Data-Free Learning of Reduced-Order Kinematics

NICHOLAS SHARP, NVIDIA, University of Toronto, Canada
CRISTIAN ROMERO, Universidad Rey Juan Carlos, Spain
ALEC JACOBSON, University of Toronto, Adobe Research, Canada
ETIENNE VOUGA, The University of Texas at Austin, USA
PAUL G. KRY, McGill University, Canada
DAVID I.W. LEVIN, University of Toronto, NVIDIA, Canada
JUSTIN SOLOMON, Massachusetts Institute of Technology, USA

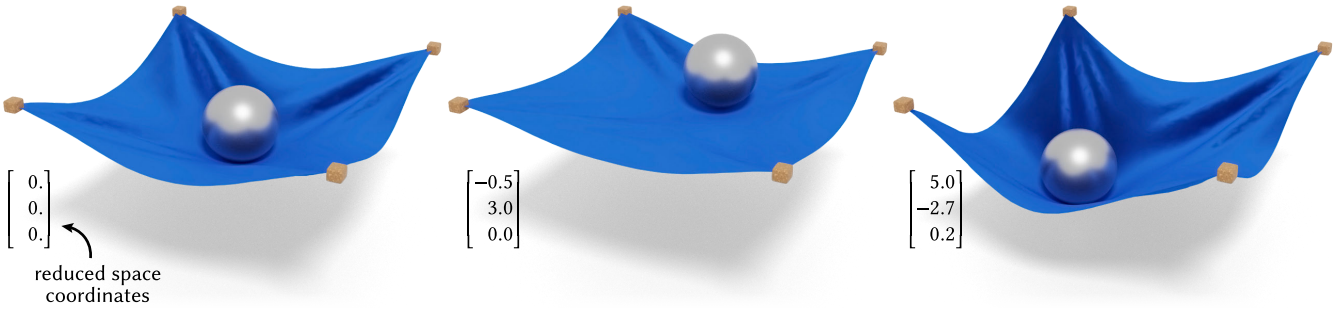

Fig. 1. We fit neural networks as reduced-order models to learn low-dimensional approximations of complex physical systems. This approach applies to a broad range of systems and requires no data as input, only a differentiable energy function and a seed state for sampling. Here, we show a 3-dimensional neural subspace for a system wherein a ball rolls around on a pinned cloth. The ball location and cloth geometry are simultaneously encoded by the subspace, which is fit automatically from a potential energy that includes gravity, cloth bending and stretching, and a collision penalty between the ball and the cloth.

Physical systems ranging from elastic bodies to kinematic linkages are defined on high-dimensional configuration spaces, yet their typical low-energy configurations are concentrated on much lower-dimensional subspaces. This work addresses the challenge of identifying such subspaces *automatically*: given as input an energy function for a high-dimensional system, we produce a low-dimensional map whose image parameterizes a diverse yet low-energy submanifold of configurations. The only additional input needed is a single seed configuration for the system to initialize our procedure; no dataset of trajectories is required. We represent subspaces as neural networks that map a low-dimensional latent vector to the full configuration space, and propose a training scheme to fit network parameters to any system of interest. This formulation is effective across a very general range of physical systems; our experiments demonstrate not only nonlinear and very low-dimensional elastic body and cloth subspaces, but also more general systems like colliding rigid bodies and linkages. We briefly explore applications built on this formulation, including manipulation, latent interpolation, and sampling.

Additional Key Words and Phrases: simulation, neural network, subspaces

1 INTRODUCTION

Physical simulation algorithms perennially achieve new heights of detail and fidelity. Modern computer graphics techniques successfully capture phenomena from elasticity to fluid motion, producing visual effects that are nearly indistinguishable from real life. With this added realism, however, comes substantial computational expense, often placing detailed physical simulation in the realm of offline computations involving many degrees of freedom.

In settings like interactive graphics, however, it is advantageous to reparameterize the system with a much smaller number of degrees of freedom which describe only the states that are actually of interest. These *subspaces*, or *reduced order models* enable downstream tasks, most traditionally fast simulation in reduced coordinates, but also other operations such as user-guided animation, interpolation, or sampling states of the system.

However, identifying such subspaces is inevitably challenging, because they must trade-off between the conciseness and expressivity. Classical reduced-order simulation methods such as linear modal analysis or modal derivatives have typically focused on perturbative motions about a rest state for a deformable object. These methods are highly effective numerical schemes for fast forward-integration of system dynamics; our approach will seek a complementary technique in two senses.

First, such approaches typically only approximate object behavior in a truncated region about the rest pose, and dramatic nonlinear motions are not well-represented in the subspace. This concern is already impactful for the classic case of deformable bodies undergoing large motions, but is a total show-stopper when seeking reduced kinematics for more general physical systems, such as rigid bodies under collision penalties. In these settings, the kinematic landscape is so nonlinear that an approximation in terms of a local expansion does not capture any significant behavior.

Second, our subspaces will parameterize *only* the desired configuration space of the system. The challenge of large motions can be mitigated in perturbative methods by using a moderately large reduced basis. However, this returns to the original problem with the full configuration space: the relevant system configurations again

lie only on narrow submanifold of the space. In contrast, we seek very low-dimensional but highly nonlinear subspaces, such that even large motions and physical systems with irregular potential landscapes can be directly parameterized; an important property for applications like animation and sampling.

This work is not the first to propose using a richer class of highly nonlinear models to fit reduced kinematics (see *e.g.* [Fulton et al. 2019; Holden et al. 2019; Shen et al. 2021; Srinivasan et al. 2021]). Our motivating goal is to do so *without* data-driven fitting; we do not require any dataset of representative simulation trajectories or states as input. Collecting such a high-quality dataset is challenging and labor-intensive, both in the sense of engineering effort and user input. It is a significant obstacle for past methods which otherwise offer excellent properties [Fulton et al. 2019; Hahn et al. 2014]. To be clear, although our method leverages tools from machine learning, it is *not* data-driven in the usual sense. Instead, it mirrors recent "overfit" neural networks [Xie et al. 2022], where models are fit in isolation to each example, and neural networks are used simply as a general and easy-to-optimize nonlinear function space.

Summary. In this paper, we apply machine learning to identify a nonlinear reduced model for physical motion. Our approach is designed around two significant properties:

- We do *not* assume input data such as simulation trajectories are provided. Instead, our method is self-supervised, taking the energy function as input and automatically sampling it to explore the low-energy subspace.
- Our method is very general, and avoids specific assumptions about *e.g.* deformable bodies. It applies broadly across systems such as rigid bodies and linkages under penalty potentials, or even multi-physics combinations of several different interacting systems.

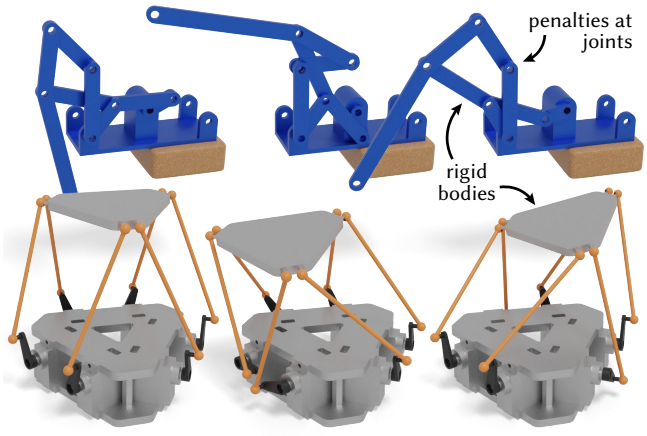

Fig. 2. Our subspaces are effective on systems beyond deformable bodies and cloth. Here, linkages are naively described by the location of each segment, under a potential with strong penalties holding joints together. Neural subspace optimization automatically discovers low-dimensional kinematic motion as we fit a 1d subspace to the Klann Linkage, *top*, and a 3d subspace to the Stewart Mechanism, *bottom*.

