FreeForm: Reduced-Order Deformable Simulation from Particle-Based Skinning Eigenmodes

Anonymous CVPR submission

Paper ID 9567

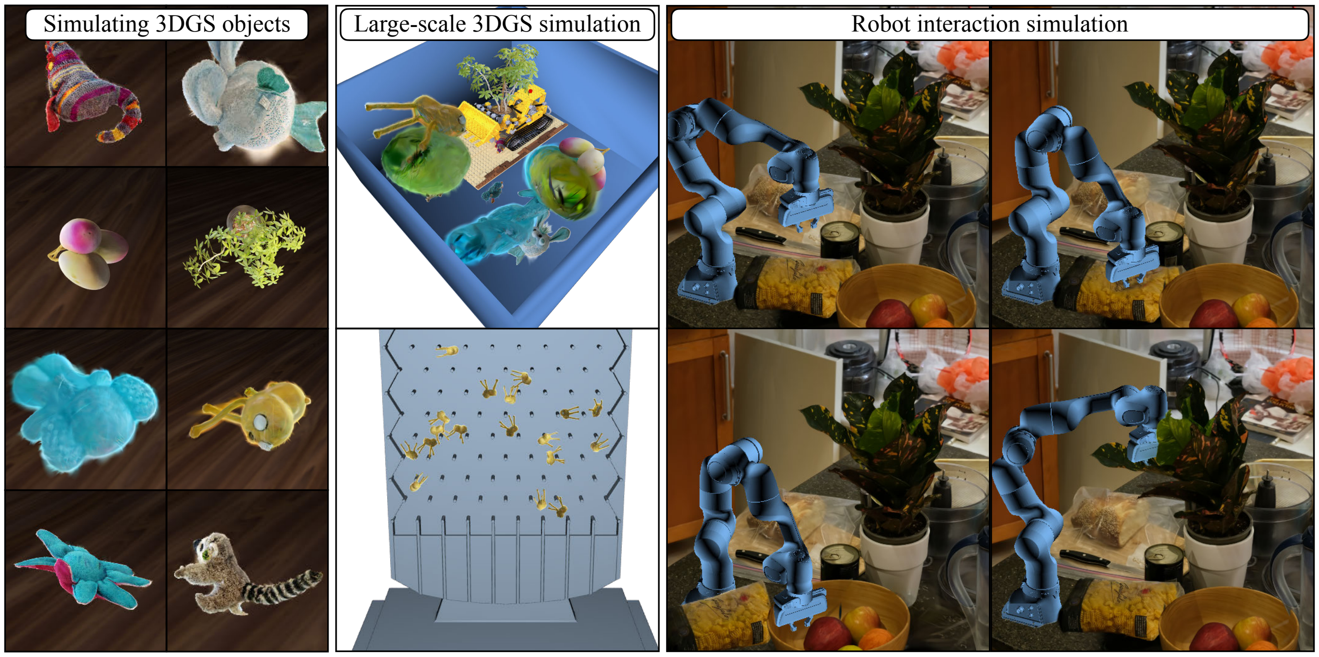

Figure 1. Left: we show results of our reduced-order elastic simulation applied to 3D Gaussian Splatting (3DGS) objects. Middle: our simulation can handle multiple interacting 3DGS objects. Right: we show the application of our method in simulating robot interaction.

Abstract

We present a novel formulation for mesh-free, reduced-order simulation of deformable hyperelastic objects. Existing work in reduced-order elastodynamic simulation represents the input geometry by either meshes, which can be difficult to obtain due to challenges in scanning and triangulating complex shapes, or by neural fields that require per-shape optimization. We propose to adopt a Reproducing Kernel Particle Method (RKPM) representation, which enables the construction of reduced-order skinning weights by solving a generalized eigensystem on the Hessian matrix of the elastic energy. We demonstrate that this formulation not only leads to a 40× training speedup compared with the per-shape optimization of neural fields, but also achieves lower simulation error when evaluated against the converged results of finite element method. We show our simulation results on a wide variety of objects in different representations including meshes and Gaussian splats, as well as the application of our method in the downstream task of robot simulation.

1. Introduction

Elastodynamic simulation of deformable objects is important and widely used in engineering, scientific computing, visual effects, and robotics. The Finite Element Method (FEM) is typically employed to this effect; however it suffers from two major limitations. First, element-based FEM requires high-quality meshes as input: this can be problematic on traditional mesh representations due to the challenges of volumetric meshing on arbitrary shapes, and may not even be well-defined for modern, imprecise point-based representations such as Gaussian Splats [22]. Second, an accurate FEM simulation at high resolution needs a similarly high number of Degrees of Freedom (DoFs) and more iterations of numerical solves, and is thus slow to compute.

Particle-based methods like the Material Point Method (MPM) and Smoothed Particle Hydrodynamics (SPH) have been proposed, which can simulate elastic objects in a mesh-free manner, and are popular in recent works that aims to simulate objects represented by 3DGS, e.g, PhysGaussian and PhysDreamer [43, 47]. However, these approaches also have limitations; for example, they are sensitive to both spatial and temporal discretizations [12, 46], potentially leading to failures under large strains.

Meanwhile, reduced-order simulation techniques [3] have been proposed to reduce computation costs, but have mainly been focused on mesh-based simulation. A recent work, Simplicits [31], addresses the problem of reduced-order simulation in the mesh-free domain, the same setting as our work. This approach, however, requires optimizing a neural field for every input object before simulation can be run. Moreover, we empirically find that Simplicits achieves suboptimal simulation accuracy, possibly due to the difficulty in variational optimization of elastic energy (see discussion in the experiments).

To address the limitation of previous work, our key insight is that the Reproducing Kernel Particle Method (RKPM) [27], a mesh-free, particle-based representation of spatially-defined functions, enjoys multiple advantages in this setting, to which it has not previously been applied to the best of our knowledge. We leverage RKPM to parameterize the deformation subspace and formulate the elastic energy in a mesh-free manner. More importantly, this explicit representation makes it possible to obtain a set of optimal skinning eigenmodes through eigenanalysis, which is more accurate and significantly faster than other comparable subspace generation techniques.

Our contributions are summarized as follows:

- We present a novel formulation of mesh-free, reduced-order elastodynamics using skinning eigenmodes of RKPM-parametrized elastic objects;
- We derive a simple, easy-to-implement mathematical expression for the Hessian matrix of the commonly used Neo-Hookean elastic energy.
- We empirically demonstrate the efficiency and effectiveness of our method on a wide variety of objects.

2. Related Work

Traditionally, in graphics and engineering, elastodynamic simulations have employed explicit mesh-based representations of objects, optimizing over per-element energies [4, 28, 39]. However, the recent popularity of NeRFs [30], Gaussian Splats [22], and signed-distance functions have given rise to new simulation techniques naturally supporting these implicit object representations.

Neural physics-based simulation Neural physics simulation is typically motivated by the need to simulate geometries or materials for which traditional pipelines fail or for which parameters or specifications are not known [21, 34, 36, 42, 43]. These methods exist on a spectrum, ranging from augmenting existing simulation algorithms with neural components [10, 35] to replacing physics simulation in its entirety with a learned representation [25, 33]. Such algorithms are of increasing usefulness due to the proliferation of new, high-fidelity geometric data representations [22, 30] and a desire to generate physically plausible motion from them, whether that be for games [21], training other neural models [26] or applications in the physical AI [1, 44].

Particle-based simulation. Alternatively, particle-based updated-Lagrangian methods, such as the Material Point Method [20, 38] and Smoothed Particle Hydrodynamics [11, 18] have been employed to simulate Gaussian Splats or NeRFs [24, 43]. These methods can handle a wide range of constitutive models. However, their sensitivity to the spatial and temporal discretizations renders them non-ideal for elastic solid simulation, with numerical fracturing under large strains and inexact boundaries as common limitations [20]; RKPM and Moving Least Squares (MLS) interpolation techniques have been explored to reduce these artifacts [8, 41]. Zong et al. [50] explored reducing the computational cost using neural fields. Martin et al. [29] proposed a fully implicit particle-based simulation method leveraging generalized MLS; however, this suffers from a high DoF count and lack of convergence with particle resolution. Feng et al. [15] reduce the DoF count using clustering, but lacks awareness of geometric details. Position-Based Dynamics and mass-spring techniques have also been employed to simulate Gaussian particles [1, 21, 48], but do not derive from a proper continuum model and as such require the use of ad-hoc material parameters. Dodik et al. [13] introduce a particle-based biharmonic formulation with motivations similar to ours, but focus on skinning control from predefined handles rather than physical simulation.

