NeuralClothSim: Neural Deformation Fields Meet the Thin Shell Theory

Navami Kairanda **Marc Habermann** **Christian Theobalt** **Vladislav Golyanik**

Max Planck Institute for Informatics, Saarland Informatics Campus

Abstract

Despite existing 3D cloth simulators producing realistic results, they predominantly operate on discrete surface representations (*e.g.*, points and meshes) with a fixed spatial resolution, which often leads to large memory consumption and resolution-dependent simulations. Moreover, back-propagating gradients through the existing solvers is difficult, and they hence cannot be easily integrated into modern neural architectures. In response, this paper re-thinks physically accurate cloth simulation: We propose NeuralClothSim, *i.e.,* a new quasistatic cloth simulator using thin shells, in which surface deformation is encoded in neural network weights in the form of a neural field. Our memory-efficient solver operates on a new continuous coordinate-based surface representation called neural deformation fields (NDFs); it supervises NDF equilibria with the laws of the non-linear Kirchhoff-Love shell theory with a non-linear anisotropic material model. NDFs are adaptive: They 1) allocate their capacity to the deformation details and 2) allow surface state queries at arbitrary spatial resolutions without re-training. We show how to train NeuralClothSim while imposing hard boundary conditions and demonstrate multiple applications, such as material interpolation and simulation editing. The experimental results highlight the effectiveness of our continuous neural formulation. See our project page: `https://4dqv.mpi-inf.mpg.de/NeuralClothSim/`.

1 Introduction

Realistic cloth simulation is a central, long-standing and challenging problem in computer graphics. It arises in game engines, computer animation, movie production, digital art, and garment digitisation, only to name a few areas. To date, it has been mostly addressed with physics-based simulators operating on explicit geometric representations, *i.e.,* meshes and particle systems. While recent simulators [24, 60, 36, 33, 32, 38] can produce realistic 3D simulations that obey various types of boundary conditions and consider secondary effects, but their operational principle remains limited in several ways. First, they work on discrete surface representations such as meshes and points inherently assuming a pre-defined spatial resolution that cannot be easily changed once the simulation is accomplished. Second, re-running with different meshing of the same initial template leads to different folds and wrinkles, which is often problematic for downstream applications. Third, explicit geometries require notoriously large amounts of storage for the detailed simulation: the memory size grows linearly with the number of points. Moreover, it is difficult to integrate simulators into learning frameworks and to edit the output 3D state without re-running the simulation.

The recent advances in physics-informed neural networks [49, 25] as well as the success of neural fields [43, 64, 62, 65], makes us question if continuous coordinate-based representations can alleviate these limitations. All these considerations motivate us to rethink the fundamentals of physically accurate cloth simulation and we introduce a new approach for cloth quasistatics, in which the surface deformation is encoded in neural network weights. The proposed neural architecture is coordinate-

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

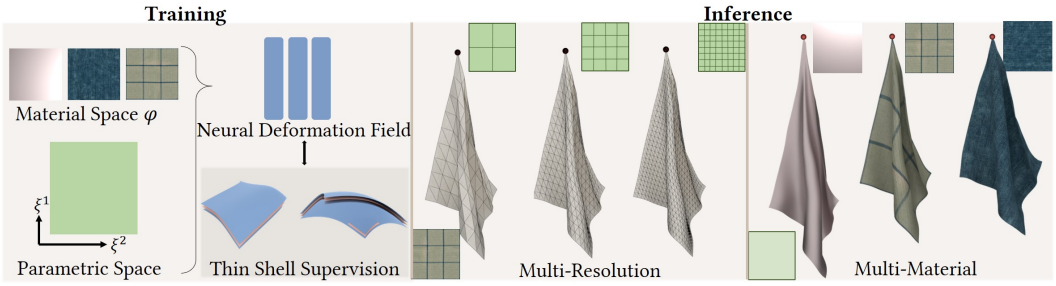

Figure 1: NeuralClothSim is the first neural cloth simulator representing surface deformation as a neural field. It is supervised for each target scenario with the laws of the Kirchhoff-Love thin shell theory with non-linear strain (left). Once trained, the simulation can be queried *continuously* and *consistently* enabling different spatial resolutions (center). NeuralClothSim can also incorporate learnt priors such as material properties that can be edited at test time (right).

based and has multiple advantages compared to previous simulators; see Fig. 1 for an overview. Our neural fields are adaptive, *i.e.,* the parameters are used to encode the deformations as they occur. As a matter of efficiency, we neither need to know the resolution in advance before the simulation nor do we require complex re-meshing schemes [45]. Realistic cloth simulation requires modelling geometric non-linearities and non-linear anisotropic elasticity. It involves large bending deformations and rigid transformations leading to non-linear point displacements. To efficiently model this, we rely on neural networks as they are good universal (non-linear) function approximators.

We model cloth simulation as a thin shell boundary-value problem with the deformation governed by the *Kirchhoff-Love shell theory*. In contrast to previous simulators using Kirchhoff-Love shell relying on isogeometric analysis [40] or subdivision surface algorithms [24, 20], we model thin shell deformations as implicit neural representations, *i.e.,* 3D deformation fields encoding cloth quasistatics. During training, our formulation supervises a neural deformation field (NDF), minimising the cloth's potential energy functional. In contrast to classical simulators [38, 36] sensitive to the finite element discretisations of the initial surface, which could lead to inconsistent folds, we generate simulations with consistent drapes, folds, and wrinkles. This is important for downstream applications that might query (*e.g.,* in the case of a renderer) or even modify (*e.g.,* like inverse methods) the simulation with adaptive sampling. Next, our representation is memory-efficient, and the simulation states are generated directly in a compressed form. In summary, our core technical contributions are as follows:

- A new continuous coordinate-based neural representation (Sec. 4.1)—and a new neural solver for cloth quasistatics based on thin shell theory that accepts boundary conditions such as external forces or guiding motions (Sec. 4.2).
- Modelling of thin shell's deformation with non-linear Kirchhoff-Love theory supervising the neural deformation fields (Sec. 4.3). Upon convergence, the equilibrium state can be queried continuously and consistently.
- Applications of the proposed neural simulator including material interpolation and fast editing of simulations according to updated simulation parameters (Sec. 5.4).

We want to point out that we do *not* claim qualitative superiority over classical cloth simulation methods and *completeness* of our formulation (*e.g.*, our method does not consider collisions). However, we believe that our new way of deeply integrating neural networks as a surface representation and solver into cloth simulation has the potential to stimulate future research in this direction, and we show that our formulation overcomes multiple fundamental limitations of existing discrete approaches.

2 Related Work

Cloth Simulation is a well-studied problem [1, 9, 21, 26, 59, 38, 34, 51], with the first methods dating back to the 1980s [2, 55]. The computational flow of the modern simulation approaches includes: Discretisation using the finite element method (FEM) [18, 45], implicit time-integration [1, 32], frictional contact [33, 42], and collision handling [46, 54, 26]. Cloth simulators model real fabric behaviors [15, 61], which is typically done by fitting constitutive material models. Liang et al. [38] and Li et al. [36] introduced differentiable cloth simulators, which were subsequently shown to be

also useful in 3D reconstruction as a physics-based prior [28, 37]. Zhang *et al*.'s approach [67, 68] enables interactive exploration of cloth parameters with progressively consistent quasistatics. Another category of methods constitutes *neural cloth simulators*. Pfaff et al. [48] proposed to learn simulations using graph neural networks. Bertiche et al. [7] is a neural simulator for static draping of garments on a virtual character. It is further extended with self-supervised approaches [51, 8] to learning garment dynamics. They leverage physics-based loss terms and do not require simulated ground-truth data. However, these methods are application-oriented rather than approaches for general cloth simulation, as the garments are skinned to the human body and garment deformations are driven by body shape and poses. Several methods for cloth simulation rely on the Kirchhoff-Love shell theory [19, 24]. The energy functionals in the theory require higher-order derivatives, which are not available for general unstructured triangle meshes. In their pioneering work, Cirak et al. [14] present Loop subdivision with control meshes that meet this additional C^1 interpolation requirement, which is extended to dynamic cloth simulation with corotational strains [56]. NURBS isogeometry [40] also enables continuity, whereas recent methods [16, 31] rely on Catmull-Clark subdivision surfaces and model the geometric non-linearity of shells. All the aforementioned cloth simulators (traditional FEM, neural, and Kirchhoff-Love) use discrete surface representation (*i.e*., meshes) with several inherent limitations. The representation is not adaptive, and simulations suffer from coarse-to-fine inconsistency and are sensitive to initial discretisation.

Neural Fields. Recent approaches parameterising surfaces as neural fields [62, 53, 44, 43, 58] offer a promising alternative to meshes. As a common theme, these methods use coordinate-based MLP for neural field parameterisation, which takes coordinates in the spatio-temporal domain and returns the task-specific property, *e.g.* occupancy or SDF values. For a detailed discussion, we refer to the survey of Xie et al. [64]. However, none of the works focus on integrating such neural fields into the cloth simulation, which is the main goal of the proposed work.

Neural Networks for Solving PDEs/ODEs. Several recent approaches [50, 49, 12, 66, 35], also dubbed Physics-Informed Neural Networks (PINNs), leverage neural networks for solving tasks that are supervised by the laws of physics; we refer to a recent survey from Hao et al. [25] for a detailed review. Chen *et al*. [11, 10] use implicit neural representation to accelerate [11] or replace [10] PDE solvers. However, they do not demonstrate thin-shell simulation. While previous works such as Rao *et al*. [50] and Zehnder *et al*. [66] applied neural implicit representations for volumetric elastodynamic problems, our approach focuses on realistic thin-shell and cloth simulation. It addresses important simulation aspects such as geometric non-linearities and the integration of non-linear anisotropic models that are crucial for simulating large deformations and rotations. Another method [65] allows the processing of neural fields encoding geometric structures. Conceptually, the most closely related to ours is the work of Bastek and Kochmann [5], however, there are important differences to our work. First, they model linear small-strain regime for Naghdi shells, whereas we model the full non-linear stretching and bending behaviour of clothes. Second, we propose several architectural improvements —periodic activation functions, periodic boundary conditions, data-driven orthotropic material model— that are necessary for producing realistic wrinkles and folds, and demonstrate generalisation to point loads, different material and boundary values. Next, we present a short background on Kirchhoff-Love theory that enables us to model a cloth deformation as a thin shell.

3 Kirchhoff-Love Thin Shell Theory for Cloth Modeling

Before we explain our method, we define our cloth representation. We characterise cloth as a thin shell and model its behaviour with the Kirchhoff-Love theory [39, 63]. A thin shell is a 3D geometry with a high ratio of width to thickness. The shell continuum can be kinematically described by the *midsurface* located in the middle of the thickness dimension and the *director*, a unit vector directed along fibres in the shell that are initially perpendicular to the midsurface. The Kirchhoff hypothesis states the director remains straight and normal, and the shell thickness $h \in \mathbb{R}$ does not change with deformation (see inset). We provide a detailed review of Kirchhoff-Love thin shell theory in App. B.

Notation. Throughout the document, we use Greek letters for indexing quantities on the midsurface, *e.g*., $\mathbf{a}_\alpha, \alpha, \beta, ... = 1, 2$, and Latin letters for indexing quantities on the shell, *e.g*., $\mathbf{g}_i, i, j, ... = 1, 2, 3$. Italic letters a, A indicate scalars, lower case bold letters $\mathbf{a}$ indicate first-order tensors (vectors), and upper case bold letters $\mathbf{A}$ indicate second-order tensors. An index can appear as a superscript or subscript. Superscripts $(\cdot)^i$ refer to contravariant components of a tensor, which scale inversely with the change of basis, whereas subscripts $(\cdot)_i$ refer to covariant components that change in the same

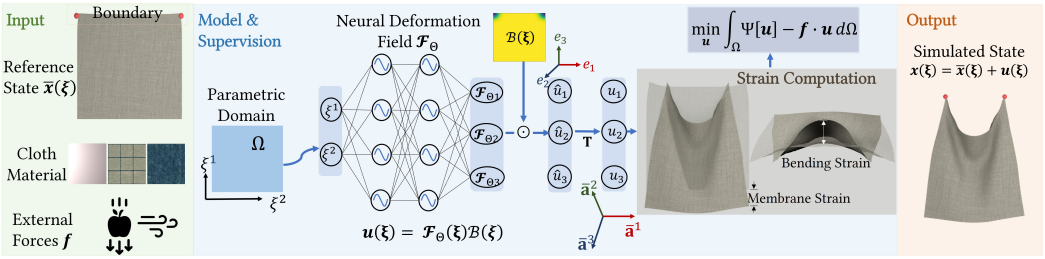

Figure 3: **NeuralClothSim** takes as input a thin shell in the reference state and its material properties, boundary motion and external forces. It then learns an NDF, *i.e.,* a coordinate-based implicit 3D deformation field. At inference, NDF can be *continuously* queried for the deformed state of the surface at equilibrium using curvilinear coordinates from the parametric domain. We use the Kirchhoff-Love thin shell modelling to supervise the cloth quasistatics with the potential energy functional.

way as the basis transforms. Moreover, we use upper dot notation for time derivatives, lower comma notation for partial derivatives with respect to the curvilinear coordinates, ξ^i, and vertical bar for covariant derivatives, *e.g.*, $\dot{\mathbf{u}} = \partial \mathbf{u}/\partial t$, $\mathbf{x}_{,\alpha} = \partial \mathbf{x}/\partial \xi^\alpha$, and $u_\alpha|_\beta$, respectively. Geometric quantities with overbar notation $\bar{(\cdot)}$ refer to the reference configuration. Additionally, Einstein summation convention of repeated indices is used for tensorial operations, *e.g.*, $\varphi_{\alpha\lambda}\varphi^\lambda_\beta = \varphi_{\alpha 1}\varphi^1_\beta + \varphi_{\alpha 2}\varphi^2_\beta$. A detailed list of notations can be found in Tab. II in Appendix B.

4 Method

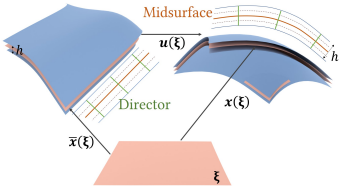

Figure 2: Kirchhoff-Love shell

We propose NeuralClothSim, *i.e.,* a new approach for continuous and consistent quasistatic cloth simulation relying the thin shell theory. We seek to generate a complex simulation state at equilibria given a cloth geometry in a reference configuration, its material properties and external forces. The physical basis for our cloth quasistatics is the nonlinear Kirchhoff-Love thin shell equations that model the stretching and bending of cloths in a unified manner. We parameterise the cloth states as a neural deformation field (NDF) defined over a continuous parametric domain (Sec. 4.1). We explicitly account for positional and periodic boundary conditions, incorporated as hard constraints (Sec. 4.2). NDF is optimised using a loss function based on the potential energy functional (Sec. 4.3). Fig. 3 provides a method overview.

4.1 Neural Deformation Field (NDF)

At the core of our approach is a *neural deformation field* (NDF), a continuous representation of cloth quasistatics, entirely parameterised by a neural network. Following Sec. 3, we model cloth geometry as a Kirchhoff-Love thin shell. Given the rest state $\bar{\mathbf{x}}(\boldsymbol{\xi})$ of a cloth, we describe the equilibrium state $\mathbf{x}(\boldsymbol{\xi})$ of its midsurface under the action of external forces $\mathbf{f}(\boldsymbol{\xi})$ and boundary constraints $\mathcal{B}_d(\boldsymbol{\xi})$ using

$$\mathbf{x}(\boldsymbol{\xi}) = \bar{\mathbf{x}}(\boldsymbol{\xi}) + \mathbf{u}(\boldsymbol{\xi}), \text{ with } \boldsymbol{\xi} := (\xi^1, \xi^2) \in \Omega. \tag{1}$$

The curvilinear coordinate space (ξ^1, ξ^2) can (but does not need to) naturally correspond to the orthotropic warp-weft structure of woven clothes. As examples, the reference state associated with a flat square cloth of side L in the xy-plane and that of a garment sleeve (radius R, length L) admitting a natural parameterisation with cylindrical coordinates are:

$$\begin{aligned} \bar{\mathbf{x}}(\boldsymbol{\xi}) &= [\xi^1, \xi^2, 0]^\top, \quad \forall (\xi^1, \xi^2) \in [0, L]^2, \\ \bar{\mathbf{x}}(\boldsymbol{\xi}) &= [R\cos\xi^1, \xi^2, R\sin\xi^1]^\top, \ \forall \xi^1 \in [0, 2\pi); \xi^2 \in [0, L]. \end{aligned} \tag{2}$$

Analytically defining surface parameterisations might not be feasible for reference geometries given as meshes. In such cases, we learn the reference parametrisation by fitting an MLP $\bar{\mathbf{x}}(\boldsymbol{\xi}; \Upsilon)$ with parameters Υ to the reference mesh. Specifically, we learn $\bar{\mathbf{x}}$ by supervising it with the ℓ_2-loss $\mathcal{L}(\Upsilon) = ||\bar{\mathbf{x}}(\hat{\boldsymbol{\xi}}; \Upsilon) - \hat{\bar{\mathbf{x}}}||_2^2$, where $\hat{\bar{\mathbf{x}}} \in \mathbb{R}^3, \hat{\boldsymbol{\xi}} \in \mathbb{R}^2$ are the vertices and texture coordinates of the given reference mesh. The advantage of this preprocessing over directly using the reference mesh is that we can continuously sample in the parametric domain by querying the MLP and compute all the geometric quantities at these points, similar to analytical access to the reference surface. Our key

idea is to regress the displacement field $\mathbf{u}(\boldsymbol{\xi})$ using an MLP $\mathcal{F}_\Theta : \Omega \rightarrow \mathbb{R}^3$ and optimise its weights Θ to minimise the total potential energy of the thin-shell cloth. Specifically, the NDF $\mathbf{u}$ is formulated as follows:

$$\mathbf{u}(\boldsymbol{\xi}; \Theta) = \mathcal{F}_\Theta(\mathcal{B}_p(\boldsymbol{\xi}))\mathcal{B}_d(\boldsymbol{\xi}), \tag{3}$$

where $\mathcal{B}_p(\boldsymbol{\xi})$ and $\mathcal{B}_d(\boldsymbol{\xi})$ are functions that respectively account for periodic and Dirichlet boundary conditions. In Sec. 4.2, we elaborate on encoding such conditions as hard constraints.

