Neurally Integrated Finite Elements for Differentiable Elasticity on Evolving Domains

GILLES DAVIET, NVIDIA, Annecy, France
TIANCHANG SHEN, NVIDIA, Toronto, Canada
NICHOLAS SHARP, NVIDIA, Seattle, United States
DAVID I.W. LEVIN, NVIDIA, Toronto, Canada

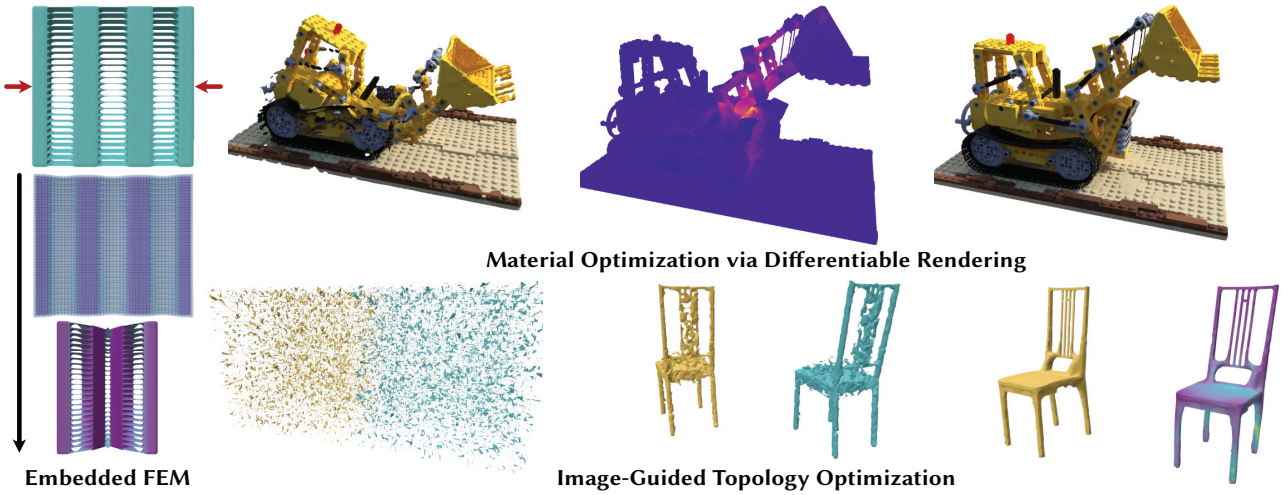

Fig. 1. Our neurally integrated, high-order mixed finite element solver (left) can accurately simulate evolving implicit geometry (including sub-grid features). It is end-to-end differentiable and can be easily combined with other differentiable tools to enable new applications such as image-guided material, shape and topology optimization (top, bottom).

We present an elastic simulator for domains defined as evolving implicit functions, which is efficient, robust, and differentiable with respect to both shape and material. This simulator is motivated by applications in 3D reconstruction: it is increasingly effective to recover geometry from observed images as implicit functions, but physical applications require accurately simulating and optimizing-for the behavior of such shapes under deformation, which has remained challenging. Our key technical innovation is to train a small neural network to fit quadrature points for robust numerical integration on implicit grid cells. When coupled with a Mixed Finite Element formulation, this yields a smooth, fully differentiable simulation model connecting the evolution of the underlying implicit surface to its elastic response. We demonstrate the efficacy of our approach on forward simulation of implicits, direct simulation of 3D shapes during editing, and novel physics-based shape and topology optimizations in conjunction with differentiable rendering.

Authors' Contact Information: Gilles Daviet, NVIDIA, Annecy, France; e-mail: gdaviet@nvidia.com; Tianchang Shen, NVIDIA, Toronto, Canada; e-mail: frshen@nvidia.com; Nicholas Sharp, NVIDIA, Seattle, United States; e-mail: nsharp@nvidia.com; David I.W. Levin, NVIDIA, Toronto, Canada; e-mail: dlevin@nvidia.com.

ACM 0730-0301/2025/04-ART20
https://doi.org/10.1145/3727874

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Reconstruction;**

Additional Key Words and Phrases: Differentiable simulation, numerical integration, topology optimization, shape reconstruction

ACM Reference Format:
Gilles Daviet, Tianchang Shen, Nicholas Sharp, and David I.W. Levin. 2025. Neurally Integrated Finite Elements for Differentiable Elasticity on Evolving Domains. *ACM Trans. Graph.* 44, 2, Article 20 (April 2025), 17 pages. https://doi.org/10.1145/3727874

1 Introduction

For more than a decade, computer vision has been making strides in improving the output fidelity of reconstruction algorithms. With the advent of differential rendering, it is now possible to create highly complex, three-dimensional geometry from two-dimensional images. The robustness and ease-of-use of these modern methods means that almost any complex real world shape can now be cast as a convincing geometric digital twin.

Increasingly, there is a demand to use this geometry in application spaces where beyond its shape, its physical responses and robustness are critically important. For instance, engineers may wish to ensure that a captured 3D bracket can support a certain load if fabricated, and roboticists want to ensure that a captured chair won't collapse under the weight of their robot in a simulated training environment.

To meet such physical constraints requires not just a geometric reconstruction solution, but one that is physically-aware. While shape optimization and system identification methods have been explored in the past, systems that can optimize over geometry, topology and material properties simultaneously are relatively unexplored. In engineering, most shape optimization methods are strongly model-driven often requiring an initial parametric shape model (something that existing reconstruction methods do not produce) while system identification approaches assume the input geometry is fixed and optimize only for physical parameters. There is a great need for algorithms that can optimize shape, topology, and material properties holistically, to maximize physical performance or improve response to physical inputs such as forces.

One approach to this problem is to use a differentiable elasticity simulator as a physical prior in conjunction with a more standard geometry reconstruction method. But state-of-the-art reconstruction approaches generate geometry that is rapidly evolving and can degenerate during the reconstruction process. This, combined with material stiffness parameters that can vary by up-to-four orders of magnitude (at times during optimization) across the object means that robust, in-the-loop simulation for reconstruction is a non-trivial task.

We propose an algorithm that directly attacks these difficulties for elastically deformable objects. Our simulator is built around a regular grid discretization and represents geometry as an implicit function on that same grid. We perform dynamic and quasi-static simulation using a mixed **finite-element method (FEM)** which supports high-order basis functions if necessary, and prevents performance degradation even when material properties are wildly varying. Crucially, we introduce a neural-network approach to per-element quadrature which allows for smooth, differentiable integration of field quantities across the implicitly-defined domain — even as it evolves during the reconstruction procedure.

We combine our novel simulator with the FlexiCubes [Shen et al. 2023] reconstruction algorithm and demonstrate its ability to directly produce geometry that is physically reinforced as to avoid excessive deformation under load. We show that the method requires no strong shape prior, predicts the geometry and topology of the output as part of the reconstruction process, and can simulate the effect of thin, sub-grid features (Figure 1). Finally, we show how each part of our method (mixed FEM, neural quadrature) is required to achieve stable and robust results.

2 Related Work

Geometry reconstruction algorithms focus on producing a consistent 3D representation of a shape from a variety of scanned or synthetic inputs. These inputs include but are not limited to photographs, rendered images, or scan data [Choy et al. 2016]. These include algorithms for producing triangle meshes [Gkioxari et al. 2019; Liu et al. 2018b, 2024], implicit surfaces [Mittal et al. 2022; Park et al. 2019], point clouds [Fan et al. 2017; Wu et al. 2020], Gaussian splats [Charatan et al. 2024; Zhang et al. 2024], and NeRFs [Hong et al. 2023; Mildenhall et al. 2020; Yu et al. 2021]. Our work focuses on providing a compatible, elastodynamics simulator that seamlessly augments geometric reconstruction algorithms to enable the production of physically-sound reconstructed geometry.

Shape and Topology Optimization [Allaire et al. 2004; Bendsøe and Sigmund 2009; Zehnder et al. 2021] are related problems from the engineering literature. While they still optimize for output geometry, they seek to optimize the shape (resp. topology) of an object with respect to some physical properties (e.g., compliance) [Wang et al. 2003] rather than purely seeking geometric or visual agreement with input. The boundary between these methods is somewhat indistinct since most topology optimization schemes can alter shape, via adding or removing material, hence shape optimization more often relies on a parameterized shape template to constrain results to a design space (e.g., Panetta et al. [2017]).

Finally, systems identification problems endeavour to identify material parameters that match observed motion and/or deformation. Typically an existing, parameterized material model is assumed and differentiable simulation is used to ascertain the parameters that harmonize simulated and observed object behavior [Huang et al. 2024; Li et al. 2023a]. Other methods avoid differentiable simulation via techniques such as modal analysis [Chen et al. 2017] but in all cases the geometry is known prior to the physics parameter optimization. For instance, methods such as PAC-NeRF [Li et al. 2023a] first estimate geometry from images then perform system identification on that fixed geometry. Practically, this means that changing geometry cannot be used to optimize physical behavior by construction. In contrast, our novel, neurally-integrated, differentiable elasticity solver is fully differentiable with respect to geometry and material parameters enabling both image-driven and mechanically optimized reconstruction seamlessly and simultaneously.

Differentiable simulations are well-studied and exist for optimizing the trajectory of rigid objects [Popović et al. 2000], fluids [McNamara et al. 2004], coupled rigid and fluid motion [Li et al. 2023b], and deformable objects [Du et al. 2021; Geilinger et al. 2020; Hu et al. 2020; Jatavallabhula et al. 2021], but less frequently support shape derivatives. Topology optimization schemes rely on meshless methods [Li et al. 2020] or finite elements [Gain et al. 2015; Schumacher et al. 2015] to compute necessary physical responses. Crucially, these previous approaches all suffer from one or more issues that make them less than ideal for general geometric optimization tasks. Standard, conforming mesh FEM (typically applied on tetrahedral or hexahedral elements) requires high mesh resolutions [Liu et al. 2018a] to capture the correct behavior of the complex geometries generated by the optimization process. Simulation methods which rely on evolving meshes can require difficult and time-consuming remeshing operations [Huang et al. 2024; Misztal and Bærentzen 2012; Wicke et al. 2010] to keep elements well-conditioned, while density-based meshless methods yield a fuzzy interface [Li et al. 2020].

Implicit functions and their neural counterparts have become the *de facto* geometry representation for shape optimization due to their ability to compactly represent complex evolving geometries [Gao et al. 2020; Shen et al. 2023]. We observe that trying to directly and exactly represent these functions using an evolving, high-resolution mesh is the main source of algorithmic complexity as well as memory and computational pressure. Rather, we are influenced by the success of high-order embedded methods in predicting complex deformations of intricate shapes using simple regular grids [Longva et al. 2020]. When coupled

with appropriately accurate quadrature schemes [Kim and Pollard 2011; Patterson et al. 2012], excellent accuracy can be obtained. However, these methods need the underlying simulated geometry *a priori*, meaning it is not obvious how to apply them to applications where the geometry may evolve.

Fixed quadrature points and weights introduce non-smoothness and ill-conditioning [Van Dijk et al. 2013] as the surface evolves past them. Inspired by recent applications of machine-learning techniques in other areas of simulation [Li et al. 2023; Tymms et al. 2020; Wang et al. 2022; Zesch et al. 2023], our solution is to build a novel finite element method around a neural integration scheme, which uses a small, per-element neural network to learn the quadrature point locations and weights as a function of the underlying implicit shape. This allows points and weights to evolve smoothly, along with the shape itself, yielding higher quality results. Learning functions within grid cells is oft-used in geometry processing (e.g., for mesh extraction [Chen et al. 2022]) however we believe this is the first time it has been applied to this particular problem.

Our differentiable simulator is built around a mixed-variational approach to elasticity [Reissner 1985; Simo and Rifai 1990]. In particular, we modify the rotation-aware extension proposed by Trusty et al. [2022] which allows for simulation performance independent of material stiffness — an important property for system identification where material parameters can range over several orders-of-magnitude. We use a novel, performant variant of the scheme that avoids a per-element singular-value-decomposition at each simulation sub-step and supports high-order elements.

Our major contribution is a differentiable simulator that enables simultaneous optimization of object shape, topology and material properties via drop-in combination with existing geometry reconstruction algorithms. In service of this we develop a novel neural quadrature scheme suitable for finite element algorithms on evolving surfaces, a fast mixed finite-element method to enable robust simulation across wide ranges of material parameters, and a gradient preconditioner to improve convergence.

3 Neurally Integrated FEM

We first recall a few preliminaries about FEM and numerical integration. We consider a material domain $\Omega \subset \mathbb{R}^3$ equipped with a displacement field $\mathbf{u}$ defined over a space[1] V_Ω, and write $\boldsymbol{F}(\mathbf{u}) := \mathbb{I}_3 + \nabla\mathbf{u}$ the deformation gradient.