Reduced-order simulation. Rather than rely on a large number of particles or nodes to represent motion, reduced approaches rely on a small set of degrees of freedom augmented with complex basis [2] or interpolating functions [40]. These methods can produce rich shape aware motion as long as appropriate basis functions can be computed, via modal analysis [3, 5, 45] or exemplars [2]. Train-

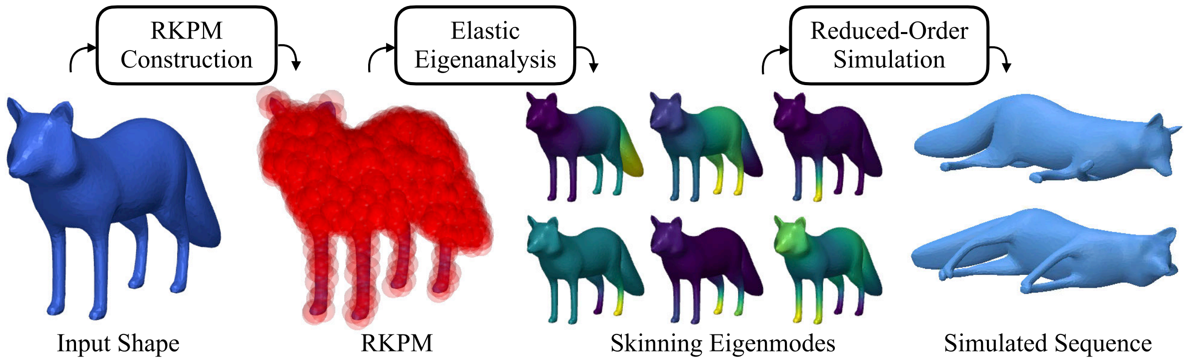

Figure 2. Overview of our method. Given an input object's shape and material properties, we first construct RKPM particles and perform eigenanalysis to derive expressive skinning weights. These weights are then used to enable reduced-order simulation at runtime.

ing a neural network to encode the reduced representation has shown great promise for modeling kinematics and dynamics of complex geometries [16, 37]. While LiCROM [6] learns a continuous reduced order model from an existing subspace — usually generated on a mesh, Simplicits [31] demonstrates representation agnostic simulation across any input domain that admits an inside-outside query, along with shape-aware deformation and good agreement with standard FEM simulations. Chang et al. [7] trains a neural field that predicts Laplace eigenfunctions for classes of parametric objects, which may also be used for reduced order simulation. Besides the requirement of parametric models, unlike Simplicits [31] and our method, this approach is not material-aware and thus only addresses homogeneous objects.

3. Methodology

Our method takes as input the geometry of an elastic object, in any representation that allows us to do integration on the volume, e.g, by sampling points from the object. First, in an optimization/training stage, we construct a set of reduced-order bases, known as *skinning weights*, for the object. Then, in a simulation stage, the skinning weights are used for a low-degree-of-freedom elastic simulation. We introduce the overall background and notation in Sec. 3.1, and our approach in Sec. 3.2. A high-level overview of the method is given in Fig. 2.

3.1. Background

Reduced-order elastodynamic simulation models the deformation of an object by a deformation map $\mathbf{x} \leftarrow \phi(\mathbf{X}, \mathbf{z})$ that maps any point $\mathbf{X} \in \Omega \subset \mathbb{R}^3$ in the object from the reference space to $\mathbf{x} \in \mathbb{R}^3$ in the deformed configuration, controlled by a number of Degrees of Freedom (DoFs) $\mathbf{z} \in \mathbb{R}^n$. In a maximal-coordinate simulation, the DoFs are simply the vertex or particle positions, while reduced-order simulations pick DoFs from a low-dimensional space for efficiency and control. One common formulation is skinning-based (or frame-based) deformation [3, 13, 17], where the DoFs $\mathbf{z}$ consist of a set of m affine transformations $\{\mathbf{Z}_j \in \mathbb{R}^{3\times 4}\}_{j=1}^m$ with associated fixed skinning weight functions $\mathbf{W} : \mathbb{R}^3 \to \mathbb{R}^m$ combined using Linear Blend Skinning (LBS):

$$\mathbf{x} = \Phi(\mathbf{X}, \mathbf{z}) = \mathbf{X} + \sum_{j=1}^{m} \mathbf{W}^j(\mathbf{X})\mathbf{Z}_j\overline{\mathbf{X}}, \qquad (1)$$

where $\overline{\mathbf{X}}$ is the homogeneous coordinates of $\mathbf{X}$, $\mathbf{W}^j$ is the j-th weight of $\mathbf{W}$.

Reduced-order simulation methods have a *fitting* or *training stage* that finds skinning weights $\mathbf{W}$ for a given object, and a *simulation stage* that, given $\mathbf{W}$, the previous state $\mathbf{z}_{t+1}$, and the environment, performs time stepping to solve for DoFs of the next state $\mathbf{z}_{t+1}$. The simulation stage follows the standard implicit time integration that minimizes the incremental potential:

$$\mathbf{z}_{t+1} = \arg\min_{\mathbf{z}} \mathrm{Ir}(\mathbf{z}, \mathbf{z}_t) + E_{\text{pot}}(\mathbf{z}) + E_{\text{ext}}(\mathbf{z}) \qquad (2)$$

where h is the timestep, $\mathrm{Ir}(\mathbf{z}, \mathbf{z}_t)$ is the inertia energy, E_{pot} is the elastic potential energy, and E_{ext} is the potential energy for external forces such as gravity and boundary conditions, both implicitly dependent on the skinning model $\mathbf{W}$.

Simplicits One baseline, Simplicits, proposes a mesh-free formulation by representing the skinning weights $\mathbf{W}$ with a neural field parametrized by network weights θ (hence the notation $\mathbf{W}_\theta, \Phi_\theta$). Simplicits aims to find the skinning weights that produce physically-plausible deformation by minimizing a combination of elastic loss and orthogonality constraints:

$$\theta^* = \arg\min_{\theta} \lambda_{\text{elastic}}\mathcal{L}_{\text{elastic}} + \lambda_{\text{ortho}}\mathcal{L}_{\text{ortho}} \qquad (3)$$

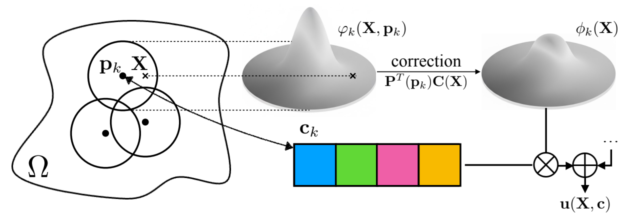
$\varphi_k(\mathbf{X}, \mathbf{p}_k)$
$\phi_k(\mathbf{X})$
$\mathbf{p}_k$
$\mathbf{X}$
correction
$\mathbf{P}^T(\mathbf{p}_k)\mathbf{C}(\mathbf{X})$
$\mathbf{c}_k$
Ω
$\mathbf{u}(\mathbf{X}, \mathbf{c})$

Figure 3. Illustration of RKPM. RKPM smoothly interpolates nodal values $\mathbf{c}_k$ to any query location $\mathbf{X}$ using the reproducing kernels ϕ_k corrected on top of the raw RBF φ_k to satisfy the reproducing condition.

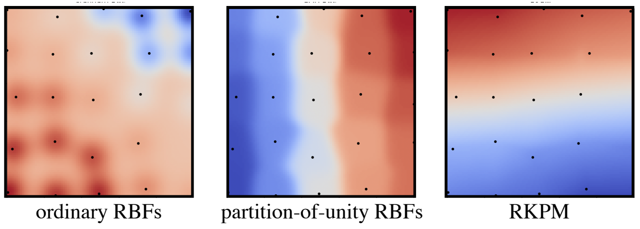

Figure 4. Choosing a suitable kernel basis is essential for high-quality reduced particle simulation. Here, we visualize the first nonzero Laplacian eigenmode on three different particle bases. Even with nicely-sampled centers, RBFs (*left*) and partition-of-unity RBFs (*middle*) have irregular nonsmooth modes, while RKPMs (*right*, ours) closely approximate the expected linear field.

The elastic loss evaluates the elastic energy of the deformation map Φ_θ over randomly sampled transformations $\mathbf{z}$ from a normal distribution of variance σ^2

$$\begin{aligned} \mathcal{L}_{\text{elastic}} &= \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\sigma\mathbf{I})}\left[E_{\text{pot}}(\mathbf{z})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\sigma\mathbf{I})}\left[\int_\Omega \Psi(\Phi_\theta(\mathbf{X},\mathbf{z}))d\mathbf{X}\right], \end{aligned} \tag{4}$$

where Ψ is the strain energy density function depending on the constitutive model (e.g., Neo-Hookean). The second term, $\mathcal{L}_{\text{ortho}}$ enforces that the skinning weights form an orthonormal basis:

$$\mathcal{L}_{\text{ortho}} = \sum_{i,j=1}^{n} \int_\Omega (\mathbf{W}_\theta^i(\mathbf{X})\mathbf{W}_\theta^j(\mathbf{X}) - \delta_{ij})^2 d\mathbf{X}. \tag{5}$$

This orthogonality constraint not only prevents the trivial solution $\mathbf{W} \equiv 0$, but also ensures a nice numerical condition for the mass matrix which plays a crucial role in solving Eq. (2) by Newton's method.

3.2. Efficient Skinning Eigenmode from RKPM

As an alternative to neural skinning weights, we propose to discretize $\mathbf{W}$ using the Reproducing Kernel Particle Method (RKPM) [27], which has several key benefits. We first recall the formulation of RKPM, and then show how this representation allows us to efficiently build a high-quality reduced-order basis for elastic deformation.