Apart from being parameter-differentiable, *i.e.,* the gradient $\nabla_\Theta \mathcal{F}_\Theta$ is defined everywhere, $\mathcal{F}_\Theta$ needs to be input-differentiable, *i.e.,* $\nabla^2_{\boldsymbol{\xi}} \mathcal{F}_\Theta$ must exist likewise, in order to compute the strains required for the Kirchhoff-Love energy functional. This restricts the activation function used in the network; only C^2-continuous non-linearities can be used. Therefore, we use periodic sine as the preferred activation function [53] as it can represent high-frequency signals (needed for folds and wrinkles) while allowing for computing higher-order derivatives. Note that unlike NDF $\mathbf{u}(\boldsymbol{\xi}; \Theta)$, we use GELU [27] activations for smoothly fitting the reference shape, $\bar{\mathbf{x}}(\boldsymbol{\xi}; \Upsilon)$. Sec. 4.3 describes the optimisation procedure to train the deformation field $\mathbf{u}(\boldsymbol{\xi}; \Theta)$.

Once trained, $\mathcal{F}_\Theta$ provides *continuous* access to the cloth quasistatics, where the network can be queried at any point in the spatial domain Ω. Based on the requirement for downstream applications, parametric input samples during inference can be *different* and their number can be *higher* than those during training, since it does not require the expensive computations of physical quantities; see Fig. 1. Thanks to our continuous formulation, at inference, different discretised meshing and texturing operations in the parametric domain Ω can be lifted from 2D to 3D using $\mathbf{u}(\boldsymbol{\xi}; \Theta)$, which will lead to consistent result irrespective of the specific discretisation (see also Fig. 6).

4.2 Boundary Conditions

A practical cloth simulator allows for imposing conditions such as a user-specified corner motion; for most garments, the simulation needs to be continuous and consistent along the seams. We seek to strictly enforce these conditions in our method. We achieve this by formulating boundary conditions as spatial distance functions, and seams as periodicity constraints along a curvilinear coordinate (such as the azimuthal angle of a cylindrically parameterised sleeve), and directly apply them to the NDF in Eq. (3).

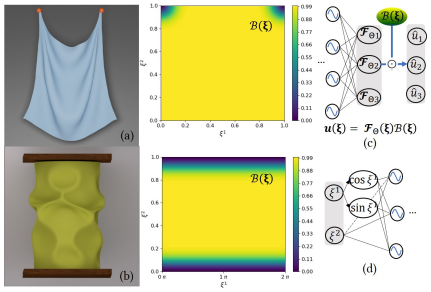

Figure 4: **Boundary conditions.** In contrast to Dirichlet conditions that alter the network output (c), we impose periodic boundaries by remapping the network input to its sine and cosine values (d).

Dirichlet Boundary Conditions. To constrain boundary positions, we require $\mathbf{u}(\boldsymbol{\xi}_{\partial\Omega}) = \mathbf{0}$ for some specified list of parameter space points $\boldsymbol{\xi}_{\partial\Omega}$ along the boundary segment $\partial\Omega$. While we elaborate on the simpler case here, it is also possible to specify complex conditions *i.e.,* $\mathbf{u}(\boldsymbol{\xi}_{\partial\Omega}) = \mathbf{b}(\boldsymbol{\xi}_{\partial\Omega})$, detailed in Appendix D. One solution is to sample points in the boundary segment and enforce the boundary conditions through separate loss terms. As shown in previous physics-informed neural networks [25], having competing objectives during training can lead to unbalanced gradients, which causes the network to often struggle with accurately learning the underlying solution. Further, there is no guarantee that the boundary conditions will always be enforced. Therefore, we propose to modify the NDF to embed *essential* boundary conditions as hard constraints [41]. Specifically, a distance function $\mathcal{B}_d(\boldsymbol{\xi})$ satisfying $\mathcal{B}_d(\boldsymbol{\xi}) = \begin{cases} 0, \text{ if } \boldsymbol{\xi} \in \partial\Omega \\ >0, \text{ otherwise if } \boldsymbol{\xi} \in \Omega \end{cases}$ ensures that any instance of deformation field $\mathbf{u}(\boldsymbol{\xi}, t; \Theta)$ automatically satisfies the boundary conditions. We set

$$\mathcal{B}_d(\xi^1, \xi^2) := 1 - e^{-((\xi^1 - \xi^1_{\partial\Omega})^2 + (\xi^2 - \xi^2_{\partial\Omega})^2)/\sigma} \text{ s.t. } (\xi^1_{\partial\Omega}, \xi^2_{\partial\Omega}) \in \partial\Omega, \quad \forall (\xi^1, \xi^2) \in \Omega \tag{4}$$

as a distance function with small support $\sigma = 0.01$. Fig. 4 provides an illustrative example.

The above formulation supports point and shape constraints in the cloth interior, *i.e.*, $\partial\Omega$ can likewise be a boundary segment inside the domain (Fig. IV-appendix). Moreover, if the initial geometry is provided as a mesh (instead of an analytical definition), point constraints can be directly provided as mesh vertices, with $(\xi^1_{\partial\Omega}, \xi^2_{\partial\Omega})$ corresponding to texture coordinates of the vertex; see Fig. 6-(right).

Periodic Boundary Conditions. In contrast to the positional or motion-dependent boundary conditions specified as per the user's desires, additional boundary conditions can arise from the geometric

cloth parametrisation. Points along the panel seams of the garment share the world-space position and velocity, though they are mapped to different values in the parametric domain. We express continuity in geometry and simulation using periodic conditions. Consider any simulation involving a sleeve: Our method needs to guarantee the additional condition due to the parametrisation, *i.e.,* $\mathbf{u}(\xi^1, \xi^2) = \mathbf{u}(\xi^1 \pm 2n\pi, \xi^2)$. Whereas the Dirichlet condition is imposed by altering the network output, we *strictly* impose periodic boundaries by modifying its input. Recall that any continuous periodic function can be written using its Fourier series. If $\mathbf{u}(\boldsymbol{\xi})$ is a periodic deformation field with period P w.r.t. the input coordinate ξ^λ, $\mathbf{u}(\boldsymbol{\xi})$ can be decomposed into a weighted sum $\{1, \sin(2n\pi\xi^\lambda/P), \cos(2n\pi\xi^\lambda/P)\}, n \in \mathbb{N}$. Due to the universal approximation power of MLP, only the first cosine and sine terms need to be considered, as the others can be expressed as the nonlinear continuous functions of $\cos(2\pi\xi^\lambda/P)$ and $\sin(2\pi\xi^\lambda/P)$ [41]. Hence, we map ξ^λ using $\xi^\lambda \mapsto \{\cos\xi^\lambda, \sin\xi^\lambda\}$ when feeding it to the MLP, enforcing periodicity of the predicted NDF along ξ^λ. This completes the definition of boundary conditions applied during both training and inference.

4.3 NDF Optimisation

We next explain optimisation in NDF learning. Note $\boldsymbol{\xi}$ for $\mathbf{u}(\boldsymbol{\xi})$ and derived quantities are dropped.

Strain Computation. To compute the geometric strains due to the thin shell deformation, we evaluate the NDF on samples from the curvilinear coordinate space Ω. We generate N_Ω points using a stratified sampling approach. This ensures that the samples are random, yet well-distributed. At each training iteration, we re-sample coordinates to learn an NDF that fully explores the continuous domain over the course of the optimisation. We evaluate NDF $\mathbf{u}(\boldsymbol{\xi})$ at all samples using Eq. 3 and this prediction (*i.e.*, $\hat{u}_i$) is assumed to be in the Cartesian coordinate system, *i.e.,* $\mathbf{u} = \hat{u}_i \mathbf{e_i}$. Our further strain computations (Eq. 6) require covariant deformation components in the reference contravariant basis, *i.e.,* $\mathbf{u} = u_\alpha \bar{\mathbf{a}}^\alpha + u_3 \bar{\mathbf{a}}^3$, therefore we use the basis transformation matrix $\mathbf{T} = [\bar{\mathbf{a}}^1\ \bar{\mathbf{a}}^2\ \bar{\mathbf{a}}^3]^{-1}$ for converting from Cartesian deformation coordinates to covariant coordinates (see Appendix B for detailed Kirchhoff-Love preliminaries). While it is possible to predict in the local contravariant basis directly, the global basis is better suited for NDF training since the local basis vectors are not normalised, and the basis varies with the input position $\boldsymbol{\xi}$, especially noticeable for reference geometries such as sleeve (Fig. I-(b)-appendix).

Next, we describe the ingredients required to evaluate the internal strain energy Ψ. Membrane strain $\boldsymbol{\varepsilon} = [\varepsilon_{\alpha\beta}]$ and bending strain $\boldsymbol{\kappa} = [\kappa_{\alpha\beta}]$ measure the in-plane stretching and the curvature change, respectively, and are defined as $\varepsilon_{\alpha\beta} := \frac{1}{2}(a_{\alpha\beta} - \bar{a}_{\alpha\beta})$, and $\kappa_{\alpha\beta} := \bar{b}_{\alpha\beta} - b_{\alpha\beta}$ where $(\bar{a}_{\alpha\beta}, a_{\alpha\beta})$, and $(\bar{b}_{\alpha\beta}, b_{\alpha\beta})$ are the metric and curvature tensors of reference and deformed midsurface. With the assumptions of Kirchhoff-Love theory and following [3], we simplify these equations to directly operate on $\mathbf{u}$ and evaluate strains as

$$\begin{aligned} \varepsilon_{\alpha\beta} &= \frac{1}{2}(\varphi_{\alpha\beta} + \varphi_{\beta\alpha} + \varphi_{\alpha\lambda}\varphi^\lambda_\beta + \varphi_{\alpha 3}\varphi_{\beta 3}), \\ \kappa_{\alpha\beta} &= -\varphi_{\alpha 3}|_\beta - \bar{b}^\lambda_\beta \varphi_{\alpha\lambda} + \varphi^\lambda_3(\varphi_{\alpha\lambda}|_\beta + \frac{1}{2}\bar{b}_{\alpha\beta}\varphi_{\lambda 3} - \bar{b}_{\beta\lambda}\varphi_{\alpha 3}), \end{aligned} \tag{5}$$

where the deformation gradients $\varphi_{\alpha\lambda}, \varphi_{\alpha 3}$ are the components of $\mathbf{u}_{,\alpha}$ such that

$$\mathbf{u}_{,\alpha} = \varphi_{\alpha\lambda}\bar{\mathbf{a}}^\lambda + \varphi_{\alpha 3}\bar{\mathbf{a}}^3, \varphi_{\alpha\lambda} := u_\lambda|_\alpha - \bar{b}_{\alpha\lambda}u_3, \text{ and } \varphi_{\alpha 3} := u_{3,\alpha} + \bar{b}^\lambda_\alpha u_\lambda. \tag{6}$$

We do not linearise the strain. Orange and teal correspond to the linear and the non-linear components, respectively. To evaluate the derivatives of geometric quantities based on NDF $\mathbf{u}$ w.r.t. inputs $\boldsymbol{\xi}$ (as part of strain computation), we use automatic differentiation of machine learning frameworks [47].

Cloth Material Model. A thin shell develops an internal potential energy due to deformation and the material's hyperelasticity. As in the cloth simulation literature [34, 67], we write the internal hyperelastic energy density as a function of the stretching and bending strains, $\Psi(\boldsymbol{\varepsilon}, \boldsymbol{\kappa}, \xi^3; z(\bar{\mathbf{x}}), \boldsymbol{\Phi}, h)$. Here, $\boldsymbol{\Phi}$ are the cloth's material parameters and $\xi^3 \in [-\frac{h}{2}, \frac{h}{2}]$ is the thickness coordinate, and $z(\bar{\mathbf{x}})$ are geometric quantities derived from the reference midsurface $\bar{\mathbf{x}}$. Our neural field-based cloth simulation is orthogonal to the research on material modelling and can, thus, be formulated with many different elastic models, as long as the elasticity can be represented as an energy density function. For example, a linear isotropic [52] stress-strain relationship leads to strain energy of the form $\Psi = \frac{1}{2}(DH^{\alpha\beta\lambda\delta}\varepsilon_{\alpha\beta}\varepsilon_{\lambda\delta} + BH^{\alpha\beta\lambda\delta}\kappa_{\alpha\beta}\kappa_{\lambda\delta})$, where D is the in-plane stiffness and B is the bending stiffness computed as $D := \frac{Eh}{1-\nu^2}$ and $B := \frac{Eh^3}{12(1-\nu^2)}$, with Young's modulus E, Poisson's

Table 1: **Quantitative evaluation.** We validate the displacements obtained with our method on the Belytschko obstacle course with analytical solutions from [6, 57]. Guo et al. [23] use different material and match the corresponding reference result. Below, we show the ablation. We highlight that our method outperforms prior works and baselines by a large margin.

Method	Square plate	Scordelis-Lo roof	Rigid-end cylinder	Free-end cylinder
Analytical	0.487	0.3024	1.825e−5	4.52e−4
Guo *et al.*	2.566*	n/a	n/a	n/a
Bastek *et al.*	n/a	0.297	n/a	n/a
Ours, full	0.487	0.3018	1.81e−5	4.58e−4
Ours, no periodicity	n/a	n/a	3.6e−9	3.13e−6
Ours, GELU	0.496	0.288	1.74e−5	5.7e−4

ratio ν, and $H^{\alpha\beta\lambda\delta} := \nu\bar{a}^{\alpha\beta}\bar{a}^{\lambda\delta} + \frac{1}{2}(1-\nu)(\bar{a}^{\alpha\lambda}\bar{a}^{\beta\delta} + \bar{a}^{\alpha\delta}\bar{a}^{\beta\lambda})$ with $\bar{a}^{\alpha\beta}$ being the contravariant metric tensors. Alternatively, we support the data-driven non-linear anisotropic material model of Clyde *et al.* [15] that has been carefully constructed to fit measured woven fabrics. We refer to Appendix B.3 for the mathematical details of the non-linear model.

Energy Optimisation. A thin shell's stable equilibrium is characterised by the principle of minimum potential energy, i.e. the sum of external potential energy owing to forces $\mathbf{f}$ and internal potential energy Ψ due to material elasticity. The total potential energy $\mathcal{E}$ reads as $\mathcal{E}[\mathbf{u}] = \int_\Omega \Psi \, d\Omega - \int_\Omega \mathbf{f}\cdot\mathbf{u}\, d\Omega$, and the stable equilibrium deformation $\mathbf{u}^*$ can be found by minimising the energy functional subject to boundary constraints $\mathbf{u}(\xi^1,\xi^2) = \mathbf{b}(\xi^1,\xi^2)$ on $\partial\Omega$. We take advantage of the variational structure of $\mathcal{E}[\mathbf{u}]$ and minimise it directly with gradient descent. All operations of our energy computation are naturally differentiable, and we estimate the integral as a sum over continuous parametric domain. For linear isotropic materials, we arrive at the following loss function to optimise the MLP weights for a physically-principled cloth simulation encoded as $\mathbf{u}^*(\boldsymbol{\xi};\Theta)$:

$$\mathcal{L}(\Theta) = \frac{|\Omega|}{N_\Omega}\sum_{i=1}^{N_\Omega}\left(\tfrac{1}{2}D\boldsymbol{\varepsilon}^\top(\boldsymbol{\xi}_i;\Theta)\mathbf{H}(\boldsymbol{\xi}_i)\boldsymbol{\varepsilon}(\boldsymbol{\xi}_i;\Theta) + \tfrac{1}{2}B\boldsymbol{\kappa}^\top(\boldsymbol{\xi}_i;\Theta)\mathbf{H}(\boldsymbol{\xi}_i)\boldsymbol{\kappa}(\boldsymbol{\xi}_i;\Theta) - \mathbf{f}^\top(\boldsymbol{\xi}_i)\mathbf{u}(\boldsymbol{\xi}_i;\Theta)\right)\sqrt{\bar{a}(\boldsymbol{\xi}_i)}, \quad (7)$$

where $\boldsymbol{\varepsilon}(\boldsymbol{\xi};\Theta) \in \mathbb{R}^4$, $\boldsymbol{\kappa}(\boldsymbol{\xi};\Theta) \in \mathbb{R}^4$ are vectorised strains computed using (5); $|\Omega| = \int_\Omega d\xi^1 d\xi^2$ is the area of the parametric domain; $\mathbf{H}(\boldsymbol{\xi}) \in \mathbb{R}^{4\times4}$ depends only on the reference surface. For data-driven materials [15], the strain energy is additionally a function of thickness coordinate ξ^3. Hence, we integrate $\mathcal{E}$ along the thickness with the Simpson's 3-point rule (similar to [16]) *i.e.*, $\mathcal{E}[\mathbf{u}] = \int_\Omega \int_{-\frac{h}{2}}^{\frac{h}{2}} \Psi \, d\xi^3 \, d\Omega - \int_\Omega \int_{-\frac{h}{2}}^{\frac{h}{2}} \mathbf{f}\cdot\tilde{\mathbf{u}}\, d\xi^3\, d\Omega$, where $\tilde{\mathbf{u}} = \mathbf{u} + \xi^3\mathbf{w}$ is the deformation for a point on the shell continuum and $\mathbf{w}$ quantifies the change in the midsurface orientation (see Appendix B.2).

5 Experimental Evaluation

We next present the qualitative and empirical results highlighting the new characteristics of our continuous neural fields, including validation (Sec. 5.1), simulation results (Sec. 5.2), comparison to prior works (Sec. 5.3), and applications (Sec. 5.4).

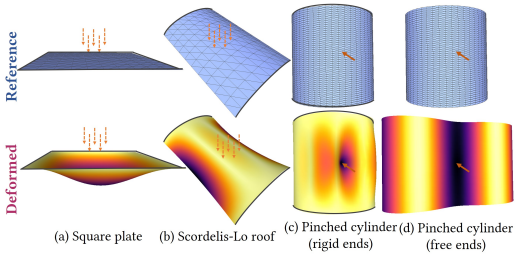

Figure 5: **Belytschko obstacle course** for which we generate accurate displacements (rescaled for better visualisation).