Provided a differentiable potential energy function describing a given physical system and a single seed state from which to begin the search, our learning algorithm automatically determines an effective nonlinear low-order model, trading off between staying in low-potential energy configurations and coverage of the configuration space. Two loss parameters are exposed to adapt our objective to the system of interest. Once fit, the parameterized kinematic subspace can be leveraged for a variety of purposes, from simulation via standard dense integrators in the subspace, to kinematic exploration and sampling. We demonstrate our approach on a variety of physical systems, including deformable bodies, cloth simulations, rigid bodies under collision, and mechanical linkages.

2 RELATED WORK

Subspaces for Simulation. Subspace simulation methods have a long history in engineering and graphics, beginning with linear modal analysis [Hildebrandt et al. 2011; James et al. 2006; James and Pai 2002; Shabana 1991; Von-Tycowicz et al. 2015]. The survey of Benner et al. [2015] provides a broad summary. Linear modes provide a concise basis expressing deformation around an object's rest state. Fast simulation methods then restrict the equations of motion to this subspace. Difficulties arise in large-deformation settings wherein the basis size must be greatly increased to approximate nonlinearity [Brandt et al. 2018]. Barbič and James [2005] augment the modal basis with second-order "modal derivatives," while still resulting in a linear deformation subspace, and Choi and Ko [2005] and Yang et al. [2015] explore rotation and higher order terms.

While modal derivatives offset some disadvantages of linear modal analysis, both techniques are limited to representing deformations centered around a rest pose. This makes representing highly nonlinear deformations and effects difficult, and obstructs the application to more general physical systems as we show in Figure 8. Snapshot methods generalize beyond a region around the rest state by collecting large databases of simulation outputs and fitting a reduced space to that data. Initial algorithms used PCA [Noor and Peters 1980] to construct an improved subspace, but the linear PCA basis still must be large to capture a wide range of deformations.

Modern neural representations such as autoencoders [Fulton et al. 2019; Shen et al. 2021] offer a potential panacea. However, such methods again rely on simulation snapshots for training, and thus resort to user-guided sampling, making these methods time consuming and compute intensive. Like us, Brandt et al. [2016] sample configuration space, but do so in a way which still only interpolates specified configurations. Even methods learning neural enrichments to linear subspaces [Romero et al. 2021] suffer from the data generation problem; no successful self-supervised, data-free learning method for nonlinear subspaces has yet been demonstrated.

Neural and Data-Driven Methods. Recent work across machine learning shows neural networks have significant potential to model complex physical systems efficiently [Gao et al. 2021; Kochkov et al. 2021; Pfaff et al. 2020; Tompson et al. 2017]. These approaches range from fitting update rules to observed data, to accelerating expensive numerical steps with data-driven proxies. The most similar of these efforts tackle problems in dynamics and deformation, often with the goal of producing efficient real-time simulators [Grzeszczuk et al.

1998; Romero et al. 2020; Zheng et al. 2021]. Applications of this work, as well as ours, include graphics, animation, robotics, design [Li et al. 2018, 2019].

The task of modeling dynamics and collisions in cloth has received particular attention [Bertiche et al. 2021, 2022; Hahn et al. 2014; Holden et al. 2019; Santesteban et al. 2022; Zhang et al. 2021]. In fact, Bertiche et al. [2021] and Santesteban et al. [2022] leverage self-supervised setups which bear some similarity to ours, although many aspects of their approach are specific to garment modeling task. Additionally, a primary challenge in our setting is avoiding collapse of the subspace, while with clothing this is automatically handled by human body shape and motion distributions.

3 METHOD

We present a straightforward approach to fit a neural network modeling low-energy kinematics of a physical system. The formulation is general, applying to a broad set of systems and capturing both linear and nonlinear subspaces. It follows widespread success fitting low-dimensional submanifolds of high-dimensional spaces using neural networks (*e.g.* [Lee and Carlberg 2020]).

3.1 Neural Subspace Maps

Consider a map f_θ, which takes a low-dimensional subspace $\mathbb{R}^d$ to the high-dimensional configuration space $\mathbb{R}^n$ of some physical system ($d \ll n$), so $f_\theta : \mathbb{R}^d \to \mathbb{R}^n$. For example, $\mathbb{R}^n$ might represent the set of all possible vertex configurations for a given triangle mesh (so, $n = 3|V|$ where $|V|$ is the number of vertices), while $\mathbb{R}^d$ parameterizes a space of deformations that move multiple vertices in tandem. The vector $\theta \in \mathbb{R}^k$ contains learnable parameters specific to the physical system, e.g. neural network weights.

Classical simulation algorithms operate on $\mathbb{R}^n$, where the potential energy $E_{\text{pot}} : \mathbb{R}^n \to \mathbb{R}$ and external forces can be evaluated directly; the expense of physical simulation then comes from the large number of variables n that must be manipulated. However, $\mathbb{R}^n$ contains many unlikely configurations, corresponding to high-energy deformations under the potential energy E_{pot}. In many settings, we can reasonably expect the kinematics to stay in the image $f_\theta(\mathbb{R}^d)$ of some map f_θ parameterizing typical configurations.

As a simple example, if we take $\theta = (A, x_0)$ for some $A \in \mathbb{R}^{n\times d}$ and $x_0 \in \mathbb{R}^n$ with $f_\theta(z) = Az + x_0$, we recover the basic setup of linear modal analysis. In this setting, x_0 is the rest state of the system, and the columns of A parameterize low-energy perturbations of x_0. However, linear models cannot fit general nonlinear kinematics.

More broadly, efficient and accurate simulation demands a map $f_\theta(\cdot)$ spanning low-energy configurations of the system. Unlike classical modal analysis, an immediate benefit of working with a more general f_θ is that f_θ can encode *nonlinear* subspaces, rather than only linear modes. In this work, we model f_θ as a neural network, with weights θ (see Section 4.2 for architectures).

Figure 7 illustrates this property on a hanging stiff cow under elastic and gravitational potentials. Traditional modal analysis is limited to linear skewing about a rest pose, whereas our neural model finds a nonlinear subspace with curved swinging motion.

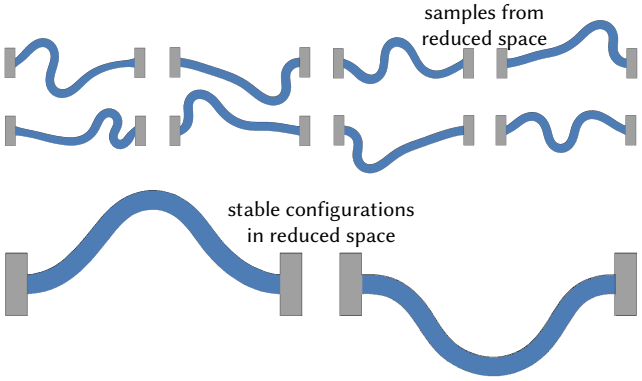

Fig. 3. We fit an 8-dimensional reduced space for a neohookean elastic bar under gravity in a compressed buckling configuration. *Top:* samples from the smooth yet irregular reduced space fit by the neural network. *Bottom:* the reduced space contains both of the stable buckled configurations.