RKPM represents any vector-valued function $\mathbf{u} : \Omega \subset \mathbb{R}^3 \to \mathbb{R}^d$ by a sum of node values $\mathbf{c} = \{\mathbf{c}_k \in \mathbb{R}^d\}_{k=1}^K$ weighted by reproducing kernels $\{\phi_k\}_{k=1}^K$ centered at positions $\{\mathbf{p}_k \in \Omega\}_{k=1}^K$,

$$\mathbf{u}(\mathbf{X};\mathbf{c}) = \sum_{k=1}^{K} \phi_k(\mathbf{X})\mathbf{c}_k. \tag{6}$$

These kernels are conceptually similar to standard Radial Basis Functions (RBF), e.g, Gaussian RBF $\varphi_k(\mathbf{X}) = \exp(-\|\mathbf{X} - \mathbf{p}_k\|^2/r^2)$, but are augmented with correction terms to satisfy certain numerical conditions:

$$\phi_k(\mathbf{X}) = \varphi_k(\mathbf{X})\mathbf{P}^T(\mathbf{p}_k)\mathbf{C}(\mathbf{X}), \tag{7}$$

where $\mathbf{P}(\mathbf{X}) = [1, x, y, z]^T$ includes monomials of $\mathbf{X}$ up to degree $D = 1$ in our case, and $\mathbf{C} : \Omega \mapsto \mathbb{R}^{\dim \mathbf{P}}$ is an introduced correction function to be solved for. The *reproducing condition* requires that these kernels reproduce polynomial functions up to degree D in Eq. (6):

$$\sum_{k=1}^{K} \phi_k(\mathbf{X})\mathbf{P}(\mathbf{p}_k) = \mathbf{P}(\mathbf{X}) \tag{8}$$

Substituting Eq. (7) into Eq. (8) yields a linear equation that allows us to solve for $\mathbf{C}(\mathbf{X})$ for any query location $\mathbf{X}$ as

$$\mathbf{M}(\mathbf{X})\mathbf{C}(\mathbf{X}) = \mathbf{P}(\mathbf{X}), \tag{9}$$

$$\text{where} \quad \mathbf{M}(\mathbf{X}) = \sum_{k=1}^{K} \varphi_k(\mathbf{X})\mathbf{P}(\mathbf{p}_k)\mathbf{P}^T(\mathbf{p}_k). \tag{10}$$

Thus the final formulation of RKPM is

$$\begin{aligned} \mathbf{u}(\mathbf{X};\mathbf{c}) &= \sum_{k=1}^{K} \overbrace{\mathbf{P}^T(\mathbf{p}_k)\mathbf{C}(\mathbf{X})\varphi_k(\mathbf{x})}^{\phi_k(\mathbf{X})}\,\mathbf{c}_k \\ \text{where} \quad &\mathbf{C}(\mathbf{X}) = \mathbf{M}^{-1}(\mathbf{X})\mathbf{P}(\mathbf{X}). \end{aligned} \tag{11}$$

Skinning Eigenmode with RKPM. Directly applying the RKPM formulation in Eq. (11) to parameterize the deformation map $\Phi(\mathbf{X}) = \mathbf{u}(\mathbf{X},\mathbf{d}) + \mathbf{X}$ allows us to evaluate the elastic potential as

$$E_{\text{pot}}^{\text{full}}(\mathbf{d}) = \int_\Omega \Psi(\mathbf{u}(\mathbf{X},\mathbf{d}) + \mathbf{X})d\mathbf{X}, \tag{12}$$

where the DoFs are 3D nodal displacements $\mathbf{d} = \{\mathbf{d}_k \in \mathbb{R}^3\}_{k=1}^K$. Simulating this full-order formulation directly for a large number of RKPM kernels would be costly. In line with existing reduced-order methods, we adopt the skinning deformation in Eq. (1) and propose to discretize the skinning weight fields $\{\mathbf{W}^j\}_{j=1}^m$, rather than the displacement $\mathbf{u}$, with RKPM. The problem boils down to determining adequate nodal values $\mathbf{c} = [\mathbf{c}_1, \dots, \mathbf{c}_K]^T \in \mathbb{R}^{K\times m}$ for each skinning function $\mathbf{W}^j(\mathbf{X}) = \sum_{k=1}^K \phi_k(\mathbf{X})\mathbf{c}_k^j$, where $\mathbf{c}_k^j$ is the j-th element of the vector $\mathbf{c}_k \in \mathbb{R}^m$.

Adapting RKPM allows us to extend the Skinning Eigenmode approach [3] to the mesh-free domain. The basic idea is to approximate the elastic potential $E_{\text{pot}}^{\text{full}}(\cdot)$ by its Hessian matrix $\mathbf{H}$ around the rest positions

$$E_{\text{pot}}^{\text{full}}(\mathbf{d}) \approx \frac{1}{2}\mathbf{d}^T\mathbf{H}\mathbf{d}, \tag{13}$$

where $\mathbf{d} \in \mathbb{R}^{3K}$ is nodal displacements flattened to a vector. Then a set of most expressive skinning weights $\mathbf{W}$ should be selected so that different columns of $\mathbf{c}$ minimize the sum of this quadratic form while keeping different channels of $\mathbf{W}$ orthogonal to each other. Following Benchekroun et al. [3], we use the simplified weight-space Hessian that prioritizes translation for skinning eigenmode: $\mathbf{H}_w = \mathbf{H}_{xx} + \mathbf{H}_{yy} + \mathbf{H}_{zz}$, which are blocks of $\mathbf{H}$ with respect to x, y, z coordinates, respectively. For orthogonality, we have

$$\delta_{ij} = \langle \mathbf{W}^i, \mathbf{W}^j \rangle = \int_\Omega \mathbf{W}^i(\mathbf{X})\mathbf{W}^j(\mathbf{X})d\mathbf{X}$$
$$= \sum_{l,m=1}^K \mathbf{c}_l^i\mathbf{c}_m^j \overbrace{\int_\Omega \phi_l(\mathbf{X})\phi_m(\mathbf{X})d\mathbf{X}}^{\mathcal{M}_{lm}} = \sum_{l,m=1}^K \mathbf{c}_l^i\mathcal{M}_{lm}\mathbf{c}_m^j \tag{14}$$

where $\mathcal{M}$ is the mass matrix of RKPM. Eq. (14) can be written as $\mathbf{c}^T\mathcal{M}\mathbf{c} = \mathbf{I}$ in matrix form. Putting these together, we have the following optimization problem:

$$\underset{\mathbf{c}\in\mathbb{R}^{K\times m}}{\arg\min}\, \mathrm{tr}(\mathbf{c}^T\mathbf{H}_w\mathbf{c}), \quad \text{subject to} \quad \mathbf{c}^T\mathcal{M}\mathbf{c} = \mathbf{I} \tag{15}$$

This problem can be efficiently solved as a generalized eigenvalue problem $\mathbf{H}_w\mathbf{v} = \lambda\mathcal{M}\mathbf{v}$, and we can use the first m generalized eigenvectors $[\mathbf{v}_1, \dots, \mathbf{v}_m]$ as $\mathbf{c}$.

Simple expression for Hessian matrix. We derive a simple expression for the weight-space Hessian matrix $\mathbf{H}_w$ for the commonly used Neo-Hookean elastic energy that allows easy analytical evaluation. We use the following version of Neo-Hookean elastic energy

$$\Psi(\mathbf{F}) = \frac{1}{2}(\lambda+\mu)(\det\mathbf{F} - \gamma)^2 + \mu\,\mathrm{tr}(\mathbf{F}^T\mathbf{F}) - E_0. \tag{16}$$

where $\mathbf{F} = \nabla\Phi$ is the deformation gradient, λ and μ are Lamé coefficients, and $\gamma = 1 + \mu/(\lambda+\mu)$, E_0 is a constant term so that $\Psi(\mathbf{I}) = 0$,

Test	m	Simplicits	Ours	MPM	SPH
Bend	6	1.20e-02	7.80e-03	1.42e-03	6.57e-04
	9	6.94e-03	4.90e-03		
	16	1.53e-03	4.10e-04		
	32	1.17e-04	**2.93e-06**		
Twist	6	2.54e-03	1.56e-04	2.34e-05	1.33e-04
	9	3.42e-04	2.95e-05		
	16	1.30e-04	**3.46e-06**		
	32	4.21e-05	6.64e-06		

Table 1. Quantitive evaluation on the standard beam deformation test. We report the normalized Mean Squared Error (MSE) of simulated point locations on two types of boundary conditions. The results are reported for different numbers m of affine transformations for Simplicits and our method. We also show the results of MPM and SPH for comparison.