5.1 Obstacle Course

A scrupulously modelled thin shell, and consequently cloth, must be able to handle inextensional bending modes, complex membrane states of stress, and rigid body motion without straining. Therefore, for validation, we use the engineering obstacle course of benchmark problems from Belytschko et al. [6], for which the analytical solutions are known for linearised functionals. Such problems were previously used in computer graphics [22] for testing the performance of finite mesh elements. Specifically, we test our method on the square plate [57], the Scordelis-Lo roof, and the pinched cylinder with rigid diaphragms and free ends examples, for which the original and our deformed shells are shown in Fig. 5. See Tab. 1 for converged numerical results. The results, which show that our method outperforms prior works by a significant margin, demonstrate our method's excellent modelling ability. We further present details

of the experiments with the square plate and pinched cylinder, including the experimental setup and visualisations of the full displacement fields, in App. C.

Scordelis-Lo Roof is a non-flat reference shape subject to complex membrane strains, i.e. an open cylindrical shell with radius $R = 25\,\mathrm{m}$, length $L = 50\,\mathrm{m}$ and subtends an angle of $80°$. It is supported with two rigid diaphragms at the ends and loaded by gravity $\mathbf{f} = [0, -90, 0]^\top$. The shell's material is given as $E = 4.32\mathrm{e}8\,\mathrm{Pa}, \nu = 0$ and thickness $h = 0.25\,\mathrm{m}$. We obtain the maximum vertical displacement u_2 at the centre of the edge (averaged over the two sides) as 0.3018, closely approximating the analytical $u_2 = 0.3024$ [6].

5.2 Qualitative Results

We next present our simulation results. The experiments are performed with the values $E = 5000\,\mathrm{Pa}, \nu = 0.25, h = 0.0012\,\mathrm{m}$ for the linear isotropic material, and with parameters from Clyde et al. [15] for the nonlinear orthotropic material. For the supplemental video, we extend the method to visualise the deformation trajectory (*i.e.*, transition from the reference to the equilibrium state). Details on boundary conditions, external forces, and time-stepping can be found in Appendix D.

Napkin. We first consider a square napkin of length $L = 1\,\mathrm{m}$, falling freely under the effect of gravitational force. The napkin has a flat reference state in the *xy*-plane given by (2), and the gravitational force field is applied along the negative *y*-axis, *i.e.,* external force density $\mathbf{f} = [0, -9.8\rho, 0]^\top$. We specify a fixed boundary condition at the top left corner to constrain the napkin movement. The meshes extracted from the trained NDF are visualised in Fig. 1-(center). Note that apart from its realism, one can also query the simulation at arbitrary resolution in the case of NeuralClothSim. Next, we perform another experiment with a napkin subject to gravity and dynamic boundary condition, *i.e.,* in which the corners move inwards. This leads to fold formation at the top, as visualised in Fig. 4-(a) and in Fig. II for varying fabrics such as cotton and silk.

Sleeve. We also consider a cylindrical shell and perform sleeve compression and twisting. In both cases, we consider the reference state (2) with $L = 1\,\mathrm{m}$ and $R = 0.25\,\mathrm{m}$. See Figs. 4-(b) and 7-(b) for visualisations. In the first case, we apply torsional motion on the sleeve, *i.e.,* a total rotation of $\frac{3\pi}{4}$ around the *y*-axis to both the top and bottom rims. The optimised NDF forms wrinkles at the centre as expected [24]. In the second case, we compress the sleeve to produce the characteristic buckling effect. There are no external forces here and the compression is entirely specified by boundary conditions. We achieve a total displacement of $0.2\,\mathrm{m}$ due to compression with the inward motion of the top and bottom rims along the cylinder axis; see Fig. 4-(right). The demonstrated simulation is a representative example of strain localisation, with noticeable diamond patterns of shell buckling.

5.3 Comparisons to Previous Methods

In this section, we compare NeuralClothSim to state-of-the-art FEM cloth simulators and physics-informed neural networks for shell structures. We do not compare to other neural simulators [8, 51], as they do not support simulating non-garment cloths, whereas ours is a general neural cloth simulator.

Cloth Simulators. Next, we validate the consistency of cloth simulations at different discretisations of the reference state. We consider two scenarios: 1) A napkin with a fixed corner under gravity simulated with our approach and DiffARCSim [38] (Fig. 6-left), and 2) a flag with two fixed corners deforming under wind and gravity simulated with our approach and DiffCloth [36] (Fig. 6-right). In both scenarios, we simulated ours and compared methods thrice, starting with a marginally perturbed meshing of the same initial geometry resulting in different mesh discretisations. In the case of NeuralClothSim, we learn the reference parameterisation by fitting an MLP $\bar{\mathbf{x}}(\boldsymbol{\xi}; \Upsilon)$ for each initial discretisation followed by NDF optimisation. We find that the simulated meshes extracted from NDF are consistent for all discretisations. In contrast, for competing FEM-based methods, simulation is sensitive to the discretisation; while multiple simulations with the same initial mesh produce identical results, slightly different meshing generates inconsistent simulations. Theoretically, a well-defined FEM-based cloth solver should lead to consistent simulation results under different discretisation at high mesh resolutions. To investigate this, we perform an additional experiment where we increase the resolution of the ARCSim simulation ($10k$ vertices) so that the computation time roughly matches ours. However, the results still contain noticeable inconsistencies (Fig. XIV-appendix), possibly due to several operations that are highly discretisation-dependent [61] (such as the bending model relying on the dihedral angles). In contrast, our method leads to consistent results already at much

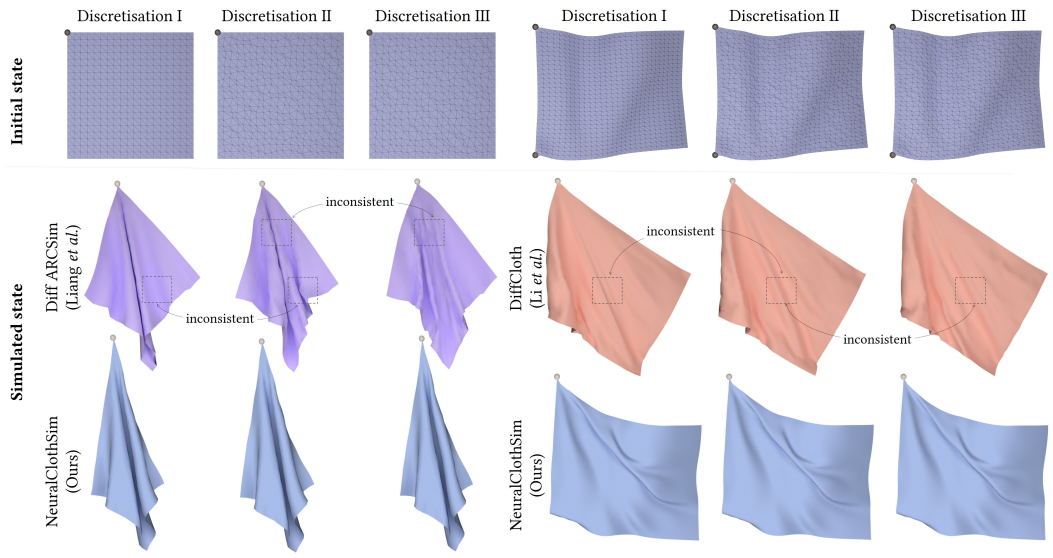

Figure 6: **Simulation consistency.** At different initial state discretisations, FEM-based simulators lead to inconsisies with often differences in the folds or wrinkles. In contrast, ours overfits an MLP to the reference mesh and encodes the surface evolution using another MLP (*continuous* neural fields). lead to inconsistent results with often occurring differences in the folds or wrinkles.

coarser discretisations (400 vertices, Fig. 6).We next evaluate the memory efficiency for simulations generated by NeuralClothSim, DiffARCSim and DiffCloth. The simulations are chosen to be of similar complexity, and qualitative results are visualised in Fig. XIII-appendix. In Fig. XV-appendix, we then plot the memory requirement as a function of spatial resolution. Memory is recorded for the simulated mesh states for the compared methods and weights of the NDF network for ours. While the memory requirement of finite-element-based methods grows linearly as the function of the number of vertices in the simulated cloth, our approach requires a constant and comparably small memory volume to store the quasistatic simulations. For better memory efficiency, existing simulators offer adaptive refinement (such as ARCSim [45]) by re-meshing at each time step (coarse triangulation is used at smooth regions, and fine meshes are used for wrinkles). However, this requires additional computation and loss of important characteristics, such as differentiability. In contrast, our approach is adaptive without the overhead and without losing correspondence and differentiability due to re-meshing.

PINNs for Shells. Bastek *et al.* [5] focus on engineering scenarios, and we compare the solutions to the Scordelis-Lo roof and a square plate in Tab. 1; both ours and theirs closely match the reference solutions. Bastek *et al.* note instabilities during training a neural network trained on a point load, therefore, define the Gaussian force kernel in their pinched hemisphere example. In contrast to theirs, we propose a new loss function for point loads (see App. C) addressing the pinched cylinder obstacle course. While Bastek *et al.* show converged results on engineering examples, their method cannot capture the high-frequency signal (folds and wrinkles) required for cloth simulation; see Fig. 7 for an illustrative example. The main reasons for their failure are 1) the linear strain and 2) that their activation function (GELU) can capture only smooth signals.

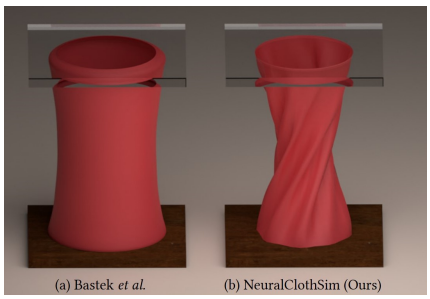

Figure 7: **Comparison to Bastek *et al.*** on sleeve twisting. While the cylinder in (a) twists without wrinkles, our result (b) is correctly wrinkled, similar to [24].

5.4 Ablation and Applications

Ablation. We evaluate the following ablated versions of our approach: 1) Contravariant coordinate system for NDF components, 2) Using a linear approximation of the strains instead of our model, 3)

Variants of boundary constraint imposition and 4) Choice of activation functions for NDF. For the latter two, we present the numerical results in Tab. 1. See Appendix E for further details.

Material-conditioning. NeuralClothSim can incorporate learnt priors: Our NDF can be directly extended by making it dependent on the material properties, *i.e.,* it can accept the material parameters as an extra input. This is possible since the material parameter space is typically low-dimensional. Once such NeuralClothSim modification is trained, we can edit the simulated geometry at test time, as shown in Fig. 1-(bottom right). We provide implementation details of conditioning in Appendix F.

Simulation editing. For high-dimensional scene parameters such as reference pose and external forces, we can edit simulations: The user can interrupt the training of NDF at any point, change the parameters and continue the training. Moreover, editing can also be done after full convergence (aka pre-training) and then fine-tuned with gradually modified design parameters. Editing an NDF has multiple advantages over NDF training from scratch: It is computationally and memory efficient, and provides access to interpolated simulations. We provide further details and results in Appendix F.

6 Discussion and Conclusion

NeuralClothSim closely matches reference values in challenging cloth deformation scenarios (e.g. the Belytschko course), thanks to compact NDF representation governed by the non-linear Kirchhoff-Love shell theory with (non-)linear orthotropic material. An extended NDF allows test-time interpolation of material properties and simulation editing. In contrast to the previous mesh-based simulators, NeuralClothSim enables querying continuous and consistent equilibrium cloth states. The shown results are physically plausible in different scenarios under time-varying external forces and boundary motions. We also see multiple avenues for future research, such as adding dynamic effects, *i.e*. inertia and damping. Our simulator currently does not support contacts and friction necessary for many applications beyond what is demonstrated here (*cf.* Appendix J on this standalone research problem).

In conclusion, we see NeuralClothSim as an exciting step towards neural-field-based continuous and differentiable cloth simulation. Inverse problems in vision could benefit from its multi-resolution consistency. While there is a long way until other functionalities such as collision handling are unlocked, we believe it can pave the way towards a new generation of physics simulation engines.

References

[1] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Annual Conference on Computer Graphics and Interactive Techniques*, 1998.

[2] Alan H. Barr. Global and local deformations of solid primitives. In *Annual Conference on Computer Graphics and Interactive Techniques*, 1984.

[3] Yavuz Basar and Wilfried B Krätzig. *Mechanik der Flächentragwerke: theorie, berechnungsmethoden, anwendungsbeispiele*. Springer-Verlag, 2013.

[4] Y Başar, M Itskov, and A Eckstein. Composite laminates: nonlinear interlaminar stress analysis by multi-layer shell elements. *Computer Methods in Applied Mechanics and Engineering*, 2000.

[5] Jan-Hendrik Bastek and Dennis M Kochmann. Physics-informed neural networks for shell structures. *European Journal of Mechanics-A/Solids*, 2023.

[6] Ted Belytschko, Henryk Stolarski, Wing Kam Liu, Nicholas Carpenter, and Jame SJ Ong. Stress projection for membrane and shear locking in shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 1985.

[7] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Pbns: physically based neural simulation for unsupervised garment pose space deformation. *ACM Transactions on Graphics (TOG)*, 2021.

[8] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Neural cloth simulation. *ACM Transactions on Graphics (TOG)*, 2022.

[9] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics*, 2002.

[10] Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit neural spatial representations for time-dependent pdes. In *International Conference on Machine Learning (ICML)*, 2023.

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

[12] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[13] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses*. 2005.

[14] Fehmi Cirak, Michael Ortiz, and Peter Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg.*, 2000.

[15] David Clyde, Joseph Teran, and Rasmus Tamstorf. Modeling and data-driven parameter estimation for woven fabrics. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, 2017.

[16] David Clyde, Joseph Teran, and Rasmus Tamstorf. Simulation of nonlinear kirchhoff-love thin shells using subdivision finite elements. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, 2017.

[17] David Corwin Clyde. *Numerical Subdivision Surfaces for Simulation and Data Driven Modeling of Woven Cloth*. University of California, Los Angeles, 2017.

[18] Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A fast finite element solution for cloth modelling. In *Proc. of The Pacific Conference on Computer Graphics and Applications*, 2003.

[19] Seth Green, George Turkiyyah, and Duane Storti. Subdivision-based multilevel methods for large scale engineering simulation of thin shells. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, 2002.

[20] Eitan Grinspun, Petr Krysl, and Peter Schröder. Charms: A simple framework for adaptive simulation. *ACM Transactions on Graphics*, 2002.

[21] Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, 2003.

[22] Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. Computing discrete shape operators on general meshes. In *Computer Graphics Forum*, 2006.

[23] Hongwei Guo, Xiaoying Zhuang, and Timon Rabczuk. A deep collocation method for the bending analysis of kirchhoff plate. *arXiv preprint arXiv:2102.02617*, 2021.

[24] Qi Guo, Xuchen Han, Chuyuan Fu, Theodore Gast, Rasmus Tamstorf, and Joseph Teran. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics*, 2018.

[25] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

[26] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. In *ACM SIGGRAPH 2008 papers*. 2008.

[27] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[28] Navami Kairanda, Edith Tretschk, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik. f-sft: Shape-from-template with a physics-based deformation model. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[29] Josef Kiendl, Ming-Chen Hsu, Michael CH Wu, and Alessandro Reali. Isogeometric kirchhoff–love shell formulations for general hyperelastic materials. *Computer Methods in Applied Mechanics and Engineering*, 2015.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Alena Kopanicakova, Rolf Krause, and Rasmus Tamstorf. Subdivision-based nonlinear multiscale cloth simulation. *SIAM Journal on Scientific Computing*, 2019.

[32] Cheng Li, Min Tang, Ruofeng Tong, Ming Cai, Jieyi Zhao, and Dinesh Manocha. P-cloth: interactive complex cloth simulation on multi-gpu systems using dynamic matrix assembly and pipelined implicit integrators. *ACM Transactions on Graphics*, 2020.

[33] Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E Brown, and Laurence Boissieux. An implicit frictional contact solver for adaptive cloth simulation. *ACM Transactions on Graphics*, 2018.

[34] Minchen Li, Danny M Kaufman, and Chenfanfu Jiang. Codimensional incremental potential contact. *ACM Transactions on Graphics*, 2021.

[35] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *International Conference on Learning Representations (ICLR)*, 2022.

[36] Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics*, 2022.

[37] Yifei Li, Hsiao-yu Chen, Egor Larionov, Nikolaos Sarafianos, Wojciech Matusik, and Tuur Stuyck. Diffavatar: Simulation-ready garment optimization with differentiable simulation. *arXiv preprint arXiv:2311.12194*, 2023.

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

[39] Augustus Edward Hough Love. *A treatise on the mathematical theory of elasticity*. Cambridge university press, 2013.

[40] Jia Lu and Chao Zheng. Dynamic cloth simulation by isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 2014.

[41] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 2021.

[42] Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. Projective dynamics with dry frictional contact. *ACM Transactions on Graphics (TOG)*, 2020.

[43] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[44] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 2022.

[45] Rahul Narain, Armin Samii, and James F O'brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics*, 2012.

[46] Miguel A Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. Implicit contact handling for deformable objects. In *Compututer Graphics Forum*, 2009.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[48] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[49] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019.

[50] Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics*, 2021.

[51] Igor Santesteban, Miguel A Otaduy, and Dan Casas. SNUG: Self-Supervised Neural Dynamic Garments. *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[52] Juan C Simo and David D Fox. On a stress resultant geometrically exact shell model. part i: Formulation and optimal parametrization. *Computer Methods in Applied Mechanics and Engineering*, 1989.

[53] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[54] Min Tang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. Pscc: Parallel self-collision culling with spatial hashing on gpus. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2018.

[55] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *SIGGRAPH Comput. Graph.*, 1987.

[56] Bernhard Thomaszewski, Markus Wacker, and Wolfgang Straßer. A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, 2006.

[57] Stephen Timoshenko, Sergius Woinowsky-Krieger, et al. *Theory of plates and shells*. McGraw-hill New York, 1959.

[58] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021.

[59] Pascal Volino and N Magnenat Thalmann. Implementing fast cloth simulation with collision response. In *Proceedings Computer Graphics International 2000*, 2000.

[60] Huamin Wang. Gpu-based simulation of cloth wrinkles at submillimeter levels. *ACM Transactions on Graphics*, 2021.

[61] Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. Data-driven elastic models for cloth: modeling and measurement. *ACM Transactions on Graphics*, 2011.

[62] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[63] Gerald Wempner and Demosthenes Talaslidis. Mechanics of solids and shells. *CRC, Boca Raton*, 2003.

[64] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum (Eurographics State of the Art Reports)*, 2022.

[65] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[66] Jonas Zehnder, Yue Li, Stelian Coros, and Bernhard Thomaszewski. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems*, 2021.