3.1 Weak-form Elasticity

The kinetic and potential energies of the system are defined as

$$E_k := \int_\Omega \rho \dot{\mathbf{u}}^2, \quad E_p := \psi(\boldsymbol{F}(\mathbf{u})) - \int_\Omega \mathbf{u}.\mathbf{g}, \quad \psi(\boldsymbol{F}) := \int_\Omega \Psi(\boldsymbol{F}),$$

with $\Psi : \mathbb{R}^{3\times3} \to \mathbb{R}$ the local elasticity potential, and $\mathbf{g}$ an external force density. Using an implicit Euler integrator with timestep Δ_t (possibly infinite in the quasistatic limit), such that $\dot{\mathbf{u}} \sim \frac{\mathbf{u}-\mathbf{u}^t}{\Delta_t}$, with $\mathbf{u}^t$ the begin-of-step velocity, the conservation of momentum over the timestep can be expressed as the minimization of the incremental potential [Kane et al. 2000]

$$\min_{\mathbf{u}\in V_\Omega} E_t(\mathbf{u}), \qquad E_t(\mathbf{u}) := \psi(\boldsymbol{F}(\mathbf{u})) + \frac{1}{2}a(\mathbf{u},\mathbf{u}) - b(\mathbf{u}),$$
$$a(\mathbf{u},\mathbf{v}) := \int_\Omega \frac{\rho}{\Delta_t^2}\mathbf{u}.\mathbf{v}, \qquad b(\mathbf{v}) := \int_\Omega \left(\mathbf{g} + \frac{\rho}{\Delta_t^2}\mathbf{u}^n\right).\mathbf{v}. \tag{1}$$

Writing the optimality condition $\partial_{\mathbf{u}} E_t = 0$ as directional (Gâteaux) derivatives over all of V_Ω yields the weak-form FEM formulation,

$$a(\mathbf{u},\mathbf{v}) + \int_\Omega \frac{\partial\Psi}{\partial\boldsymbol{F}}(\boldsymbol{F}(\mathbf{u})) : \nabla\mathbf{v} = b(\mathbf{v}) \quad \forall\mathbf{v} \in V_\Omega. \tag{2}$$

We can then choose a finite subspace for V_Ω, typically polynomials defined over the elements of a mesh $\mathcal{M}$, and write Equation (2) for each function $\mathbf{v}_i$ of our discrete basis. This yields as many nonlinear scalar equations, that can be solved for instance using a quasi-Newton method [e.g., Smith et al. 2018]. Doing so assumes being able to evaluate integrals over Ω, or in practice over any element K of the mesh $\mathcal{M}$. As analytical expressions may not be available, we resort to approximate quadrature rules, that is, sets of points and weights (w_p^K), $(\mathbf{y}_p^K)$ such that for any polynomial P of degree less than or equal to d, $\int_K P = \sum_p w_p^K P(\mathbf{y}_p^K)$. The integer d is called the *order* of the formula. Quadrature rules have been derived and tabulated for common elements (such as tetrahedra and hexahedra) at all practical polynomial orders [Cools 2003]. However, standard quadrature rules are only applicable when the mesh $\mathcal{M}$ coincides with the material domain Ω, and generating good-quality conforming volumetric meshes from a surface is both expensive and not easily differentiable. In the next paragraphs we show how we can avoid building such a mesh altogether and cheaply generate good non-conforming quadrature formulas for surfaces that are implicitly defined by a **signed-distance function (SDF)** discretized over a hexahedral mesh — such as in marching cubes [Lorensen and Cline 1998], OpenVDB [Museth et al. 2013], or FlexiCubes [Shen et al. 2023] grids.

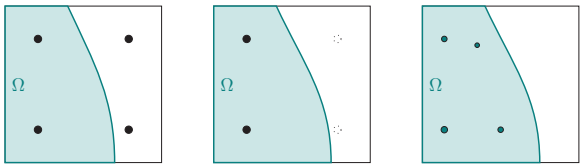

Fig. 2. Illustration in 2D of order-2 quadrature points for a boundary voxel for, from left to right, Full, Clip, and Neural quadrature formulae. The neural quadrature points and weights are updated smoothly when the boundary moves, while other formulas experience jumps.

3.2 Optimized Quadrature Rules

A first observation is that for the weak-form FEM described in Equation (2), we do not need the exact domain geometry; we only need the capacity to numerically integrate functions over elements with good accuracy. Let us consider a mesh element K and a domain Ω such that $K \not\subseteq \Omega$; we want to compute integrals over the part of K where there is material, i.e., $K \cap \Omega$. One first possibility, which we will refer to as the Clip quadrature – see Figure 2, would be to multiply the integrand, or equivalently the quadrature point

[1] typically a subspace of the Sobolev space $H^1(\Omega)^3$.

weights, with the domain indicator function χ_Ω,

$$\int_{K\cap\Omega} f = \int_K f\chi_\Omega \sim \sum_p w_p \chi_\Omega(\mathbf{y}_p) f(\mathbf{y}_p), \quad \chi_\Omega := \begin{cases} 1 \text{ on } \Omega, \\ 0 \text{ elsewhere.} \end{cases}$$

However, the indicator function χ_Ω is highly nonlinear and the quadrature quality will quickly degrade, leading to unstable simulations. Moreover, χ_Ω is non-differentiable with zero gradient almost everywhere, hindering the computation of meaningful derivatives of the integration result with respect to the domain. Instead, following Patterson et al. [e.g., 2012], we opt to derive new quadrature points and weights that can accurately integrate polynomials at a chosen order d on the actual material domain. Using an optimization point of view, we express this quest as

$$\min_{w_p, \mathbf{y}_p} Q_K, \quad Q_K := \sqrt{\sum_{P\in\mathcal{B}_P^d} \left(\int_{K\cap\Omega} P - \sum_p w_p P(\mathbf{y}_p) \right)^2}, \quad (3)$$

with $\mathcal{B}_P^d$ a basis for polynomials of degree d. Monomials may be used to define $\mathcal{B}_P^d$, in which case minimization (3) is known as moment fitting [Bremer et al. 2010]. For symmetry reasons, we prefer to use Lagrange polynomials defined on the Lobatto–Gauss–Legendre nodes as our basis.

3.3 Neural Quadrature Rule Prediction

In our settings of interest, the material domain $\Omega \cap K$ is implicitly defined as the region of K where the SDF φ^K is negative. Moreover, the function φ^K is itself discretized as a finite set of nodal values (φ_j^K), $\varphi^K(\mathbf{x}) = \sum_j \varphi_j^K N_j^\varphi(\mathbf{x})$, with the shape functions (N_j^φ) assumed identical for all nodes — trilinear in our case.[2] Our problem thus reduces to finding, from a set of input SDF values (φ_j^K), quadrature points $(\mathbf{y}_p^K)$ and weights (w_p^K) that are an approximate solution to the minimization problem (3).

In principle, one could use direct numerical optimization techniques. However, we want the following desirable properties for our quadrature rule generation scheme:

(a) it should be extremely cheap, as it will need to be performed for every partially filled element of the mesh, each time the boundary is evolved within a shape optimization loop;
(b) the resulting $(\mathbf{y}_p^K, w_p^K)$ should be continuous with respect to φ_j^K, with easily-accessible and well-behaved gradients;
(c) the number of quadrature points should be fixed, both for controlling the cost of the simulation and, in Mixed FEM settings, for satisfying an element-compatibility condition;
(d) for numerical conditioning the ratio of weights between the different quadrature points should be limited.

On the other hand, our applications do not require

(e) the quadrature rule to be extremely accurate,

as our object reconstruction objective implies uncertainty about the exact location of the domain boundary anyway.

Most moment-fitting approaches [Bremer et al. 2010; Longva et al. 2020; Patterson et al. 2012] aim to minimize the nonlinear problem (3) to high accuracy, i.e., achieve (e) at the detriment of (a) and often (c). Moreover, (b) is usually out of reach for non-convex optimization problems with local minima. Müller et al. [2013] keep the number and position of the quadrature points fixed and optimize for the weights only, yielding a linear problem that achieves both (b) and (c). However, the resulting weights may be null or negative, contradicting (d), and this restricted optimization space limits accuracy for a given number of quadrature points — see also Appendix A.

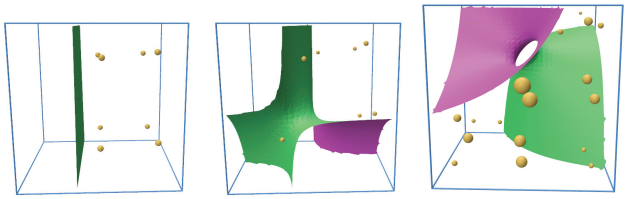

Fig. 3. Learned quadrature points (yellow) for integrating over the part of the unit voxel defined by trilinear interpolation of the corner SDF values (visualized by the green isosurface). Left and middle depict 8-point order-2 quadrature, right is 27 point order-4. Size is proportional to the quadrature point weight.

Instead, we propose to train a small neural network to learn the mapping $(\varphi_j) \mapsto (\mathbf{y}_p, w_p)$. Precisely, we fit a network which takes as input a stacked vector of implicit SDF values at a single cell's corners, and outputs the quadrature point locations and weights within the cell. At simulation time we simply need to run inference for all current voxels, stacked as a single tensor; this is extremely cheap, achieving (a). Criteria (b) and (c) are satisfied by construction, and conditioning (d) can be controlled as an additional training loss term. We train this network once for integration order 2 and 4, and use it for all experiments.

Architecture, Losses, and Training. We choose the network architecture to be a simple multilayer perceptron with $N_{MLP} = 5$ fully-connected layers of size $W_{MLP} = 64$ for order 2 and $W_{MLP} = 128$ for order 4, and ReLU activations on hidden layers. Network inputs are normalized such that the gradient of φ is unit at the cell center, and network outputs are parameterized as offsets from and multipliers for the usual Gauss–Legendre points and weights. We define the loss function as the sum

$$\mathcal{L}_{\text{QuadNet}} := Q_K + 10^1 Q_\square + \gamma_\star Q_\star \quad (4)$$

where Q_K is from Equation (3), with target integrals $\int_{\Omega\cap K} \mathcal{B}_P^d$ computed using brute force uniform integration at high resolution, $Q_\square$ is quadratic barrier enforcing quadrature coordinates to stay in $[0,1]^3$, and $Q_\star$ is a conditioning term penalizing the log ratio of the maximum to minimum quadrature weight. We generate a training set of 2^{24} voxels, each consisting of 8 random corner SDF values, and train for 64k iterations with the AdamW optimizer with batch size 2^{18} chosen to fit our specific GPU memory. Training the order-2 network takes about 1.5 hours on an NVIDIA GeForce RTX 3080Ti GPU — which was also used to generate all of the examples in the remainder of this article — while training the order-4 network takes about 30 hours on an NVIDIA A40 GPU. Figure 3 shows the inferred quadrature points for selected voxel configurations, and Appendix A provides more evaluations and training details.

[2] As we are only concerned with the 0-isosurface, the discretization does not need to preserve the eikonal property of the SDF.

4 Neurally Integrated Mixed FEM

Mixed FEM consists in discretizing the elasticity equations over multiple fields, rather than just the displacement $\mathbf{u}$, which can significantly improve numerical convergence properties [Brezzi and Fortin 1991; Frâncu et al. 2021; Ko et al. 2017; Simo and Rifai 1990]. This is of particular interest to us as we want to embed our solver in a shape reconstruction loop, and as such, desire to obtain a good approximation of the final result even when truncating the solve to a few Newton iterations and regardless of the material stiffness.

The rotation-aware Mixed FEM formulation described by Trusty et al. [2022] boasts this property, however as presented it is limited to linear displacements and piecewise-constant strains and stresses. Below we propose a four-field extension of this mixed formulation to arbitrary finite elements, and show how it can be used in conjunction with our Neural Quadrature integration strategy. Unless otherwise mentioned, this formulation will serve as the basis for our differentiable elasticity simulations.

4.1 Generalized Four-field Mixed FEM

We denote by T_Ω the space of square-integrable 3×3 tensor fields, and define SO_Ω, Sym_Ω and Skew_Ω, the subspaces of T_Ω whose values are rotations, symmetric tensors, and skew-symmetric tensors, respectively. We introduce two additional primal fields, the symmetric strain $\boldsymbol{S}\in\mathrm{Sym}_\Omega$ and rotation $\boldsymbol{R}\in\mathrm{SO}_\Omega$, related to the deformation gradient through the constraint $\boldsymbol{C}(\mathbf{u},\boldsymbol{R},\boldsymbol{S}) := \boldsymbol{F}(\mathbf{u})-\boldsymbol{R}\boldsymbol{S}=0$. Being rotation-independent, the local elastic potential Ψ can now be measured directly on $\boldsymbol{S}$ rather than $\boldsymbol{F}$. Minimization of the incremental potential (1) can be expressed as the constrained optimization

$$\min_{\substack{\mathbf{u}\in V_\Omega,\,\boldsymbol{S}\in\mathrm{Sym}_\Omega,\,\boldsymbol{R}\in\mathrm{SO}_\Omega\\ \boldsymbol{C}(\mathbf{u},\boldsymbol{R},\boldsymbol{S})=0}} \frac{1}{2}a(\mathbf{u},\mathbf{u})-b(\mathbf{u})+\psi(\boldsymbol{S})$$

or equivalently as a saddle point of the associated Lagrangian,

$$\min_{\mathbf{u}\in V_\Omega,\,\boldsymbol{S}\in\mathrm{Sym}_\Omega,\,\boldsymbol{R}\in\mathrm{SO}_\Omega}\ \max_{\boldsymbol{\sigma}\in T_\Omega} \mathcal{L}(\mathbf{u},\boldsymbol{S},\boldsymbol{R},\boldsymbol{\sigma}),$$

$$\mathcal{L}(\mathbf{u},\boldsymbol{S},\boldsymbol{R},\boldsymbol{\sigma}) := \frac{1}{2}a(\mathbf{u},\mathbf{u})-b(\mathbf{u})+\Psi(\boldsymbol{S})+c(\mathbf{u},\boldsymbol{S},\boldsymbol{R},\boldsymbol{\sigma}),$$

$$c(\mathbf{u},\boldsymbol{S},\boldsymbol{R},\boldsymbol{\sigma}) := \int_\Omega \boldsymbol{C}(\mathbf{u},\boldsymbol{R},\boldsymbol{S}):\boldsymbol{\sigma}.$$

Solutions of problem (1) must thus satisfy $\partial\mathcal{L}=0$, that is,

$$a(\mathbf{u},\mathbf{v})+c_{,\mathbf{u}}(\mathbf{v},\boldsymbol{\sigma})-b(\mathbf{v})=0 \qquad \forall\mathbf{v}\in V_\Omega, \tag{5}$$

$$\psi_{,\boldsymbol{S}}(\boldsymbol{S};\boldsymbol{\tau})+c_{,\boldsymbol{S}}(\boldsymbol{R};\boldsymbol{\tau},\boldsymbol{\sigma})=0 \qquad \forall\boldsymbol{\tau}\in\mathrm{Sym}_\Omega, \tag{6}$$

$$c_{,\boldsymbol{R}}(\boldsymbol{S};\boldsymbol{Q},\boldsymbol{\sigma})=0 \qquad \forall\boldsymbol{Q}\in\mathrm{SO}_\Omega, \tag{7}$$

$$c(\mathbf{u},\boldsymbol{R},\boldsymbol{S},\boldsymbol{\lambda})=0 \qquad \forall\boldsymbol{\lambda}\in T_\Omega, \tag{8}$$

where the forms $\psi_{,\boldsymbol{S}}$ and $c_{,q}$ are directional derivatives, i.e.,

$$\psi_{,\boldsymbol{S}}(\boldsymbol{S};\boldsymbol{\tau}) := \int_\Omega \frac{\partial\Psi}{\partial\boldsymbol{S}}(\boldsymbol{S}):\boldsymbol{\tau}, \qquad c_{,\mathbf{u}}(\mathbf{u},\boldsymbol{\lambda}) := \int_\Omega \nabla\mathbf{u}:\boldsymbol{\lambda},$$

$$c_{,\boldsymbol{R}}(\boldsymbol{S};\boldsymbol{R},\boldsymbol{\lambda}) := \int_\Omega \boldsymbol{R}\boldsymbol{S}:\boldsymbol{\lambda}, \qquad c_{,\boldsymbol{S}}(\boldsymbol{R};\boldsymbol{S},\boldsymbol{\lambda}) := \int_\Omega \boldsymbol{R}\boldsymbol{S}:\boldsymbol{\lambda}.$$

Note that at equilibrium the Lagrange multiplier $\boldsymbol{\sigma}$ coincides with the first Piolat–Kirchhoff stress tensor, $\frac{\partial\Psi}{\partial\boldsymbol{F}}$ [Bonet and Wood 2008].