Proposition 1 *For the Neo-Hookean elastic energy above, the (i, j)-th element of the weight-space Hessian matrix $\mathbf{H}_w$ with RKPM discretization in Eq. (6) simplifies to*

$$(\mathbf{H}_w)_{ij} = \int_\Omega (\lambda(\mathbf{X}) + 4\mu(\mathbf{X}))\nabla\phi_i(\mathbf{X})^T\nabla\phi_j(\mathbf{X})d\mathbf{X}.$$

The proof is given in the supplementary material, and similar expressions can be derived for other commonly used elastic energy functions. Moreover, for homogeneous materials where λ and μ are constant across the domain, the Hessian matrix simplifies to

$$(\mathbf{H}_w)_{ij} = (\lambda + 4\mu)\overbrace{\int_\Omega \nabla\phi_i(\mathbf{X})^T\nabla\phi_j(\mathbf{X})d\mathbf{X}}^{\mathbf{L}_{ij}}, \tag{17}$$

where $\mathbf{L}$ is the weak-form Laplace matrix of RKPM. In this case, the elastic Hessian $\mathbf{H}_w$ shares the same eigenmodes as the Laplace matrix $\mathbf{L}$, which is known to be the minimizer of the Dirichlet energy for any input scalar fields. In the heterogeneous case, our proposed RKPM-based skinning eigenmode can be regarded as a material-aware Laplace eigenmode, and connects back to this important concept in classical modal analysis.

4. Evaluation

Standard beam test. We start with evaluation results on a cuboid beam, a standard test in the simulation of elastodynamics. We follow the same experiment setup as Simplicits [31], using a beam of shape $5\text{m} \times 1\text{m} \times 1\text{m}$, Young's Modulus $5\times10^6\text{Pa}$, Poisson ratio 0.45, and density $1\times10^3\text{kg/m}^3$. We apply two types of boundary conditions for the beam: (1) fixing the leftmost 0.5m of the beam and letting the right side bend freely (named 'Bend'); (2) fixing

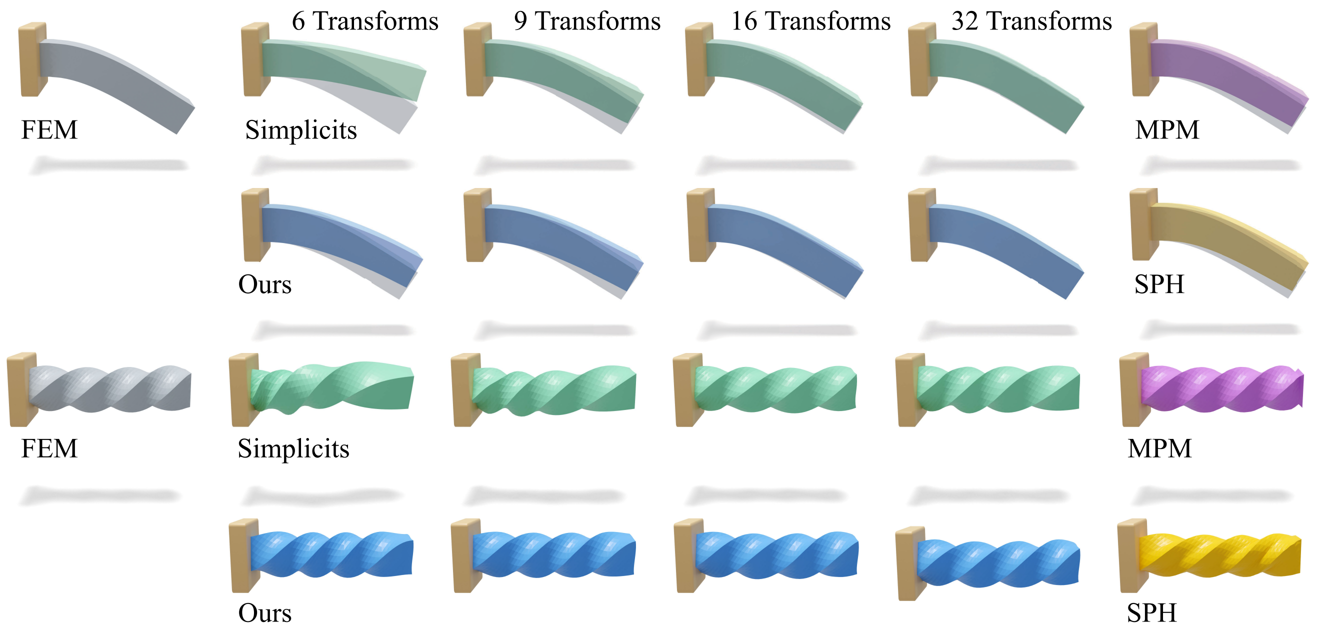

Figure 5. Visual comparison on standard beam test. For the case of bending cantilever beam, FEM solution is overlaid semi-transparently on top of the simulated result from all competing methods to aid visual comparison.

the leftmost end of the beam and twisting the right end of the beam by up to 720° (named 'Twist').

We compare our method with three other types of mesh-free methods, Simplicits [31], MPM and SPH, against simulation results with Finite Element Methods (FEM) on tetrahedral meshes, which is regarded as the gold standard for elastodynamic simulation. We use the Mean Squared Error (MSE) of simulated point locations across the whole shape and all frames, normalized by the bounding box size of the reference shape. Simplicits and ours are reduced-order methods, so we also report results with different numbers m of skinning functions or affine transformations. The experiment results are reported in Table 1.

With both boundary conditions, Simplicits and our method show steady improvement in accuracy as the number of affine transformations increases, meaning an increase in the number of DoFs allowed in the simulation. Our method consistently outperforms Simplicits with the same DoFs. When the number of transformations reaches a certain threshold, our simulation results can match or surpass the accuracy of MPM and SPH, two full-order simulation methods with different formulations.

Thingi10K and Simready datasets. To evaluate our method on more diverse input examples, we take 20 shapes from the Thingi10K dataset [49] and 19 shapes from the Simready Dataset [32]. The shapes we select need to satisfy several geometric properties, such as being manifold, oriented, without self-intersection, and enclosing clear volumes in order to obtain tetrahedral meshes for FEM simulation ground truth. We use TetWild [19] for the tetrahedralization of those filtered shapes from both datasets. In this experiment, we focus on comparing our method with Simplicits [31], since it is the only method with the same problem setting as ours.

The Thingi10K dataset is an online collection of 3D models covering a wide range of categories including both imaginary and real-world objects. The shapes come in different scales, so we normalize their bounding box sizes to be 1. Depending on the semantic meaning of the object, we manually assign Young's Modulus of around $10^5\mathrm{Pa}$ to organic shapes and $10^8\mathrm{Pa}$ to other categories, while keeping Poisson ratio and density consistent across the dataset.

The Simready dataset consists of meshes of real-world objects created by artists, where the shapes are provided in metric scales. We utilize VoMP [9] to predict the volumetric physical parameters including Young's Modulus, Poisson ratio, and density, and use these properties for both the FEM ground truth simulation and the methods in comparison.

We report the simulation results in three types of boundary conditions: (1) fixing one side of the objects; (2) pulling the objects in different directions on 4 farthest points sampled from the surface; (3) pulling the objects on two sides along its longest axis. The results are reported in Table 2. To evaluate simulation accuracy, we report the normalized Mean Squared Error (MSE) and the maximum displacement error in each simulation sequence averaged over all the examples in the dataset. We also compare the aver-

Method	Fix Side		Pull Farthest		Pull Boundary		Training time (s)
	MSE	Max	MSE	Max	MSE	Max	
Simplicit	8.97e-03	2.64e-02	5.58e-02	1.66e-01	3.37e-02	6.30e-02	121.44 ± 10.15
RKPM	**6.87e-03**	**2.14e-02**	**3.75e-02**	**1.19e-01**	**3.11e-02**	**5.96e-02**	**3.19** ± 2.48
Improvement	34.2%	18.2%	29.8%	27.3%	37.5%	38.9%	97.4%
Simplicit	2.16e-09	2.55e-09	9.38e-04	2.77e-03	8.83e-04	2.60e-03	117.45 ± 1.13
RKPM	**1.01e-09**	**1.29e-09**	**4.75e-04**	**1.40e-03**	**4.16e-04**	**1.26e-03**	**3.49** ± 2.39
Improvement	18.9%	18.3%	38.1%	40.1%	45.4%	44.4%	97.0%

Table 2. Quantitive evaluation on the Thingi10K and Simready Datasets. We report the normalized Mean Squared Error (MSE) and maximum error across all the examples for each boundary condition. We also show the training time of our method compared to Simplicits, and the improvement of our method in percentage.