3.2 Objective Function

For any choice of subspace architecture f_θ, we will fit the parameters θ via stochastic gradient descent on an objective function. We must then identify an objective function which drives f_θ to be a suitable, high-quality kinematic subspace.

Our key observation is that we can optimize for θ directly using the analytical description of the system—in particular, the potential energy function $E_{\text{pot}}(\cdot)$—rather than requiring training data. This approach sidesteps the data collection needed for supervised models: We do not assemble a dataset of motion trajectories or even forward-integrate the dynamics of the system during training.

We might seek low-energy subspaces of a system by minimizing the expected potential energy of randomly-sampled subspace configurations z as follows:

$$\mathbb{E}_{z\sim\mathcal{N}}\Big[E_{\text{pot}}(f_\theta(z))\Big]. \tag{1}$$

Here, $\mathcal{N}$ denotes the Gaussian distribution over $\mathbb{R}^d$ with mean 0 and variance $I_{d\times d}$; $\mathbb{E}$ denotes the expectation with respect to a random variable. Note we have not put a scale on our latent variable z, so we are using $\mathcal{N}$ to capture a reasonable range of values.

Minimizing Equation 1 with respect to θ, however, yields an uninteresting map f_θ: The optimal f_θ maps all latent variables z to the lowest-energy configuration, i.e., the minimizer of E_{pot}.

To combat the degeneracy above, we also expect our subspaces to span some sizable region of configuration space $\mathbb{R}^n$. To accomplish this goal, we could attempt to impose isometry up to scale on f_θ, e.g. by enforcing that

$$|f_\theta(z) - f_\theta(z')|_M \approx \sigma|z - z'|$$

for typical $z, z' \in \mathbb{R}^d$. Here the distance in configuration space $\mathbb{R}^n$ is measured with respect to the system's mass matrix $M \in \mathbb{R}^{n\times n}$: $|x|_M^2 := x^\top Mx$. Equivalently, we write:

$$\log \frac{|f_\theta(z) - f_\theta(z')|_M}{\sigma|z - z'|} \approx 0 \tag{2}$$

Enforcing strict equality in Equation 2 for all z, z' is a stiff constraint; indeed, one can show that changing $\approx$ to $=$ above forces f_θ to be affine (see Proposition B.1 in the supplemental material). Hence, we instead use a soft penalty to avoid degeneracies:

$$\mathbb{E}_{z,z' \sim \mathcal{N}} \left[\left(\log \frac{|f_\theta(z) - f_\theta(z')|_M}{\sigma |z - z'|} \right)^2 \right]. \tag{3}$$

Intuitively, this expression prefers maps $f_\theta(\cdot)$ whose Lipschitz constant is roughly σ everwhere. Similar formulations have recently been leveraged in other contexts by Du et al. [2021].

Combining these terms and using $\lambda \in \mathbb{R}$ as a weight, we optimize for the parameters θ as follows:

$$\min_\theta \mathbb{E}_{z,z' \sim \mathcal{N}} \left[E_{\text{pot}}(f_\theta(z)) + \lambda \left(\log \frac{|f_\theta(z) - f_\theta(z')|_M}{\sigma |z - z'|} \right)^2 \right] \tag{4}$$

In this formulation, our latent subspaces map from a roughly unit-scaled region about the origin, with low-energy configurations concentrated near 0. To accelerate subspace fitting in practice, rather than sampling pairs z, z' to evaluate Equation 3, we estimate it pairwise on all z samples in a training batch.

The hyperparameter σ adjusts the size of the subspace. Small σ yields subspaces that are tightly concentrated around low-energy configurations, while large σ can force the map's image to include higher-energy states. The weighting parameter λ should be chosen to ensure that Equation 2 roughly holds. Hyperparameters inevitably arise due to the wide variety of scaling and units used in physical energies—Table 1 gives values for all experiments; see Section C.1 for additional discussion.

3.3 Reduction to Modal Analysis

When we take our parameters λ and σ to the extreme, our general model reduces to a classical linear method for modal analysis in physical simulation. In particular, in Appendix A we derive the following proposition:

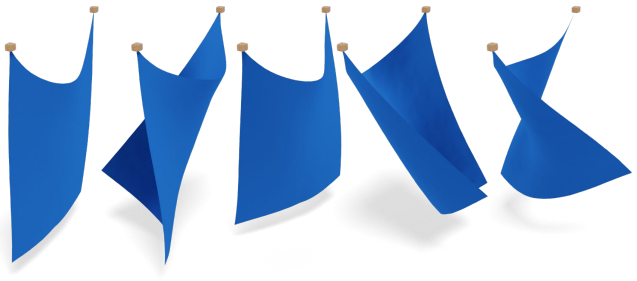

Fig. 4. States from an 8d reduced-order subspace of a thick hanging cloth.

Proposition 3.1. *Suppose f_θ has the capacity to represent affine functions. Then, as $\sigma \to 0$ and $\lambda \to \infty$, the solution to Equation 4 satisfies*

$$\begin{cases} f(z) = Az + b \\ \quad b = \arg\min_b E_{pot}(b) \\ \quad A = \sigma \cdot \textsc{top-d-generalized-eigenvectors}(M, H(b)). \end{cases} \tag{5}$$

In words, as we push to preserve geometry of the configuration space exactly ($\lambda \to \infty$) and to prioritize small neighborhoods ($\sigma \to 0$), we recover a linearization about the minimum-energy state.

3.4 Subspace Simulation

Although we focus primarily on simply encoding the kinematic subspace, if desired our subspaces f_θ can also be used for forward simulation via time integration in the latent space. In principle any integration scheme is compatible with our approach, we leverage a simple implicit Euler scheme.

We optimize to obtain the subspace configuration $\hat{z}$ in the next timestep [Hahn et al. 2012] as:

$$\hat{z} = \arg\min_z \left[\frac{1}{2h^2} |f_\theta(z) - \bar{q}|_M^2 + E_{\text{pot}}(f_\theta(z)) \right], \tag{6}$$

where h is the timestep and $\bar{q}$ is an inertial guess computed from previous configurations. The optimization is performed in the neural space z via substitution into the optimization formulation [Fulton et al. 2019] and solved using L-BFGS [Liu and Nocedal 1989].

The performance characteristics of this integration are very different from past methods; an advantage is that integration is performed in a small, dense space amenable to fast vectorized computation, while a disadvantage is that the nonlinearity of our subspaces may demand many optimization steps for accuracy. In general such integration is significantly faster than simulation in the full space, but does not outperform existing specialized subspace integrators. This work does not accelerate energy function evaluation, but could be used in conjunction with methods such as An et al. [2008].

3.5 Conditional Subspaces

A benefit of our neural subspace formulation is that subspaces can easily be conditioned on auxiliary data such as material parameters and external constraints. Conditional parameters can be adjusted to adapt the subspace to a family of systems, and even can be varied dynamically at runtime.

More precisely, we can generalize f_θ to incorporate conditional parameters as additional inputs to the neural subspace map as

$$f_\theta : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^n \qquad q \leftarrow f_\theta([z, c]) \tag{7}$$

where the conditional parameters are a vector $c \in \mathbb{R}^m$, and $[z, c]$ denotes vector concatenation. During training, we additionally sample from the space of system-defined valid conditional parameters to evaluate the expectation in Equation 4. In Figure 5 we show an elastic bar conditioned on both the location of boundary conditions, and the material stiffness.