[67] Jiayi Eris Zhang, Jérémie Dumas, Yun Fei, Alec Jacobson, Doug L James, and Danny M Kaufman. Progressive simulation for cloth quasistatics. *ACM Transactions on Graphics*, 2022.

[68] Jiayi Eris Zhang, Jérémie Dumas, Yun Fei, Alec Jacobson, Doug L James, and Danny M Kaufman. Progressive shell qasistatics for unstructured meshes. *ACM Transactions on Graphics (TOG)*, 2023.

NeuralClothSim: Neural Deformation Fields Meet the Thin Shell Theory —Appendices—

Navami Kairanda Marc Habermann Christian Theobalt Vladislav Golyanik

Max Planck Institute for Informatics, Saarland Informatics Campus

Table of Contents

A Implementation Details **15**

B Kirchhoff-Love Thin Shell Theory **15**

B.1 Geometric Preliminaries 16

B.2 Kirchhoff-Love Shell Kinematics 16

B.3 Material Elasticity Model 18

B.4 Equilibrium Deformation 19

B.5 Tensor Algebra 20

B.6 Proof of Strain Computation 20

C Belytschko Obstacle Course **22**

C.1 Square Plate 22

C.2 Scordelis-Lo Roof 22

C.3 Pinched Cylinder 23

D Simulation Details **23**

D.1 Cloth Trajectory Visualisation 24

D.2 Napkin 24

D.3 Sleeve 25

D.4 Skirt 25

E Ablations **25**

E.1 Activation Function 25

E.2 NDF Coordinate System 26

E.3 Non-linearity of Strains 26

E.4 Boundary Constraints 26

F Applications **27**

F.1 Material-conditioned NDFs 27

F.2 NDF Editing 27

G Performance **28**

G.1 Runtime 28

G.2 Sampling Strategy 29

G.3 Simulation Reproducibility 29

Table II: **Notations.** We omit separation of quantities in undeformed (overbar, *e.g.* $\bar{\mathbf{x}}$) and deformed configurations. Moreover, we list the tensors with their covariant components but omit the contravariant and mixed variant versions. Instead of subscripts (covariant components), they are represented with superscripts or a mix of superscripts and subscripts.

Symbol	Description	Symbol	Description
$\xi^\alpha \in \Omega, \boldsymbol{\xi} = (\xi^1, \xi^2)$	Curvilinear coordinates	$t \in [0, T]$	Time
$\xi^3 \in [-\frac{h}{2}, \frac{h}{2}], h \in \mathbb{R}$	Thickness coordinate	$\dot{\mathbf{u}} : \Omega \times [0, T] \to \mathbb{R}^3$	Velocity
$\mathbf{x} : \Omega \to \mathbb{R}^3$	Midsurface representation	$\mathcal{I} : [0, T] \to \mathbb{R}$	Initial distance function
$\mathbf{a}_\alpha : \Omega \to \mathbb{R}^3$	Midsurface tangent vectors	$\mathcal{B} : \Omega \to \mathbb{R}$	Boundary condition
$\mathbf{a}_3 : \Omega \to \mathbb{R}^3$	Unit normal to midsurface	$\mathcal{F}_\Theta : \Omega \to \mathbb{R}^3$	Neural deformation field (NDF)
$a_{\alpha\beta} : \Omega \to \mathbb{R}$	Metric tensor on midsurface	$\rho \in \mathbb{R}$	Mass density
$b_{\alpha\beta} : \Omega \to \mathbb{R}$	Curvature tensor on midsurface	$E \in \mathbb{R}$	Young's modulus
$\mathbf{r} : \Omega \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}^3$	Shell representation	$\nu \in \mathbb{R}$	Poisson's ratio
$\mathbf{g}_i : \Omega \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}^3$	Tangent base vectors on shell	$k_{11}, k_{12}, k_{22}, G_{12}$	Infinitesimal strain parameters
$g_{ij} : \Omega \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}$	Metric tensor on shell	$\mu_{ji}, \alpha_{ji}, d_j$	Nonlinear material response
$\mathbf{f} : \Omega \to \mathbb{R}^3$	External force	$\mathbf{d}_1, \mathbf{d}_2$	Warp/weft material directions
$\mathbf{u} : \Omega \to \mathbb{R}^3$	Midsurface deformation	$\tilde{E}_{\alpha\beta} : \Omega \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}$	Orthotropic strain
$\tilde{\mathbf{u}} : \Omega \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}^3$	Shell deformation	$v_{\alpha\vert\beta} : \Omega \to \mathbb{R}$	Covariant derivative
$\mathbf{w} : \Omega \to \mathbb{R}^3$	Midsurface orientation change	$\Gamma^\lambda_{\alpha\beta} : \Omega \to \mathbb{R}$	Christoffel symbol
$\varphi_{\alpha\beta} : \Omega \to \mathbb{R}$	Deformation gradients	$\mathcal{E} : \Omega \to \mathbb{R}$	Potential energy
$E_{ij} : \Omega \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}$	Green-Lagrange strain	$\Psi : \Omega \to \mathbb{R}$	Hyperelastic strain energy
$\varepsilon_{\alpha\beta} : \Omega \to \mathbb{R}$	Membrane strain	$\kappa_{\alpha\beta} : \Omega \to \mathbb{R}$	Bending strain

H Additional Comparisons **29**

H.1 Runtime 31

H.2 Multi-Resolution Consistency 31

I Collision **32**

J Extended Discussion and Limitations **33**

Sections referenced with numbers refer to the main matter. All referenced figures and equations are per default from this document, unless they are followed by the "(main matter)" mark.

A Implementation Details

We implement NeuralClothSim in PyTorch [47] and compute the geometric quantities on the reference shape and on the NDF using its tensor operations; the first and second-order derivatives are calculated using automatic differentiation. Our network architecture for NDF is an MLP with sine activations (SIREN) [53] with five hidden layers and 512 units in each layer. We empirically set SIREN's frequency parameter to $\omega_0 = 30$ for all experiments (we observed that choosing $\omega_0 = 1$ does not permit folds). Although we sample from $(\xi^1, \xi^2) \in \Omega, t \in [0, T], T = 1$, we normalise samples to $(\xi^1, \xi^2, t) \in [0, 1]^3$ when feeding the input to MLP as per the initialisation principle of SIREN. Note that all physical quantities are computed in the original domains $\Omega, [0, T]$ and the gradients are tracked in their scaled versions. For training, we use $N_\Omega = 20 \times 20$ and $N_t = 20$. At test time, we sample much higher for visualisation, usually with $N_\Omega = 100 \times 100$ and $N_t = 30$. For material conditioning, we use a single random material sample per training iteration. NeuralClothSim's training time amounts to $\sim 10 - 30$ minutes for most experiments, and the number of training iterations equals $\sim 2000 - 5000$. We use ADAM [30] optimiser with a learning rate of 10^{-4} and run our simulator on a single NVIDIA Quadro RTX 8000 GPU with 48 GB of global memory.

B Kirchhoff-Love Thin Shell Theory

In this section, we briefly review the Kirchhoff-Love thin shell theory following [14, 29]; detailed treatment of the subject can be found in [3]. We already introduced physical and mathematical

notations in Sec. 3-(main matter). A detailed list of notations can be found in Tab. II. We next present concepts from the differential geometry of surfaces to explain the midsurface and director (Appendix B.1). We then follow with the shell parameterisation and computation of strain measures on and off the midsurface (Appendix B.2). Further, we present the hyperelastic material models that relate the strains to the internal stress (Appendix B.3) and finally review the energy principles for equilibrium deformation (Appendix B.4). Moreover, we provide a proof of the simplified strain formulation in Appendix B.6 as well as additional results from tensor algebra that are relevant for the computations (Appendix B.5).

B.1 Geometric Preliminaries

In Kirchhoff-Love shell theory, the shell midsurface completely determines the strain components throughout the thickness. Therefore, we review those aspects of the differential geometry of surfaces that are essential for understanding the shell theory.

Let us represent the midsurface as a 2D manifold in the 3D space, as shown in Fig. I. It can be described by a smooth map, $\mathbf{x} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$ on the parametric domain Ω. Any position $\mathbf{x}(\xi^1, \xi^2)$ on the surface is uniquely identified using the convective curvilinear coordinates $(\xi^1, \xi^2) \in \Omega$. As positions can be specified using Cartesian coordinates $\mathbf{x} = x_i \mathbf{e}_i$, it follows that the invertible maps $x_i = x_i(\xi^1, \xi^2)$ and $\xi^\alpha = \xi^\alpha(x_1, x_2, x_3)$ exist. We define a local covariant basis to conveniently express local quantities on the surface. Such a basis is constructed using $\mathbf{a}_\alpha$, the set of two vectors tangential to the curvilinear coordinate lines ξ^α:

$$\mathbf{a}_\alpha := \mathbf{x}_{,\alpha}. \tag{8}$$

To measure the distortion of length and angles, we compute the covariant components of the symmetric metric tensor (also known as the first fundamental form):

$$a_{\alpha\beta} = a_{\beta\alpha} := \mathbf{a}_\alpha \cdot \mathbf{a}_\beta. \tag{9}$$

The corresponding contravariant components of the surface metric tensors denoted by $a^{\alpha\lambda}$ can be obtained using the following identity:

$$a^{\alpha\lambda} a_{\lambda\beta} = \delta_{\alpha\beta}, \tag{10}$$

where $\delta_{\alpha\beta}$ stands for the Kronecker delta. $a^{\alpha\lambda}$ can be used to compute the contravariant basis defined as $\mathbf{a}^\alpha \cdot \mathbf{a}_\beta = \delta_{\alpha\beta}$, as follows:

$$\mathbf{a}^\alpha = a^{\alpha\lambda} \mathbf{a}_\lambda. \tag{11}$$

While the covariant base vector $\mathbf{a}_\alpha$ is tangent to the ξ^α line, the contravariant base vector $\mathbf{a}^\alpha$ is normal to $\mathbf{a}_\beta$ when $\alpha \neq \beta$. Generally, $\mathbf{a}_\alpha$ and $\mathbf{a}^\alpha$ need not be unit vectors.

The shell director coincides with $\mathbf{a}_3$, the unit normal to the midsurface, and, therefore, computed as the cross product of the tangent base vectors:

$$\mathbf{a}_3 := \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1 \times \mathbf{a}_2|}, \quad \mathbf{a}^3 = \mathbf{a}_3. \tag{12}$$

The second fundamental form—which measures the curvature of the midsurface—can be defined with $\mathbf{a}_3$ as:

$$b_{\alpha\beta} := -\mathbf{a}_\alpha \cdot \mathbf{a}_{3,\beta} = -\mathbf{a}_\beta \cdot \mathbf{a}_{3,\alpha} = \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3. \tag{13}$$

Finally, the surface area differential $d\Omega$ relates to the reference coordinates via the determinant of the metric tensor:

$$d\Omega = \sqrt{a}\, d\xi^1\, d\xi^2, \text{ where } \sqrt{a} := |\mathbf{a}_1 \times \mathbf{a}_2|. \tag{14}$$

B.2 Kirchhoff-Love Shell Kinematics

The Kirchhoff-Love model proposes a reduced kinematic parameterisation of a thin shell characterised by a 2D midsurface and shell director. It relies on the Kirchhoff hypothesis, *i.e.,* the director initially perpendicular to the midsurface remains straight and normal, and the shell thickness $h \in \mathbb{R}$ does not change with deformation; see Fig. I.

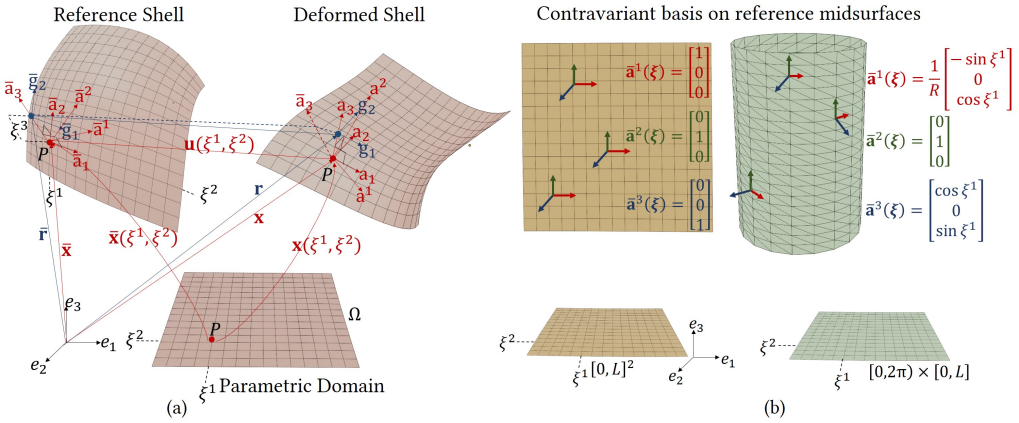

Figure I: **(a) Kirchhoff-Love thin shell.** A thin shell can be kinematically described by the midsurface (here: reference and deformed midsurfaces) and the director (here, $\bar{\mathbf{a}}_3$). Any material point P on the midsurface is then parameterised with curvilinear coordinates (ξ^1, ξ^2), whereas a point on the shell continuum requires an additional thickness coordinate ξ^3. Geometric quantities on the midsurface (off the midsurface or on the shell continuum) are coloured red (blue). **(b) Contravariant basis for midsurfaces in the reference configuration.** While a local contravariant basis coincides with the global Cartesian coordinate system for a planar reference shell, such a basis varies in magnitude and direction across any circular section of the cylinder. Local basis relies on the surface parameterisation, therefore the derived basis vectors need not be normalised (notice how $\bar{\mathbf{a}}^1(\boldsymbol{\xi})$ scales inversely with the radius).

The position vector $\bar{\mathbf{r}}$ of a material point in the reference configuration of the shell continuum can be parametrised with curvilinear coordinates (ξ^1, ξ^2) and thickness coordinate ξ^3 as:

$$\bar{\mathbf{r}}(\xi^1, \xi^2, \xi^3) = \bar{\mathbf{x}}(\xi^1, \xi^2) + \xi^3 \bar{\mathbf{a}}_3(\xi^1, \xi^2), \text{ s.t. } -\frac{h}{2} \leq \xi^3 \leq \frac{h}{2}, \tag{15}$$

where $\bar{\mathbf{x}}(\xi^1, \xi^2)$ represents the midsurface.

The shell adopts a deformed configuration under the action of applied forces $\mathbf{f}$. Analogously, the deformed position vector $\mathbf{r}$ is represented as

$$\mathbf{r}(\xi^1, \xi^2, \xi^3) = \mathbf{x}(\xi^1, \xi^2) + \xi^3 \mathbf{a}_3(\xi^1, \xi^2), \text{ s.t. } -\frac{h}{2} \leq \xi^3 \leq \frac{h}{2}, \tag{16}$$

where the deformed director $\mathbf{a}_3$ coincides with the unit normal.

As a consequence, the overall deformation of the Kirchhoff-Love shell is fully described by the displacement field $\mathbf{u}(\xi^1, \xi^2)$ of the midsurface, *i.e.,*

$$\mathbf{x}(\xi^1, \xi^2) = \bar{\mathbf{x}}(\xi^1, \xi^2) + \mathbf{u}(\xi^1, \xi^2). \tag{17}$$

Analogous to the deformation field $\mathbf{u} := \mathbf{x} - \bar{\mathbf{x}}$ of the midsurface, we define $\mathbf{w}$ as the difference vector of unit normals to the midsurface, *i.e.,*

$$\mathbf{w} := \mathbf{a}_3 - \bar{\mathbf{a}}_3 = w_\lambda \bar{\mathbf{a}}^\lambda + w_3 \bar{\mathbf{a}}^3 = w^\lambda \bar{\mathbf{a}}_\lambda + w^3 \bar{\mathbf{a}}_3. \tag{18}$$

Using this formulation, deformations on the shell continuum can be described by the field $\tilde{\mathbf{u}}(\xi^1, \xi^2, \xi^3) = \mathbf{u}(\xi^1, \xi^2) + \xi^3 \mathbf{w}(\xi^1, \xi^2)$. The difference vector $\mathbf{w}$ describes the change in the orientation of the midsurface, enabling us to quantify bending. A simplified way to compute the components w_i of $\mathbf{w}$ is provided in (51).

The tangent base vectors at a point on the shell continuum are denoted by $\mathbf{g}_i := \mathbf{r}_{,i}$ and expressed by those of the midsurface as:

$$\begin{aligned} \mathbf{g}_\alpha &= \mathbf{a}_\alpha + \xi^3 \mathbf{a}_{3,\alpha}, \\ \mathbf{g}_3 &= \mathbf{a}_3. \end{aligned} \tag{19}$$

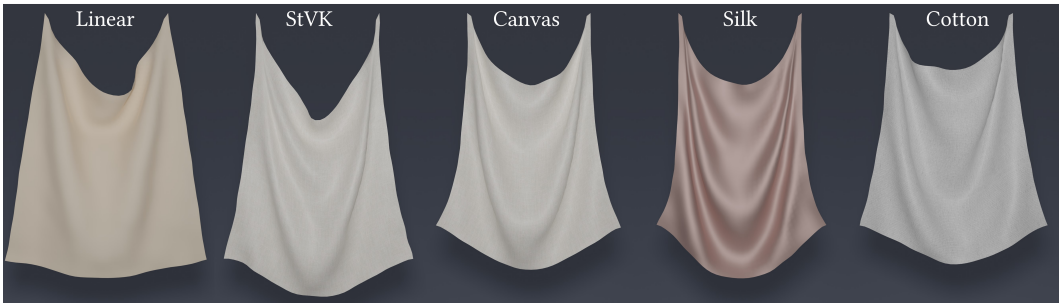

Figure II: **Material model.** Simulation of stable equilibria of $1\,\mathrm{m} \times 1\,\mathrm{m}$ napkin with corners held $60\,\mathrm{cm}$ apart. From left to right, we visualise linear isotropic, linear anisotropic St.Venant-Kirchhoff (canvas), and non-linear anisotropic canvas, silk and cotton materials from Clyde et al. [15].