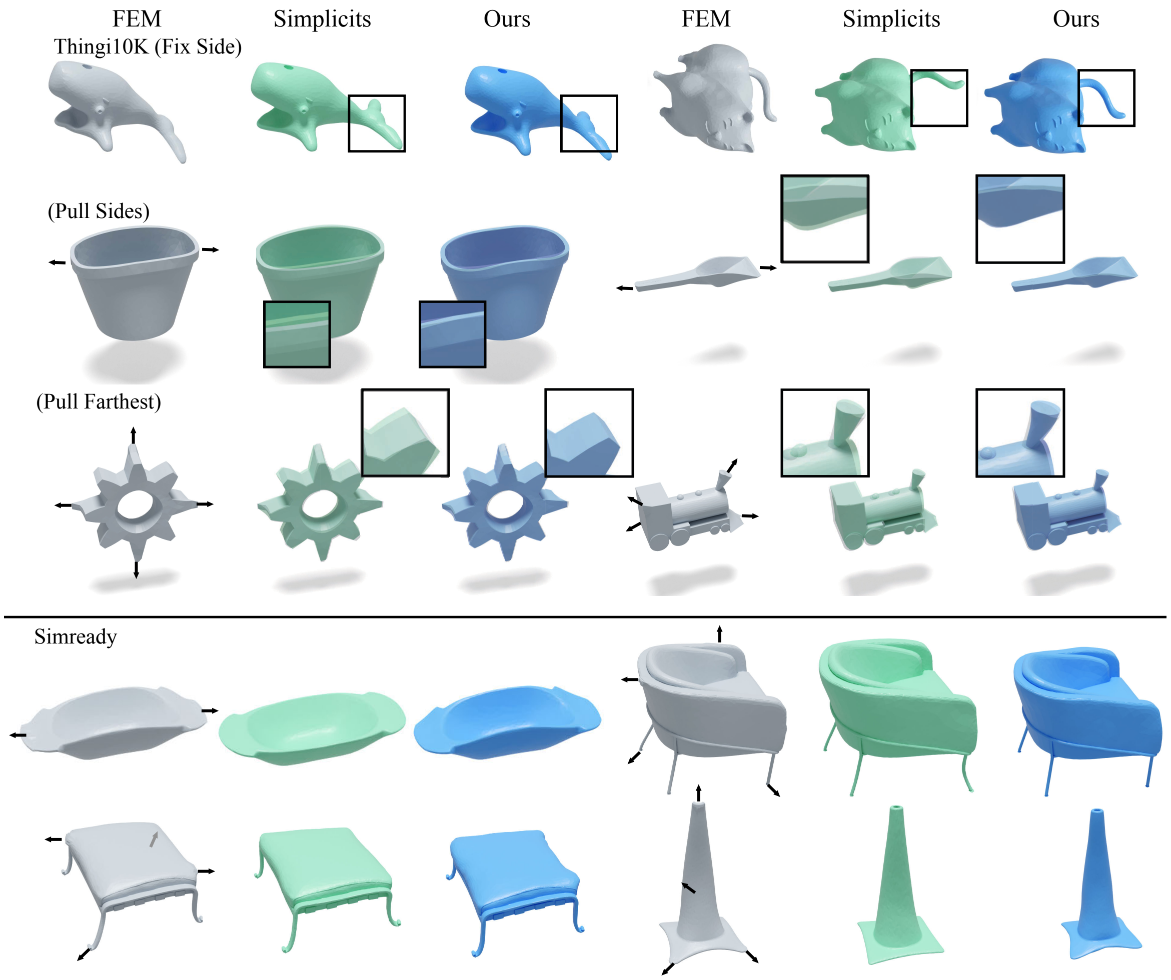

Figure 6. Comparison on Thingi10k and Simready dataset. In "Thingi10K (Fix Side)", the leftmost sides of the objects are fixed. In "Pull Sides" and "Pull Farthest", black arrows around the FEM results in grey visualize the locations and directions of applied moving boundary conditions, and we overlay FEM results on top of the results from Simplicits in green and our method in blue, along with zoom-in views to highlight the discrepancy from the converged FEM solution. See Table 2 and the attached video for the increased fidelity of our simulations.

age training time of our method, including the computation of Hessian matrix and mass matrix followed by eigendecomposition in Eq. (15), with Simplicits. Our method achieves consistently better simulation accuracy than Simplicits, and achieves around 40x faster training speed than Simplicits, thanks to our eigenanalysis formulation.

Loss type	Sampling	Drop	Twist	Time (s)
Simplicits	(Random)	1.53e-3	1.30e-3	114.28
Random $\mathbf{z}$	Random	1.58e-2	2.50e-2	160.12
Random $\mathbf{z}$	Grid	1.24e-2	7.29e-3	412.38
Hessian	Random	4.86e-3	5.07e-4	103.66
Hessian	Grid	4.45e-4	3.49e-5	145.96
Ours	(Grid)	**4.10e-4**	**3.46e-5**	**3.93**

Table 3. Ablation results on different training strategy. We compare variants of our method trained using different loss functions and integration point sampling methods. We also highlight the efficiency of our eigenanalysis formulation over gradient-based optimization in terms of training time.

Ablation study on training. We conduct ablation studies on differences in training strategy between our method and Simplicits to justify our design choices. We use the standard beam test in Table 1, where all competing methods are allowed $m = 16$ affine transformations, and use the RKPM discretization except the original Simplicits method. All models, except ours, are trained using iterative optimization for the same number of iterations. The results are reported in Table 3.

Besides using neural field or RKPM, two other key differences between Simplicts and our method are the loss variants and point sampling method for training. We first compare the expected elastic energy over randomly sampled transformations used in Simplicits in Eq. (4) with the Hessian approximation in Eq. (15) in our method. The results show that the Hessian loss achieves a better simulation accuracy for RKPM parameterization. Second, for the Monte Carlo integration of elastic energy in Eq. (4) and (12), we compare the strategy of sampling random points in different iterations, as in Simplicits, with using the same points sampled from a uniform grid, as in our method. The results show that our uniform grid sampling strategy performs better given sufficient grid resolution.

Lastly, our method has almost the same formulation as the "Hessian - Grid" baseline in the table, only except that the baseline uses a gradient-based iterative optimization of loss terms like Eq. (3), whereas our method solves a generalized eigenvalue problem using highly efficient linear algebra routines. As a result, these two variants are on par in accuracy, while our method uses a significantly lower training time. In addition, the optimization in Eq. (3) imposes orthogonality as a soft penalty constraint, whereas the output skinning weights $\mathbf{W}$ from eigen-decomposition satisfy exact orthogonality (up to numerical precision). This is beneficial to the numerical condition of the system Hessian during simulation (Eq. 2).

Test	Sampling	Simplicits	Ours
Drop	Grid	1.53e-3	**4.10e-4**
	Random	3.69e-3	9.42e-4
Twist	Grid	1.30e-4	**3.46e-6**
	Random	3.88e-4	1.14e-5

Table 4. Ablation studies on sampling method of integration points in simulation stage. We test with 5k integration points sampled from a uniform grid or random uniform distribution.

Ablation study on test-time sampling. Simplicits randomly samples integration points inside the object domain in different training iterations, while our method computes the elastic energy on a limited set of sample points due to the eigen-decomposition. As a result, one may wonder whether our simulation result is more sensitive to different integration samples at test time. In Table 4, we show a comparison of our method and Simplicits using 5k points from a uniform grid vs. randomly sampled points. It turns out that although both Simplicits and our method both show variation in accuracy due to sampling differences, our method still achieves lower error in both cases.

Qualitative results. We demonstrate qualitative simulation results of our method in several different scenarios Figure 1. We first show a set of 3DGS objects dropped on a table individually, and then two large-scale simulation scenes, where 13 different splats being dropped in a container and 18 gaussian splat dog toys falling through a plinko machine, respectively. We also show an application of simulating a robot arm interacting with several 3DGS objects. Our results are best viewed in videos, and please refer to our supplementary materials for more results and detail.

5. Limitation and Future Work

Although the explicit RKPM discretization offers clear advantages over implicit neural bases, it also has several limitations. Some are shared with all reduced representations, while others are specific to RKPM. As a reduced-order model, high-frequency details such as wrinkles are difficult to capture because the global basis represents smooth, low-frequency deformation. Large nonlinear effects, including sharp contact, are likewise challenging since the basis is smooth and derived from a linearization around the rest state. As with most reduced models, we do not model topology changes such as fracture by default. Finally, RKPM depends on kernel radius, sampling density, and particle distribution, which requires careful implementation to ensure basis quality. These limitations suggest several promising directions for future work.

References

[1] Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Suenderhauf. Physically embodied gaussian splatting: A realtime correctable world model for robotics. In *8th Annual Conference on Robot Learning*, 2024. 2

[2] Jernej Barbic and Doug L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM SIGGRAPH 2005 Papers*, 2005. 2

[3] Otman Benchekroun, Jiayi Eris Zhang, Siddhartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. Fast complementary dynamics via skinning eigenmodes, 2023. 2, 3, 5

[4] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)*, 33(4):154, 2014. 2

[5] Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. Hyper-reduced projective dynamics. *ACM Trans. Graph. (Proc. of SIGGRAPH)*, 37(4):80:1–80:13, 2018. 2

[6] Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M. Chiaramonte, Kevin Carlberg, and Eitan Grinspun. Licrom: Linear-subspace continuous reduced order modeling with neural fields, 2023. 3

[7] Yue Chang, Otman Benchekroun, Maurizio M Chiaramonte, Peter Yichen Chen, and Eitan Grinspun. Shape space spectra. *ACM Transactions on Graphics (TOG)*, 44(4):1–16, 2025. 3

[8] Xiao-Song Chen, Chen-Feng Li, Geng-Chen Cao, Yun-Tao Jiang, and Shi-Min Hu. A moving least square reproducing kernel particle method for unified multiphase continuum simulation. *ACM Trans. Graph.*, 39(6), 2020. 2

[9] Rishit Dagli, Donglai Xiang, Vismay Modi, Charles Loop, Clement Fuji Tsang, Anka He Chen, Anita Hu, Gavriel State, David IW Levin, and Maria Shugrina. Vomp: Predicting volumetric mechanical property fields. *arXiv preprint arXiv:2510.22975*, 2025. 6