4 ARCHITECTURES AND TRAINING

In principle, any neural architecture could be used to represent our subspace map f_θ. In this work we focus on the problem formulation itself, and consider only simple multi-layer perceptrons (MLPs).

4.1 Seeded Subspace Exploration

One important modification is needed to effectively train our subspaces, accounting for the difficulties of stiff physical energies, which are very different from typical data-driven machine learning losses. The challenge is that when starting from a randomly-initialized neural subspace map, merely finding *any* point on the low-energy submanifold in configuration space amounts to a surprisingly hard optimization problem. Consider the case of an elastic body: samples from a randomly-initialized subspace network yield configurations with vertices randomly positioned in space, leading to extremely large energies and many inverted elements.

Although much recent work has tackled robust simulation of such systems in the classical setting [Kim 2020; Lin et al. 2022; Smith et al. 2018, 2019], here we have the added challenge that the system degrees of freedom are parameterized by a highly nonlinear neural network sampled stochastically at each optimization step. As a simple solution, we propose a training procedure which explores the configuration space outward from an initial *seed* configuration $q_{\text{seed}} \in \mathbb{R}^n$ provided as input to the method. Precisely, during training only we parameterize the neural subspace map as

$$f_\theta(z) \coloneqq \rho \text{MLP}_\theta(z) + (1-\rho) q_{\text{seed}} \tag{8}$$

where ρ is a scheduling parameter which linearly increases from $0 \to 1$ as training proceeds. Crucially, at the conclusion of training, this seed state is entirely absent and the resulting network is an ordinary MLP. Likewise, we also modulate the scale parameters σ in Equation 3, multiplying by a factor of ρ because at initialization f_θ is a constant map to the seed state which cannot possibly achieve the target scale.

Intuitively, this training procedure "grows" the subspace outward as fitting proceeds, initially expanding about the seed state but ultimately gaining the freedom to parameterize an arbitrary map. This approach does demand q_{seed} as an additional input to the method, but in our systems this proved to be no additional burden: a suitable state was already implicit in the definition of the system, *e.g.* the rest state of an elastic body, or the pinned-joint configuration of a linkage. Also, we emphasize that the formulation in Section 3 does not make any assumptions involving the seed state. It does not need to be a rest state or minimal-energy configuration, any somewhat low-energy initial state will do, and the resulting subspaces have little dependence on the choice of seed. For example, in systems like Figure 3 which have no single distinguished configuration, any choice will yield similar subspaces.

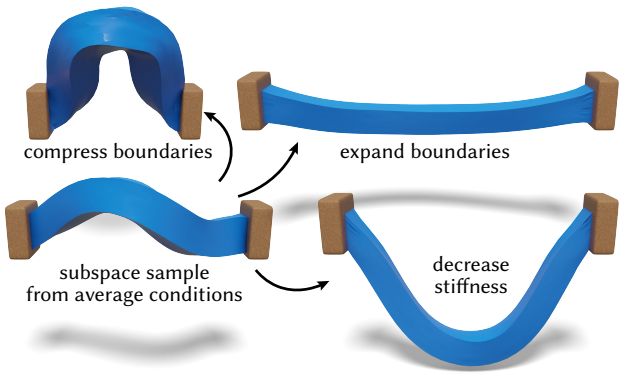

Fig. 5. Neural subspaces are easily conditioned on parameters like boundary conditions or material properties. Here, we learn a subspace for an elastic bar, conditioned on the material stiffness ratio and locations of pinned endpoints. For each parameter choice, our subspace captures plausible kinematics.

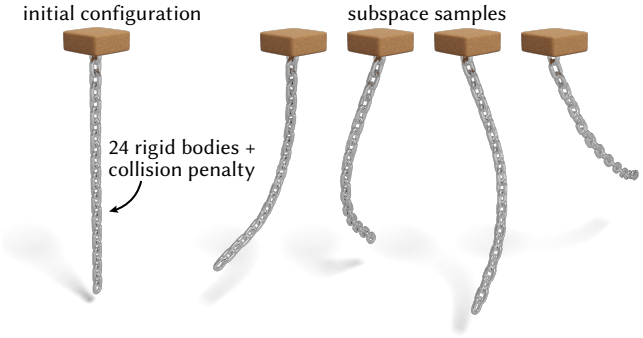

Fig. 6. A chain of rigid bodies, held together by pairwise signed-distance collision evaluated at vertices. Despite the highly irregular potential landscape, our neural subspace fits an 8d subspace spanning the large-scale continuum behaviors of the system.

4.2 Implementation Details

We implement all physical systems and neural networks in JAX [Bradbury et al. 2018], leveraging automatic differentiation to compute derivatives. Our neural networks use ELU activations and 5 hidden layers. The width of the hidden layers is adjusted from $64 - 128$ based on the scale of the problem. Table 1 gives hyperparameters for all examples in this work, and Section C.1 in the supplement provides further details about selecting parameters when applying our method to new and different physical systems. An implementation is included in the supplement and at github.com/nmwsharp/neural-physics-subspaces.

We use the Adam optimizer [Kingma and Ba 2014] for training, and a learning rate of 10^{-4} for 10^6 training steps, with batch size 32. After each $250k$ training iterations the learning rate is decayed by a factor of 0.5. Models are trained and evaluated on a single RTX 3090 GPU. Memory usage is modest ($<$ 1GB/model), and training times range from 1 minute for small systems to 1 hour for large systems. Runtime performance when exploring or sampling the subspace is extremely fast; a single forward pass of our networks takes $< 1ms$ and is dominated by pipeline latencies. If the simulation is time-stepped in the subspace, performance is dominated by the cost of evaluating and differentiating the system's energy function, generally 10s of milliseconds per timestep for the systems shown. A scaling study of fitting and evaluation is included in the supplemental material.

Table 1. Parameters and dimensions for the experiments appearing in this work. All networks are MLPs, with the latent dimension, network size, and scaling parameter σ adapted based on the degrees of freedom and desired range of motion for the kinematic space. See Section C.1 for extended discussion.

Name	Figure	Energy	Full dim	Reduced dim	Condition dim	MLP size	λ	σ
Cloth ball	Figure 1	cloth + penalty	6069	3	0	5x128	1.0	0.05
Klann linkage	Figure 2	rigid + penalty	84	1	0	5x64	1.0	1.0
Stewart platform	Figure 2	rigid + penalty	168	6	0	5x128	0.5	1.0
Bistable bar	Figures 3, 10	neohookean	830	8	0	5x128	0.5	10^{-3}
Hanging sheet	Figure 4	cloth	6072	8	0	5x128	0.1	0.1
Conditional bar	Figure 5	neohookean	942	8	2	5x128	1.0	1.0
Chain	Figures 6, 8	rigid + collision	288	8	0	5x128	1.0	1.0
Mini Chain	Figure 11	rigid + collision	24	4	0	5x128	1.0	0.3
Hanging cow	Figure 7	neohookean	14676	3	0	5x128	0.5	0.1
Cantilever	Figure 8	neohookean	1500	2	0	5x128	0.5	0.5
Elephant	Figure 9	neohookean	1926	8	0	5x64	1.0	10^{-4}

5 EVALUATIONS

5.1 Physical Systems

Here we summarize the systems/energies considered in this work. See Table 1 for problem sizes and parameter choices.