The corresponding covariant components of the metric tensor are then obtained using

$$g_{ij} := \mathbf{g}_i \cdot \mathbf{g}_j. \tag{20}$$

To measure strain, we use the symmetric Green-Lagrange strain tensor $\mathbf{E} = E_{ij}\bar{\mathbf{g}}^i \otimes \bar{\mathbf{g}}^j$, since it discards the rotational degrees of freedom from tangent base vector $\mathbf{g}_i$ while retaining the stretch and shear information. It is defined as the difference between the metric tensors on the deformed and undeformed configurations of the shell, *i.e.,*

$$E_{ij} := \frac{1}{2}(g_{ij} - \bar{g}_{ij}). \tag{21}$$

Using (19) and (20), note that transverse shear strain measuring the shearing of the director vanishes ($E_{\alpha 3} = 0$) and the stretching of the director is identity, *i.e.,* $E_{33} = 1$; hence, the strain simplifies to

$$E_{\alpha\beta} = \varepsilon_{\alpha\beta} + \xi^3 \kappa_{\alpha\beta}, \tag{22}$$

with membrane strain measuring the in-plane stretching defined as

$$\varepsilon_{\alpha\beta} := \frac{1}{2}(a_{\alpha\beta} - \bar{a}_{\alpha\beta}), \tag{23}$$

and bending strain measuring the change in curvature defined as

$$\kappa_{\alpha\beta} := \bar{b}_{\alpha\beta} - b_{\alpha\beta}. \tag{24}$$

B.3 Material Elasticity Model

Our NeuralClothSim is orthogonal to the research on material modelling and can, thus, be formulated with many different elastic behaviours. We demonstrate results with a simple linear isotropic model [14], and the data-driven anisotropic non-linear model from Clyde *et al*. [15], as well as the St Venant-Kirchhoff variant of the Clyde's model.

Linear Isotropic Material. Given the material Young's modulus E, Poisson's ratio ν, a linear isotropic stress-strain relationship leads to hyperelastic strain energy density [52] of the form

$$\Psi = \frac{1}{2}(DH^{\alpha\beta\lambda\delta}\varepsilon_{\alpha\beta}\varepsilon_{\lambda\delta} + BH^{\alpha\beta\lambda\delta}\kappa_{\alpha\beta}\kappa_{\lambda\delta}), \tag{25}$$

where D is the in-plane stiffness and B is the bending stiffness computed as

$$D := \frac{Eh}{1-\nu^2} \text{ and } B := \frac{Eh^3}{12(1-\nu^2)}, \tag{26}$$

and

$$H^{\alpha\beta\lambda\delta} := \nu\bar{a}^{\alpha\beta}\bar{a}^{\lambda\delta} + \frac{1}{2}(1-\nu)(\bar{a}^{\alpha\lambda}\bar{a}^{\beta\delta} + \bar{a}^{\alpha\delta}\bar{a}^{\beta\lambda}). \tag{27}$$

Here, Ψ is the sum of the membrane strain energy density (the first term) and the bending strain energy density (the second term).

Non-linear Orthotropic Material. While the linear isotropic model is simple and sufficient to demonstrate our NeuralClothSim formulation, a data-driven model with estimated fabric material parameters can generate highly realistic cloth simulations. Therefore, we additionally demonstrate our method with the non-linear anisotropic material model from Clyde *et al*. [15], and its simplification to St. Venant-Kirchhoff model [4]. We show the simulation results with the material model with varying materials, such as cotton and silk in Fig. II, and describe the model next.

Clyde *et al*. present an orthotropic constitutive model that accurately represents the anisotropy introduced by the warp and weft structure of woven cloth. More concretely, they write the hyperelastic strain energy density as $\Psi(\mathbf{E}, \mathbf{D}, \mathbf{\Phi})$ where $\mathbf{E}$ is the Green-Lagrange strain (21), $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3]$ is the reference configuration warp/weft orthotropy $(\mathbf{d}_1, \mathbf{d}_2)$ and normal $(\mathbf{d}_3)$ directions, and $\mathbf{\Phi}$ being the fabric parameters. We follow the technique of [4] to determine the material directions $\mathbf{D}$. Orthotropy directions are computed as tangents to the midsurface with the warp direction $\mathbf{d}_1$ coinciding with the normalised covariant base vector, *i.e*.,

$$\mathbf{d}_1 = \frac{\bar{\mathbf{a}}_1}{\|\bar{\mathbf{a}}_1\|}, \mathbf{d}_3 = \bar{\mathbf{a}}_3, \text{ and } \mathbf{d}_2 = \mathbf{d}_3 \times \mathbf{d}_1. \tag{28}$$

Next, the orthotropic components $\tilde{E}_{ij}$ of the strain are obtained by expressing $\mathbf{E}$ in the material basis, with $\tilde{\mathbf{E}} = \mathbf{D}^\top \mathbf{E} \mathbf{D}$. Due to the Kirchhoff-Love kinematic assumptions, any stretches and shears in the out-of-plane direction $\mathbf{d}_3$ vanish, *i.e*. $\tilde{E}_{i3} = \tilde{E}_{3i} = 0$. Finally, the Clyde model's strain energy density intuitively separates the distinct deformation modes and is defined as,

$$\Psi = \frac{k_{11}}{2}\eta_1(\tilde{E}_{11}^2) + k_{12}\eta_2(\tilde{E}_{11}\tilde{E}_{22}) + \frac{k_{22}}{2}\eta_3(\tilde{E}_{22}^2) + G_{12}\eta_4(\tilde{E}_{12}^2), \tag{29}$$

where $\{k_{11}, k_{12}, k_{22}, G_{12}\}$ describe the cloth's infinitesimal (linear) strain behaviour, whereas the function η_j describes the nonlinear response to larger strains with,

$$\eta_j(x) = \sum_{i=1}^{d_j} \frac{\mu_{ji}}{\alpha_{ji}}((x+1)^{\alpha_{ji}} - 1). \tag{30}$$

We obtain the values for material parameters $\mathbf{\Phi} = \{k_{11}, k_{12}, k_{22}, G_{12}, \mu_{ji}, \alpha_{ji}, d_j\}_{i=1, j\in[1,\dots,4]}^{d_j}$ from [17] and model the silk, canvas and cotton fabrics. Additionally, we can arrive at the orthotropic (linear) St. Venant-Kirchhoff model [4] by simply choosing $\eta_j(x) = x$ for all j (see Fig. II).

When optimising the NDF, the non-linear response (30) gives unpredictable results for strains outside the fitting dataset. For a reasonable strain extrapolation, we use quadratic Taylor expansion around the closest valid strain, as proposed in [17] (see [34]-supplement for the derivatives). Towards this, we leverage the strain cutoffs $\tilde{E}_{\alpha\beta}^{\min}$, and $\tilde{E}_{\alpha\beta}^{\max}$ provided as part of the material dataset.

B.4 Equilibrium Deformation

Under the action of external forces and boundary conditions, a thin shell deforms and achieves an equilibrium configuration. Its stable equilibrium state is characterised by the principle of minimum potential energy, which is the sum of external potential energy owing to applied forces and internal potential energy due to material elasticity.

While all the geometric quantities in (8)–(27) are defined at each material point $(\xi^1, \xi^2) \in \Omega$, the energy is integrated over the parametric domain Ω. Considering the total potential energy of the shell $\mathcal{E}[\mathbf{u}]$ is given by the sum of elastic potential energy Ψ and the potential energy due to the external force density $\mathbf{f}$, we obtain:

$$\mathcal{E}[\mathbf{u}] = \int_\Omega \Psi \, d\Omega - \int_\Omega \mathbf{f} \cdot \mathbf{u} \, d\Omega. \tag{31}$$

Next, the stable equilibrium deformation of the shell can be found by minimising the potential energy functional subject to boundary constraints:

$$\begin{aligned} &\mathbf{u}^* = \arg\min_{\mathbf{u}} \mathcal{E}[\mathbf{u}], \text{ subject to} \\ &\mathbf{u}(\xi^1, \xi^2) = \mathbf{b}(\xi^1, \xi^2) \text{ on } \partial\Omega. \end{aligned} \tag{32}$$

The above definition of hyperelastic energy of the shell requires the displacement field $\mathbf{u} \in H^2(\Omega \mapsto \mathbb{R}^3)$ that must necessarily have square-integrable first and second derivatives.

B.5 Tensor Algebra

We provide additional results from tensor algebra [63]) that are relevant for strain computations on shells (see Sec. B.6).

Based on the coordinate system for the tensor components, a tensor can be covariant (*e.g.*, $A_{\alpha\beta}$), contravariant (*e.g.*, $A^{\alpha\beta}$) and may even have mixed character, *i.e.*, partly contravariant and partly covariant in different indices (*e.g.*, A^{β}_{α}). For computing $\varphi^{\lambda}_{\beta}$, and φ^{λ}_{3} in (5)-(main matter), we use the following rule from shell theory that transforms a covariant tensor to a mixed one:

$$\begin{aligned} A^{\alpha}_{\beta} &= A_{\beta\lambda}\bar{a}^{\lambda\alpha}, \\ A^{\alpha}_{3} &= A_{\lambda 3}\bar{a}^{\lambda\alpha}. \end{aligned} \tag{33}$$

A tensor of n-th order has n indices. For example, v_α is first-order, and $H^{\beta\alpha\lambda\delta}$ is fourth. For computing the covariant derivatives of the first-order tensor $u_\rho|_\alpha$ and the second-order tensor $\varphi_{\alpha\lambda}|_\beta$ in (5), we use the following rules:

$$\begin{aligned} v_\alpha|_\beta &= v_{\alpha,\beta} - v_\lambda \Gamma^{\lambda}_{\alpha\beta}, \text{ and} \\ A_{\alpha\beta}|_\gamma &= A_{\alpha\beta,\gamma} - A_{\lambda\beta}\Gamma^{\lambda}_{\alpha\gamma} - A_{\alpha\lambda}\Gamma^{\lambda}_{\beta\gamma}, \end{aligned} \tag{34}$$

where $\Gamma^{\lambda}_{\alpha\beta}$ is the Christoffel symbol given by

$$\Gamma^{\lambda}_{\alpha\beta} := \bar{\mathbf{a}}^{\lambda} \cdot \bar{\mathbf{a}}_{\alpha,\beta}. \tag{35}$$

Some tensors arising in the kinematic description of Kirchhoff-Love thin shells are symmetric with respect to indices α and β, *i.e.,* $A_{\alpha\beta} = A_{\beta\alpha}$. We exploit the symmetry for efficient computations of the following tensors: $a_{\alpha\beta}$, $b_{\alpha\beta}$, $\varepsilon_{\alpha\beta}$, $\kappa_{\alpha\beta}$, and $\Gamma^{\lambda}_{\alpha\beta}$.

In the case of linear elastic material, we also exploit the symmetry of fourth-order symmetric tensor $\mathbf{H}$:

$$H^{\alpha\beta\lambda\delta} = H^{\beta\alpha\lambda\delta} = H^{\beta\alpha\delta\lambda} = H^{\alpha\beta\delta\lambda} = H^{\lambda\delta\alpha\beta}.$$

This property means that only six independent components (after applying symmetry) need to be computed (*i.e.,* H^{1111}, H^{1112}, H^{1122}, H^{1212}, H^{1222}, and H^{2222}).

B.6 Proof of Strain Computation

According to the Kirchhoff-Love theory, the Green-Lagrange strain associated with the deformation of a thin shell is decomposed into the stretching and bending strains of the midsurface. One could compute them using Eqs. (23) and (24), written in terms of the reference state $\bar{\mathbf{x}}$ and the deformed state $\mathbf{x}$ of the midsurface. As an easier alternative, we directly evaluate strains with the NDF $\mathbf{u}$ of the midsurface using (5)-(main matter). Next, we prove that the two formulations are identical following [3]).

Lemma B.1 (Deformation gradient)**.** *Deformation gradient* $\mathbf{u}_{,\alpha}$ *can be written as* $\mathbf{u}_{,\alpha} = \varphi_{\alpha\lambda}\bar{\mathbf{a}}^{\lambda} + \varphi_{\alpha 3}\bar{\mathbf{a}}^{3}$ *where the components of the gradients* $\varphi_{\alpha\lambda}, \varphi_{\alpha 3}$ *are defined as*

$$\begin{aligned} \varphi_{\alpha\lambda} &:= u_\lambda|_\alpha - \bar{b}_{\alpha\lambda}u_3, \text{ and} \\ \varphi_{\alpha 3} &:= u_{3,\alpha} + \bar{b}^{\lambda}_{\alpha}u_\lambda. \end{aligned} \tag{36}$$

Proof. Given deformation field $\mathbf{u}$ of the midsurface described in contravariant basis as $\mathbf{u} = u_\lambda\bar{\mathbf{a}}^{\lambda} + u_3\bar{\mathbf{a}}^{3}$, we compute the deformation gradient as follows:

$$\begin{aligned} \mathbf{u}_{,\alpha} = \mathbf{u}|_\alpha &= u_\lambda|_\alpha\bar{\mathbf{a}}^{\lambda} + u_\lambda\bar{\mathbf{a}}^{\lambda}|_\alpha + u_3|_\alpha\bar{\mathbf{a}}^{3} + u_3\bar{\mathbf{a}}^{3}|_\alpha \\ &= u_\lambda|_\alpha\bar{\mathbf{a}}^{\lambda} + u_\lambda\bar{b}^{\lambda}_{\alpha}\bar{\mathbf{a}}^{3} + u_{3,\alpha}\bar{\mathbf{a}}^{3} - u_3\bar{b}_{\alpha\lambda}\bar{\mathbf{a}}^{\lambda}, \end{aligned} \tag{37}$$

where we use the following identities from the shell theory [3] to arrive at the bottom part of the previous equation:

$$\begin{aligned} \bar{\mathbf{a}}^{3}|_\alpha &= \bar{\mathbf{a}}^{3}_{,\alpha} = -\bar{b}_{\alpha\lambda}\bar{\mathbf{a}}^{\lambda} = -\bar{b}^{\lambda}_{\alpha}\bar{\mathbf{a}}_{\lambda}, \\ \bar{\mathbf{a}}^{\lambda}|_\alpha &= \bar{b}^{\lambda}_{\alpha}\bar{\mathbf{a}}^{3}. \end{aligned} \tag{38}$$

Finally, we rewrite them as

$$\mathbf{u}_{,\alpha} = \varphi_{\alpha\lambda}\bar{\mathbf{a}}^{\lambda} + \varphi_{\alpha 3}\bar{\mathbf{a}}^{3}. \tag{39}$$

□

Theorem B.2 (Membrane strain)**.** *Membrane strain (quantifying/measuring in-plane stretching) can be written as a function of the deformation gradient in the following form:*

$$\varepsilon_{\alpha\beta} = \frac{1}{2}(\varphi_{\alpha\beta} + \varphi_{\beta\alpha} + \varphi_{\alpha\lambda}\varphi^{\lambda}_{\beta} + \varphi_{\alpha 3}\varphi_{\beta 3}). \tag{40}$$

Proof. We start with membrane strain given as the difference of metric tensors (first fundamental form) (23):

$$\begin{aligned} \varepsilon_{\alpha\beta} &:= \frac{1}{2}(a_{\alpha\beta} - \bar{a}_{\alpha\beta}), \\ \varepsilon_{\alpha\beta} &= \frac{1}{2}(\mathbf{a}_{\alpha}\cdot\mathbf{a}_{\beta} - \bar{\mathbf{a}}_{\alpha}\cdot\bar{\mathbf{a}}_{\beta}). \end{aligned} \tag{41}$$

Substituting the tangent basis vectors $\mathbf{a}_{\alpha}$

$$\mathbf{a}_{\alpha} = \mathbf{x}_{,\alpha} = \bar{\mathbf{x}}_{,\alpha} + \mathbf{u}_{,\alpha} = \bar{\mathbf{a}}_{\alpha} + \mathbf{u}_{,\alpha} \tag{42}$$

gives us updated strain in terms of deformation $\mathbf{u}$:

$$\varepsilon_{\alpha\beta} = \frac{1}{2}(\bar{\mathbf{a}}_{\alpha}\cdot\mathbf{u}_{,\beta} + \bar{\mathbf{a}}_{\beta}\cdot\mathbf{u}_{,\alpha} + \mathbf{u}_{,\alpha}\cdot\mathbf{u}_{,\beta}). \tag{43}$$

Assuming the following identities from Kirchhoff-Love shell hypothesis [3]:

$$\begin{aligned} &\bar{\mathbf{a}}_{\alpha}\cdot\bar{\mathbf{a}}^{\beta} = \delta^{\beta}_{\alpha}, \bar{\mathbf{a}}^{3} = \bar{\mathbf{a}}_{3}, \bar{\mathbf{a}}_{\alpha}\cdot\bar{\mathbf{a}}^{3} = \bar{\mathbf{a}}^{\alpha}\cdot\bar{\mathbf{a}}_{3} = 0, \bar{\mathbf{a}}^{3}\cdot\bar{\mathbf{a}}^{3} = 1, \text{ and} \\ &\bar{a}^{\alpha\beta} = \bar{\mathbf{a}}^{\alpha}\cdot\bar{\mathbf{a}}^{\beta}, A^{\alpha}_{\beta} = A_{\beta\lambda}\bar{a}^{\lambda\alpha}, \end{aligned} \tag{44}$$

and considering the above lemma for the deformation gradient $\mathbf{u}_{,\alpha}$, we finally obtain the target formulation for strain:

$$\varepsilon_{\alpha\beta} = \frac{1}{2}(\varphi_{\alpha\beta} + \varphi_{\beta\alpha} + \varphi_{\alpha\lambda}\varphi^{\lambda}_{\beta} + \varphi_{\alpha 3}\varphi_{\beta 3}). \tag{45}$$

□

Theorem B.3 (Bending strain)**.** *Bending strain (measuring the change in curvature) can be written as a function of the deformation gradient in the following form:*

$$\kappa_{\alpha\beta} \approx -\varphi_{\alpha 3}|_{\beta} - \bar{b}^{\lambda}_{\beta}\varphi_{\alpha\lambda} + \varphi^{\lambda}_{3}(\varphi_{\alpha\lambda}|_{\beta} + \frac{1}{2}\bar{b}_{\alpha\beta}\varphi_{\lambda 3} - \bar{b}_{\beta\lambda}\varphi_{\alpha 3}). \tag{46}$$

Proof. The bending strain of the midsurface is defined as the difference of curvature tensors (second fundamental form) in the reference and deformed configurations:

$$\begin{aligned} \kappa_{\alpha\beta} &:= \bar{b}_{\alpha\beta} - b_{\alpha\beta} \\ \kappa_{\alpha\beta} &= \mathbf{a}_{\alpha}\cdot\mathbf{a}_{3,\beta} - \bar{\mathbf{a}}_{\alpha}\cdot\bar{\mathbf{a}}_{3,\beta} \end{aligned} \tag{47}$$

