loss had converged. We train MGN for 3.8 million iterations. But note that training of MGN works a bit differently: (i) we train on 512 rollouts with 8000-step long clips. The downsampling strategy introduced in Sec. A is applied here as well, so we get 200 frames of data per clip. This is on the same scale as 250 frames of data per clip used in training the `flag_dynamic` example in MGN Pfaff et al. [40]. (ii) training uses batch size of 1 clip. Turning to the loss function: we use the impulse loss exclusively, and we found that force loss is not particularly useful in this case. We believe this is because when the source cloth is constructed from triangular meshes, elastic and damping forces that act on each particle comes from edges which has no one-to-one correspondence to springs in the surrogate. Therefore learning per-spring elastic and damping force from the net force applied onto each particle becomes difficult. Impulse loss on the other hand is more robust to the change in geometric construction.

C Additional Results

C.1 Generalizability to Novel Initial and Boundary Conditions

Here we showcase more results from the experiment in which we assess the generalizability of different neural models to novel initial and boundary conditions.

Fig. 12 shows the architecture of the 512×6 MLP-based Neural ODE that we compare with SpringTime as a baseline. Without topological structure of the sheet explicitly injected into the architecture as a prior knowledge, the neural network needs to learn it by itself.

Fig. 13 shows the source and surrogate sheets reaching four different steady states after 8-second rollouts from different initial and boundary conditions. For each rollout, all surrogates start from the same condition as the source. Our method yields substantially lower steady-state errors than Neural ODE, whose prediction disintegrates after 8 seconds; MGN performs comparably to SpringTime but requires considerably more training time. Our method still has room for improvement: in the first row, the bottom-left cloth corner is poorly fitted, and in the third row the mass-spring cloth stretches farther downward than the ground truth.

C.2 Ablation Studies

Dual-pass vs single-pass training. As an alternative to the dual-pass training curriculum, we learned stiffness and damping constants jointly in a single pass for 128 iterations with batch size 32 on mass-spring cloth data. Compared with Fig. 7, stiffness RMSE increased from 14.3 to 22.8, and damping RMSE increased from 1.9 to 8.5. This is expected because elastic and damping forces are entangled, as discussed in Sec. 3.3.

Force-and-impulse loss vs position loss. Position-based loss directly penalizes step-wise Euclidean distance between target and predicted particle positions and is common in neural constitutive modelling [7, 31, 55]. We tried learning stiffness with position loss $||\mathbf{x} - \hat{\mathbf{x}}||^2$ on mass-spring cloth data, using $\lambda_{\text{k neg.}} = 100$ because position losses tend to be larger than force or impulse losses. Other hyperparameters match the force-and-impulse setting. With position loss, stiffness RMSE increased from 14.3 to 26.3. Force loss works well on mass-spring source cloth because it is highly sensitive to stepwise differences between predicted and target elastic and damping forces, whereas position loss integrates per-step errors over time and makes fine-scale parameter estimation harder.

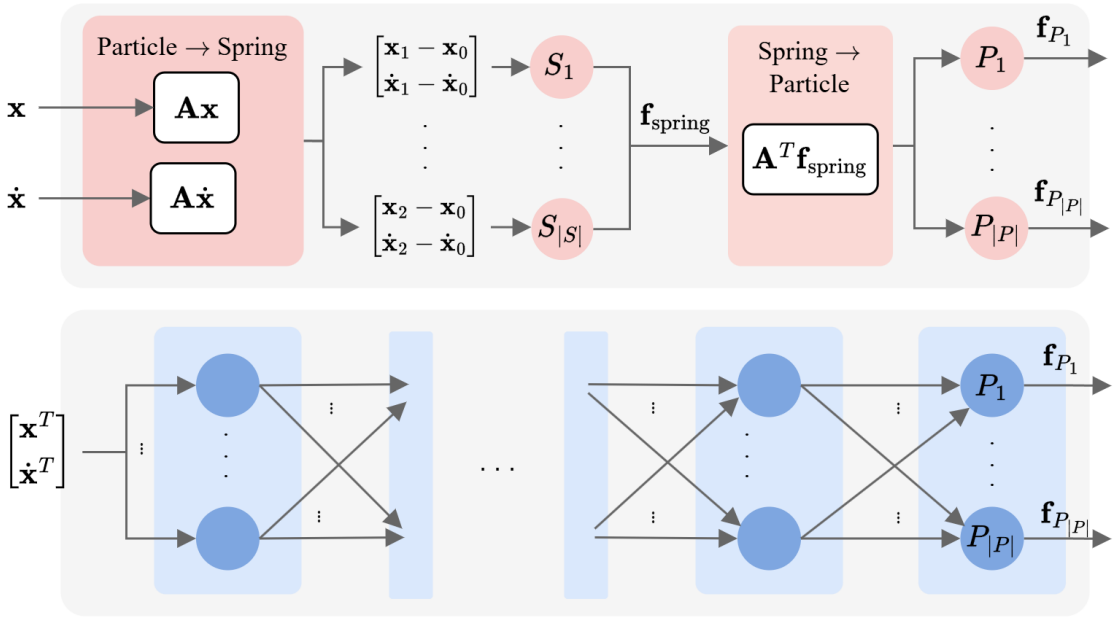

Figure 12: Architecture of SpringTime (top) and the MLP-based Neural ODE baseline (bottom). Neural ODE has no springs; its final-layer neurons directly predict constitutive forces acting on each surrogate particle at each time step. Hidden layers and connections are omitted for brevity.