FEM. We use the finite element method (FEM) for the simulation and learning of deformable objects, discretizing the continuum in 2D and 3D examples using triangular and tetrahedral elements, respectively. We aggregate the contributions of all elements under a stable neo-Hookean material model [Smith et al. 2018] as the total potential energy.

Cloth model. We also model thin cloth sheets discretized as triangular surface meshes. Our experiments make use of a simple energy model with a bending term defined at edges [Grinspun et al. 2003], and a constant-strain Saint Venant–Kirchhoff (StVK) stretching term on faces. In Figure 4 we compute a subspace for a pinned thick hanging sheet. For now, we do not model cloth self-collisions, although Figure 1 shows basic cloth-object interactions (Figure 1).

Penalty Functions. We can also include penalties to enforce constraints in the system. For instance, we use penalties to prevent collisions between objects and to enforce joint constraints in articulated mechanisms. Our penalty functions are defined as:

$$w_{eq}|C_{eq}(q)|^2 + w_{ineq}|\min(C_{ineq}(q), 0)|^2 \tag{9}$$

where C_{eq}, C_{ineq} are the equality and inequality constraint functions with the constraints to be imposed, and $w_{eq}, w_{ineq} \in \mathbb{R}^+$ are weighting factors. These penalties must not go to infinity, because we must optimize through them during training even when sampling violates the constraint. These penalties are added to the energy function, and otherwise our subspace fitting is applied as normal, fitting local constraints without any special treatment. We demonstrate learning with penalty functions in Figures 1 & 6, where inequality penalties and signed distance model collision, and Figure 2, where an equality penalty holds linkage joints together.

Rigid Bodies. We also consider rigid body dynamics, where each body's state is described by $\mathbb{R}^{12}$ unconstrained coefficients, interpreted as the entries of a 3×4 transformation matrix $[R|t]$. An additional potential term $|(R^T R - I)|_2^2$ encourages the rotation component to be orthogonal via a Frobenius norm. Collisions are implemented as naive penalties, testing all vertices of one shape against an analytical signed distance of the other.

Figure 6 and Figure 8 show a hanging chain modeled with 24 independent rigid bodies and signed distance function collision terms preventing separation between adjacent links. This system is difficult to simulate classically, as even small-timestep implicit integrators get stuck on the energy landscape, yet the expected low-dimensional nonlinear continuum dynamics emerge automatically as we fit our subspace.

Mechanisms and Linkages. By combining rigid body dynamics with penalties holding joints together, we can also generate subspace models for complex linkages. Reduced mechanism models have applications in engineering [Boukouvala et al. 2013; Lee and Chen 2013]. As before, these linkages are naively modeled as free-floating rigid objects, with strong penalties at joints; no angular or relative parameterizations are used. The learned subspace for the system automatically finds a low-dimensional parameterization for linkage motion. Figure 2 demonstrates this behavior on the planar Klann linkage and 3D Stewart mechanisms, both of which have applications in robotics.

5.2 Comparisons and Applications

In Figure 7 we quantitatively evaluate $d = 3$ subspaces computed on a stiff deformable object pinned at a single point, and measure the strain due to nonrigid deformation. We compare our method with linear modal analysis, modal derivatives and the principal component basis (PCA) of a small dataset recorded in an interactive session. The linear approaches cannot represent rotations, and instead shear and scale the shape, while our approach fits a nonlinear rotation. Note that recent supervised approaches have also tackled rotations by learning local motions coupled with a differentiable physics layer [Srinivasan et al. 2021]. Also, see Section C.2 for an additional comparison to a baseline supervised approach.

In Figure 8 we perform a side-by-side test of our subspaces, linear modal analysis, and modal derivatives on a heterogeneous deformable bar and rigid body chain. For all methods we compute a

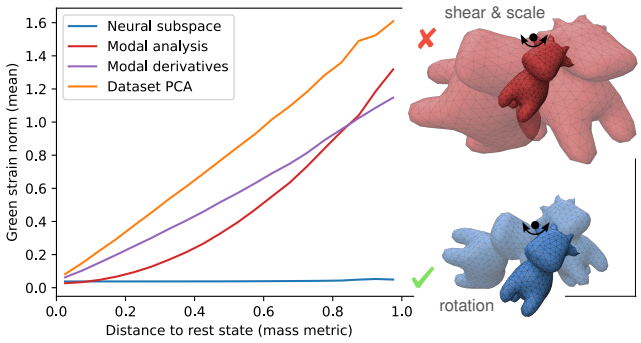

Fig. 7. Our neural maps encode general nonlinear subspaces, greatly increasing expressivity at low dimension count. Here, we fit a 3d subspace to a stiff 3D deformable body under gravity, pinned at a single point, and measure the rigidity of the resulting subspace. Linear and quadratic approaches cannot encode the rotating motion in this low-dimensional subspace, whereas our method easily fits it.

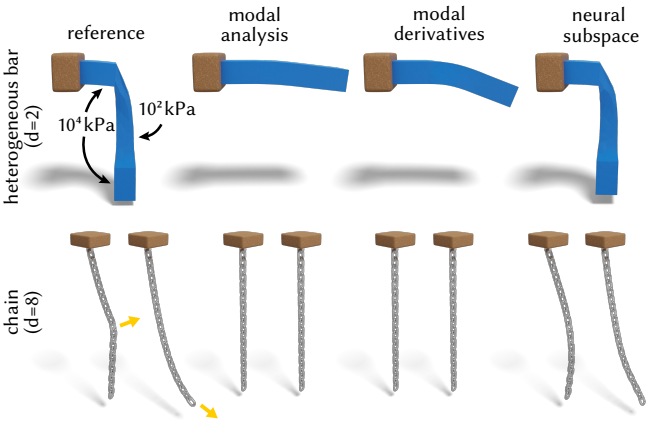

Fig. 8. A comparison of data-free subspace generation methods: ordinary linear modal analysis, modal derivatives [Barbič and James 2005], and our approach, applied to a heterogeneous 3D bar and a rigid body chain with collision penalties. Each frame applies the same external load and visualizes the equilibrium response in the subspace. For the chain, the classic local methods contain no useful motions even at $d = 8$.

subspace of matching dimension, then apply equivalent external loads and visualize the resulting subspace equilibrium. In both experiments, our subspace much more-closely matches the expected reference physics, demonstrating the effectiveness of our approach for fitting low-dimensional yet expressive subspaces.

Recent work on generating rich nonlinear reduced spaces requires an input dataset of representative configurations or trajectories [Fulton et al. 2019; Holden et al. 2019; Shen et al. 2021]. In these methods, data collection is a laborious, problem-specific process that requires humans in the loop or a scripted procedure to identify typical trajectories, as noted in *e.g.* [Fulton et al. 2019, Sec 4.4] and [Shen et al. 2021, Sec 6.1-6.2]. In contrast, our approach does not require an input dataset. We generally do not expect our approach to outperform

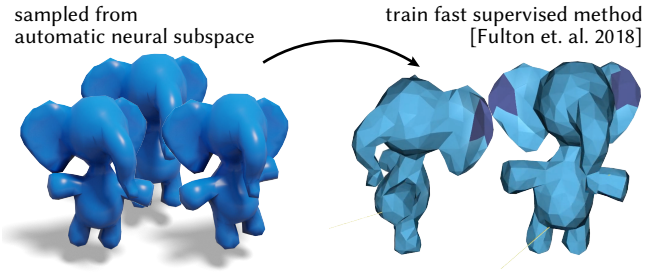

Fig. 9. Our approach enables automatic sampling of diverse low-energy states of a system, useful for downstream data-driven applications. The AutoDef algorithm [Fulton et al. 2019] offers fast, high-quality deformable simulation in reduced space, but requires a laborious process to collect training data. We first fit our subspace automatically to the deformable body of interest, then sample from the subspace to train [Fulton et al. 2019], sidestepping the need for data collection. See supplement for details.

a supervised method trained on a sufficiently large and high-quality dataset; if a dataset is available it should certainly be used.

