Data-Free Learning of Reduced-Order Kinematics

ANONYMOUS AUTHOR(S)

SUBMISSION ID: 478

![Image of neural network models](image_url)

**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 to begin 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 a 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 challenging systems like colliding rigid bodies and linkages, which resist conventional local analysis. 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, interpolating between configurations, 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]). 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 several different interacting systems.

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 parameters 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.

Fig. 2. Our subspaces are surprisingly effective on systems far beyond the usual deformable bodies and cloth. Here, linkages are naïvely described by the location of each segment, under a potential with strong penalty barriers 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.

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]. Barbi and James [2005] augment the modal basis with second-order “modal derivatives,” while still resulting in a linear deformation subspace, and Yang et al. [2015] explore 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 8. Snapshot methods generalize beyond a rest space by collecting large databases of simulation outputs and fitting a reduced space from 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 deformations of a physical system. The formulation is general, applying to a broad set of systems and capturing both linear and nonlinear subspaces. It follows in the footsteps of widespread success in machine learning using neural networks to encode significant low-dimensional submanifolds of high-dimensional spaces (e.g., [Lee and Carlberg 2020]).

3.1 Neural Subspace Maps

Consider a map \( f_\theta \), 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 small set of deformation modes that move multiple vertices in tandem. The vector \( \theta \in \mathbb{R}^q \) contains learnable parameters specific to the physical system, e.g. neural network weights.

Classical simulation algorithms operate on \( \mathbb{R}^n \), where 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 at each step. But, \( \mathbb{R}^n \) includes unlikely configurations given the potential energy \( E_{\text{pot}} \), corresponding to high-energy deformations. Hence, in many settings, we can reasonably expect the kinematics to stay in the image \( f_\theta(\mathbb{R}^d) \) of some map \( f_\theta \) 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 \( f_\theta \) is that \( f_\theta \) can encode nonlinear subspaces, rather than only linear modes. In this work, we model \( f_\theta \) as a neural network, with weights \( \theta \) (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.

3.2 Objective Function

Beyond network architectures determining the role of the parameters \( \theta \), the key to identifying an effective parameterization \( f_\theta \) involves the choice of an objective function. In particular, given a set of candidate parameters \( \theta \), we need a means of determining whether the image of \( f_\theta \) is useful for simulation. We can then identify an effective \( \theta \) by stochastic gradient descent.

Our key observation is that we can optimize for \( \theta \) 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:

\[
\mathbb{E}_{Z \sim \mathcal{N}} \left[ E_{\text{pot}}(f_\theta(z)) \right].
\]

Here, \( \mathcal{N} \) denotes the Gaussian distribution over \( \mathbb{R}^d \) with mean 0 and variance \( \lambda_d \cdot I \). \( \mathbb{E} \) denotes the expectation with respect to a random variable. Note we have not (yet) put a scale on our latent variable \( z \). Note we have not (yet) put a scale on our latent variable \( z \).

Equivalently, we write:

\[
\mathbb{E}_{Z \sim \mathcal{N}} \left[ \frac{1}{2} \| f_\theta(z) - f_\theta(z') \|_M \right],
\]

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 \( M \in \mathbb{R}^{n \times n} \):

\[
|x|_M^2 = x^T M x.
\]

Equivalently, we write:

\[
\frac{1}{2} \mathbb{E}_{Z \sim \mathcal{N}} \left[ \frac{1}{2} \| f_\theta(z) - f_\theta(z') \|_M \right].
\]

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

\[
\mathbb{E}_{Z \sim \mathcal{N}} \left[ \frac{1}{2} \left( \frac{1}{2} \| f_\theta(z) - f_\theta(z') \|_M \right)^2 \right].
\]

Intuitively, this expression prefers maps \( f_\theta(\cdot) \) whose Lipschitz constant is roughly \( \frac{1}{2} \) everywhere. 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 \( \theta \) as follows:

![Diagram](image.png)

Fig. 3. We fit an 8-dimensional reduced space for a neo-hookean elastic bar 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.
When we take our parameters wide variety of scaling and units used in physical energies—Table 1

Fig. 4. States from an simulation via time integration in the latent space. In principle any