[10] Gilles Daviet, Tianchang Shen, Nicholas Sharp, and David I.W. Levin. Neurally integrated finite elements for differentiable elasticity on evolving domains. *ACM Trans. Graph.*, 44(2), 2025. 2

[11] Mathieu Desbrun and Marie-Paule Cani. Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*, page 61–76, Berlin, Heidelberg, 1996. Springer-Verlag. 2

[12] Mathieu Desbrun and Marie-Paule Cani. Space-Time Adaptive Simulation of Highly Deformable Substances. Research Report RR-3829, INRIA, 1999. 2

[13] Ana Dodik, Vincent Sitzmann, Justin Solomon, and Oded Stein. Robust biharmonic skinning using geometric fields. *ACM Transactions on Graphics*, 2024. 2, 3

[14] Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. A multi-scale model for simulating liquid-fabric interactions. *ACM Trans. Graph.*, 37(4), 2018. 1

[15] Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, and Yin Yang. Pie-nerf: Physics-based interactive elastodynamics with nerf, 2023. 2

[16] Lawson Fulton, Vismay Modi, David Duvenaud, David Levin, and Alec Jacobson. Latent-space dynamics for reduced deformable simulation. 2019. 3

[17] Benjamin Gilles, Guillaume Bousquet, Francois Faure, and Dinesh K Pai. Frame-based elastic models. *ACM transactions on graphics (TOG)*, 30(2):1–12, 2011. 3

[18] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3): 375–389, 1977. 2

[19] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4):60–1, 2018. 6

[20] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, New York, NY, USA, 2016. Association for Computing Machinery. 2, 1

[21] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. *arXiv preprint arXiv:2401.16663*, 2024. 2

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2

[23] Tassilo Kugelstadt, Jan Bender, José Antonio Fernández-Fernández, Stefan Rhys Jeske, Fabian Löschner, and Andreas Longva. Fast corotated elastic sph solids with implicit zero-energy mode control. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–21, 2021. 2

[24] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *ArXiv*, abs/2303.05512, 2023. 2

[25] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 2

[26] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8576–8588, 2024. 2

[27] Wing Kam Liu, Sukky Jun, and Yi Fei Zhang. Reproducing kernel particle methods. *International journal for numerical methods in fluids*, 20(8-9):1081–1106, 1995. 2, 4

[28] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, pages 49–54, 2016. 2

[29] Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. Unified simulation of elastic rods, shells, and solids. *ACM Transactions on Graphics (TOG)*, 29(4):1–10, 2010. 2

[30] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *The European Conference on Computer Vision (ECCV)*, 2020. 2

[31] Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David IW Levin. Simplicits: Mesh-free, geometry-agnostic elastic simulation. *ACM Transactions on Graphics (TOG)*, 43(4):1–11, 2024. 2, 3, 5, 6

[32] NVIDIA Developer. Simready assets. `https://developer.nvidia.com/omniverse/simready-assets`, 2025. Accessed: 2025-06-13. 6

[33] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. 2

[34] Cristian Romero, Dan Casas, Jesús Pérez, and Miguel A. Otaduy. Learning Contact Corrections for Handle-Based Subspace Dynamics. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 40(4), 2021. 2

[35] Cristian Romero, Dan Casas, Maurizio M. Chiaramonte, and Miguel A. Otaduy. Contact-centric deformation learning. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 41(4), 2022. 2

[36] Boxiang Rong, Artur Grigorev, Wenbo Wang, Michael J. Black, Bernhard Thomaszewski, Christina Tsalicoglou, and Otmar Hilliges. Gaussian garments: Reconstructing simulation-ready clothing with photorealistic appearance from multi-view video, 2024. 2

[37] Nicholas Sharp, Cristian Romero, Alec Jacobson, Etienne Vouga, Paul G Kry, David IW Levin, and Justin Solomon. Data-free learning of reduced-order kinematics. 2023. 3

[38] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995. Particle Simulation Methods. 2

[39] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models, 1987. *URL http://design. osu. edu/carlson/history/PDFs/ani-papers/terzopoulos-deformable. pdf*. 2

[40] Yu Wang, Alec Jacobson, Jernej Barbic, and Ladislav Kavan. Linear subspace design for real-time shape deformation. *ACM Trans. Graph.*, 34(4):57:1–57:11, 2015. 2

[41] Lukas Westhofen, Stefan Jeske, and Jan Bender. A comparison of linear consistent correction methods for first-order sph derivatives. *Proceedings of the ACM on Computer Graphics and Interactive Techniques (SCA)*, 6(3), 2023. 2, 1

[42] William F. Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey R. Allen. Learning 3d particle-based simulators from rgb-d videos, 2023. 2

[43] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023. 2

[44] Jie Xu, Eric Heiden, Iretiayo Akinola, Dieter Fox, Miles Macklin, and Yashraj Narang. Neural robot dynamics. In *9th Annual Conference on Robot Learning*, 2025. 2

[45] Yin Yang, Dingzeyu Li, Weiwei Xu, Yuan Tian, and Changxi Zheng. Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph.*, 34(6), 2015. 2

[46] Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. Continuum foam: A material point method for shear-dependent flows. *ACM Trans. Graph.*, 34(5), 2015. 2, 1

[47] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*. Springer, 2024. 2

[48] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. *European Conference on Computer Vision (ECCV)*, 2024. 2

[49] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 6

[50] Zeshun Zong, Xuan Li, Minchen Li, Maurizio M Chiaramonte, Wojciech Matusik, Eitan Grinspun, Kevin Carlberg, Chenfanfu Jiang, and Peter Yichen Chen. Neural stress fields for reduced-order elastoplasticity and fracture. *arXiv preprint arXiv:2310.17790*, 2023. 2

FreeForm: Reduced-Order Deformable Simulation from Particle-Based Skinning Eigenmodes

Supplementary Material

6. Further Discussion on MPM and SPH

As mentioned in Sec. 2, many particle-based physics simulation methods have been proposed. MPM and SPH are particularly attractive, as they can handle a wide variety of material models, including plasticity effects, topology and phase changes, and, as is our focus here, elastodynamics.

However, this versatility regarding topology changes is also what makes MPM and SPH sensitive to spatial discretization, as the interaction stencils change over time. For MPM, two particles will interact if and only if they share a least one common grid node; for SPH, if only if the support of their kernels overlap. Inevitably, as the material is increasingly stretched, particles will eventually get further apart than this critical distance, and numerical fracture will happen, as illustrated in Figure 7. For SPH, particles becoming locally co-dimensional can also lead to numerical conditioning issues, with the velocity gradient becoming singular and requiring special care [41]. For MPM elastic bodies, deformation gradient estimation can be made more robust by leveraging a rest pose mesh [20], or even by rasterizing forces from a Lagrangian model [14], when such a representation is available. However, integration accuracy will still suffer when the number of particles per cell is not sufficient, while a too coarse grid will exhibit locking; this makes picking the grid resolution difficult for uneven particle distributions.

The same sampling criteria apply to our RKPM kernel centers; however, our method only needs to worry about the rest pose, for which it is easier to control the sample distribution and kernel width, while MPM and SPH need to have the particles remain well distributed at each timestep. Resampling particles over time can avoid those issues [46]; however, this is not really practical when simulating a predefined number of Gaussian splats, for instance.

Moreover, while so-called implicit variants of MPM and SPH have been proposed, most still treat advection as an explicit step and are therefore subject to the Courant–Friedrichs–Lewy (CFL) condition. In contrast, our total-Lagrangian approach, with shape functions remaining fixed over time and implicit time stepping, does not have a constraint on the size of time step.

7. Implementation Details

7.1. Our Method

In this section, we provide implementation details with regard to our proposed method.

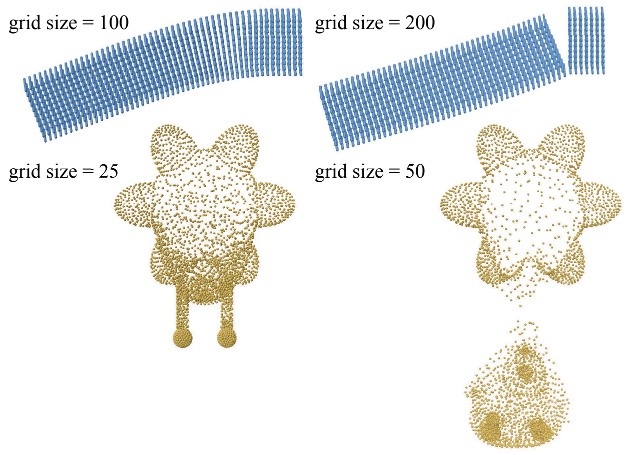

Figure 7. The Material Point Method (MPM) is versatile, but presents challenges for deformable body simulation due to the occurrence of unintended numerical fracture depending on the grid resolution (shown in top-left corner).

RKPM construction. Given an input object, our method first constructs a set of RKPM kernels around the object shape. We start by dense sampling integration points around the object. Unless otherwise stated, we sample points on a uniform grid inside the object bounding box, and then reject points outside the object for shapes with well-defined an inside/outside test functions, e.g., a watertight mesh. For shapes where the inside/outside test cannot be easily applied, e.g. 3DGS, we directly use the given points as integration points (after downsampling if necessary to avoid out-of-memory error).