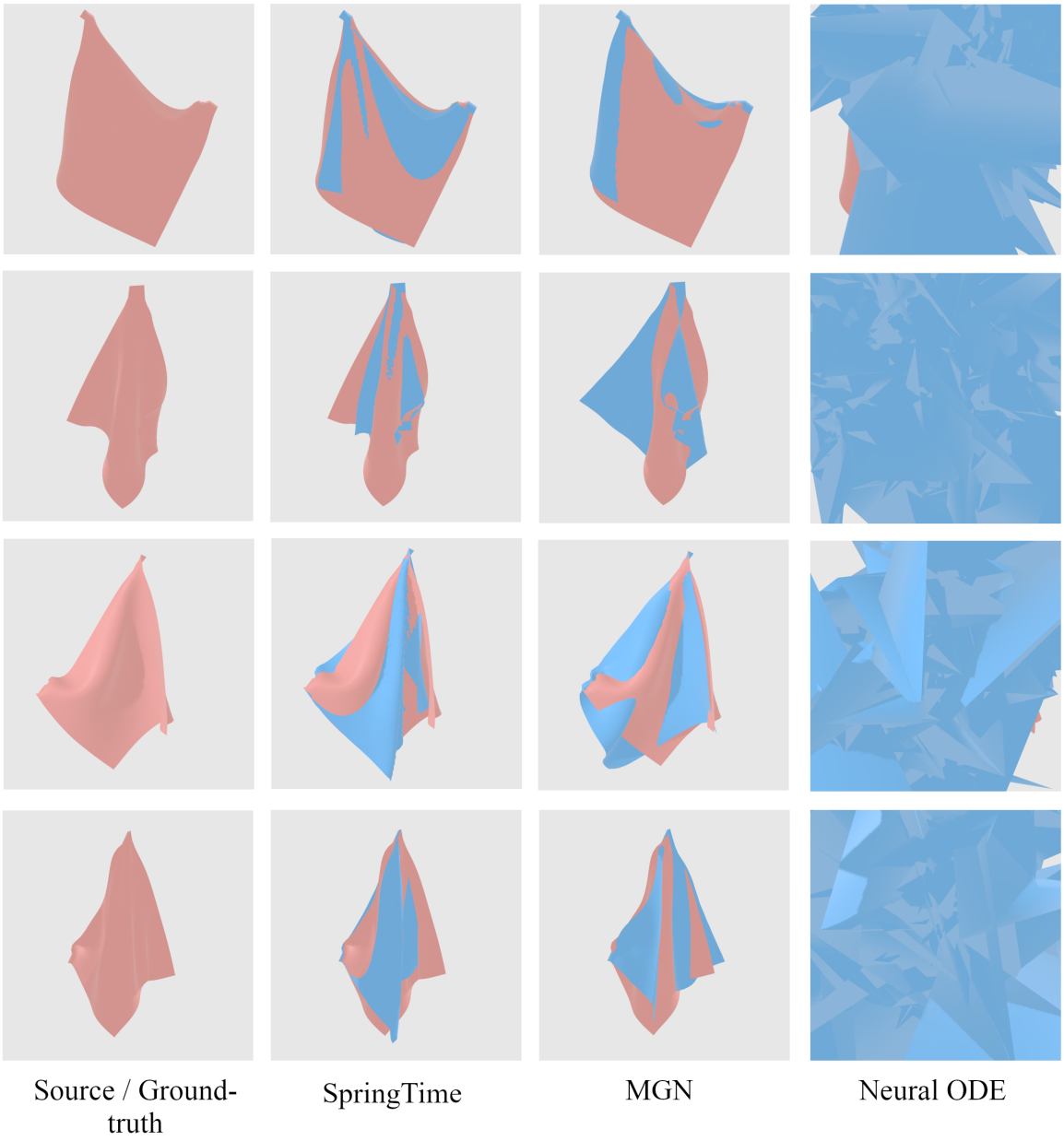

Figure 13: Generalization of SpringTime to novel initial and boundary conditions: steady states reached by the source cloth, SpringTime, MGN and Neural ODE-based surrogate after 8 seconds, in four different test rollouts. Ground-truth cloth is in red; surrogates are in blue.