Using (42) and (18), we rewrite strain using deformation gradients as:

$$\kappa_{\alpha\beta} = (\bar{\mathbf{a}}_{\alpha} + \mathbf{u}_{,\alpha})\cdot(\bar{\mathbf{a}}_{3,\beta} + \mathbf{w}_{,\beta}) - \bar{\mathbf{a}}_{\alpha}\cdot\bar{\mathbf{a}}_{3,\beta} \tag{48}$$

Further simplification and applying identity (38) leads to:

$$\kappa_{\alpha\beta} = \bar{\mathbf{a}}_{\alpha}\cdot\mathbf{w}_{,\beta} + \mathbf{u}_{,\alpha}\cdot\mathbf{w}_{,\beta} - \bar{b}^{\lambda}_{\beta}\mathbf{u}_{,\alpha}\cdot\bar{\mathbf{a}}_{\lambda} \tag{49}$$

Using Lemma B.1 for deformation gradients $\mathbf{u}_{,\alpha}$ and $\mathbf{w}_{,\beta}$, *i.e,* $\mathbf{w}_{,\beta} = (w_{\lambda}|_{\beta} - \bar{b}_{\lambda\beta}w_3)\bar{\mathbf{a}}^{\lambda} + (w_{3,\beta} + \bar{b}^{\lambda}_{\beta}w_{\lambda})\bar{\mathbf{a}}^{3}$ and with the shell identities of (44), we arrive at:

$$\begin{aligned} \kappa_{\alpha\beta} = w_{\alpha}|_{\beta} - \bar{b}_{\alpha\beta}w_3 - \bar{b}^{\lambda}_{\beta}\varphi_{\alpha\lambda} + \varphi^{\lambda}_{\alpha}(w_{\lambda}|_{\beta} - \bar{b}_{\lambda\beta}w_3) \\ + \varphi_{\alpha 3}(w_{3,\beta} + \bar{b}^{\lambda}_{\beta}w_{\lambda}) \end{aligned} \tag{50}$$

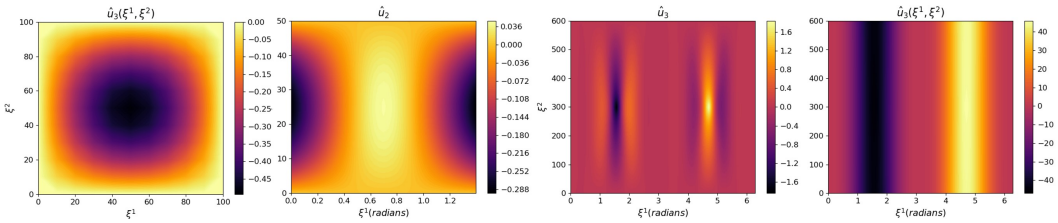

Figure III: **Belytschko obstacle course: Visualisation of the deformation fields predicted by our NeuralClothSim.** From the left to the right: Square plate, Scordelis-Lo roof, pinched cylinder with fixed boundary conditions and pinched cylinder with free boundary conditions. For the pinched cylinder, the results are re-scaled versions with $E = 30$.

With the Kirchhoff-Love normal hypothesis and neglecting cubic terms, we can approximate the components of $\mathbf{w}$ as the following:

$$\begin{aligned} w_3 &\approx -\frac{1}{2} w_\lambda w^\lambda = -\frac{1}{2}\varphi_{\alpha 3}\varphi_3^\alpha, \\ w_\alpha &\approx -\varphi_{\alpha 3} + \varphi_\alpha^\lambda \varphi_{\lambda 3}. \end{aligned} \tag{51}$$

First eliminating the component w_3 and subsequently w_α, we arrive at the target strain formulation:

$$\begin{aligned} \kappa_{\alpha\beta} &\approx w_\alpha|_\beta - \bar{b}_\beta^\lambda \varphi_{\alpha\lambda} + \frac{1}{2}\bar{b}_{\alpha\beta} w_\lambda w^\lambda + w^\lambda|_\beta \varphi_{\alpha\lambda} + \bar{b}_\beta^\lambda w_\lambda \varphi_{\alpha 3} \\ \kappa_{\alpha\beta} &\approx -\varphi_{\alpha 3}|_\beta - \bar{b}_\beta^\lambda \varphi_{\alpha\lambda} + \varphi_3^\lambda(\varphi_{\alpha\lambda}|_\beta + \frac{1}{2}\bar{b}_{\alpha\beta}\varphi_{\lambda 3} - \bar{b}_{\beta\lambda}\varphi_{\alpha 3}). \end{aligned} \tag{52}$$

□

C Belytschko Obstacle Course

In the following, we provide detailed information for reproducing the Belytschko obstacle course experiments from Sec. 5.1-(main matter). We visualise the NDF along the direction of applied load in Fig. III that closely matches the reference solutions [6].

C.1 Square Plate

In the first test case, we consider a simple bending problem of a flat square shell [14]. It is simply supported at all edges and is subject to a uniform load. The plate has a side length of $L = 100\,\mathrm{m}$ and a thickness $h = 1\,\mathrm{m}$ and, therefore, falls under the scope of Kirchhoff-Love thin shell theory. The material parameters are given as $E = 1\mathrm{e}7\,\mathrm{Pa}$ and $\nu = 0$. We represent the reference geometry with (2)-(main matter) and impose Dirichlet boundary constraints by constructing a distance function to the plate edges. This is followed by training an NDF to solve for quasi-static displacement minimising the total potential energy (31) subject to uniformly distributed external load $\mathbf{f} = [0, 0, -1]^\top$. With the simply supported constraints along the boundary defined by $\partial\Omega = \{(\xi^1, 0), (0, \xi^2), (\xi^1, L), (L, \xi^2)\}$, we define NDF as follows:

$$\begin{aligned} \mathbf{u}(\boldsymbol{\xi}; \Theta) &= \mathcal{F}_\Theta(\boldsymbol{\xi})\mathcal{B}(\boldsymbol{\xi}), \\ \text{s.t. } \mathcal{B}(\boldsymbol{\xi}) &:= \xi^1\xi^2(L - \xi^1)(L - \xi^2). \end{aligned} \tag{53}$$

We train with the loss (7)-(main matter) for 2500 iterations and illustrate the solution in Fig. 5-(main matter) where the displacement is scaled up by a factor of 50. The maximum displacement u_3 at the centre of the plate is found to be 0.487 after convergence and exactly matches the reference solution [57]. Fig. III-(left) shows the obtained NDF along the z-axis for the square plate.

C.2 Scordelis-Lo Roof

The reference geometry of the Scordelis-Lo roof is given by the following parametric expression:

$$\begin{aligned} &\bar{\mathbf{x}}(\boldsymbol{\xi}) = [R\cos(\xi^1 + 50°), R\sin(\xi^1 + 50°), \xi^2]^\top, \\ &\forall \xi^1 \in [0, 80°); \xi^2 \in [0, L], \text{ with } R = 25\,\mathrm{m}, L = 50\,\mathrm{m}, \text{ and } h = 0.25\,\mathrm{m}. \end{aligned} \tag{54}$$

Concerning the boundary conditions, the structure is supported with a rigid diaphragm along the edges, *i.e.,* $\partial\Omega = \{(\xi^1, 0), (\xi^1, L)\}$. The material properties are set as $E = 4.32\text{e}8\,\text{Pa}, \nu = 0$ and a uniformly distributed load $\mathbf{f} = [0, -90, 0]^\top$ is applied to it.

We optimise NDF under boundary conditions as follows:

$$\begin{aligned} \mathbf{u}(\boldsymbol{\xi}; \Theta) &= [\mathcal{F}_{\Theta 1}\mathcal{B}(\boldsymbol{\xi}), \mathcal{F}_{\Theta 2}\mathcal{B}(\boldsymbol{\xi}), \mathcal{F}_{\Theta 3}]^\top, \\ \text{s.t. } \mathcal{B}(\boldsymbol{\xi}) &:= \xi^2(L - \xi^2). \end{aligned} \tag{55}$$

Fig. III-(second on the left) visualises the computed NDF along the y-axis.

C.3 Pinched Cylinder

Pinched Cylinder. Finally, we consider the pinched cylinder problem, *i.e.,* one of the most severe tests for both inextensional bending modes and complex membrane states. As shown in Fig. 5-(main matter), a cylindrical shell is pinched with two diametrically opposite unit loads applied at the middle of the shell. We consider two cases: First, a shell with ends supported by rigid diaphragms [6] (similar to Scordelis-Lo roof), and second, a cylinder with free ends [57]. We define the cylinder geometry with (2)-(main matter), where $R = 300\,\text{m}, L = 600\,\text{m}$ and the thickness is set to $h = 3\,\text{m}$; the material properties are given as $E = 3\text{e}6\,\text{Pa}, \nu = 0.3$. In contrast to the previous test geometries, which required specifying only the Dirichlet boundary conditions, we additionally account for the periodicity constraint along the circular cross-sections. To model this, we define NDF as $\mathcal{F}_\Theta(\cos\xi^1, \sin\xi^1, \xi^2)$, instead of the default case $\mathcal{F}_\Theta(\xi^1, \xi^2)$. A crucial challenge of pinched cylinder test case is modelling load at singular points in the sample space. To achieve this, we adapt the potential energy functional (31)-(main matter)—described previously for uniformly distributed forces —to the point load setting, rewriting it as $\mathcal{E}^{\text{pot}}[\mathbf{u}] = \int_\Omega \Psi\, d\Omega - \sum_{\Omega_0} \mathbf{f} \cdot \mathbf{u}$, where Ω_0 is the set of points of the load application. We apply point loads $\mathbf{f} \in \{[0, 0, 1]^\top, [0, 0, -1]^\top\}$ at diametrically opposite points $\Omega_0 = \{(90^\circ, 300), (270^\circ, 300)\}$. In the case of distributed load, we previously proposed computing external and hyperelastic strain energy at an identical set of stratified samples in the parametric domain. We depart from this setting for point loads: At each training iteration, we sample all points from Ω_0 for external energy, whereas random stratified samples are used for computing strain energy. To speed up the convergence, we set $E = 30\,\text{Pa}$ instead of the original value $E = 3\text{e}6\,\text{Pa}$; this simply scales the displacement field in the linear setting as shown in [5]. As mentioned, the constrained cylinder is supported with a rigid diaphragm along the edges, *i.e.,* $\partial\Omega = \{(\xi^1, 0), (\xi^1, L)\}$, therefore, we optimise NDF using

$$\begin{aligned} \mathbf{u}(\boldsymbol{\xi}; \Theta) &= [\mathcal{F}_{\Theta 1}\mathcal{B}(\boldsymbol{\xi}), \mathcal{F}_{\Theta 2}, \mathcal{F}_{\Theta 3}\mathcal{B}(\boldsymbol{\xi})]^\top, \\ \text{s.t. } \mathcal{B}(\boldsymbol{\xi}) &:= \xi^2(L - \xi^2). \end{aligned} \tag{56}$$

Next, we consider a pinched cylinder with free ends, *i.e.* $\partial\Omega = \emptyset$. Without any boundary constraints, the cylinder can move rigidly due to the applied force, and such rigid body motion should be factored out. Therefore, to suppress it, we restrict the displacement of the point under the load in directions other than the direction of the force vector. We achieve this by enforcing $\hat{u}_1 = 0, \hat{u}_2 = 0$ at load points. The NDF parametrisation factoring out the rigid motion reads as:

$$\begin{aligned} \mathbf{u}(\boldsymbol{\xi}; \Theta) &= [\mathcal{F}_{\Theta 1}\mathcal{B}_1(\boldsymbol{\xi})\mathcal{B}_2(\boldsymbol{\xi}), \mathcal{F}_{\Theta 2}\mathcal{B}_1(\boldsymbol{\xi})\mathcal{B}_2(\boldsymbol{\xi}), \mathcal{F}_{\Theta 3}]^\top, \\ \text{s.t. } \mathcal{B}_1(\boldsymbol{\xi}) &:= 1 - e^{-((\xi^1 - 90^\circ)^2 + (\xi^2 - 300)^2)/\sigma}, \text{ and} \\ \mathcal{B}_2(\boldsymbol{\xi}) &:= 1 - e^{-((\xi^1 - 270^\circ)^2 + (\xi^2 - 300)^2)/\sigma}. \end{aligned} \tag{57}$$

In both examples with the pinched cylinder, we monitor the displacements under the loading point. As shown in Fig. 5-(main matter) and Tab. 1-(main matter), it qualitatively and quantitatively converges to the reference solution. Fig. III-(second from the right, and the rightmost) shows the obtained NDFs along the z-axis for the two cases of the pinched cylinder.

D Simulation Details

We first describe the extension of NeuralClothSim for visualising trajectory to equilibria. Then, similar to the previous section, we provide boundary and loading conditions for all experiments here.

D.1 Cloth Trajectory Visualisation

We next visualise the transition from the reference to the equilibrium state: We extend the NDF (3) to $\mathbf{u}(\boldsymbol{\xi}, t; \Theta)$ modelling time-dependent deformations, $\forall t \in [0, T]$, with $T = 1$. For a smooth and physically-plausible interpolation from the initial state $\bar{\mathbf{x}}(\boldsymbol{\xi})$ to the deformed $\mathbf{x}(\boldsymbol{\xi}, T)$, we impose initial conditions and a temporal regulariser.

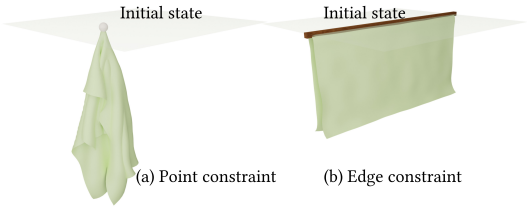

Figure IV: Napkin simulation upon convergence under gravity with *non-boundary* constraints.

Initial Cloth Configuration. If we optimise the time-dependent NDF only with the potential energy loss (7), the model finds the converged equilibrium states of the underlying cloth model $\forall t \in [0, T]$. To start from the initial undeformed cloth state, initial conditions leading to zero displacement and velocity need to be explicitly incorporated. Hence, we use the function $\mathcal{I}(t) := t^2$ as an additional multiplying factor in (3) leading to $\mathbf{u}(\boldsymbol{\xi}, 0) = \mathbf{0}$, and $\dot{\mathbf{u}}(\boldsymbol{\xi}, 0) = \mathbf{0}$.

Temporal Smoothing. Without any temporal prior, the transition from the reference to the equilibrium state will be too swift and not smooth. Therefore, we use an additional regularisation loss $\mathcal{L}_t(\Theta) := \frac{|\Omega|}{N_\Omega N_t} \sum_{i=1}^{N_\Omega} \sum_{j=1}^{N_t} \frac{1}{2} \rho |\dot{\mathbf{u}}(\boldsymbol{\xi}_i, t_j; \Theta)|^2$ constraining the cloth velocity $\dot{\mathbf{u}}$. Specifically, the smooth optimised trajectory $\mathbf{u}^*(\boldsymbol{\xi}, t; \Theta)$ is obtained with the final loss $\mathcal{L} + \mathcal{L}_t$, where, similar to $\mathcal{L}_t(\Theta)$, the physics loss $\mathcal{L}(\Theta)$ is now evaluated over the entire parametric-temporal domain. Additionally, for some examples, such as sleeve compression/torsion, we drive changes in the deformation trajectory by imposing time-varying Dirichlet boundary conditions. We note that the time-stepping is performed purely for visualisation, and we do not model the simulation dynamics that would require taking into account inertial and damping effects. For linear isotropic material, we set $\rho = 0.144\,\mathrm{kg\,m^{-2}}$ and for the non-linear orthotropic material from Clyde et al. [15].