Unfortunately, directly applying Newton iterations to Equations (5)–(8) would lead to numerical difficulties. Indeed, the elasticity Hessian may be indefinite, and there is no coercive potential for the rotation variable; see Appendix B for details. To remedy this problem, we re-inject the constraint $\boldsymbol{C}$ into Equations (6)–(7) using an Augmented–Lagrangian-like penalization term ϵ,

$$a(\mathbf{u},\mathbf{v})+c_{,\mathbf{u}}(\mathbf{v},\boldsymbol{\sigma})-b(\mathbf{v})=0 \qquad \forall\mathbf{v}\in V_\Omega \tag{9}$$

$$\psi_{,\boldsymbol{S}}(\boldsymbol{S};\boldsymbol{\tau})+c_{,\boldsymbol{S}}\left(\boldsymbol{R};\boldsymbol{\tau},\boldsymbol{\sigma}+\varepsilon C\left(\mathbf{u},\boldsymbol{R},\boldsymbol{S}\right)\right)=0 \qquad \forall\boldsymbol{\tau}\in\mathrm{Sym}_\Omega \tag{10}$$

$$c_{,\boldsymbol{R}}\left(\boldsymbol{S};\boldsymbol{Q},\boldsymbol{\sigma}+\varepsilon C\left(\mathbf{u},\boldsymbol{R},\boldsymbol{S}\right)\right)=0 \qquad \forall\boldsymbol{Q}\in\mathrm{SO}_\Omega \tag{11}$$

$$c(\mathbf{u},\boldsymbol{R},\boldsymbol{S},\boldsymbol{\lambda})=0 \qquad \forall\boldsymbol{\lambda}\in T_\Omega. \tag{12}$$

The penalization parameter ε has the dimension of an elastic modulus, and in practice we set it equal to the typical stress $\hat{\sigma} := \rho\hat{g}\hat{L}$, with $\hat{L}$ and $\hat{g}$ typical length and acceleration, respectively. We proceed to solve system (9–12) using projected Newton iterations; we describe how to do so efficiently in Appendix B. Differences with the original approach from Trusty et al. [2022] are outlined in Section B.4.

4.2 Combination with Neural Quadrature

Our Mixed FEM solver does not overly restrict the choice of quadrature formulas, as long as they are of sufficient accuracy. As outlined in Appendix B.2, it mandates for efficiency that the quadrature points used to integrate Equations (10)–(12) coincide with the degrees of freedom of the strain spaces; but we can freely pick the location of those Lagrange polynomial nodes. We can thus combine the Mixed FEM formulation with our Neural Quadrature from Section 3.3. For hexahedral elements we use polynomials of similar degree k for the displacement and tensor spaces, meaning that we can use the same formula of order $d=2k$ for all of our integrals. While the displacement space V_Ω is continuous with nodes positioned according to the mesh, the strain, rotation and stress spaces use a discontinuous Lagrange polynomial basis with nodes collocated to the quadrature points $(\mathbf{y}_p^K)$ inferred in each element. Note that in practice the tensor fields will only be evaluated at said nodes, so we do not need to consider general interpolation.

5 Forward Simulation Results

We have implemented our FEM and Mixed FEM solvers using the `warp.fem` module from the NVIDIA Warp [Macklin 2022] library, which allows us to conveniently express the linear and bilinear forms described in Sections 3.1 and B.1 and provides auto-differentiated numerical integration code with respect to all of the domain and material parameters. Below we assess the efficiency of neural quadrature and Mixed FEM solver, first on a simple dumbbell geometry then on more complex topologies.

5.1 Dumbbell

We define a continuous dumbbell SDF as the union of three analytical cylinders, the middle one being of smaller radius than the other two. We then discretize this SDF on regular grids at resolutions varying from 8^3 to 64^3. We generate quadrature formulas of order 2 (8 points) and 4 (27 points) from this discrete SDF using the Full (regular Gauss–Legendre points and weights), Clip (filtering-out points in the SDF exterior), and our Neural approaches to perform

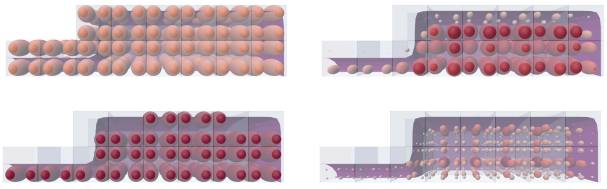

Fig. 4. Visualization of quadrature points generated for an SDF discretized on a grid at resolution 16^3, using order-2 Full (top left) and Clip (bottom left) quadratures, and our Neural quadrature at order 2 (top right) and 4 (bottom right). The non-empty voxels are shown in blue, and the SDF isosurface (extracted at resolution 64^3) is shown in purple. For clarity, only one octant is shown.

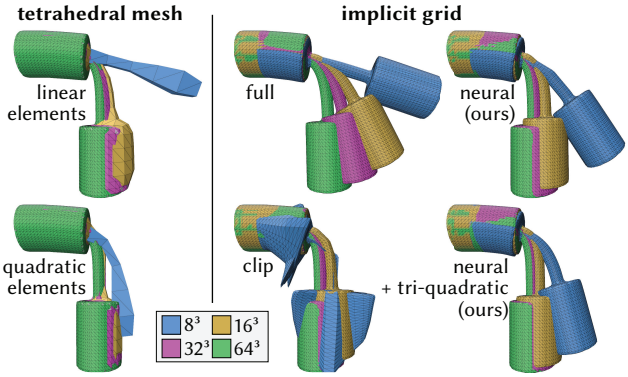

Fig. 5. Comparison of equilibrium behavior for the dumbbell across several SDF grid resolutions and discretizations. *Left*, a tetrahedral mesh is first extracted from the grid via the FlexiCubes algorithm, and simulated with linear and quadratic elements. *Right*, our grid-based simulations are performed with full and clipped quadrature on tri-linear elements, as well as our neural quadrature on both tri-linear and tri-quadratic elements. Color denotes the grid resolution at which the mesh is extracted or the simulation is performed, respectively. The grid simulation is interpolated to a high-resolution surface for visualization.

both displacement-only and Mixed FEM simulations. The deformation is visualized by embedding the isosurface extracted using dual marching-cubes at 64^3 resolution within the simulation grid, as illustrated in Figure 4. We use the Stable Neo–Hookean elastic model from Smith et al. [2018] with Poisson ratio $\nu = 0.4$.

Cantilever. The first experiment consists in clamping one end and letting the (soft) dumbbell sag under gravity, studying the impact of the grid resolution on the achieved equilibrium shape for different quadrature formulae. We also compare our non-conforming approach with the simulation of a tetrahedral mesh generated from the discrete grid using the algorithm from Shen et al. [2023]. Figure 5 shows the equilibrium shapes obtained using our Mixed FEM formulation — we also performed this experiment with displacement-only FEM and obtained visually identical results. At the fine 64^3 resolution, all experiments converge to the same shape. At coarse resolutions however, the Full quadrature and the linear tetrahedral mesh underestimate the deformation the most, while the Clip quadrature suffers from instabilities. The quadratic tetrahedral mesh and the tri-quadratic embedded simulation

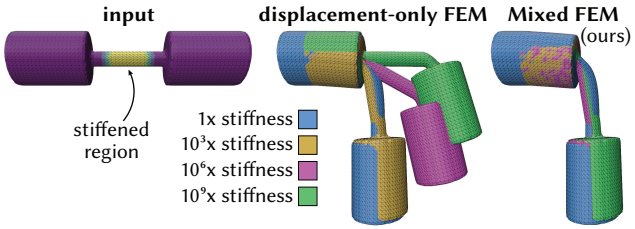

Fig. 6. At high stiffness ratio, displacement-only FEM suffers from slow convergence, while Mixed FEM does not. Here, the center region of the dumbbell is stiffened by an increasing factor, and in each case the Newton loop is truncated after 250 iterations.

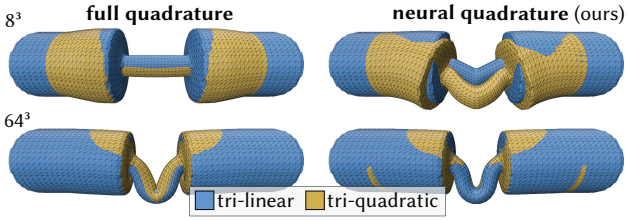

Fig. 7. Comparison of buckling shapes across resolution (*by row*), element types (*by color*), and quadrature strategies (*by column*).

(using order-4 neural quadrature) perform best, followed by the order-2 neural quadrature with trilinear displacements.

Large stiffness ratios. We now increase the elastic modulus of the middle region of the dumbbell (Figure 6) with stiffness ratios up to $10^9\times$. At lower ratios, displacement-only and Mixed FEM perform identically, but for the higher ratios classic FEM suffers from high damping of the rotational mode in the stiff regions, and remains far from the converged shape even after 250 Newton iterations. This is consistent with the results from Trusty et al. [2022].

Buckling. Finally, Figure 7 looks at the impact of changing the polynomial degree of the displacement field for the Full and Neural quadrature formulae, at 8^3 and 64^3 resolution. We observe little impact for the coarse Full and fine Neural buckling simulations; the former fails to take into account the thinner part for both degrees, while the latter results in identical converged shapes. The tri-quadratric displacement is most interesting for the coarse Neural simulation, with an equilibrium shape much closer to the high-resolution solution than with tri-linear displacements.

5.2 Complex Geometries

Heterogeneous material. We simulate a slab of material with heterogeneities roughly the size of one voxel, so that the embedding grid is effectively dense (top left). As shown in Figure 8, using the regular Full quadrature, the material behaves as if it was homogeneous, with globally uniform strain. The Clip quadrature also yields incorrect behavior, as the strain is no longer transmitted away from the dense clamped regions. Our neural quadrature successfully captures the intricacies of the material at no additional cost.

Interactive editing. As our framework allows simulation of arbitrarily complex and evolving material topology without the need

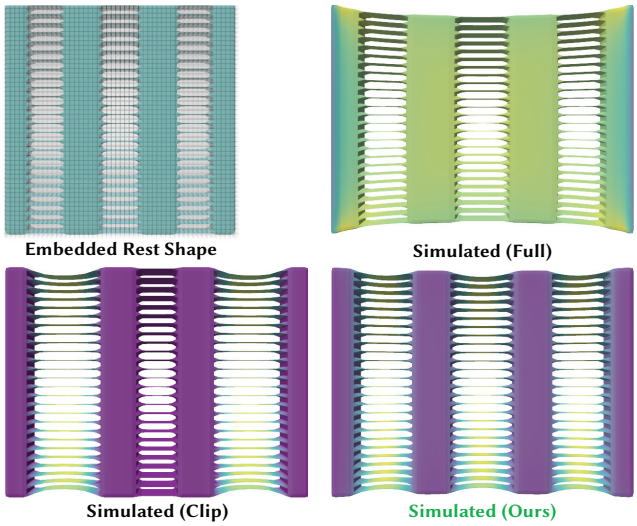

Fig. 8. A slab with sub voxel-sized features simulated with Full, Clip, and Neural quadrature formulas. Shading denotes the norm of the strain tensor S relative to the current configuration.

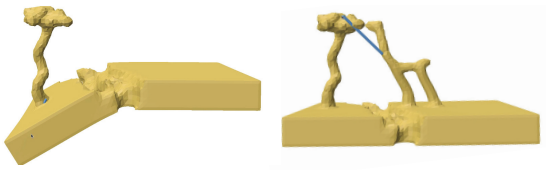

Fig. 9. Interactive sculpting and simulation of physics-enabled clay.

for expensive remeshing, a natural application is a physics-ready virtual playground where the user may interactively add or subtract material and immediately see how it responds to applied forces (Figure 9 and video).

6 Physics-aware Reconstruction

Not only can our neural quadrature handle evolving material domains, it does so in a differentiable way. We exploit this ability to demonstrate physics-aware mesh reconstruction from multiple views. But first we describe how we can efficiently compute the adjoint of our simulations.