We also present two preliminary applications which show the complementary value of our low-dimensional data-free scheme. Because our subspaces densely map on to the desired submanifold of configuration space, we can perform user-guided animation in the subspace. The supplemental video shows a looping animation of the hanging chain constructed by choosing a set of keyframes in the latent space and applying a cyclic Catmull-Rom spline interpolation. Additionally, in Figure 9, we use our automatic method as a *sampler* for a downstream specialized supervised method, overcoming the primary limitation of needing to collect a dataset.

6 CONCLUSION

This work introduces a promising approach for fitting kinematic subspaces directly to physical systems, without gathering datasets of trajectories.

Limitations. Our subspace training procedure inherits both the difficulties of optimizing deep neural networks and of numerically integrating stiff physical systems.

Local minima may also result from isolated, locally stable configurations. For example, the inset figure shows an elastic bar pinned at both ends, where by-chance the training procedure has found a local minimum that respects the boundaries but has a 360° turn.

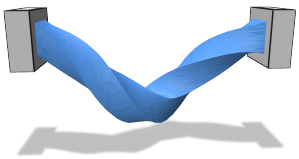

Fig. 10. A twisted subspace.

More broadly, our subspaces may not perfectly reproduce desired configurations due to such local minima or simply insufficient model capacity, leading to artifacts that can be seen in some of our simulations (*e.g.* the cloth in Figures 1 & 4, and asymmetries in Figure 5). We find that the training procedure in Section 4.1 generally avoids such artifacts, but it cannot guarantee to eliminate them. Future work could develop numerical methods tailored to this hybrid problem.

Here we mainly consider low-dimensional subspaces, fitting higher dimensional subspaces $d > 10$ with our method does not necessarily capture additional effects, perhaps because Equation 3 becomes less effective as dimension increases. More fundamentally, traditional subspaces methods are accompanied by theoretical analysis, while our neural networks currently have no corresponding physical theory to quantify which effects are captured and which are truncated, beyond the limiting models in Section 3.3.

Future Work. Here we used MLP architectures to encode the subspace map, but other network architectures could offer additional properties, such as equivariant networks [Bronstein et al. 2021] which build in rigid-invariance, or set-based networks [Qi et al. 2017; Wang et al. 2019] to model systems like particle fluids.

Future work could seek smoother subspaces, *e.g.* via Lipschitz regularization of the MLP [Arjovsky et al. 2017], for faster implicit timestep convergence (Equation 6) at large step sizes. We could also expand our models for increased semantic control of subspaces, or to learn models that generalize over many physical systems at once.

7 ACKNOWLEDGEMENTS

The Bellairs Workshop on Computer Animation was instrumental in the conception of the research presented in this paper.

The authors acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fields Institute for Mathematics, and the Vector Institute for AI. This research is funded in part by the National Science Foundation (HCC-2212048), NSERC Discovery (RGPIN–2022–04680), the Ontario Early Research Award program, the Canada Research Chairs Program, a Sloan Research Fellowship, the DSI Catalyst Grant program and gifts by Adobe Systems. The MIT Geometric Data Processing group acknowledges the generous support of Army Research Office grants W911NF2010168 and W911NF2110293, of Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grants IIS-1838071 and CHS-1955697, from the CSAIL Systems that Learn program, from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, from a gift from Adobe Systems, and from a Google Research Scholar award.

REFERENCES

S. S. An, T. Kim, and D. L. James. 2008. Optimizing Cubature for Efficient Integration of Subspace Deformations. *ACM Trans. Graph.* 27, 5, Article 165 (dec 2008), 10 pages. https://doi.org/10.1145/1409060.1409118

M. Arjovsky, S. Chintala, and L. Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 214–223. https://proceedings.mlr.press/v70/arjovsky17a.html

J. Barbič and D. L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (jul 2005), 982–990. https://doi.org/10.1145/1073204.1073300

P. Benner, S. Gugercin, and K. Willcox. 2015. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review* 57, 4 (2015), 483–531.

H. Bertiche, M. Madadi, and S. Escalera. 2021. PBNS: physically based neural simulation for unsupervised garment pose space deformation. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–14.

H. Bertiche, M. Madadi, and S. Escalera. 2022. Neural Cloth Simulation. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14.

F. Boukouvala, Y. Gao, F. Muzzio, and M. G. Ierapetritou. 2013. Reduced-order discrete element method modeling. *Chemical Engineering Science* 95 (2013), 12–26.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. 2018. JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax

C. Brandt, E. Eisemann, and K. Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

C. Brandt, C. von Tycowicz, and K. Hildebrandt. 2016. Geometric flows of curves in shape space for processing motion of deformable objects. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 295–305.

M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478* (2021).

M. G. Choi and H.-S. Ko. 2005. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics* 11, 1 (2005), 91–101.

Y. Du, K. Collins, J. Tenenbaum, and V. Sitzmann. 2021. Learning signal-agnostic manifolds of neural fields. *Advances in Neural Information Processing Systems* 34 (2021), 8320–8331.

L. Fulton, V. Modi, D. Duvenaud, D. I. W. Levin, and A. Jacobson. 2019. Latent-space Dynamics for Reduced Deformable Simulation. *Computer Graphics Forum* (2019).

H. Gao, L. Sun, and J.-X. Wang. 2021. Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids* 33, 7 (2021), 073603.

D. Girard. 1987. Un algorithme simple et rapide pour la validation croisée généralisée sur des problèmes de grande taille. *Rapport de recherche RR 665-M, TIM3-IMAG, Université de Grenoble* (1987).

E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. 2003. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Citeseer, 62–67.

R. Grzeszczuk, D. Terzopoulos, and G. Hinton. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 9–20.

F. Hahn, S. Martin, B. Thomaszewski, R. Sumner, S. Coros, and M. Gross. 2012. Rig-space physics. *ACM transactions on graphics (TOG)* 31, 4 (2012), 1–8.

F. Hahn, B. Thomaszewski, S. Coros, R. W. Sumner, F. Cole, M. Meyer, T. DeRose, and M. Gross. 2014. Subspace clothing simulation using adaptive bases. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–9.

K. Hildebrandt, C. Schulz, C. v. Tycowicz, and K. Polthier. 2011. Interactive surface modeling using modal analysis. *ACM Transactions on Graphics (TOG)* 30, 5 (2011), 1–11.