D.2 Napkin

The force in this experiment is defined as $\mathbf{f} = [0, -9.8\rho, 0]^\top$ and the boundary conditions read $\partial\Omega = \{(0, 0), (0, L)\}$. The result of a napkin droop with a fixed corner is shown in Fig. 1-(main matter). NDF in this experiment is parametrised as follows:

$$\begin{aligned} \mathbf{u}(\boldsymbol{\xi}, t; \Theta) &= \mathcal{F}_\Theta(\boldsymbol{\xi}, t) \mathcal{I}(t) \mathcal{B}_{\text{top_left_corner}}(\boldsymbol{\xi}), \\ \text{s.t. } \mathcal{B}_{\text{top_left_corner}}(\xi^1, \xi^2) &:= 1 - e^{-((\xi^1)^2 + (\xi^2 - L)^2)/\sigma}. \end{aligned} \tag{58}$$

The experimental result of a napkin droop with moving corners is shown in Fig. 3-(main matter). The boundary condition read $\partial\Omega = \{(0, L), (L, L)\}$. The NDF parametrisation in this scenario is as follows:

$$\begin{aligned} &\mathbf{u} = \mathcal{F}_\Theta \mathcal{I} \mathcal{B}_{\text{top_left}} \mathcal{B}_{\text{top_right}} + (1 - \mathcal{B}_{\text{top_left}}) \mathcal{B}_{\text{motion}} - (1 - \mathcal{B}_{\text{top_right}}) \mathcal{B}_{\text{motion}}, \\ &\text{s.t. } \mathcal{B}_{\text{top_left}}(\xi^1, \xi^2) := 1 - e^{-((\xi^1)^2 + (\xi^2 - L)^2)/\sigma}, \\ &\mathcal{B}_{\text{top_right}}(\xi^1, \xi^2) := 1 - e^{-((\xi^1 - L)^2) + (\xi^2 - L)^2)/\sigma}, \text{ and} \\ &\mathcal{B}_{\text{motion}}(t) := [0.2t, 0, 0]^T. \end{aligned} \tag{59}$$

The experimental result for napkin droop with fixed edges is shown in Fig. VII. The boundary conditions are defined as $\partial\Omega = \{(\xi^1, 0), (0, \xi^2)\}, \forall (\xi^1, \xi^2) \in [0, L]^2$ and the NDF parameterisation reads

$$\begin{aligned} \mathbf{u} &= \mathcal{F}_\Theta \mathcal{I} \mathcal{B}_{\text{left_edge}} \mathcal{B}_{\text{right_edge}} \\ \text{s.t. } \mathcal{B}_{\text{left_edge}}(\xi^1) &:= 1 - e^{-(\xi^1)^2/\sigma}, \text{ and} \\ \mathcal{B}_{\text{right_edge}}(\xi^2) &:= 1 - e^{-(\xi^2)^2/\sigma}. \end{aligned} \tag{60}$$

D.3 Sleeve

In the experiment with a sleeve, no external force is exerted: $\mathbf{f} = [0, 0, 0]^T$. The boundary region is defined by: $\partial\Omega = \{(\xi^1, 0), (\xi^1, L)\}, \forall \xi^1 \in [0, 2\pi)$. The NDF is parametrised as follows:

$$\begin{aligned}
&\mathbf{u} = \mathcal{F}_\Theta \mathcal{I} \mathcal{B}_{\text{bottom_rim}} \mathcal{B}_{\text{top_rim}} + (1 - \mathcal{B}_{\text{bottom_rim}}) \mathcal{B}_{\text{motion}} - (1 - \mathcal{B}_{\text{top_rim}}) \mathcal{B}_{\text{motion}}, \\
\text{s.t. } &\mathcal{B}_{\text{bottom_rim}}(\xi^2) := 1 - e^{-(\xi^2)^2/\sigma}, \\
&\mathcal{B}_{\text{top_rim}}(\xi^2) := 1 - e^{-(\xi^2 - L)^2/\sigma}, \text{ and} \\
&\mathcal{B}_{\text{motion}}(t) := [0, 0.1t, 0]^\top.
\end{aligned} \tag{61}$$

Sleeve twist is achieved by introducing rotation displacement $\theta = \frac{3\pi}{4}$. The NDF in this scenario is parametrised as follows:

$$\begin{aligned}
&\mathbf{u} = \mathcal{F}_\Theta \mathcal{I} (1 - \mathcal{B}_{\text{bottom_rim}})(1 - \mathcal{B}_{\text{top_rim}}) - \mathcal{B}_{\text{bottom_rim}} \mathcal{B}_{\text{bottom_motion}} + \mathcal{B}_{\text{top_rim}} \mathcal{B}_{\text{top_motion}}, \\
\text{s.t. } &\mathcal{B}_{\text{bottom_rim}}(\xi^2) := e^{-(\xi^2)^2/\sigma}, \\
&\mathcal{B}_{\text{top_rim}}(\xi^2) := e^{-(\xi^2 - L)^2/\sigma}, \\
&\mathcal{B}_{\text{bottom_motion}}(\xi^1, t) := \begin{bmatrix} R(\cos(\xi^1 - \theta t) - \cos \xi^1) \\ 0 \\ R(\sin(\xi^1 - \theta t) - \sin \xi^1) \end{bmatrix}, \text{ and} \\
&\mathcal{B}_{\text{top_motion}}(\xi^1, t) := \begin{bmatrix} R(\cos(\xi^1 + \theta t) - \cos \xi^1) \\ 0 \\ R(\sin(\xi^1 + \theta t) - \sin \xi^1) \end{bmatrix}.
\end{aligned} \tag{62}$$

We demonstrate sleeve torsion in Fig. V-(left) and buckling in V-(right).

D.4 Skirt

See Fig. XIII for the experimental results with skirt. The reference skirt geometry is defined as:

$$\begin{aligned}
\bar{\mathbf{x}}(\boldsymbol{\xi}) &= [r \cos \xi^1, \xi^2, r \sin \xi^1]^T, \ \forall \xi^1 \in [0, 2\pi); \xi^2 \in [0, L], \\
\text{s.t. } r(\xi^2) &:= \frac{(R_{\text{top}} - R_{\text{bottom}})\xi^2}{L} + R_{\text{bottom}}.
\end{aligned} \tag{63}$$

The skirt deform in this experiment under gravity, *i.e.,* $\mathbf{f} = [0, -9.8\rho, 0]^T$; the boundary region is given by $\partial\Omega = \{(\xi^1, L)\}, \forall \xi^1 \in [0, 2\pi)$. NDF is parametrised as follows:

$$\begin{aligned}
\mathbf{u} &= \mathcal{F}_\Theta \mathcal{I} (1 - \mathcal{B}_{\text{top_rim}}), \\
\text{s.t. } \mathcal{B}_{\text{top_rim}}(\xi^2) &:= e^{-(\xi^2 - L)^2/\sigma}.
\end{aligned} \tag{64}$$

The conditions for skirt twisting (angular displacement) are similar to those of the sleeve twist (applied at the top rim) in Sec. D.3.

E Ablations

E.1 Activation Function

Experimental results for a sleeve twist with different activation functions in the NDF network are shown in Fig. V. While ReLU lacks support for higher-order derivatives leading to artefacts, a network with GELU activation can only represent low-frequency deformations. Our usage of sine activation [53] overcomes these limitations and successfully represents fine folds.

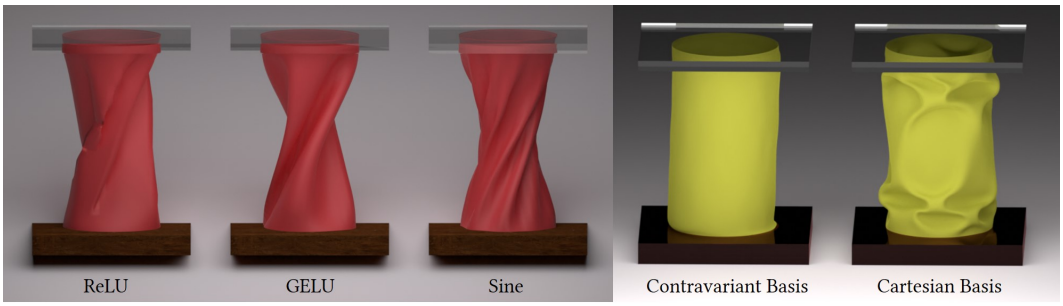

Figure V: **Activations (left).** Results of our method with different activation functions (ReLU, GELU and Siren). **Contravariant *vs* Cartesian basis (right).** Prediction of NDF output in the Cartesian coordinate system is well conditioned compared to the local contravariant coordinate system.

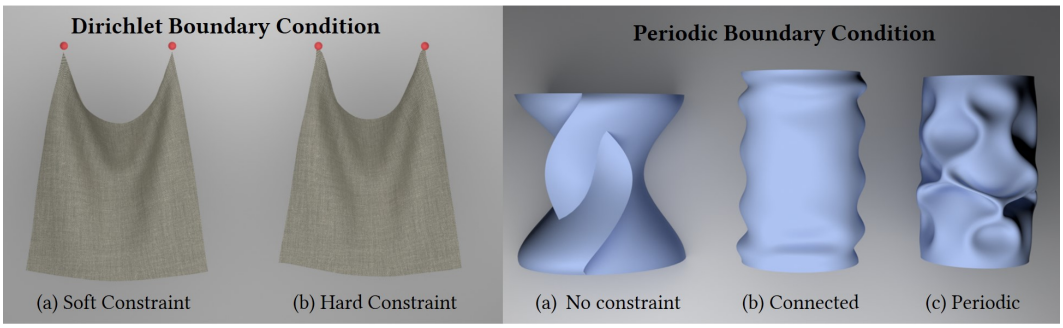

Figure VI: Ablation study for boundary conditions, with Dirichlet (top) and periodic (bottom) boundary conditions.

E.2 NDF Coordinate System

In the Kirchhoff-Love formulation, strain energy computation is performed in the local contravariant (or covariant) basis. This leaves us with an obvious choice of predicting covariant components of the NDF in a locally varying contravariant basis (Fig. I-(b)). Hence, (a) we predict NDF in a contravariant basis and use it directly in strain calculation (ablated version), and (b) we predict NDF in a global basis and transform its components to a local basis before strain calculation. The second case leverages the knowledge of local basis (which is not guessed) and leads to better convergence (Fig. V).

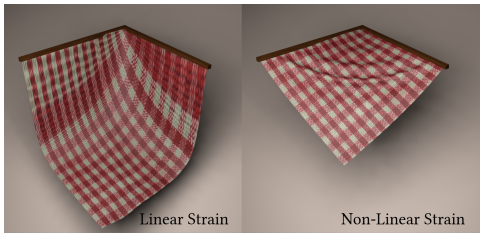

Figure VII: **Linear *vs* non-linear strain.** We demonstrate napkin drooping under a downward force. Kirchhoff-Love strain is inherently highly non-linear.

E.3 Non-linearity of Strains

In the small-strain regime, linearised kinematics is often employed. However, accurate simulation of cloth quasistatics requires modelling of both rigid motion and non-linear deformation. Kirchhoff-Love membrane and bending strains are non-linear functions of the displacement field and non-linear strain calculation is decisive for obtaining realistic results. Thus, we evaluate the linear approximation of Kirchhoff-Love strain by omitting the non-linear terms in (5)-(main matter). In Fig. VII, we show that a linear approach leads to significant inaccuracies in modelling cloth bending under gravity.

E.4 Boundary Constraints

We perform ablations on the Dirichlet and periodic boundary conditions. We try a soft constraint variation, in which we impose the boundary condition as a loss term in addition to the Kirchhoff-Love

energy. This requires empirically determining the optimal loss weight, takes much longer to train and does not guarantee satisfying boundary constraints, as shown in Fig. VI-(left). Our approach with hard constraints avoids all these problems. In the second example, we simulate the compression of a cylindrical sleeve as described in Appendix D.3. As seen in Fig. VI-(right), at $\xi^1 = \pi$, cylinder (a:) is disconnected if no constraint is specified; (b:) is connected with $\xi^1 \mapsto \cos \xi^1$; (c:) fully models continuity and differentiability forming folds with $\xi^1 \mapsto \{\cos \xi^1, \sin \xi^1\}$.

F Applications

F.1 Material-conditioned NDFs

For simplicity, we choose the linear elastic materials, *i.e.* $\mathbf{\Phi} := \{\rho, h, E, \nu\}$. Conditioning on $\mathbf{\Phi}$ allows us to adjust at test time mass density ρ, cloth thickness h, as well as the linear isotropic elastic properties of the material, *i.e.,* Young's modulus E and the Poisson's ratio ν. The updated NDF—which is now a function of material as well—reads: $\mathbf{u}(\boldsymbol{\xi}, t, \mathbf{\Phi}; \Theta) = \mathcal{F}_\Theta(\boldsymbol{\xi}, t, \mathbf{\Phi})\mathcal{I}(t)\mathcal{B}(\boldsymbol{\xi})$, where $\mathbf{\Phi} \in [\mathbf{\Phi}_{\min}, \mathbf{\Phi}_{\max}]$ is the continuous range of material parameters. At each training iteration, we uniformly (at random) re-sample $\mathbf{\Phi}$ to explore the entire material domain. At test time, novel simulation can be generated with a single forward pass for any material $\mathbf{\Phi}$ in the valid material range. Unlike latent space conditioning in other fields and problems, the material space conditioning in NeuralClothSim has a direct physical (semantic) interpretation.

As an example, we train an NDF conditioned on cloth thickness and $\mathbf{\Phi} \in \{\rho, E, \nu\} \times [h_{\min}, h_{\max}]$ with $h_{\min} = 0.0005\,\mathrm{m}$ and $h_{\max} = 0.0025\,\mathrm{m}$. We visualise the simulated result for $h = \{0.0005, 0.0015, 0.0025\}$, in Fig. 1-(bottom right).

F.2 NDF Editing

In movie and game production, a 3D artist's workflow includes updating design parameters, which requires multiple repeated simulations from scratch. Such scene parameters include reference state geometry, external forces, and material properties. Material parameters typically constitute a low-dimensional space, so we propose to condition the NDF on material properties. However, other inputs such as shape and pose of reference state, as well as external force, are high-dimensional. Instead of learning simulations over the entire scene space, we offer simulation editing the following way: the user can interrupt the training of NDF at any point, change the scene parameters and continue training for successive improvement. On the other hand, editing can also be done after full convergence (aka pre-training) and then fine-tuned with gradually modified design parameters. Editing an NDF provides multiple advantages over training a new NDF from scratch: It is computationally and memory efficient and allows access to interpolated simulations.

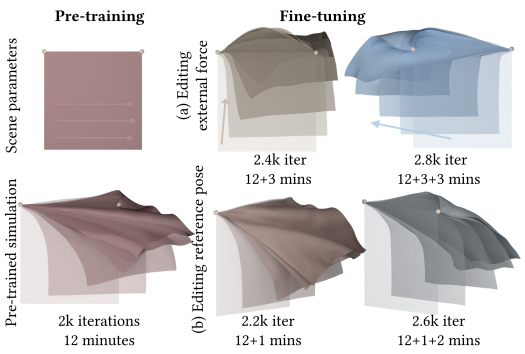

Figure VIII: **Simulation editing with NeuralClothSim.** We show an example of a simulation pre-trained with a fixed reference state and external force. Once converged, we fine-tune the NDF with smoothly varying external force (top) or the pose of the reference geometry (bottom) in each iteration. Fine-tuning a pre-trained NDF with updated design parameters is faster and offers querying of physically-plausible intermediate simulations.

In the following, we demonstrate editing of the following scene parameters: (a) external force, and (b) reference state geometry. The key idea is to use the modified scene parameters in the loss function and update the NDF weights with gradient-based optimisation. Specifically, given a cloth geometry $\bar{\mathbf{x}}$, external forces $\mathbf{f}$, we train an NDF to obtain a simulation $\mathbf{u}^*$ parameterised by network weights Θ^*, as described in the main method. As an editing objective, we would like to arrive at a novel simulation corresponding to external force $\mathbf{f}^I$ and/or reference geometry $\bar{\mathbf{x}}^I$ with $I \in \mathbb{N}$ training iterations. Here, I is much smaller than the iterations needed for the convergence of the original simulation. We can then fine-tune the pre-trained NDF over iterations $i \in \{0, ..., I\}$ by minimising the loss function, $\mathcal{L}(\Theta; \mathbf{f}^i, \bar{\mathbf{x}}^i)$ to obtain edited and interpolated simulations $\mathbf{u}^i, \Theta^i$. Here, we assume

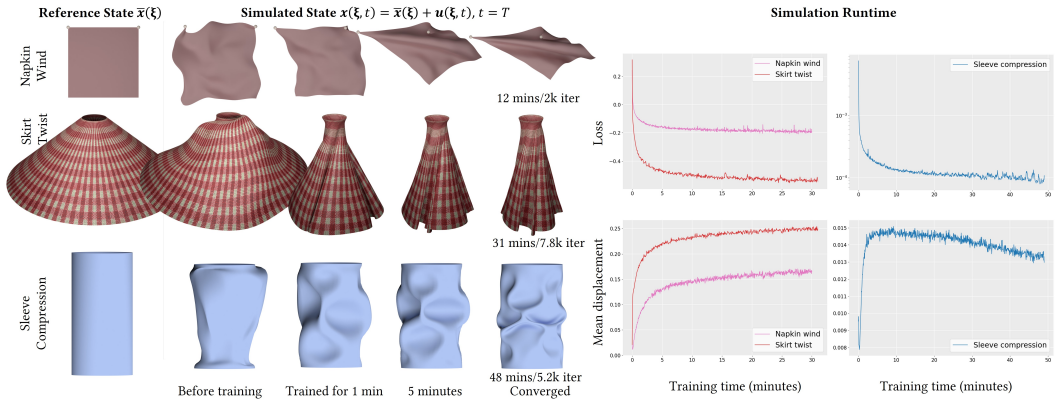

Figure IX: **Runtime analysis of NeuralClothSim.** On the left, we visualise the evolution of the last frame ($T = 1$) over the training iterations. On the right, the plot shows NDF convergence as a function of training time leading to refined simulations.

a smooth transition of external force or the reference shape from the initial to the edited value, which can be obtained, for example, by linear interpolation, *i.e.,* $\mathbf{f}^i = \text{lerp}(\mathbf{f}, \mathbf{f}^I, \frac{i}{I})$.

We next demonstrate results for dynamic editing of the pre-trained simulations. We conduct two experiments, *i.e.,* editing external forces and editing the 6DoF pose of the reference geometry; the results are visualised in Fig. VIII. First, a short simulation of a napkin is pre-trained as an NDF with a fixed reference state and external force, which takes ≈12 minutes. In the first example (top row), we gradually vary the direction and magnitude of the external force by linearly interpolating between the original and the final forces. This leads to the motion of cloth towards the instantaneous force direction. In the second example (bottom row), we smoothly vary the reference poses and the corresponding position of the handles, generating novel edited simulations. Editing reference pose leads to the motion of the cloth towards a fixed force direction but originates from varying initial poses. Note how the change in the input scene parameters propagates to the entire simulation. Notably, fine-tuning is much faster and takes ≈2 minutes, leading to a time-saving of ≈83%. We show two intermediate simulations in Fig. VIII, and other edited simulations corresponding to each iteration can be queried as well. As the simulation is parameterised by the weights of a neural network (instead of meshes), our proposed way of simulation editing is memory-efficient.

G Performance

The performance of a cloth simulator—such as computation time—is a crucial aspect of its usability. This work focuses on the fundamental challenges of developing an implicit neural quasistatic simulator with new characteristics. Our method does *not* outperform the classical simulators in all aspects as they are well-engineered and highly optimised. Next, we provide a detailed analysis of NeuralClothSim's performance.

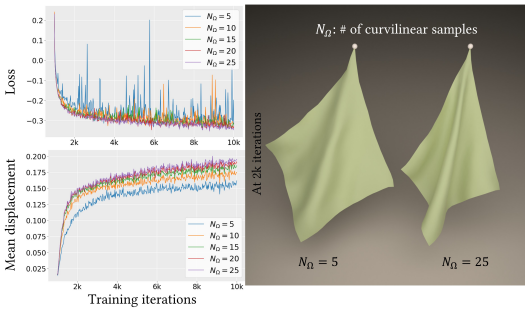

Figure X: **Analysis of the sampling strategies.** We show the influence of the number of training points on the performance of our method.

G.1 Runtime

NeuralClothSim encodes the cloth equilibrium state as an NDF, and, consequently, the bulk of computation time lies in the NDF training (*i.e.,* optimisation of the network weights). At inference, extracting the simulated states from NDF as meshes or point clouds requires a single forward pass and is, therefore, fast. In Fig. IX, we provide a runtime analysis of three representative simulations as a function of training time. On the left,

we visualise the evolution of the last frame (*i.e.*, equilibria state), showing the refinement of the simulated state with increased training time. Before training, the simulation state is the sum of the reference state and random noisy output from NDF. Within a few minutes of training, NDF generates a reasonable simulated state, which then converges within 30 minutes to one hour; see our supplementary video for the evolution of simulation states over training iterations. On the top-right of Fig. IX, we plot loss values as a function of training time, which shows that our training is stable. As NeuralClothSim is an instance of a physics-informed neural network with a physics loss only (but no data term), the loss is not expected to converge to zero. We monitor the mean NDF over all sampled spatio-temporal points (Fig. IX-(bottom right)) as an additional cue on the simulation refinement. Along with the loss, saturation in mean NDF can be used as a stopping criterion. Note that all our experiments are carried out on a single NVIDIA Quadro RTX 8000 GPU.