With the integration points determined, we then apply Farthest Point Sampling (FPS) to select around 1k points as RKPM kernel centers to ensure that kernels are equidistantly distributed. We set each Gaussian kernel radius r to be the minimal distance to reach two other centers, so that the space around the kernels is well-covered by RKPM.

Eigenanalysis. After RKPM kernels are constructed, we assemble the Hessian matrix $\mathbf{H}_w$ according to Prop. 1 and $\mathcal{M}$ according to Eq. (14). We perform generalized eigendecomposition using `torch.lobpcg`[1] on CUDA in double precision. We take the eigenvectors associated with m smallest eigenvalues (excluding the constant mode associ-

[1]Pytorch document `https://docs.pytorch.org/docs/stable/generated/torch.lobpcg.html`

ated with zero eigenvalue) as the nodal values for our skinning weight estimation. In the main text, we always report timing for eigenanalysis with $m = 32$.

Simulation. Once the skinning weights are determined, we run simulation of deformable objects in the same approach as Simplicits [31]. Our implementation is built on top of the open-source Kaolin[2] library based on the Warp language and PyTorch, and we further boost the runtime performance with more efficient kernel launching via CUDA graph captured in Warp and PyTorch while maintaining the same simulation results. In the paper, we test both our method and Simplicits with the same enhanced implementation for fairness.

We use Newton's method with line search based on Wolfe conditions to solve implicit time-stepping in Eq. (2), allowing up to 20 updates per time step with a convergence tolerance of 10^{-8}. To solve the linear system in Newton's method, we use the direct solver implemented in PyTorch[3].

7.2. Baseline Methods

Simplicits. We use the recommended implementation in Kaolin. The neural field for skinning weights is a 6-layer MLP (excluding the input and output layers) of layer width 64, trained for 10k iterations using the Adam optimizer, with a learning rate of 10^{-3}. The elasticity and orthogonality loss weights are set to 0.1 and 10^6, respectively. At run time, we adopt the same simulation implementation as our method for Simplicits.

MPM and SPH. For MPM, we use the GPU-based warp-mpm[4] implementation. For the standard beam test, we use $\mathrm{dt} = 10^{-4}$s, whereas $\mathrm{dt} = 10^{-3}$s leads to numerical explosion. For SPH, we use the elasticity model in Kugelstadt et al. [23] implemented in SPlisHSPlasH[5], and $\mathrm{dt} = 0.01$s for the beam test.

Finite Element Methods. Finite Element Methods are widely adopted and regarded as the standard approach for simulating elasticity. In this work, we use converged FEM simulation results as the gold standard reference for evaluating various methods. Our full-DoF FEM simulation is implemented based on `warp.fem`[6]. The simulation uses the same Neo-Hookean elasticity model in Eq. (16) on tetrahedral meshes. The solver adopts the backward Euler method for implicit time stepping, solved by Newton's method.

[2]https://github.com/NVIDIAGameWorks/kaolin
[3]https://docs.pytorch.org/docs/stable/generated/torch.linalg.solve.html
[4]https://github.com/zeshunzong/warp-mpm/
[5]https://github.com/InteractiveComputerGraphics/SPlisHSPlasH
[6]https://nvidia.github.io/warp/modules/fem.html

m	Simplicits	**Ours**	FEM	MPM	SPH
6	5.01	3.01	427.2	23.1	37.8
9	3.95	3.71			
16	5.09	5.42			
32	10.0	10.7			

Table 5. Comparison of runtime in milliseconds (ms) for a simulation step of $\mathrm{dt} = 0.01$s on average. The timing results are reported for the beam-bending experiment.

7.3. Runtime Comparison

We report the time used to simulate a period of 0.01s in the beam-bending experiment by various methods in Table 5. Our method and Simplicits adopt the same solver and therefore reach similar runtime performance. FEM is a full-DoF simulation that yields accurate results but takes orders of magnitude longer to run. MPM and SPH also achieve competitive runtime performance, but are still slower than our reduced-order formulation (and are subject to other limitations as discussed in Sec. 6).

8. Proof of Proposition 1

In this section, we provide a proof of Proposition 1. Consider the deformation map $\Phi(\mathbf{X}, \mathbf{d}) = \mathbf{u}(\mathbf{X}, \mathbf{d}) + \mathbf{X}$, where the displacement $\mathbf{u}(\mathbf{X}, \mathbf{d})$ is parameterized by RKPM in Eq. (6) and the DoFs are the nodal displacements $\mathbf{d} = \{\mathbf{d}_k \in \mathbb{R}^3\}_{k=1}^K = \{(\mathbf{d}_k^x, \mathbf{d}_k^y, \mathbf{d}_k^z)^T\}_{k=1}^K$.

$$\begin{aligned}\Phi(\mathbf{X}, \mathbf{d}) &= \begin{bmatrix}\Phi^x(\mathbf{X}, \mathbf{d})\\ \Phi^y(\mathbf{X}, \mathbf{d})\\ \Phi^z(\mathbf{X}, \mathbf{d})\end{bmatrix}\\ &= \sum_{k=1}^K \phi_k(\mathbf{X})\mathbf{d}_k + \mathbf{X},\end{aligned} \tag{18}$$

The deformation gradient $\mathbf{F}(\mathbf{X}, \mathbf{d}) \in \mathbb{R}^{3\times3}$ is the spatial derivative of Φ at each point $\mathbf{X} \in \Omega$, given by

$$\begin{aligned}\mathbf{F}(\mathbf{X}, \mathbf{d}) = \nabla\Phi(\mathbf{X}, \mathbf{d}) &= \begin{bmatrix}\nabla\Phi^x(\mathbf{X}, \mathbf{d})^T\\ \nabla\Phi^y(\mathbf{X}, \mathbf{d})^T\\ \nabla\Phi^z(\mathbf{X}, \mathbf{d})^T\end{bmatrix}\\ &= \sum_{k=1}^K \mathbf{d}_k \nabla\phi_k(\mathbf{X})^T + \mathbf{I}\\ &= \overbrace{\begin{bmatrix}\mathbf{d}_1 & \dots & \mathbf{d}_K\end{bmatrix}}^{\mathbf{d}^T\in\mathbb{R}^{3\times K}} \overbrace{\begin{bmatrix}\nabla\phi_1(\mathbf{X})^T\\ \vdots\\ \nabla\phi_K(\mathbf{X})^T\end{bmatrix}}^{\nabla\boldsymbol{\phi}(\mathbf{X})\in\mathbb{R}^{K\times3}} + \mathbf{I}.\end{aligned} \tag{19}$$

where ∇ denotes the derivative with respect to the point $\mathbf{X}$. With a slight abuse of notation, we write

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{X}) &= \begin{bmatrix}\phi_1(\mathbf{X}) \\ \dots \\ \phi_K(\mathbf{X})\end{bmatrix} \in \mathbb{R}^K, \\ \mathbf{d} &= \begin{bmatrix}\mathbf{d}_1^T \\ \vdots \\ \mathbf{d}_K^T\end{bmatrix} = \begin{bmatrix}\mathbf{d}^x, \mathbf{d}^y, \mathbf{d}^z\end{bmatrix} \in \mathbb{R}^{K\times 3},\end{aligned} \tag{20}$$

where $\mathbf{d}^x, \mathbf{d}^y, \mathbf{d}^z \in \mathbb{R}^K$ are the nodal displacements in the x, y, z directions, respectively. Then we have

$$\begin{aligned}\mathbf{F}(\mathbf{X}, \mathbf{d}) &= \begin{bmatrix}\mathbf{F}_{11} & \mathbf{F}_{12} & \mathbf{F}_{13} \\ \mathbf{F}_{21} & \mathbf{F}_{22} & \mathbf{F}_{23} \\ \mathbf{F}_{31} & \mathbf{F}_{32} & \mathbf{F}_{33}\end{bmatrix} \\ &= \begin{bmatrix}(\mathbf{d}^x)^T \\ (\mathbf{d}^y)^T \\ (\mathbf{d}^z)^T\end{bmatrix}\begin{bmatrix}\partial_x \boldsymbol{\phi}(\mathbf{X}) & \partial_y \boldsymbol{\phi}(\mathbf{X}) & \partial_z \boldsymbol{\phi}(\mathbf{X})\end{bmatrix} + \mathbf{I} \\ &= \mathbf{d}^T \nabla \boldsymbol{\phi}(\mathbf{X}) + \mathbf{I},\end{aligned} \tag{21}$$

where $\partial_x \boldsymbol{\phi}(\mathbf{X}) = \begin{bmatrix}\partial \phi_1(\mathbf{X})/\partial x & \dots & \partial \phi_K(\mathbf{X})/\partial x\end{bmatrix}^T$, and similarly for $\partial_y \boldsymbol{\phi}(\mathbf{X})$ and $\partial_z \boldsymbol{\phi}(\mathbf{X})$.