D. Holden, B. C. Duong, S. Datta, and D. Nowrouzezahrai. 2019. Subspace Neural Physics: Fast Data-Driven Interactive Simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '19)*. Association for Computing Machinery, New York, NY, USA, Article 6, 12 pages. https://doi.org/10.1145/3309486.3340245

M. F. Hutchinson. 1989. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation* 18, 3 (1989), 1059–1076.

D. L. James, J. Barbič, and D. K. Pai. 2006. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 987–995.

D. L. James and D. K. Pai. 2002. DyRT: Dynamic Response Textures for Real Time Deformation Simulation with Graphics Hardware. *ACM Trans. Graph.* 21, 3 (jul 2002), 582–585. https://doi.org/10.1145/566654.566621

T. Kim. 2020. A Finite Element Formulation of Baraff-Witkin Cloth. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 171–179.

D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. 2021. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences* 118, 21 (2021).

C.-H. Lee and J.-S. Chen. 2013. Proper orthogonal decomposition-based model order reduction via radial basis functions for molecular dynamics systems. *International journal for numerical methods in engineering* 96, 10 (2013), 599–627.

K. Lee and K. T. Carlberg. 2020. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.* 404 (2020), 108973.

Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. 2018. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566* (2018).

Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake. 2019. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 1205–1211.

H. Lin, F. M. Chitalu, and T. Komura. 2022. Isotropic ARAP energy using Cauchy-Green invariants. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14.

D. C. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *MATHEMATICAL PROGRAMMING* 45 (1989), 503–528.

A. K. Noor and J. M. Peters. 1980. Reduced basis technique for nonlinear analysis of structures. *Aiaa journal* 18, 4 (1980), 455–462.

T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. 2020. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409* (2020).

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

C. Romero, D. Casas, J. Pérez, and M. Otaduy. 2021. Learning contact corrections for handle-based subspace dynamics. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–12.

C. Romero, M. A. Otaduy, D. Casas, and J. Perez. 2020. Modeling and estimation of nonlinear skin mechanics for animated avatars. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 77–88.

I. Santesteban, M. A. Otaduy, and D. Casas. 2022. SNUG: Self-Supervised Neural Dynamic Garments. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

A. A. Shabana. 1991. *Theory of vibration*. Vol. 2. Springer.

S. Shen, Y. Yang, T. Shao, H. Wang, C. Jiang, L. Lan, and K. Zhou. 2021. High-Order Differentiable Autoencoder for Nonlinear Model Reduction. *ACM Trans. Graph.* 40, 4, Article 68 (jul 2021), 15 pages. https://doi.org/10.1145/3450626.3459754

B. Smith, F. D. Goes, and T. Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (mar 2018), 15 pages. https://doi.org/10.1145/3180491

B. Smith, F. D. Goes, and T. Kim. 2019. Analytic eigensystems for isotropic distortion energies. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 1–15.

S. G. Srinivasan, Q. Wang, J. Rojas, G. Klár, L. Kavan, and E. Sifakis. 2021. Learning active quasistatic physics-based models from data. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. 2017. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*. PMLR, 3424–3433.

C. Von-Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt. 2015. Real-Time Nonlinear Shape Interpolation. *ACM Trans. Graph.* 34, 3, Article 34 (may 2015), 10 pages. https://doi.org/10.1145/2729972

Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38, 5 (2019), 1–12.

Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. 2022. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum* (2022). https://doi.org/10.1111/cgf.14505

Y. Yang, D. Li, W. Xu, Y. Tian, and C. Zheng. 2015. Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph* 34, 6 (2015).

M. Zhang, T. Y. Wang, D. Ceylan, and N. J. Mitra. 2021. Dynamic Neural Garments. *ACM Trans. Graph.* 40, 6, Article 235 (dec 2021), 15 pages. https://doi.org/10.1145/3478513.3480497

M. Zheng, Y. Zhou, D. Ceylan, and J. Barbic. 2021. A deep emulator for secondary motion of 3d characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5932–5940.

Supplemental Material

A DERIVATION OF PROPOSITION 3.1

In the limit $\lambda \to \infty$, our metric-preserving constraint in Equation 4 is imposed precisely, forcing the subspace function to be linear and subject to mass orthonormality constraints (see Proposition B.1 in Appendix B):

$$f_\theta(z) = Az + b \quad \text{where} \quad A^T M A = \sigma^2 I_{d\times d}. \tag{10}$$

where $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$, and $\theta = (A, b)$.

If we take the limit of the scaling factor $\sigma \to 0^+$, the entries of the matrix A become small due to the mass weighted orthonormality constraints in Equation 10. Hence, for sufficiently small σ, we can expect that the outputs of our subspace function f in Equation 10 will not stay far from the value of the inhomogeneous term b.

To understand behavior of our model in this perturbative regime, we can approximate our original formulation in Equation 4 using a truncated second order Taylor expansion of the potential energy $E_{\text{pot}}(\cdot)$ centered at b:

$$\min_{A,b} \mathbb{E}_{z\sim\mathcal{N}}\left[E_{\text{pot}}(b) + g(b)^T Az + \frac{1}{2} z^T A^T H(b) Az\right] \quad \text{s.t.} \quad A^T M A = \sigma^2 I. \tag{11}$$

where $g(x)$ and $H(x)$ are the gradient and Hessian of the potential energy E_{pot}, respectively.

Since $\mathcal{N}$ has mean zero, the linear term vanishes from the problem above. Moreover, recognizing it as the Girard-Hutchinson trace estimator [Girard 1987; Hutchinson 1989], we can manipulate the quadratic term as follows:

$$z^T A^T H(b) Az = tr(z^T A^T H(b) Az) = tr(A^T H(b) Azz^T),$$

since a scalar is its own trace and by the property $tr(AB) = tr(BA)$. Since $\mathcal{N}$ has the identity matrix as its covariance, we arrive at the following simplification of Equation 11:

$$\min_{A,b}\left[E_{\text{pot}}(b) + \frac{1}{2} tr\left(A^T H(b) A\right)\right] \quad \text{s.t.} \quad A^T M A = \sigma^2 I. \tag{12}$$

As $\sigma \to 0$, the second term of the objective in Equation 12 becomes negligible, and the potential energy term dominates. Fixing b and optimizing for A yields a generalized eigenvalue problem.

Hence, we have motivated that as $\sigma \to 0$ and $\lambda \to \infty$, we recover Equation 5. This formulation is exactly the classical linear method for modal analysis, discussed e.g. in Shabana [1991].

B AFFINE PROPERTY

Proposition B.1. *Suppose $f(z) : \Omega \to \mathbb{R}^n$ is C^1 on an open, connected domain $\Omega \subseteq \mathbb{R}^d$. Then, $f(z)$ satisfies Equation 2 with equality for all $z, z' \in \Omega$ if and only if $f(z) = Az + b$ for some $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$ with $A^T M A = \sigma^2 I_{d\times d}$.*

Proof. We start with the Lipschitz constant expression:

$$|f(z_a) - f(z_b)|_M = \sigma |z_a - z_b| \quad \forall z_a, z_b \in \Omega,$$

where σ is the positive Lipschitz constant. Squaring this expression implies

$$|f(z_a) - f(z_b)|_M^2 = \sigma^2 |z_a - z_b|^2 \quad \forall z_a, z_b \in \Omega.$$

Since $f \in C^1(\Omega)$, taking the derivative w.r.t. z_a implies

$$J(z_a)^T M \left(f(z_a) - f(z_b)\right) = \sigma^2 (z_a - z_b) \qquad J(z) \coloneqq \left(\frac{\partial f}{\partial z}\right).$$

Taking the derivative of this expression w.r.t. z_b implies

$$J(z_a)^T M J(z_b) = \sigma^2 I_{d\times d} \quad \forall z_a, z_b \in \Omega. \tag{13}$$

Since the expression holds in the $z_a = z_b$ case, applying the expression above multiple times shows

$$\left(J(z_a) - J(z_b)\right)^T M \left(J(z_a) - J(z_b)\right) = 0.$$

Since $M \succeq 0$, we thus have

$$J(z_a) = J(z_b) = \text{const.} \quad \forall z_a, z_b \in \Omega.$$

Since f has a constant Jacobian in Ω, it is automatically affine:

$$f(z) = Az + b \quad \forall z \in \Omega.$$

Moreover, $A^T M A = \sigma^2 I_{d\times d}$ thanks to Equation 13.