6.1 Simulation Adjoint

We consider a loss function $\mathcal{L}(\mathbf{p}, \mathbf{q})$ to be minimized, with $\mathbf{p}$ the vector of material and/or shape parameters that we want to optimize, and with $\mathbf{q}$ the simulation state; $\mathbf{q} := \mathbf{u}$ for displacement-only FEM, and $\mathbf{q} := (\mathbf{u}, \boldsymbol{S}, \boldsymbol{R}, \boldsymbol{\sigma})$ for our Mixed FEM formulation from Section 4. As $\mathbf{q}$ is the result of a forward simulation, it depends in turn on the parameters $\mathbf{p}$. Performing gradient-based optimization therefore requires evaluating

$$\frac{d\mathcal{L}}{d\mathbf{p}} = \frac{\partial \mathcal{L}}{\partial \mathbf{p}} + \frac{\partial \mathcal{L}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \mathbf{p}}.$$

One may choose to use full auto-differentiation of the simulator code for all of the above terms. However, evaluating $\frac{\partial \mathbf{q}}{\partial \mathbf{p}}$ requires backtracking through the whole simulation loop — potentially comprising many solver iterations — which is costly both in wall time and memory usage. We avoid this overhead by combining auto-differentiated and analytical adjoint computations: by definition, $\mathbf{q}$ must satisfy an equilibrium condition, either Equation (2) for displacement-only FEM or Equations (9)–(12) for Mixed FEM. For brevity of notation, let us write this equilibrium condition as $\mathbf{f}(\mathbf{p}, \mathbf{q}) = 0$; the implicit function theorem allows us to express the loss gradient as

$$\frac{d\mathcal{L}}{d\mathbf{p}} = \frac{\partial \mathcal{L}}{\partial \mathbf{p}} + \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{q}} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{q}} \right)^{-1}}_{\frac{\partial \mathcal{L}}{\partial \mathbf{f}}} \frac{\partial \mathbf{f}}{\partial \mathbf{p}}.$$

As $\frac{\partial \mathbf{f}}{\partial \mathbf{q}}$ can be recognized as the Hessian of the incremental energy potential, computing $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$ amounts to solving one linear system similar similar to one Newton iteration from the forward pass.[3] The right-multiplication of $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$ by $\frac{\partial \mathbf{f}}{\partial \mathbf{p}}$ is then achieved through auto-differentiation of the linear form assembly code, which is directly provided by the `warp.fem` library [Macklin 2022] with which our solver is implemented.

Note that in our framework, we do not have to give special treatment to shape derivatives versus material parameter derivatives. The simulator is aware of the material domain through quadrature points and weights, for which the adjoint computation does not require particular considerations. We can then get the derivatives with respect to the implicit surface by backpropagating through the MLP network from Section 3.3.

6.2 Physics-aware Reconstruction Framework

We leverage the FlexiCubes [Shen et al. 2023] discrete implicit surface representation, which consists of SDF values and displacements at nodes of a regular grid, plus per-cell parameters adjusting the isosurface — effectively, per-cell, per-vertex SDF values with variable vertex positions. This representation has been shown to perform well in conjunction with differentiable rasterization [Laine et al. 2020], with stronger ability at capturing sharp features than tet-based alternatives [Shen et al. 2021].

Previously Shen et al. [2023] showed decoupled shape and material optimization, first recovering geometry via FlexiCubes and next optimizing for material properties using a differentiable simulator. In stark contrast we now describe a fully-coupled single stage pipeline wherein gradients from our simulator directly affect reconstructed geometry. We adjust the physical behavior of the reconstructed object by varying not only material parameters, but also its rest shape; all the while ensuring that renderings of said rest shape remain close to the target.

Loss functions and preconditioning. On top of the geometric and rendering-based reconstruction losses described by Shen et al. [2023], which we regroup concisely as $\mathcal{L}_{\text{FC}}$, we add a new physics loss function $\mathcal{L}_{\text{phys}}$ based on the displacement $\mathbf{u}$ and stress $\boldsymbol{\sigma}$ fields

[3] To compute the exact gradient, we should not perform SPD projection of the elasticity Hessian. However in practice, this allows for a simpler and more efficient solve with little impact on the descent direction, so we use it in the backwards pass as well.

resulting from the simulation over a timestep Δ_t,

$$\mathcal{L}_{\text{phys}}(\Delta_t, \ell_{\mathbf{u}}, \ell_{\boldsymbol{\sigma}}) := \sqrt[p]{\int_{\Omega} \frac{1}{|\det \mathrm{d}\Omega|} \left(\ell_{\mathbf{u}} \|\mathbf{u}\|^p + \ell_{\boldsymbol{\sigma}} \|\boldsymbol{\sigma}\|^p\right)},$$

where $\ell_{\mathbf{u}}$ and $\ell_{\boldsymbol{\sigma}}$ are constant scaling factors for the displacement and stress terms, and the loss power p allows us to skew the global loss towards either the average or the maximum local loss (in practice we always use $p = 8$). The $\frac{1}{|\det \mathrm{d}\Omega|}$ term scales the local loss inversely to the infinitesimal domain measure to prevent the empty domain from being a trivial optimum.

We emphasize that the $\mathcal{L}_{\text{FC}}$ and $\mathcal{L}_{\text{phys}}$ losses both affect the shape of the reconstructed model and will oppose each other; tuning the $\ell_{\mathbf{u}}$ and $\ell_{\boldsymbol{\sigma}}$ coefficient allow biasing the result towards better reconstruction fidelity or physical performance. Moreover, activating $\mathcal{L}_{\text{phys}}$ right from the beginning of the optimization is not productive; the initial guess of the FlexiCubes reconstruction consists in random SDF values, leading to many disconnected material pieces, so that running the physical simulation at such early stage is not meaningful. Instead, we run the first 30% of the optimizer iterations with $\mathcal{L}_{\text{FC}}$ only, then add $\mathcal{L}_{\text{phys}}$. To ensure a smooth transition, we also increase the simulation timestep Δ_t progressively.

In practice, performing gradient descent on $\mathcal{L}_{\text{phys}}$ tends to produces bumpy or fractured surfaces that, while yielding low values of the physics loss, are not visually pleasing. We overcome this issue by preconditioning the grid parameters that are being optimized for (vertex displacement and SDF value) with a smoothing function. To this effect, we apply a convolution with a Gaussian blur kernel before passing those parameters to the FlexiCubes reconstruction and Mixed FEM simulation. In a similar fashion, we can optionally enforce symmetry of the optimized shape by applying a symmetric preconditioner to the raw grid parameters.

Finally, adding a loss term $\mathcal{L}_{|e|}$ penalizing the total sum of edge lengths of the extracted triangular mesh is helpful for reducing the appearance of unwanted geometry like floaters or protruding details — under the condition that this term remains small compared to the reconstruction and physics losses.

6.3 Physics-aware Reconstruction Results

For the following examples, we render synthetic views of a target mesh and use the physics-aware reconstruction framework described above to reconstruct an implicit surface with desirable physical characteristics. We emphasize that during this process, the optimizer has no knowledge of the target mesh topology or 3d positions, i.e., has no strong prior.

Stress minimization. We first apply our method to optimize the shape of an aluminum hook so that stress under some predefined load is minimized (i.e., we use $\mathcal{L}_{\text{phys}}$ with $\ell_{\mathbf{u}} = 0$ and $\ell_{\boldsymbol{\sigma}} = 1$). We use the Stable Neo–Hookean material from Smith et al. [2018] with Young Modulus $E_Y = 10$GPa, Poisson ratio $\nu = 0.33$, and volumetric mass $\rho = 2700\text{kg.m}^{-3}$, and a FlexiCubes grid with resolution 64. A force of 6kN is applied to the curved portion of the hook while clamping the top of the slit; see Figure 10. Over the course of the optimization the physics loss $\mathcal{L}_{\text{phys}}$ is reduced by more than an order of magnitude, with the maximum stress on the surface being similarly reduced.

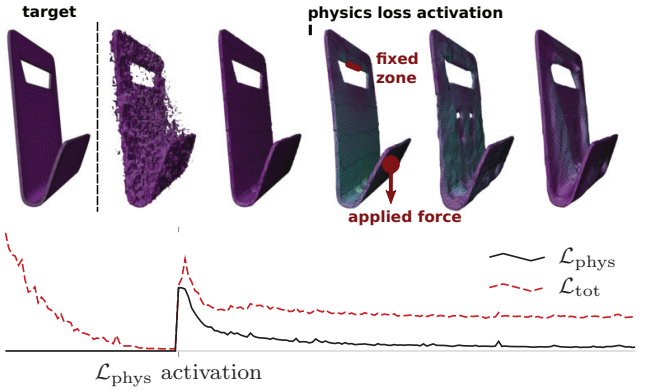

Fig. 10. Bracket topology optimized to minimize stress given a predefined load. Top: target model (leftmost), then timeline of combined shape reconstruction and stress minimization. The physics-aware loss and prescribed force get activated on the fourth image from the left; shading indicates surface stress intensity. Bottom: evolution of the physics and total losses over iterations.

Soft chairs. To evaluate our method on more challenging material topology and nonlinear effects, we select 18 representative chair models from the Pix3D dataset [Sun et al. 2018] and equip them with a rubber-like material, with volumetric mass $\rho = 1000\text{kg.m}^{-3}$, Young modulus $E_Y = 10MPa$ and Poisson ratio $\nu = 0.47$. Emulating the effect of one person sitting on the chair, we apply a downward force of 2.5kN on the seat and a backward force of 0.5kN on the backrest, with a random perturbation of 10% of the force direction and point of application at each iteration. Since we do not know in advance the 3d location of those features, we define our forces in a volumetric fashion over a predefined region of the reconstruction bounding box and scale them according to the actual amount of material in the region. The bottom 5% of each object is kept fixed. We use a timestep $\Delta_t = 3$s, loss scaling parameters $\ell_{\mathbf{u}} = 1$ and $\ell_{\boldsymbol{\sigma}} = 0.25$, and a FlexiCubes resolution of 64. We run the optimization for 1,000 gradient descent iterations, and for each of those run five Newton steps of Mixed FEM simulation, which in total takes about 15 to 25 minutes per model (depending on the number of active voxels) on a pair of NVIDIA GeForce RTX 3080Ti GPUs.

The results are depicted in Figure 11. While applying the forces to the chairs reconstructed without the physics-aware loss usually leads to a complete collapse, the chairs reconstructed with $\mathcal{L}_{\text{phys}}$ demonstrate much stronger resistance and are easily able to recover their original shape once the perturbations cease being applied. The optimization generally reinforces the chair legs and the seat–backrest junction, but with variations depending on the actual topology, such as the presence (or not) of armrests. Since we perform reconstruction from images without any strong shape prior — starting from random SDF values, topological changes are mandatory in the first stages of the optimization. Even once the reconstructed geometry has started to resemble the target, we keep observing emerging topological changes compared to the mesh used as source; some of which are highlighted in Figure 12. We emphasize that even in the rare cases where the genus is not modified by the physics loss, the changes to the surface are drastic enough that shape-differentiable simulators working on

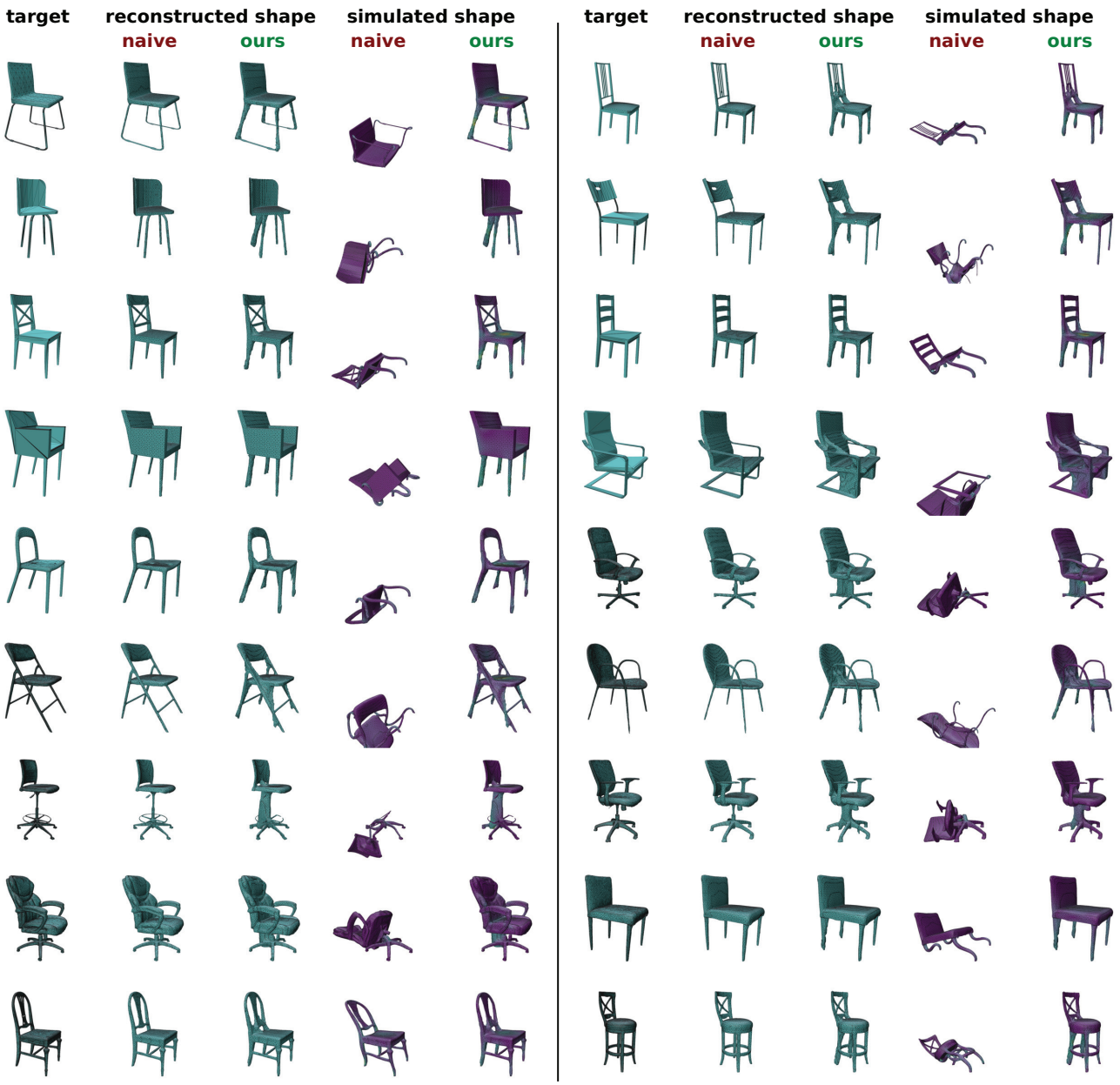

Fig. 11. Physics-aware image-based reconstruction of chair models from the Pix3d dataset such that they can sustain prescribed forces despite being made of a very soft material. For each model, from left to right, target shape, reconstructed shape without (naive) then with (ours) physics-aware loss, simulation of the reconstructed shape without (naive) then with (ours) physics-aware loss. We assume a homogeneous material with density $\rho = 1000kg.m^{-3}$, Young modulus $E_Y = 10$MPa and Poisson ratio $\nu = 0.47$, and apply a downward force of 2.5kN on the seat and a backward force of 0.5kN on the backrest. On simulation pictures, hue indicates relative stress intensity.

conforming meshes would require frequent volumetric remeshing [Huang et al. 2024; Tozoni et al. 2021], which our implicit approach avoids entirely.