Similar to classical methods [13], simulation with our approach is not unique, as bifurcation due to buckling can lead to solutions with different folds and wrinkles. Among them, the selection of the simulation outcome depends on the NDF convergence. Specifically, the randomness in training samples and weight initialisation introduces desirable optimisation path variations. In all cases, we observe NDF training to be numerically stable.

G.2 Sampling Strategy

Next, we study the influence of the number of training points on the performance of our method. Input samples to NDF include curvilinear N_Ω and temporal stratified N_t coordinates (for trajectory visualisation) over which the loss is computed at each training iteration. For a napkin of size $\Omega = [0, 1]^2$, we simulate for $t \in [0, 1]$ by training NDF for 10k iterations with number of sampling points $N_\Omega \in \{5, 10, 15, 20, 25\}$. Computation times for all experiments are comparable (and slightly higher for the higher number of samples) as they share the GPU memory and are processed in parallel. Fig. X shows the qualitative and quantitative performance. We observe that higher N_Ω leads to faster learning, as seen in the qualitative result in the top row and the mean displacement plot in the bottom row. Furthermore, it leads to stable optimisation, as seen in the loss plot. Future work could explore advanced sampling techniques for improved performance.

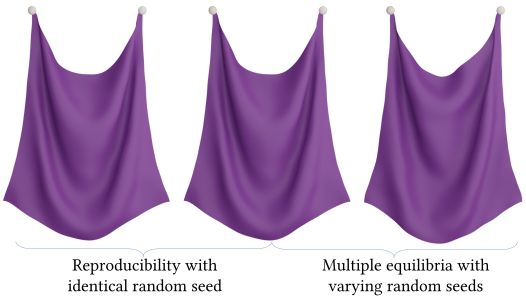

Figure XI: NDF weight initialisation allows us to control the simulation outcome. We can generate multiple valid equilibrium solutions or reproduce a simulation.

G.3 Simulation Reproducibility

Next, we investigate whether NeuralClothSim simulations are deterministic. Cloth simulation does not have a single ground truth; rather, it can have multiple equilibria solutions under the same input parameters (template, material, and boundary conditions). While FEM-based cloth simulators are designed to be deterministic, in practice, there are several factors—such as numerical precision and parallel computing—that can lead to slight variations in the simulation results between runs. We note that a mesh-based simulator running the same simulation scenario on different machines generates non-identical results (but reproducible ones on the same machine). Interestingly, we can replicate such behaviour by employing the sensitivity of our method to the initialisation of the neural network weights. We conducted two experiments leading to the following observations: 1) We can obtain reproducible results if we set the random seed leading to the same network initialisation (Fig. XI-(left)), and 2) We observe non-identical results if we do not set the random seed (Fig. XI-(right)).

H Additional Comparisons

NeuralClothSim is the first step towards neural implicit cloth quasistatics. Although less mature compared to FEM-based simulators, it offers several desired characteristics; See Table III for a comparison between existing cloth simulators and our approach.

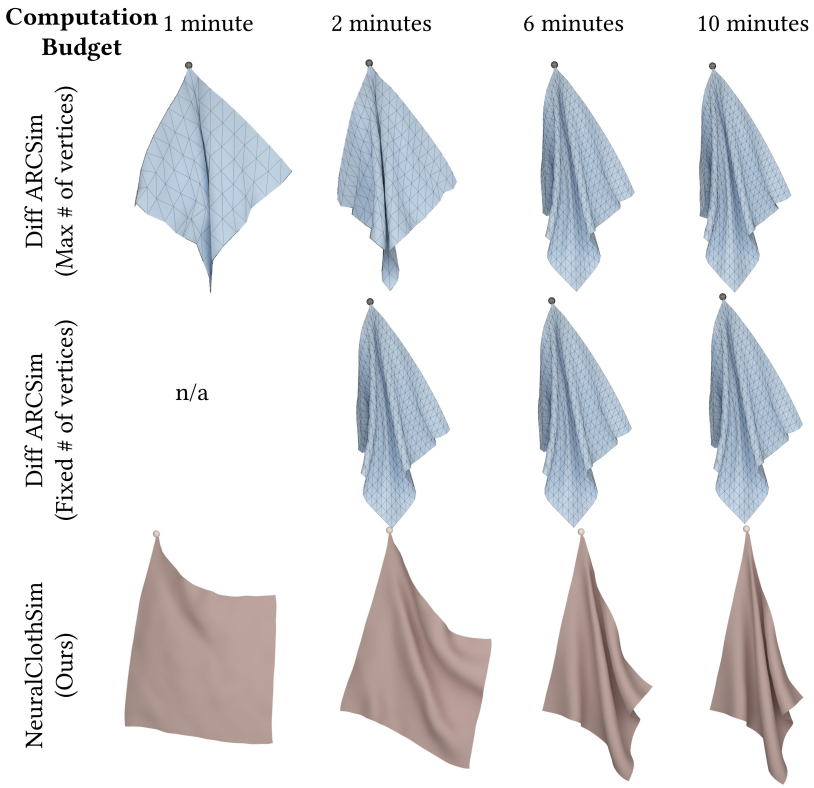

Figure XII: **Runtime comparison of DiffARCSim [38] and our approach.** Like most classical simulators, DiffARCSim integrates forward in time, solving for a 3D deformation field at each time step, in contrast to our approach which optimises for the 4D spatio-temporal NDF. With decreasing computational budget, DiffARCSim produces converged simulated states of the cloth at low resolutions or only early frames at high resolutions. On the other hand, NeuralClothSim offers partially converged simulations at arbitrary resolutions as the computational budget decreases.

Table III: **Conceptual comparison of our NeuralClothSim to previous state-of-the-art cloth simulators.** Our approach enables highly desired properties such as surface continuity, and consistent simulations (folds/wrinkles) at different discretisations of the initial mesh, material conditioning and simulation editing for updated parameters.

	Continuous	Consistency	Sim. Editing	Mat. Interpolation
Narain et al. [45]	✗	✗	✗	✗
Liang et al. [38]	✗	✗	✗	✗
Li et al. [36]	✗	✗	✗	✗
Zhang et al. [67]	✗	✓	✓	✗
Ours	✓	✓	✓	✓

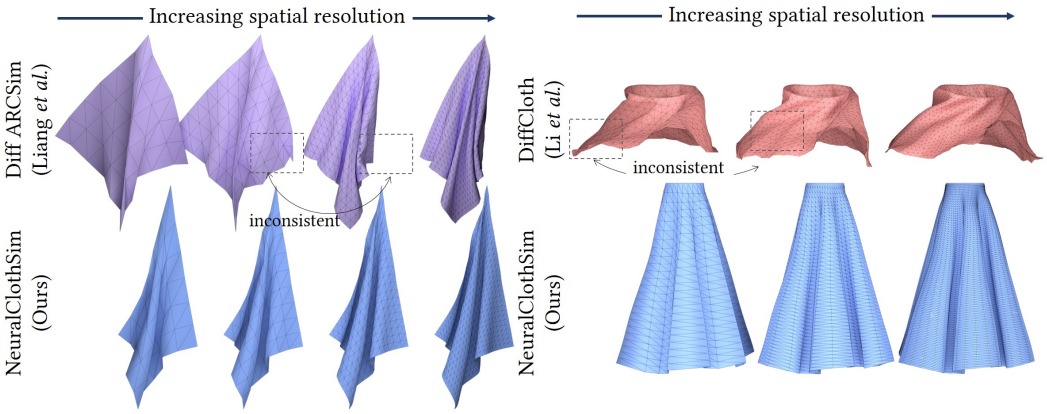

Figure XIII: **Spatial and temporal surface consistency of state-of-the-art differentiable simulators and our approach.** Classical simulators such as ARCSim [38] and DiffCloth [36] reproduce simulation outcomes when re-running at the same resolution. However, changing spatio-temporal resolution requires multiple runs and generates possibly different folds or wrinkles instead of refining (or previewing) the geometry. Since we learn a continuous neural parameterised model, a converged (or partially converged) NDF provides consistent simulation when queried at different spatio-temporal inputs. Note that NeuralClothSim does not provide consistent refinement as a function of computation time (no speed *vs* fidelity trade-off), but rather consistent simulation with respect to the spatio-temporal sampling (at a given computational budget).

H.1 Runtime

We compare the runtime of our method to those of the FEM-based simulator DiffARCSim [38, 45]. Since our approach does not support collisions, we turn off collision handling in DiffARCSim due to the computational overheads for a fair comparison. We simulate a napkin sequence, and our quasistatic result and the dynamic simulated state (after 1 s) from DiffARCSim are visualised in Fig. XII. For the same computation budget (runtime), we show the best simulated states for both methods. Therefore, we present two sets of results for DiffARCSim, *i.e.,* simulated states for the given computational budget 1) with maximum mesh resolution (Fig. XII-(top row)) and 2) with fixed mesh resolution (Fig. XII-(middle row)). We notice that both methods refine the simulated states with increased runtime. With a decreasing computational budget, DiffARCSim produces converged simulated states of the cloth at low resolutions or only early frames at high resolutions. On the other hand, NeuralClothSim offers partially converged simulations at arbitrary resolutions as the runtime decreases.

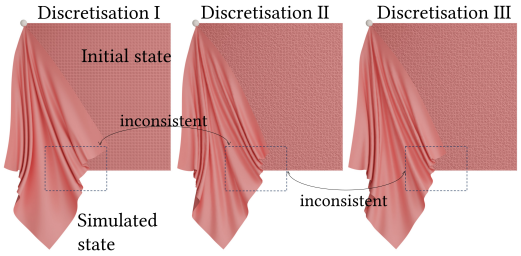

Figure XIV: Visualisation of the inconsistencies observed in the results by FEM-based ARCSim [45], even at high resolutions. Our method leads to consistent results for much coarser discretisations (Fig. 6-main).

H.2 Multi-Resolution Consistency

Next, we show the comparison of NeuralClothSim to DiffARCSim, and DiffCloth [36] in terms of the multi-resolution simulation consistency. We simulate 1) a napkin with a fixed corner under gravity, with our approach and ARCsim (Fig. XIII, top two rows) and 2) a twisting and twirling motion of the skirt with our approach and DiffCloth (Fig. XIII, two bottom rows). The compared simulators operate on meshes of pre-defined resolution (as provided initially). Hence, they need to run from scratch for different mesh resolutions, and the simulation outcome are not guaranteed to be the same across these runs under different discretisations. Thus, increasing (or decreasing) spatial resolution can result in

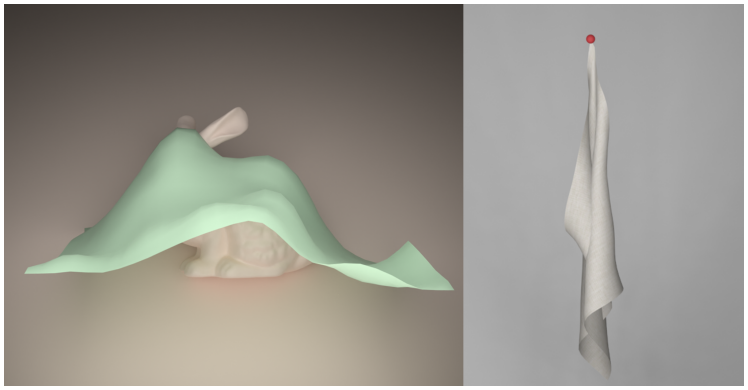

Figure XVI: **Limitations.** Our approach does not handle collisions, contacts and frictions at the moment, since the focus of this work is on the fundamental challenges of developing a neural cloth simulator. These examples show inaccuracies due to the simplifications made in one possible extension (65).

different folds or wrinkles instead of refining simulations at coarser resolutions. Unlike DiffARCSim and DiffCloth, our method provides consistent simulation at arbitrary resolutions. The same 3D points remain unaltered in meshes extracted from NDF at different resolutions. We emphasise that we do not claim consistent refinement as a function of runtime but rather a consistent equilibrium state with respect to spatial sampling (at a given computational budget). This means that both converged or partially converged NDF provide consistent quasistatics when queried at different spatial inputs.

Our comparison deviates from the literature, as the primary reason for using different spatio-temporal resolutions is to adjust runtime and memory usage. For example, the recent method of [67] produces artefact-free previewing geometries (at various approximation levels) by biasing their solutions with shell forces and energies evaluated on the finest-level model. This approach offers a trade-off between runtime *vs* resolution while maintaining simulation consistency. In contrast, with NeuralClothSim, simulation is consistent at arbitrary resolutions at any moment during the NDF training, which, we believe, is still beneficial for many downstream tasks. Of course, ARCSim and DiffCloth also support very high resolutions, which eventually enables browsing the simulations at different mesh resolutions (while maintaining mesh consistency across the levels); however, at the cost of high memory consumption. Moreover, in their case, methods for inverse problems that estimate the simulation parameters from simulated states cannot use adaptive, *e.g.*, coarse-to-fine and importance sampling. In contrast, our continuous formulation offers clear advantages in this regard.

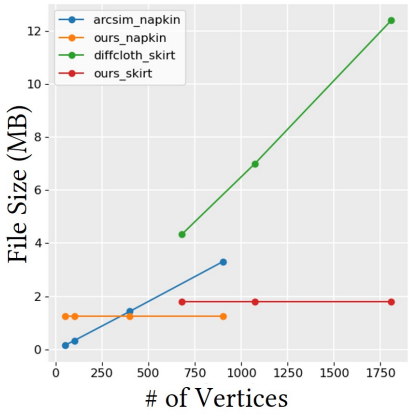

Figure XV: **Memory efficiency**. We plot the memory requirements for simulations generated by ours and DiffARCSim [38], and DiffCloth [36]. The simulations are chosen to be of similar complexity and are visualised in Fig. XIII. The constant memory requirement of our approach is due to the compressing property of the MLP weights that encode the simulations.

I Collision

In a preliminary experiment, we model collisions with external objects following earlier neural methods [51, 8], *i.e.,* we define an additional loss term $\mathcal{L}_{\text{collision}}(\Theta)$ that penalises collisions, leading

to:

$$\mathcal{L} = \mathcal{L}_{\text{physics}}(\Theta) + \lambda \mathcal{L}_{\text{collision}}(\Theta), \text{ with}$$
$$\mathcal{L}_{\text{collision}}(\Theta) = \frac{|\Omega|}{N_\Omega N_t} \sum_{i=1}^{N_\Omega} \sum_{j=1}^{N_t} \max(\epsilon - \text{SDF}(\mathbf{x}(\boldsymbol{\xi}_i, t_j; \Theta)), 0), \tag{65}$$

where $\text{SDF}(\mathbf{x})$ is the signed distance to the object, ϵ is a small safety margin between cloth and object to ensure robustness, λ is the weight for the collision term, and $\mathcal{L}_{\text{physics}}(\Theta)$ is our main thin-shell loss in Eq. (7)-(main matter). We set $\lambda = 1000$, $\epsilon = 0.001$, and use a pre-trained SDF network encoding signed distance function. Specifically, we employ the method of Sitzmann et al. [53] to fit an SDF network on an oriented point cloud, where an Eikonal regularisation is used in addition to the SDF and normal loss. Fig. XVI-(left) visualises a simulation result for a piece of cloth falling on the Stanford bunny; see our supplementary video for the full simulation. We observe that the cloth coarsely respects the object contours, although constraints in Eq. (65) are soft and do not guarantee physically realistic deformations. Difficulties in training PINN with multiple loss terms were previously reported in the literature [25] and future research is necessary to further investigate collision handling in the context of NDFs.

J Extended Discussion and Limitations

This article addresses the fundamental challenges of cloth simulation with NDFs. All in all, we find the proposed design and the obtained experimental results very encouraging and see multiple avenues for future research. Our current quasistatic approach is the first step towards implicit neural simulation. It would be a promising direction to add dynamic effects such as inertia and damping in this setting. Moreover, our simulator does not handle contacts, friction and collisions which will be necessary for many potential applications beyond those demonstrated in this article. This is, however, a standalone research question in the new context. Several limitations of NeuralClothSim originate from NDF modelling as a single MLP: First, MLP weights have a global effect on the simulation, whereas the movement of mesh vertices affects only the local neighbourhood. While this global nature offers continuity and differentiability, we believe exploring alternative network parameterisations that bring the best of both representations could bring improvements in future. Second, our results are currently empirical: While we observe expected results in all our experiments, there are no convergence guarantees or upper bounds on accuracy. Finally, periodic boundary conditions aid mainly with simple geometries; the extension to more complex garments needs further exploration. Future work could also explore modelling different types of human clothing with the help of the proposed implicit neural framework.

Summa summarum, NeuralClothSim is the first step towards neural implicit cloth simulation, which we believe can become a powerful addition to the class of cloth simulators. Inverse problems in vision and graphics could also benefit from its consistency (*e.g.*, multi-resolution data generation), and adaptivity.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We mention the claims clearly in the abstract and introduction. We support the claims with theory and sufficient results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We state the out-of-scope items at the end of the introduction; we discuss the limitations in 'Discussion and Conclusion' section. Further, we provide additional material in the Appendix, which evaluates the current limitations and computational efficiency and discusses directions for future research.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We do not have *new* theoretical results. However, we employ theories from the mechanics and engineering community, which we now bring into the world of the neural fields. For these, we provide sufficient theoretical background and proofs of theorems (please see supplemental) and refer to the appropriate sources.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper should be fully reproducible, as the architecture and the experiments are fully described. We do not contribute any dataset.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the source code as part of the supplemental document and plan to release it publicly, if the paper is accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the training details in the experiment evaluation section. Full details are presented in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the numerical approximations and simplifications of the underlying physics and the material model of the cloth, works in cloth simulation do not provide a comparison to ground-truth, and focus rather on realism over accuracy. However, we validate the model on test cases where the analytical solutions are known.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide runtime computation and GPU usage for our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

Justification: Our work has no negative societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: We do not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We do not contribute any dataset. We document the code and model well, and share code as part of submission.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.