To prove the reverse direction, note that the relationship

$$|A(z_a - z_b)|_M \quad s.t. \quad A^T M A = \sigma^2 I_{d\times d}$$

implies

$$\sigma |z_a - z_b|.$$

□

C ADDITIONAL DETAILS

This sections gathers additional implementation and experimental details.

C.1 Selecting Hyperparameters

Like most learning-based approaches, our method requires a choice of hyperparameters to weight the objective function, in our case the penalty strength λ and metric scaling parameter σ. To be clear, adjusting two parameters is a modest burden as neural networks go, and our networks train very quickly (Section 4.2); we discuss hyperparameter selection in-depth here to facilitate the application of our method to new physical systems. Importantly, although hyperparameters may need new settings for new classes of systems (*e.g.* cloth *vs.* kinematic mechanisms), they can be reused to fit many instances of a particular system (*e.g.* many different cloth systems). Table 1 gives hyperparameters for all examples in this work.

The hyperparameter σ should be chosen based on the desired behavior of the subspace. Large values yield a subspace which spans extreme states, while smaller values concentrate the subspace tightly around low-energy configurations. The diversity of the subspace also affects the ideal neural network size. Subspaces with large σ which span a larger kinematic range may also require a larger network to accurately resolve the subspace, whereas smaller networks may be sufficient for a subspace with small σ that only represents a narrow range of motions.

Note also that σ depends on the physical units in which the configuration is measured. We recommend initially choosing a large value for σ and visualizing randomly-sampled system configurations during training, recalling that Section 4.1 linearly grows the subspace diversity as training proceeds. For example, if the subspace spans a suitable range of configurations 1/3 of the way through

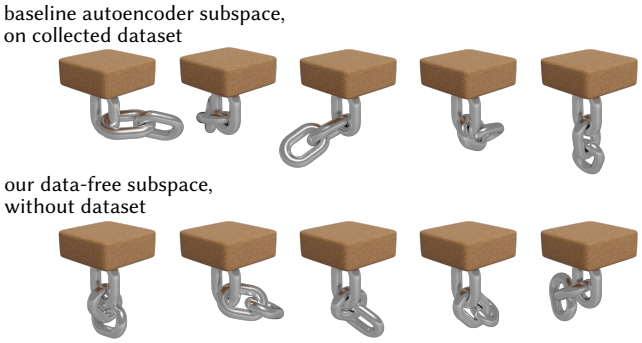

Fig. 11. A comparison of samples from the latent space of an autoencoder trained on a manually-collected dataset (*top row*), and our data-free approach on the same system (*bottom row*). Both use the same latent dimension ($d = 4$).

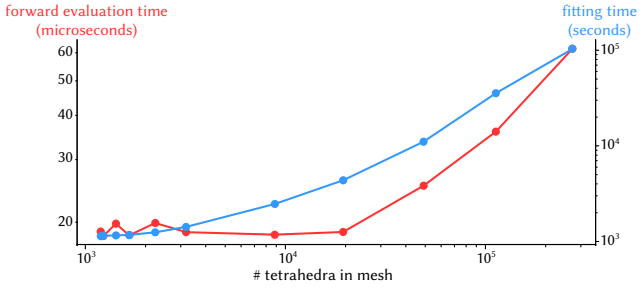

Fig. 12. We evaluate the performance scaling of our method, fitting elastic deformation subspaces to the same object tetrahedralized at various resolutions. Each data point is a fitted subspace at a different mesh resolution; the left red axis gives the time for a single forward evaluation of the subspace map, while the right blue axis gives the fitting time.

training, then $\sigma \leftarrow 1/3\sigma$ is a reasonable choice of parameter, and training can be repeated with this value.

The hyperparameter λ weights the approximately-isometric objective; it should be chosen ensure Equation 3 has an effect, but also does not dominate the objective and enforce a restrictive affine subspace (see Section B). This is easily assessed by measuring the unitless ratio in Equation 2 during training, if is far from 1 then λ should be increased, and if it very close to 1 (*e.g.* within 10^{-3}) then λ should be reduced.

C.2 Data-Free vs. Supervised Methods

The primary advantage of our formulation is that it fits a subspace using only potential energy function for a system, and does not require a training dataset. Nonetheless, it is useful to consider how the quality of the subspaces compares with a baseline supervised method if a dataset were available. To that end, we gather a dataset by interactively simulating the chain from Figure 6, here with fewer links to facilitate real-time robust full simulation. The resulting dataset contains 40k sampled states of the system. Our method is used to fit a subspace with a 5×128 MLP from a $d = 4$ latent space, which does not require the dataset. As a simple baseline model, we train an autoencoder, where the decoder is an MLP identical to our subspace model, and the encoder is a matching 5×128 hidden layer MLP. The autoencoder is fit via reconstruction loss, along with a weak regularizer to encourage a 0-centered latent space. Figure 11 shows samples from the resulting spaces.

C.3 Performance Scaling

To measure the performance scaling of our method, we fit a series of subspaces to an elastic deformation system where the shape from Figure 7 is discretized at various mesh resolutions ranging from $\approx$ 1k to $\approx$ 150k degrees of freedom. Figure 12 gives the corresponding time cost, measured on the same setup as in Section 4.2. Our method scales well to larger mesh sizes, especially for forward evaluation.

We naively use the exact same training scheme from Section 4.2 here and throughout this work. Likewise, all problem scales use the same 5×128 MLP model, increasing only the output dimension of the last layer to match the degrees of freedom for the system. In practice one might adjust model sizes and training schedules for problems with vastly ranging orders of magnitude to tune performance. Furthermore, approaches such as adaptive cubature [An et al. 2008] are well-suited to accelerate potential energy evaluation for high-resolution deformable models, which could greatly accelerate the fitting procedure.

C.4 Experiment Details

Sampling for [Fulton et al. 2019]. Section 5.2 shows a preliminary application where our data-free subspace is used to sample data for an existing downstream supervised approach, sidestepping the need for dataset collection. In particular, we automatically generate training data for the AutoDef method [Fulton et al. 2019], a recent approach which offers fast deformable simulations but requires significant effort to collect training data. To do so, we first fit our subspace as usual to a single elephant mesh from the AutoDef experiment set, using the training parameters listed in Table 1. Then, we randomly sample 1000 simulation states by taking random sinusoidal motions in latent space, and applying our fitted subspace map $q \leftarrow f_\theta(z)$ to get the corresponding system configurations. The states are encoded as displacements from the rest pose as expected by the AutoDef formulation. These displacements are then used in-place of a manually collected training dataset to fit AutoDef as described in Fulton et al. [2019]. The original AutoDef work proposes a nontrivial pipeline of user interaction to generate training data; we find that in this initial experiment substituting our automatically-sampled unsupervised population yields comparable results without the need to manually collect data.