Stability. Previously we kept the bottom of the chairs fixed, which is justified given the strong downward applied force. Here, inspired by similar experiments in Guo et al. [2024] and Ni et al. [2024], we show that our technique can also be leveraged to increase the stability envelope of the reconstructed models. We replace the bilateral clamping with an unilateral constraint modeling the ground–chair contact, and update our Newton loop with an active-set formulation. We use a much stiffer material so that the chair behaves rigidly ($E = 100$GPa), apply a downward-and-backward-pointing force on the backrest, and pick a model that

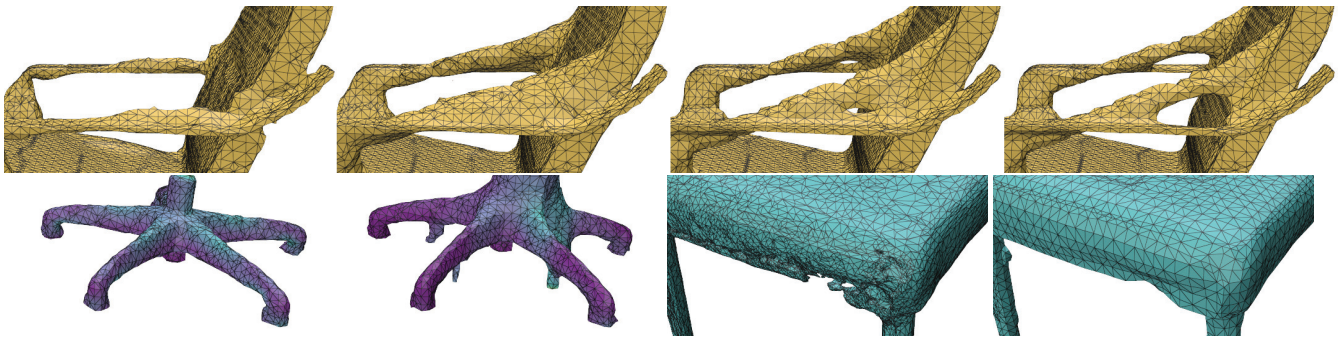

Fig. 12. Details of some topology changes in our soft chairs example. Top: Additional backrest support being grown then carved out. Bottom, left: Additional feet being grown for support on an office chair. Right: Repairing of a damaged target shape.

Fig. 13. Optimizing the stability such that the chair remains stable to a force applied on the backrest. From left to rotation, reconstructed shape without (naive) then with (ours) physics-aware loss, simulated reconstruction without (naive) then with (ours) physics-aware loss. The yellow ball on the rest geometries indicates the position of the center of mass.

Fig. 14. Concurrent optimization of the shape and Young modulus of the chair. From left to right, target shape, simulated reconstructions without (naive) and with (ours) physics-aware loss. Shading indicates regions that are made stiffer.

looks propitious to toppling. Figure 13 shows that the physics-aware loss will add material to the front of the chair such that the center of mass moves forward and resists the applied push.

Material optimization. Up until now we only allowed the optimizer to modify the shape of the model, keeping the material homogeneous; here we also allow modification of the Young Modulus. This makes the problem somewhat easier, as now the physics loss and reconstruction loss can act on orthogonal parameters. In our framework, we can just set $\ell_{\mathbf{u}}$ and ℓ_{σ} to small values so that the optimizer will favor $\mathcal{L}_{\text{FC}}$ over $\mathcal{L}_{\text{phys}}$ for the shape parameters. Figure 14 shows the result of this process under the constraint that the Young modulus should not be increased more than $10^4\times$ and with additional L_1 regularization of the stiffening parameter. Unsurprisingly, the legs of the chair and the junction between legs and seat are the regions that the optimizer prioritize for stiffening.

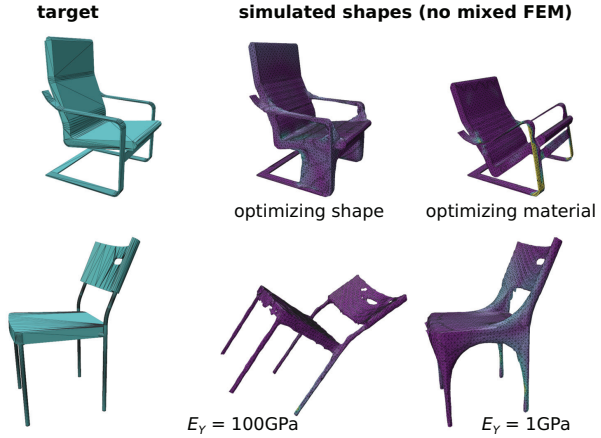

Fig. 15. Results using displacement-only FEM instead of our Mixed FEM. Top: from left to right: target shape, simulation of shape reconstructed with topology optimization, simulation of shape reconstructed with material stiffness optimization. Bottom: stability optimization, from left to right: target shape, simulation of shape reconstructed with $E_Y = 100$GPa, then with $E_Y = 1$GPa.

Fig. 16. Simulation of the reconstructed elastic dumbbell from Section 5.1 without material optimization (left), with material optimization and displacement-only FEM (middle), with material optimization and our mixed FEM (right). Hue shows material stiffness scaling, from 1× (purple) to 250× (yellow). Slower convergence of displacement-only FEM with high stiffness ratios causes the optimizer to underestimate the physics loss $\mathcal{L}_{\text{phys}}$, leading to insufficient stiffening and more sagging when re-simulating the reconstructed object.

Ablation studies. Having demonstrated the physics-aware reconstruction capabilities of our framework, we proceed to study the importance of its individual components, and show results in

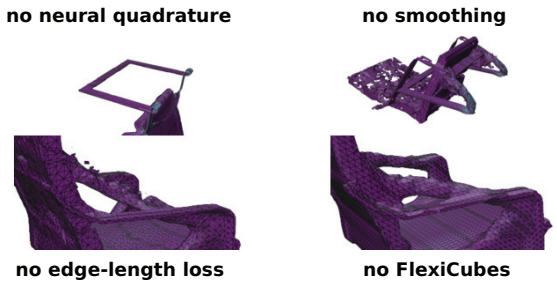

Fig. 17. Simulation of reconstructed shapes with various parts of our framework removed: top left: no Neural quadrature; top right: no smoothing preconditioner; bottom left: no edge-length loss; bottom-right: using dual marching-cube instead of FlexiCubes (cropped details).

Figure 17. First, replacing the neural quadrature with Full or Clip quadrature formulas hinder convergence entirely. Indeed, the gradient of $\mathcal{L}_{\text{phys}}$ with respect to the vertex SDF values becomes zero, so that the optimizer will only move the vertex positions, whose motion is constrained by the FlexiCubes parameterization to stay under one voxel-size. Next, when removing the smoothing preconditioner, the optimizer does manage to reduce the physics loss, but the reconstructed surface is hardly usable. Removing the edge length loss $\mathcal{L}_{|e|}$ makes the reconstructed surfaces bumpier, and tend to produce floaters. Finally, sharp features are no longer well captured when limiting the optimized parameters to the vertex SDF values, i.e., falling back to a standard dual marching-cube.

We also show in Figure 15 that our method keeps working when using displacement-only FEM rather than Mixed FEM, and thus, should be compatible with other differentiable simulators that support hexahedral elements [e.g., Huang et al. 2024]; but with degraded robustness. For our initial soft chair optimization problem, classic FEM yields a reconstruction similar to the Mixed FEM case from Figure 11. For the material optimization experiment the results are still reasonable, but, due to the slower convergence of displacement-only FEM with high stiffness ratios, the physics loss is underestimated and the reconstructed model is subject to more sagging than in the Mixed FEM version from Figure 14; see also Figure 16 for a simpler variant of this experiment. Results are worst for the stability optimization example; here, we need to reduce the stiffness by two orders of magnitude to obtain a stable reconstruction.

Quadratic elements. We verify that our method also works with higher-order elements. Figure 18 demonstrates optimization of an elastic bridge model under prescribed load on a 48^3 grid, using tri-quadratic displacements and the order-4 27-points learned quadrature. As the resolution of the FlexiCubes grid is already high enough to resolve thin features of the target shape, the higher-order result remains qualitatively similar to the trilinear version.

Physics-aware photogrammetry. Our physics-aware shape reconstruction formulation may also be integrated into more complex photogrammetry pipelines to allow for the joint optimization of shape, lighting, and both physical and rendering materials. We leverage NVDIFFREC from Munkberg et al. [2022], which supports FlexiCubes as its geometry representation. For this example we use again the setup described in Section 6.2; this time, instead of defining our rendering target as random synthetic views of a known mesh, we use a fixed list of 100 images and corresponding camera transforms from the NeRF dataset [Mildenhall et al. 2020]. We first optimize the rendering loss L_{render} from the NVDIFFREC pipeline without modification. After the shape starts converging, we begin blending in the physics loss function L_{phys}. We can again optimize both the shape and material parameters of our reconstructed object to minimize sagging under a prescribed force (Figure 19).

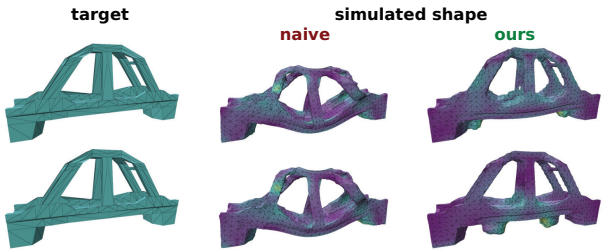

Fig. 18. From left to right: target shape and simulation of the shapes reconstructed without (naive) and with (ours) physics-aware loss. Top: using with tri-quadratic elements and the order-4 "neural" quadrature; bottom: using trilinear elements and the order-2 "neural" quadrature.

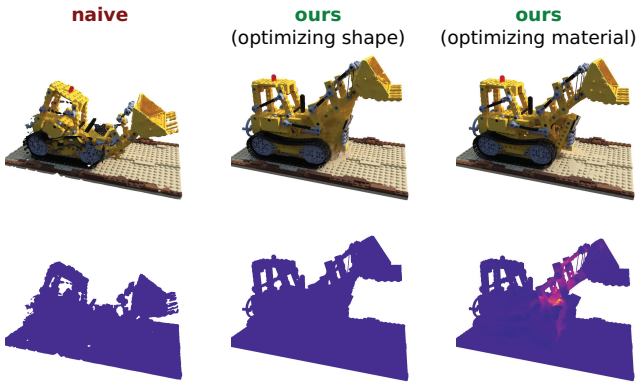

Fig. 19. Physics-aware multiview reconstruction of the Bulldozer scene from NeRF synthetic dataset. Simulation results from left to right: initial shape before L_{phys} is applied, optimizing geometry, optimizing material parameters. Shading indicates regions that are made stiffer.

7 Limitations and Future Work

While our neural quadrature rule can compute integrals over the possibly complex material domain induced by the SDF, the displacement degrees of freedom are still those of the underlying continuous shape functions. As such, even if the SDF defines two disconnected material regions within a given voxel, the simulation won't allow them to separate arbitrarily. This is detrimental for our application as this means that small disconnected pieces of material ("floaters") adjacent to the bulk of the object will tend to stick to the surface instead of falling down — meaning that they will have a low displacement loss and the optimizer won't be eager to prune them. While our edge-length loss helps reduce this phenomenon, other formulations such as the perimeter regularizer from Maestre et al. [2023] would be worth investigating. A possibly more satisfying solution that we intend to explore in

future work is the of addition of new degrees of freedom to the disconnected voxel configurations, in the vein of XFEM [Koschier et al. 2017; Moës et al. 1999] or CPIC [Hu et al. 2018]. Additionally, we do not consider evaluation of integrals over the boundary of the domain, as these are not needed for the presented tasks. If desired, isosurface meshes can easily be extracted and embedded for integration.

Another limitation of our physics-aware reconstruction algorithm is the lack of global convergence, meaning that differences in the initial random SDF values lead to variations in the final optimized shape. The end result is also very sensitive to the choice of loss function; exploring the definition of more perceptual losses would be an interesting area of research. Our current reconstruction speed is also not yet suitable for interactive applications; we would like to bridge this gap in the future.

In future work, we also want to explore whether other architectures could improve the accuracy and efficiency of our technique. While we have focused on solid elasticity in this work, we also want to take advantage of the analogy between quadrature points and Particle-in-Cell integration to investigate differentiable initialization of MPM simulations from implicit surfaces. Combined with particle-based techniques like PAC-NeRF [Li et al. 2023a], this would allow computing derivatives of the simulation end-state with respect to the material occupancy function, with applications to single-pass reconstruction, identification and shape optimization of plastic materials.