Apply the strain energy density and integrate over the domain Ω by the Monte Carlo method, we get the total elastic potential energy as

$$E_{\text{pot}}(\mathbf{d}) = \int_\Omega \Psi(\mathbf{F}(\mathbf{X}, \mathbf{d}))d\mathbf{X} \approx \sum_i v_i \Psi(\mathbf{F}(\mathbf{X}_i, \mathbf{d})), \tag{22}$$

where $\mathbf{X}_i$ is the i-th sample point, and v_i is the weight of the i-th sample point. Then its weight-space Hessian matrix

$$\mathbf{H}_w = \mathbf{H}_{xx} + \mathbf{H}_{yy} + \mathbf{H}_{zz} \tag{23}$$

contains the Hessian of E_{pot} with respect to the nodal displacements $\mathbf{d}^x, \mathbf{d}^y, \mathbf{d}^z$ around the rest position $\mathbf{d} = \mathbf{0}$, respectively. Take $\mathbf{H}_{xx}$ as an example, and denote $\text{Hess}(\cdot, \cdot)$ as the Hessian of the first argument with respect to the second argument.

$$\begin{aligned}\mathbf{H}_{xx} &= \text{Hess}(E_{\text{pot}}, \mathbf{d}^x) = \text{Hess}(\sum_i v_i \Psi(\mathbf{F}_i), \mathbf{d}^x) \\ &= \sum_i v_i \,\text{Hess}(\Psi(\mathbf{F}_i), \mathbf{d}^x),\end{aligned} \tag{24}$$

where $\mathbf{F}_i = \mathbf{F}(\mathbf{X}_i, \mathbf{d})$ is a shorthand for the deformation gradient at the i-th sample point. Notice that $\mathbf{F}$ is an affine transformation of $\mathbf{d}$, so the chain rule for Hessian matrix gives

$$\text{Hess}(\Psi(\mathbf{F}_i), \mathbf{d}^x) = \mathbf{J}_i^T \,\text{Hess}(\Psi(\mathbf{F}_i), \mathbf{F}_i)\mathbf{J}_i, \tag{25}$$

where $\mathbf{J}_i$ is the Jacobian matrix of $\text{vec}(\mathbf{F}_i)$ with respect to $\mathbf{d}^x$ around $\mathbf{d} = 0$ and $\text{Hess}(\Psi(\mathbf{F}_i), \mathbf{F}_i)$ is the Hessian of strain energy density with respect to the flattened $\text{vec}(\mathbf{F}_i) \in \mathbb{R}^9$ around $\mathbf{F}_i = \mathbf{I}$. For the Neo-Hookean energy in Eq. (16), we have the gradient

$$\begin{gathered}\text{grad}(\Psi(\mathbf{F}_i), \mathbf{F}_i) = \mu \mathbf{F}_i + (\lambda + \mu)(J - \gamma)J\mathbf{F}_i^{-T}, \\ J = \det(\mathbf{F}_i)\end{gathered} \tag{26}$$

and the Hessian

$$\begin{gathered}\text{Hess}(\Psi(\mathbf{F}_i), \mathbf{F}_i) = \mu \mathbf{I} \\ +(\lambda + \mu)(2J - \gamma)\,\text{vec}(J\mathbf{F}_i^{-T})^T \,\text{vec}(\mathbf{F}_i^{-T}) \\ -(\lambda + \mu)(J - \gamma)J(\mathbf{F}_i^{-1} \otimes \mathbf{F}_i^{-T})\mathbf{K}, \\ \mathbf{K} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.\end{gathered} \tag{27}$$

where $\otimes$ denotes the Kronecker product. When $\mathbf{F}_i = \mathbf{I}$, the Hessian simplifies to

$$\begin{gathered}\text{Hess}(\Psi, \mathbf{F}_i) = \mu \mathbf{I} + (\lambda + \mu)(2 - \gamma)\mathbf{K}_1 \\ -(\lambda + \mu)(1 - \gamma)\mathbf{K}, \\ \mathbf{K}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.\end{gathered} \tag{28}$$

Here λ and μ are Lamé coefficients depending on the sample point $\mathbf{X}_i$, which we omit for simplicity. On the other hand, from Eq. (21) we know that

$$\text{vec}(\mathbf{F}_i) = \begin{bmatrix}\mathbf{F}_{11} \\ \mathbf{F}_{12} \\ \mathbf{F}_{13} \\ \mathbf{F}_{21} \\ \mathbf{F}_{22} \\ \mathbf{F}_{23} \\ \mathbf{F}_{31} \\ \mathbf{F}_{32} \\ \mathbf{F}_{33}\end{bmatrix} = \begin{bmatrix}\partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^x + 1 \\ \partial_y \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^x \\ \partial_z \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^x \\ \partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^y \\ \partial_y \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^y + 1 \\ \partial_z \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^y \\ \partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^z \\ \partial_y \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^z \\ \partial_z \boldsymbol{\phi}(\mathbf{X}_i)^T \mathbf{d}^z + 1\end{bmatrix}. \tag{29}$$

Thus the Jacobian matrix $\mathbf{J}_i$ is given by

$$\mathbf{J}_i = \frac{\partial\,\mathrm{vec}(\mathbf{F}_i)}{\partial \mathbf{d}^x} = \begin{bmatrix} \partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T \\ \partial_y \boldsymbol{\phi}(\mathbf{X}_i)^T \\ \partial_z \boldsymbol{\phi}(\mathbf{X}_i)^T \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{9\times K}. \tag{30}$$

Substituting Eq. (28) and Eq. (30) into Eq. (24), we get

$$\begin{aligned} \mathrm{Hess}(\Psi(\mathbf{F}_i), \mathbf{d}^x) = \mu \sum_{s=x,y,z} \partial_s \boldsymbol{\phi}(\mathbf{X}_i) \partial_s \boldsymbol{\phi}(\mathbf{X}_i)^T \\ +(\lambda+\mu)(2-\gamma)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T \\ +(\lambda+\mu)(\gamma-1)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T \end{aligned} \tag{31}$$

With $\lambda = 1 + \mu/(\lambda+\mu)$, we have

$$\begin{aligned} \mathrm{Hess}(\Psi(\mathbf{F}_i), \mathbf{d}^x) = (\lambda+2\mu)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T \\ +\mu\partial_y \boldsymbol{\phi}(\mathbf{X}_i)\partial_y \boldsymbol{\phi}(\mathbf{X}_i)^T + \mu\partial_z \boldsymbol{\phi}(\mathbf{X}_i)\partial_z \boldsymbol{\phi}(\mathbf{X}_i)^T. \end{aligned} \tag{32}$$

Similarly,

$$\begin{aligned} \mathrm{Hess}(\Psi(\mathbf{F}_i), \mathbf{d}^y) = (\lambda+2\mu)\partial_y \boldsymbol{\phi}(\mathbf{X}_i)\partial_y \boldsymbol{\phi}(\mathbf{X}_i)^T \\ +\mu\partial_x \boldsymbol{\phi}(\mathbf{X}_i)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T + \mu\partial_z \boldsymbol{\phi}(\mathbf{X}_i)\partial_z \boldsymbol{\phi}(\mathbf{X}_i)^T, \\ \mathrm{Hess}(\Psi(\mathbf{F}_i), \mathbf{d}^z) = (\lambda+2\mu)\partial_z \boldsymbol{\phi}(\mathbf{X}_i)\partial_z \boldsymbol{\phi}(\mathbf{X}_i)^T \\ +\mu\partial_x \boldsymbol{\phi}(\mathbf{X}_i)\partial_x \boldsymbol{\phi}(\mathbf{X}_i)^T + \mu\partial_y \boldsymbol{\phi}(\mathbf{X}_i)\partial_y \boldsymbol{\phi}(\mathbf{X}_i)^T. \end{aligned} \tag{33}$$

In conclusion,

$$\begin{aligned} \mathbf{H}_w &= \mathbf{H}_{xx} + \mathbf{H}_{yy} + \mathbf{H}_{zz} \\ &= \sum_i \sum_{s=x,y,z} v_i\,\mathrm{Hess}(\Psi(\mathbf{F}_i), \mathbf{d}^s) \\ &= \sum_i \sum_{s=x,y,z} v_i(\lambda+4\mu)\partial_s \boldsymbol{\phi}(\mathbf{X}_i)\partial_s \boldsymbol{\phi}(\mathbf{X}_i)^T \\ &= \sum_i v_i(\lambda+4\mu)\nabla\boldsymbol{\phi}(\mathbf{X}_i)\nabla\boldsymbol{\phi}(\mathbf{X}_i)^T \\ &\approx \int_\Omega (\lambda(\mathbf{X})+4\mu(\mathbf{X}))\nabla\boldsymbol{\phi}(\mathbf{X})\nabla\boldsymbol{\phi}(\mathbf{X})^T d\mathbf{X}, \end{aligned} \tag{34}$$

The (i,j)-th element of $\mathbf{H}_w$ is

$$(\mathbf{H}_w)_{ij} = \int_\Omega (\lambda(\mathbf{X})+4\mu(\mathbf{X}))\nabla\phi_i(\mathbf{X})^T\nabla\phi_j(\mathbf{X})d\mathbf{X}. \tag{35}$$

This completes the proof of Proposition 1.