8 Conclusion

We have presented a neural integration technique that improves the quality of voxel-based implicit volume simulations at negligible additional runtime cost, and shown how to combine it with a Mixed FEM solver to efficiently perform elasticity simulation on continuously evolving domains. Our technique allows straightforward differentiation of the simulation results with respect to the implicit volume parameters, making itself particularly suitable for topology optimization tasks, and providing a first foray into physics-enabled shape reconstruction.

References

Grégoire Allaire, François Jouve, and Anca-Maria Toader. 2004. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.* 194, 1 (2004), 363–393. https://doi.org/10.1016/j.jcp.2003.09.032

Martin P. Bendsøe and Ole Sigmund. 2009. *Topology Optimization.* World Scientific.

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

James Bremer, Zydrunas Gimbutas, and Vladimir Rokhlin. 2010. A nonlinear optimization procedure for generalized Gaussian quadratures. *SIAM Journal on Scientific Computing* 32, 4 (2010), 1761–1788. https://doi.org/10.1137/080737046

Franco Brezzi and Michel Fortin. 1991. Mixed and hybrid finite element method. *Springer Series in Computational Mathematics; Vol. 15* (01 1991), 350. https://doi.org/10.1007/978-1-4612-3172-1

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. 2024. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 19457–19467.

Desai Chen, David I. Levin, Wojciech Matusik, and Danny M. Kaufman. 2017. Dynamics-aware numerical coarsening for fabrication design. *ACM Trans. Graph.* 34, 4, Article 84 (2017). https://doi.org/10.1145/3072959.3073669

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022. Neural dual contouring. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.

Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 2016. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. (2016). arXiv:cs.CV/1604.00449

Ronald Cools. 2003. An encyclopaedia of cubature formulas. *Journal of Complexity* 19, 3 (2003), 445–453. https://doi.org/10.1016/S0885-064X(03)00011-6. Oberwolfach Special Issue.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. DiffPD: Differentiable projective dynamics. *ACM Trans. Graph.* 41, 2, Article 13 (Nov. 2021), 21 pages. https://doi.org/10.1145/3490168

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. 2017. A point set generation network for 3D object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 605–613.

Mihai Frâncu, Arni Asgeirsson, Kenny Erleben, and Mads J. L. Rønnow. 2021. Locking-proof tetrahedra. *ACM Trans. Graph.* 40, 2, Article 12 (April 2021), 17 pages. https://doi.org/10.1145/3444949

Arun L. Gain, Glaucio H. Paulino, Leonardo S. Duarte, and Ivan F. M. Menezes. 2015. Topology optimization using polytopes. *Computer Methods in Applied Mechanics and Engineering* 293 (2015), 411–430. https://doi.org/10.1016/j.cma.2015.05.007

Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2020. Learning deformable tetrahedral meshes for 3D reconstruction. In *Advances In Neural Information Processing Systems.*

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Trans. Graph.* 39, 6, Article 190 (Nov. 2020), 15 pages. https://doi.org/10.1145/3414685.3417766

Georgia Gkioxari, Jitendra Malik, and Justin Johnson. 2019. Mesh R-CNN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 9785–9795.

Minghao Guo, Bohan Wang, Pingchuan Ma, Tianyuan Zhang, Crystal Elaine Owens, Chuang Gan, Joshua B. Tenenbaum, Kaiming He, and Wojciech Matusik. 2024. Physically compatible 3D object modeling from a single image. *arXiv preprint arXiv:2405.20510* (2024).

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. 2023. LRM: Large reconstruction model for single image to 3D. *arXiv preprint arXiv:2311.04400* (2023).

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable programming for physical simulation. *ICLR* (2020).

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.* 37, 4, Article 150 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201293

Zizhou Huang, Davi Colli Tozoni, Arvi Gjoka, Zachary Ferguson, Teseo Schneider, Daniele Panozzo, and Denis Zorin. 2024. Differentiable solver for time-dependent deformation problems with contact. *ACM Trans. Graph.* (2024).

Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, et al. 2021. gradSim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)* (2021). https://openreview.net/forum?id=c_E8kFWfhp0

C. Kane, J. E. Marsden, M. Ortiz, and M. West. 2000. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *Internat. J. Numer. Methods Engrg.* 49, 10 (2000), 1295–1325.

Junggon Kim and Nancy S. Pollard. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Transactions on Graphics (TOG)* 30, 5 (2011), 1–19.

Yeongbin Ko, Phill-Seung Lee, and Klaus-Jürgen Bathe. 2017. A new 4-node MITC element for analysis of two-dimensional solids and its formulation in a shell element. *Computers & Structures* 192 (2017), 34–49. https://doi.org/10.1016/j.compstruc.2017.07.003

Dan Koschier, Jan Bender, and Nils Thuerey. 2017. Robust eXtended finite elements for complex cutting of deformables. *ACM Trans. Graph.* 36, 4, Article 55 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073666

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics* 39, 6 (2020).

Jiao Li, Yanjin Guan, Guangchun Wang, Guilong Wang, Haiming Zhang, and Jun Lin. 2020. A meshless method for topology optimization of structures under multiple load cases. In *Structures*, Vol. 25. Elsevier, 173–179.

Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. 2023a. PAC-NeRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512* (2023).

Yue Li, Stelian Coros, and Bernhard Thomaszewski. 2023. Neural metamaterial networks for nonlinear material design. *ACM Trans. Graph.* 42, 6, Article 186 (Dec. 2023), 13 pages. https://doi.org/10.1145/3618325

Zhehao Li, Qingyu Xu, Xiaohan Ye, Bo Ren, and Ligang Liu. 2023b. DiffFR: Differentiable SPH-based fluid-rigid coupling for rigid body control. *ACM Trans. Graph.* 42, 6, Article 179 (Dec. 2023), 17 pages. https://doi.org/10.1145/3618318

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018a. Narrow-band topology optimization on a sparsely populated grid. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.

Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. 2018b. Paparazzi: Surface editing by way of multi-view image processing. *ACM Transactions on Graphics* (2018).
Minghua Liu, Chong Zeng, Xinyue Wei, Ruoxi Shi, Linghao Chen, Chao Xu, Mengqi Zhang, Zhaoning Wang, Xiaoshuai Zhang, Isabella Liu, et al. 2024. MeshFormer: High-quality mesh generation with 3D-guided reconstruction model. *arXiv preprint arXiv:2408.10198* (2024).
Andreas Longva, Fabian Löschner, Tassilo Kugelstadt, José Antonio Fernández-Fernández, and Jan Bender. 2020. Higher-order finite elements for embedded simulation. *ACM Trans. Graph.* 39, 6, Article 181 (Nov. 2020), 14 pages. https://doi.org/10.1145/3414685.3417853
William E. Lorensen and Harvey E. Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal Graphics: Pioneering Efforts that Shaped the Field*. 347–353.
Miles Macklin. 2022. Warp: A high-performance Python framework for GPU simulation and graphics. https://github.com/nvidia/warp. (March 2022). NVIDIA GPU Technology Conference (GTC).
Juan Montes Maestre, Ronan Hinchet, Stelian Coros, and Bernhard Thomaszewski. 2023. ToRoS: A topology optimization approach for designing robotic skins. *ACM Trans. Graph.* 42, 6, Article 194 (Dec. 2023), 11 pages. https://doi.org/10.1145/3618382
Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)* 23, 3 (2004), 449–456.
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*.
Marek Krzysztof Misztal and Jakob Andreas Bærentzen. 2012. Topology-adaptive interface tracking using the deformable simplicial complex. *ACM Trans. Graph.* 31, 3, Article 24 (June 2012), 12 pages. https://doi.org/10.1145/2167076.2167082
Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. 2022. AutoSDF: Shape priors for 3D completion, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 306–315.
Nicolas Moës, John Dolbow, and Ted Belytschko. 1999. A finite element method for crack growth without remeshing. *Internat. J. Numer. Methods Engrg.* 46, 1 (1999), 131–150.
Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting triangular 3D models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 8280–8290.
Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. 2013. OpenVDB: An open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH 2013 Courses*. 1–1.
B. Müller, F. Kummer, and M. Oberlack. 2013. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *Internat. J. Numer. Methods Engrg.* 96, 8 (2013), 512–528. https://doi.org/10.1002/nme.4569
Junfeng Ni, Yixin Chen, Bohan Jing, Nan Jiang, Bin Wang, Bo Dai, Puhao Li, Yixin Zhu, Song-Chun Zhu, and Siyuan Huang. 2024. PhyRecon: Physically plausible neural scene reconstruction. *arXiv preprint arXiv:2404.16666* (2024).
Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (2e ed.). Springer, New York, NY, USA.
Julian Panetta, Abtin Rahimian, and Denis Zorin. 2017. Worst-case stress relief for microstructures. *ACM Trans. Graph.* 36, 4, Article 122 (July 2017), 16 pages. https://doi.org/10.1145/3072959.3073649
Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.
Taylor Patterson, Nathan Mitchell, and Eftychios Sifakis. 2012. Simulation of complex nonlinear elastic bodies using lattice deformers. *ACM Trans. Graph.* 31, 6, Article 197 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366216
Jovan Popović, Steven M. Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. 209–217.
E. Reissner. 1985. On mixed variational formulations in finite elasticity. *Acta Mechanica* 56, 3 (1985), 117–125.
Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. 2015. Microstructures to control elasticity in 3D printing. *ACM Trans. Graph.* 34, 4, Article 136 (July 2015), 13 pages. https://doi.org/10.1145/2766926
Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep marching tetrahedra: A hybrid representation for high-resolution 3D shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS'21)*.
Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Trans. Graph.* 42, 4, Article 37 (July 2023), 16 pages. https://doi.org/10.1145/3592430
J. C. Simo and M. S. Rifai. 1990. A class of mixed assumed strain methods and the method of incompatible modes. *Internat. J. Numer. Methods Engrg.* 29, 8 (June 1990), 1595–1638. https://doi.org/10.1002/nme.1620290802
Breannan Smith, Fernando de Goes, and Theodore Kim. 2018. Stable Neo-Hookean flesh simulation. *ACM Trans. Graph.* 37, 2, Article 12 (Mar. 2018), 15 pages. https://doi.org/10.1145/3180491
Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B. Tenenbaum, and William T. Freeman. 2018. Pix3D: Dataset and methods for single-image 3D shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*.
Davi Colli Tozoni, Yunfan Zhou, and Denis Zorin. 2021. Optimizing contact-based assemblies. *ACM Trans. Graph.* 40, 6, Article 269 (Dec. 2021), 19 pages. https://doi.org/10.1145/3478513.3480552
Ty Trusty, Danny Kaufman, and David I. W. Levin. 2022. Mixed variational finite elements for implicit simulation of deformables. In *SIGGRAPH Asia 2022 Conference Papers (SA'22)*. Association for Computing Machinery, New York, NY, USA, Article 40, 8 pages. https://doi.org/10.1145/3550469.3555418
Chelsea Tymms, Siqi Wang, and Denis Zorin. 2020. Appearance-preserving tactile optimization. *ACM Trans. Graph.* 39, 6, Article 212 (Nov. 2020), 16 pages. https://doi.org/10.1145/3414685.3417857
Nico P. van Dijk, Kurt Maute, Matthijs Langelaar, and Fred van Keulen. 2013. Level-set methods for structural topology optimization: A review. *Structural and Multidisciplinary Optimization* 48 (2013), 437–472.
Michael Yu Wang, Xiaoming Wang, and Dongming Guo. 2003. A level set method for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering* 192, 1 (2003), 227–246. https://doi.org/10.1016/S0045-7825(02)00559-5
Ying Wang, Jasper Verheul, Sang-Hoon Yeo, Nima Khademi Kalantari, and Shinjiro Sueda. 2022. Differentiable simulation of inertial musculotendons. *ACM Trans. Graph.* 41, 6, Article 272 (Nov. 2022), 11 pages. https://doi.org/10.1145/3550454.3555490
Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–11.
Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. 2020. PQ-NET: A generative part Seq2Seq network for 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 829–838.
Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*.
Jonas Zehnder, Yue Li, Stelian Coros, and Bernhard Thomaszewski. 2021. NTopo: Mesh-free topology optimization using implicit neural representations. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 10368–10381. https://proceedings.neurips.cc/paper_files/paper/2021/file/55d99a37b2e1badba7c8df4ccd506a88-Paper.pdf
Ryan S. Zesch, Vismay Modi, Shinjiro Sueda, and David I. W. Levin. 2023. Neural collision fields for triangle primitives. In *SIGGRAPH Asia 2023 Conference Papers (SA'23)*. Association for Computing Machinery, New York, NY, USA, Article 76, 10 pages. https://doi.org/10.1145/3610548.3618225
Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. 2024. GS-LRM: Large reconstruction model for 3D Gaussian splatting. *arXiv preprint arXiv:2404.19702* (2024).

Appendices

A Neural Quadrature Training and Evaluation

For a quadrature of target order d, we use $n_Q := (d/2+1)^3$ quadrature points per voxel and a test polynomial basis $\mathcal{B}_P^d$ of cardinality $n_P := (d+1)^3$, chosen as the 3D tensor product of 1D Lagrange polynomials with Lobatto–Gauss–Legendre nodes (x_i^{LGL}),

$$\mathcal{B}_P^d := \left\{P_{ijk}, 0 \le i,j,k \le d\right\},$$
$$P_{ijk}(x,y,z) := P_i^{\mathrm{LGL}}(x) P_j^{\mathrm{LGL}}(y) P_k^{\mathrm{LGL}}(z),$$
$$P_i^{\mathrm{LGL}}(x) := \frac{\prod_{l\neq i}(x - x_l^{\mathrm{LGL}})}{\prod_{l\neq i}(x_i^{\mathrm{LGL}} - x_l^{\mathrm{LGL}})}.$$

We note $n_\varphi := 2^3$ the number of input SDF values per voxel, and (N_j^φ) the n_φ corresponding trilinear interpolation functions.

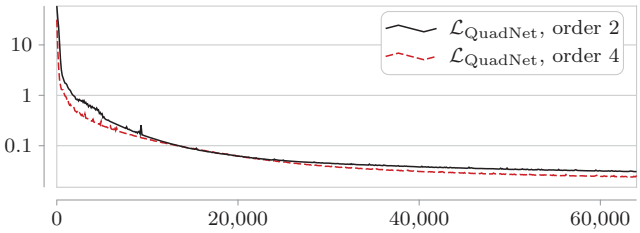

Fig. 20. Convergence of MLP training over iterations.

ALGORITHM 1: Ground truth integrals generation

Input: φ: Tensor of voxel corner SDF values, size $n_B \times n_\varphi$;
Output: Υ^{GT}: ground-truth integral values, size $n_B \times n_P$.
Parameters: n_{GT} : resolution of brute-force integration
$\Upsilon^{\text{GT}} = 0$;
$h := 1/n_{\text{GT}}$;
foreach *voxel b in batch* **do**
  for $0 \leq i, j, k < n_{GT}$ **do**
    `// Interpolate at uniformly sampled locations`
    $x, y, z := (i + 0.5)h, (j + 0.5)h, (k + 0.5)h$;
    $\varphi_{xyz} := \sum_{l<n_\varphi} \varphi_l N_l^\varphi(x, y, z)$;
    if $\varphi_{xyz} < 0.0$ **then** `// In SDF interior`
      foreach *polynomial* P_l *in* $\mathcal{B}_P^d$ **do**
        $\Upsilon^{\text{GT}}_{b,l} = \Upsilon^{\text{GT}}_{b,l} + P_l(x, y, z)h^3$
      end
    end
  end
end

ALGORITHM 2: MLP input normalization layer

Input: φ: Tensor of voxel corner SDF values, size $n_B \times n_\varphi$;
Output: $\overline{\varphi}$: MLP output tensor, size $n_B \times n_\varphi$.
foreach *voxel b in batch* **do**
  `// Normalize SDF gradient at voxel center`
  $\mathbf{g}_b := \sum_{j<n_\varphi} \varphi_{b,j} \nabla N_j^\varphi(0.5, 0.5, 0.5)$;
  $\overline{\varphi_b} = \varphi_b / \left(\|\mathbf{g}_b\| + 10^{-8}\right)$
  `// Shift full and empty voxels closer to origin`
  if $\min_j \varphi_{b,j} > 1$ **then**
    $\overline{\varphi_b} = \overline{\varphi_b} - \min_j \varphi_{b,j} + 1$;
  if $\max_j \varphi_{b,j} < -1$ **then**
    $\overline{\varphi_b} = \overline{\varphi_b} - \max_j \varphi_{b,j} - 1$;
end

Finally, we denote by n_B the batch size, i.e., the number of voxels being simultaneously evaluated.

We reiterate that as our network works independently for each voxel, we do not need to construct a dataset containing macroscopic shapes: for our training data, we simply generate a set of $2^{24} \times 8$ random values sampled from a normal distribution.

For each voxel, the $n_\varphi = 8$ corner values do not need to represent a proper SDF satisfying the eikonal equation; we actually want the network to be robust to improper SDFs, and the first Normalization layer of our network (Algorithm 2) will remap the input such that the gradient at the voxel center is unitary. This Normalization layer is followed by a standard 5-layers-deep MLP with

ALGORITHM 3: MLP output remapping layer

Input: $\mathbf{x}$: MLP output tensor, size $n_B \times n_Q \times 4$;
Output: $\mathbf{y}, \mathbf{w}$: tensors of quadrature point coordinates and weights, size $n_B \times n_Q \times 3$ and $n_B \times n_Q$.
Data: $\mathbf{y}^{\text{GL}}$, w^{GL}: 3D Tensor product of 1D Gauss–Legendre points and weights of order d, size $n_Q \times 3$, n_Q
$\mathbf{y}_{\cdot} = \mathbf{y}^{\text{GL}} + \tanh\left(\mathbf{x}_{\cdot,\cdot,0\ldots2}\right)$;
$w_{\cdot} = w^{\text{GL}} \exp\left(\mathbf{x}_{\cdot,\cdot,3}\right)$;

ALGORITHM 4: Quadrature evaluation layer

Input: $\mathbf{y}$: tensor of quadrature point coordinates, size $n_B \times n_Q \times 3$;
  w: tensor of quadrature point weights, size $n_B \times n_Q$.
Output: Tensor Υ of integral values for all Lagrange polynomials in the test basis $\mathcal{B}_P^d$, size $n_B \times n_P$.
foreach *polynomial* P_l *in* $\mathcal{B}_P^d$ **do**
  $\Upsilon_{\cdot,l} = \sum_{q<n_Q} w_{\cdot,q} P_l(\mathbf{y}_{\cdot,q})$
end

ALGORITHM 5: Loss layer

Input: $\mathbf{y}$: tensor of quadrature point coordinates, size $n_B \times n_Q \times 3$;
  w: tensor of quadrature point weights, size $n_B \times n_Q$; Υ^{GT}: ground-truth integral values, size $n_B \times n_P$.
Output: $\mathcal{L}_{\text{QuadNet}}$: scalar loss.
Parameters: $\gamma_\square$: scaling factor for quadrature point interior loss; $\gamma_\star$: scalar factor for the conditioning loss
$\Upsilon := \text{EvalQuadrature}(\mathbf{y}, \mathbf{w})$; `// Algorithm 4`
$Q_K := \|\Upsilon - \Upsilon^{\text{GT}}\|_2^2$;
$Q_\square := \|\mathbf{y} - \text{clamp}(\mathbf{y}, min = 0, max = 1)\|_2$;
$Q_\star := \left\|\log(\max_{q<n_Q} w_{\cdot,q}) - \log(\min_{q<n_Q} w_{\cdot,q})\right\|_1$;
$\mathcal{L}_{\text{QuadNet}} := Q_K + \gamma_\square Q_\square + \gamma_\star Q_\star$;

ReLu activations, then a final Remapping layer (Algorithm 3) yielding the final quadrature points and weights for each voxel.

For each randomly generated voxel, we generate the ground truth data through brute-force integration at resolution 32^3 of all Lagrange polynomials of our chosen basis $\mathcal{B}_P^d$ multiplied by the indicator function of the SDF interior $\varphi < 0$ (Algorithm 1). This ground truth is compared to the integrals computed using the inferred quadrature points (Algorithm 4) as part of our overall training loss (Algorithm 5), which also incorporates penalties for points drifting outside the voxel and large weight ratios.

We additionally generate at test set of 2^{10} distinct random voxels and associated ground truth integral values, and periodically evaluate the loss function and those as the network is training; resulting curves are shown in Figure 20.

We choose the training batch size for the AdamW optimizer as the highest that can fit in our GPU memory; in our case, $n_B = 2^{18}$. Once training is finished, we evaluate the quality of our network using yet another validation set of random voxels and ground truth integrals pair — this time, with random values sampled according to a uniform distribution. We visualize separately the integration error and conditioning loss for different order and values of the conditioning penalty coefficient $\gamma_\star$ in Figure 21. Generally, we want to pick $\gamma_\star$ such that it yields a reasonable conditioning but not at the detriment of integration accuracy; in our examples, we

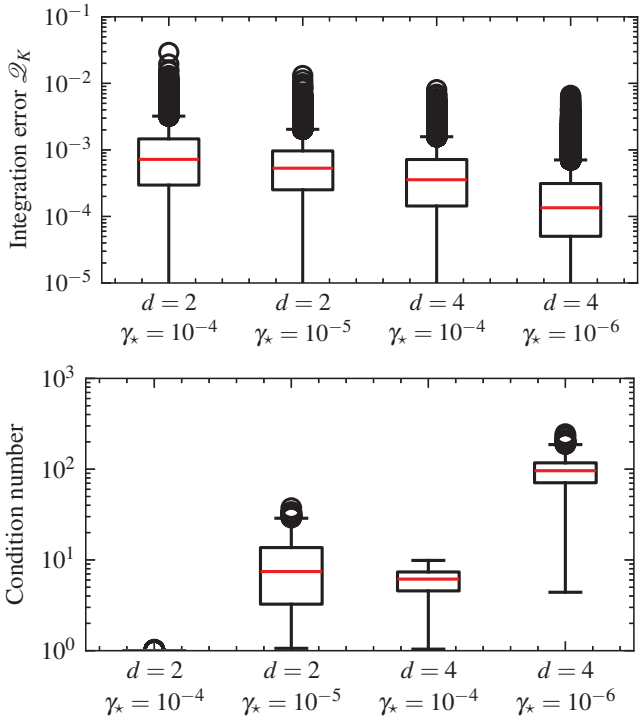

Fig. 21. Statistics of integration error and conditioning $\max w_j / \min w_j$ over 1,000 random voxels for networks trained with order d and conditioning loss scaling factor $\gamma_\star$.

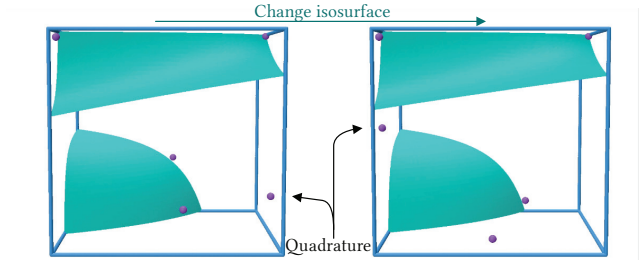

Fig. 22. Using a linear program to select a sparse subset of quadrature points can lead to large jumps in quadrature point position for small changes in isovalues.

use networks trained with $\gamma_\star = 10^{-5}$ for order-2 and $\gamma_\star = 10^{-6}$ for order-4 network.

An advantage of our neural quadrature scheme is that it produces, by construction, smooth, differentiable output. Sparse subset selection methods via linear programming [Longva et al. 2020] produce accurate, efficient quadrature schemes but the under constrained nature of the subset selection problem, and the lack of additional regularizers, lead to large jumps in quadrature point location as a function of isosurface value (Figure 22), making them unsuitable for gradient-based optimization.

Figure 23 shows that our neural integration rule is also significantly more accurate than the Clip quadrature of the same degree, and more accurate than the weight-only moment-fitting approach from Müller et al. [2013] when using the same number of

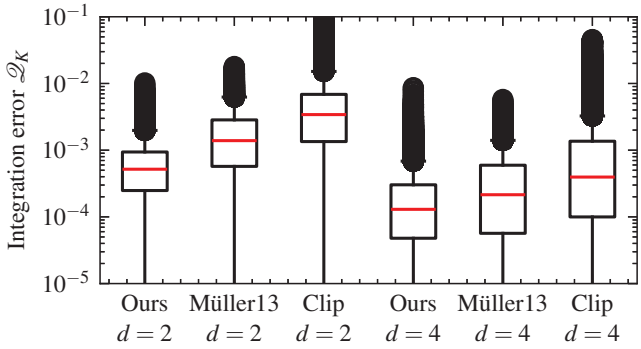

Fig. 23. Statistics of integration error over 1,000 random voxels for our neural quadrature, the weight-only moment-fitting technique from Müller et al. [2013], and the Clip quadrature at order d.

quadrature points, though the gap reduces with integration order. Note that the conditioning number of the Clip a quadrature will be infinite for any partially filled voxel, and that the method from Müller et al. [2013] may produce null or negative weights, while our neural approach ensures positivity and reasonable weight ratios. On an NVIDIA GeForce RTX 3080Ti GPU, computing the order-4 quadrature points for 2^18 voxels takes 6ms for the Clip quadrature; 16ms for our neural quadrature; and more than 1s for the technique from Müller et al. [2013], which requires evaluating the target moments using high-resolution integration, here with 2^12 samples per voxel.

B Four-field Mixed FEM Implementation

B.1 Newton Optimizer

We proceed to solve system (9–12) using projected Newton iterations to compute the step direction $(\delta\mathbf{u}, \delta S, \delta R, \delta\boldsymbol{\sigma})$, with $\delta R \in \mathrm{Skew}_\Omega$ such that $R^{k+1} = R^k + R^k\delta R$. Linearizing the residual around the current iterate $(\mathbf{u}^k, S^k, R^k, \boldsymbol{\sigma}^k)$ yields the linear forms

$$\psi^k_{,S} := \psi_{,S}\left(S^k; \cdot\right), \qquad c^k := c\left(\mathbf{u}^k, S^k, R^k, \cdot\right),$$

and the bilinear forms

$$c^k_{,S} := c_{,S}\left(R^k; \cdot, \cdot\right),$$
$$c^k_{,R}(\delta R, \boldsymbol{\lambda}) := \int_\Omega R^k\delta R S^k : \boldsymbol{\lambda}^T,$$
$$h^k(\delta S, \boldsymbol{\tau}) := \int_\Omega \delta S : \Pi\left(\frac{\partial^2\Psi}{\partial S^2}\left(S^k\right)\right) : \boldsymbol{\tau} + \varepsilon\int_\Omega \delta S : \boldsymbol{\tau}, \text{ and}$$
$$\epsilon^k(\delta R, \boldsymbol{\omega}) := \varepsilon\int_\Omega (\delta R S^k) : (\boldsymbol{\omega} S^k),$$

where the Π operator removes negative eigenvalues from the Hessian of Ψ. At each Newton iteration, we thus solve

$$\begin{aligned}
a(\mathbf{u}^k + \delta\mathbf{u}, \mathbf{v}) + c_{,\mathbf{u}}(\mathbf{v}, \boldsymbol{\sigma}^k + \delta\boldsymbol{\sigma}) &= b(\mathbf{v}) && \forall \mathbf{v} \in V_\Omega,\\
h^k(\delta S, \tau) + c^k_{,S}(\boldsymbol{\tau}, \boldsymbol{\sigma}^k + \delta\boldsymbol{\sigma} + \varepsilon C^k) &= -\psi^k_{,S}(\boldsymbol{\tau}) && \forall \boldsymbol{\tau} \in \mathrm{Sym}_\Omega,\\
\epsilon(\delta R, \boldsymbol{\omega}) + c^k_{,R}(\boldsymbol{\omega}, \boldsymbol{\sigma}^k + \delta\boldsymbol{\sigma} + \varepsilon C^k) &= 0 && \forall \boldsymbol{\omega} \in \mathrm{Skew}_\Omega,\\
c_{,\mathbf{u}}(\delta\mathbf{u}, \boldsymbol{\lambda}) + c^k_{,S}(\delta S, \boldsymbol{\lambda}) + c^k_{,R}(\delta R, \boldsymbol{\lambda}) &= -c^k(\boldsymbol{\lambda}) && \forall \boldsymbol{\lambda} \in T_\Omega,
\end{aligned} \tag{13}$$

with C^k the current constraint residual, $C^k := C(\mathbf{u}^k, S^k, R^k)$.

We equip our Newton loop with a backtracking line-search; however, as we are optimizing under equality constraints, iterates will not remain perfectly feasible and thus we can't directly use the incremental potential (1) as our objective function. Instead we adopt the merit function

$$\phi(\mathbf{u}, \boldsymbol{S}, \boldsymbol{R}) := \frac{1}{2}a(\mathbf{u}, \mathbf{u}) - b(\mathbf{u}) + \psi(\boldsymbol{S}) + \int_\Omega E_Y \|C(\mathbf{u}, \boldsymbol{R}, \boldsymbol{S})\|,$$

combined with Armijo's acceptance rule [Nocedal and Wright 2006].

Rotation regularization. In practice, we find that the robustness of convergence can be increased by augmenting the penalization term ε in the bilinear form ϵ^k with an additional rotation regularization coefficient $\vartheta > 0$. Indeed, we want the skew-symmetric update $\delta\boldsymbol{R}$ to remain small enough that it still represents a valid rotation increment. In our examples we choose ϑ equal to the residual rotational stress norm, $\vartheta = \|\boldsymbol{R}^T\sigma - \sigma^T\boldsymbol{R}\|$. Note that this additional regularization term does not change the problem solution. We do not add the ϑ regularization term when assembling the linear system corresponding to the backward step in the simulation adjoint computation.

B.2 Discrete Mixed Elements

After choosing discrete basis for our element spaces and performing numerical integration for all the terms of Equation (13), computing the Newton step direction amounts to solving the linear system

$$\begin{bmatrix} A & & & C_{,\mathbf{u}}^T \\ & H^k & & C_{,\boldsymbol{S}}^{k,T} \\ & & E & C_{,\boldsymbol{R}}^{k,T} \\ C_{,\mathbf{u}} & C_{,\boldsymbol{S}}^k & C_{,\boldsymbol{R}}^k & \end{bmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta\boldsymbol{S} \\ \delta\boldsymbol{R} \\ \delta\boldsymbol{\sigma} \end{pmatrix} = \begin{pmatrix} \mathbf{b} - A\mathbf{u}^k - C_{,\mathbf{u}}^T\boldsymbol{\sigma}^k \\ -\boldsymbol{\psi}^k - C_{,\boldsymbol{S}}^{k,T}\left(\boldsymbol{\sigma}^k + \varepsilon\mathbf{c}^k\right) \\ -C_{,\boldsymbol{R}}^{k,T}\left(\boldsymbol{\sigma}^k + \varepsilon\mathbf{c}^k\right) \\ -\mathbf{c}^k \end{pmatrix}. \tag{14}$$

While the linear system will always have this general shape whatever our choice of discrete spaces, the latter will impact the sparsity pattern of the matrices and our options for solving it. As is standard in finite-element elasticity, we assume that our basis functions $(N_i)_K$ over each mesh element K are Lagrange polynomials defined over a set of nodes $(\mathbf{x}_i)_K$, meaning that $N_i(\mathbf{x}_j) = \delta_j^i$. Due to different continuity requirements, we use distinct basis functions and nodes for the displacement space V_Ω and the tensor spaces T_Ω, SO_Ω, Sym_Ω and Skew_Ω, as explained below.

For the displacement space V_Ω we need H^1-compatible elements, i.e., continuity of the basis functions across neighboring elements. We restrict our choice to usual P_d Lagrange elements or so-called "serendipity" S_d elements that do not contain interior nodes. For P_1 and S_1 this means usual trilinear shape functions with one node at each grid vertex; for S_2, one node at each vertex and one node at the middle of each edge; and so on.

For the T_Ω, SO_Ω, Sym_Ω and Skew_Ω spaces however, we only need to discretize L^2_Ω (our mixed formulation does not require evaluating the derivatives of the stress or strain fields). Continuity across elements is not required and we get the liberty to locate the tensor field nodes $(\mathbf{x}_i)_K$ anywhere within K. To get an insight about how to pick their positions, we look once again at numerical integration. To integrate a function f on element K we resort to a discrete quadrature formula with weights $(w_p)_K$ and evaluation points $(\mathbf{y}_p)_K$, that is,

$$\int_K f \sim \sum_p w_p f(\mathbf{y}_p).$$

As discussed in Section 3.2, the weights and points are typically picked such that the formula is exact for polynomials up to a given order. Now when evaluating the matrix A for a bilinear form defined from two sets of basis functions $(N_i)_K$ and $(N_j)_K$, we get

$$A_{i,j} := \int_K N_i N_j f \sim \sum_p w_p N_i(\mathbf{y}_p) N_j(\mathbf{y}_p) f(\mathbf{y}_P),$$

meaning that if we pick our nodes and quadrature points such that $(\mathbf{x}_i)_K = (\mathbf{x}_j)_K = (\mathbf{y}_p)$, then $A_{i,j} = \sum_p w_p \delta_i^j f(\mathbf{y}_p)$, i.e., the matrix A becomes block diagonal. For a discontinuous Lagrange polynomial basis of chosen degree d, we thus pick the nodes $(\mathbf{x}_i)_K$ to correspond to the points of a quadrature formula that maximizes the order of accuracy for this number of points. In particular for quadrilateral or hexahedral elements, this is the usual Gauss–Legendre points, which yield a quadrature formula that is exact for polynomials up to order $2d$. For triangle and tetrahedral elements, we get quadrature formulas of order $2d$ for $d \leq 1$, while for higher degree polynomials we rely on numerical optimization.

By using this choice of quadrature points to define the Lagrange nodes for the discrete subspaces T_Ω, SO_Ω, Sym_Ω and Skew_Ω, we thus render the matrices H^k, E, $C_{,\boldsymbol{S}}^k$ and $C_{,\boldsymbol{R}}^k$ block-diagonal, while keeping a good order of accuracy for the numerical integration.

To finalize the Mixed FEM discretization, it remains to relate the displacement and tensor spaces. For hexahedral elements, we use serendipity elements S_d of degree d for the displacement and tensor products of element-wise discontinuous Lagrange polynomials (with Gauss–Legendre nodes) of the same degree d for tensors. For tetrahedral elements, we use continuous Lagrange polynomials P_d of degree d for the displacements and discontinuous Lagrange polynomials of degree $d-1$ for the tensor spaces. This means that for hexahedral elements, the quadrature formula defined by the tensor nodes will be enough to integrate accurately all bilinear forms, while for tetrahedral elements we will need to use a distinct, higher-order quadrature formula for the displacement forms.

B.3 Solving the Linear System

While it is possible to directly solve Equation (14) using a saddle-point solver, in practice we find it more efficient to follow Trusty et al. [2022] and perform double condensation. Here for the sake of clarity we focus on the current Newton iteration and drop the k index for matrices and vectors.

First, thanks to the non-zero Augmented–Lagrangian penalization coefficient ε and the semi-definite projection of the elasticity Hessian, H and E are positive definite (as long as S in non-singular). As they are also block-diagonal, they are easily invertible, and we can eliminate the $\delta\boldsymbol{S}$ and $\delta\boldsymbol{R}$ unknowns to obtain

$$\begin{bmatrix} A & C_{,\mathbf{u}}^T \\ C_{,\mathbf{u}} & -\Lambda \end{bmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta\boldsymbol{\sigma} \end{pmatrix} = \begin{pmatrix} \mathbf{b} - A\mathbf{u} - c_{,\mathbf{u}}^T\boldsymbol{\sigma} \\ \boldsymbol{\lambda} \end{pmatrix},$$

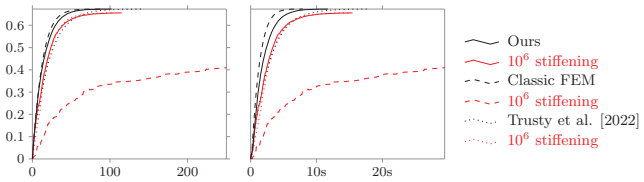

Fig. 24. Mean displacement of the dumbbell from Figure 6 as a function of the Newton iteration number (left) and wall time (right) for three FEM variants, using a homogeneous material (black) or with a 10^6 stiffer center region (red).

$$\Lambda := C_{,S} H^{-1} C_{,S}^T + C_{,\boldsymbol{R}} E^{-1} C_{,\boldsymbol{R}}^T,$$
$$\boldsymbol{\lambda} := -\mathbf{c} + C_{,S} H^{-1} \left(\boldsymbol{\psi} + C_{,S}^T \boldsymbol{\sigma} \right) + C_{,\boldsymbol{R}} E^{-1} C_{,\boldsymbol{R}}^T \boldsymbol{\sigma}.$$

Finally, Λ is also block diagonal and positive definite; it can thus also be efficiently inverted, allowing to eliminate $\delta\boldsymbol{\sigma}$ and assemble the Schur complement system

$$\left[A + C_{,\mathbf{u}}^T \Lambda^{-1} C_{,\mathbf{u}} \right] \delta\mathbf{u} = \mathbf{b} - A\mathbf{u} + C_{,\mathbf{u}}^T \Lambda^{-1} \boldsymbol{\lambda}. \tag{15}$$

We proceed to solve the symmetric positive semi-definite system (15) using a Jacobi-preconditioned Conjugate Gradient solver.

B.4 Performance Considerations

Overall, we find the per-iteration cost of displacement-only and Mixed FEM to be roughly similar. For most of our examples, the cost is dominated by the linear system solve, where both methods share similar unknowns (the per-node displacements) and sparsity stencil after the double condensation step. The assembly of the linear system of Mixed FEM requires multiple sparse matrix–matrix products, but most of those are block–diagonal and cheap to assemble: thanks to our choice of quadrature formula the tensor shape functions are only evaluated at their nodes, where they are Kronecker–delta valued. Conversely, integrating the elasticity Hessian bilinear form for classic FEM requires general interpolation and couples all displacement nodes over a 2^3 voxels stencil.

One particularity of our method, in contrast to the Mixed FEM technique from Trusty et al. [2022], is that we explicitly track the rotation $\boldsymbol{R}$ in a separate field. As such we do not need to perform a polar decomposition to extract the symmetric part $\boldsymbol{S}$ of the deformation gradient when evaluating the constraint residual. While as noted by Smith et al. [2018], this decomposition also gives access to the principal strain basis, which is required to perform the analytical positive semi-definite projection of the elasticity Hessian, in our case the symmetric strain tensor $\boldsymbol{S}$ is already known, so a 3×3 symmetric eigen-decomposition is sufficient – and only needs to be done once per Newton iteration rather that for each tentative state in the linesearch as in the approach of Trusty et al. [2022].

Moreover, we observe in practice that the convergence of our Mixed FEM solver is not significantly affected by more conservative semi-definite approximations; the eigenvalues are going to be shifted by the constraint penalization term ε anyway. In particular, we find that simply ensuring positivity by scaling the volume Hessian term from Neo-Hookean elasticity models according to its minimum eigenvalue, estimated without building the strain basis as per [Smith et al. 2018, Equation (30)], works well.

Another difference is that Trusty et al. [2022] express the constraint by placing the nonlinearity on the other side, i.e., as $\boldsymbol{R}^T \boldsymbol{F} = \boldsymbol{S}$ rather than $\boldsymbol{F} = \boldsymbol{R}\boldsymbol{S}$. This leads to the derivative form $c_{,\mathbf{u}}$ being dependent on $\boldsymbol{R}$ and thus $C_{,\mathbf{u}}$ needing to be re-assembled at each iteration, which can be costly for high-order polynomials; we only need to do this assembly once in our approach. Conversely, we need to reassemble C_S and $C_{\boldsymbol{R}}$ at each iteration, but those, being block-diagonal, are much cheaper to compute.

While the technique described by Trusty et al. [2022] is limited to linear stresses with piecewise-constant rotations and strains, for comparison purposes we have implemented a direct generalization of their method to hexahedral elements using the same choice of basis functions described in Appendix B.2, and report results in Figure 24. For the high-resolution homogeneous dumbbell example, we see that classic FEM is slightly faster than our Mixed FEM, both in terms of iteration count and wall time. In turn, on this example our Mixed FEM is slighly faster than our adaption of the variant from Trusty et al. [2022]. Note we have implemented the three variants in our GPU-based framework, without specific optimizations; results are close enough that different orderings could be observed with other implementations. For the dumbbell with highly stiff center however, we clearly see the displacement-only FEM lagging behind the mixed techniques.

Finally, in contrast with displacement-only and the technique from Trusty et al. [2022], our approach is not fully parameter–free; we need at least the penalization parameter ε to construct the reduced linear system (15). Indeed, developing a penalization-free double-condensation step for our approach is hindered by the fact that for a non-isotropic strain $\boldsymbol{S}$, the matrices $C_{,S}$ and $C_{,\boldsymbol{R}}$ are not orthogonal, so that Λ^{-1} cannot be easily constructed from their pseudo-inverses. A potential solution would be to derive an analytical eigensystem for Λ, but we reserve this investigation for future work. While a too strong regularization would slow down convergence, we did not observe this behavior when using our suggested heuristic and setting ε equal to the typical stress; we see in Figure 24 that the convergence curves for our Mixed FEM approach match the non-penalized techniques. Empirically we also obverse that the additional regularization terms help stabilize simulations when the Newton loop is aggressively truncated, as for the interactive clay example depicted in Figure 9.

Received 12 October 2024; revised 7 February 2025; accepted 25 March 2025