3.4 Subspace Simulation

In words, as we push to preserve geometry of the con

gurations, while large

ting parameter

forces the map’s image to include higher-energy states. The weight-

gurations concentrated near

ted region about the origin, with low-energy con

The hyperparameter \( \sigma \) can be tuned to adjust the size of the

Small values of \( \sigma \) produce subspaces that are tightly con-

hould be chosen to ensure that Equation 2 roughly

It is inevitable that these hyperparameters arise due to the

large \( \sigma \) can

force the map’s image to include higher-energy states. The weight-

In this formulation, our latent subspaces map from a roughly unit-

The hyperparameter \( \sigma \) can be tuned to adjust the size of the

The hyperparameter \( \sigma \) can be tuned to adjust the size of the

3.3 Reduction to Modal Analysis

When we take our parameters \( \lambda \) and \( \sigma \) 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:

PROPOSITION 3.1. Suppose \( f_0 \) has the capacity to represent affine

functions. Then, as \( \sigma \to 0 \) and \( \lambda \to \infty \), the solution to Equation 4

satisfies

\[
\begin{align*}
 \mathbf{f}(z) &= \mathbf{A}z + b \\
 b &= \arg \min_b E_{\text{pot}}(b) \\
 \mathbf{A} &= \sigma \cdot \text{TOP-d-GENERALIZED-EIGENVECTORS}(

\mathbf{M}, \mathbf{H}(b)).
\end{align*}
\]

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 \( f_0 \) 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:

\[
\hat{z} = \arg \min_z \frac{1}{2h^2} \left[ | \mathbf{f}_0(z) - \hat{q} |_M^2 + E_{\text{pot}}(\mathbf{f}_0(z)) \right],
\]

where \( h \) is the timestep and \( \hat{q} \) is an inertial guess computed from

previous configurations. The optimization is performed in the neural

space \( z \) via substitution into the optimization formulation [Fulton


The performance characteristics of this integration are very differ-

ent 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 inte-

gration is significantly faster than simulation in the full space, but
does not outperform existing specialized subspace integrators.

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 \( f_0 \) to incorporate conditional

parameters as additional inputs to the neural subspace map as

\[
\mathbf{f}_0 : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^n \quad q \leftarrow \mathbf{f}_0([z, c])
\]

where the conditional parameters are a vector \( c \in \mathbb{R}^m \), and \([z, c]\)
denotes vector concatenation. During training, we additionally sam-

dle 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 \( f_\theta \). In this work we focus on the problem formulation itself, and consider only simple multi-layer perceptrons (MLPs).

4.1 Seeded Subspace Exploration

However, 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. Untangling such a configuration in the classical physical setting is already difficult [Smith et al. 2018], and 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 \( q_{\text{seed}} \in \mathbb{R}^d \) provided as input to the method. Precisely, during training only we parameterize the neural subspace map as

\[
    f_\theta(z) := \rho \text{MLP}_\theta(z) + (1 - \rho)q_{\text{seed}}
\]

where \( \rho \) is a scheduling parameter which linearly increases from 0 \( \rightarrow \) 1 as training proceeds. Crucially, at the conclusion of training this seed state is entirely absent, and resulting network is an ordinary MLP. Likewise, we also modulate the scale parameters \( \sigma \) in Equation 3, multiplying by a factor of \( \rho \) because at initialization \( f_\theta \) 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 \( q_{\text{seed}} \) 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, and we do not require it to be a special rest state or minimal-energy configuration. For instance, in systems like Figure 3 which have no single distinguished configuration, any choice will yield similar subspaces.

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 \( \rightarrow \) 256 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. We also include an implementation in the supplement, to be made public after acceptance.

We use the Adam optimizer [Kingma and Ba 2014] for training, and a learning rate of \( 10^{-4} \) for \( 10^5 \) 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 simulating runtime performance varies drastically based on the cost of evaluating and differentiating the energy function for simulating, generally 10s of milliseconds for the systems shown.

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 (SVK) stretching term on faces. Figure 4 shows a basic example, where we compute a subspace for a pinned hanging sheet. For now, we do not model self-collisions in cloth, although we do consider basic cloth-object interactions (Figure 1).

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.
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:

\[ w_{eq}[C_{eq}(q)]^2 + w_{ineq} \min(C_{ineq}(q), 0)]^2 \]

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

Rigid Bodies. As a physical parameterization outside the usual realm of typical local subspaces, we also consider rigid body dynamics parameterized. Each body’s state is described by \( \mathbb{R}^{12} \) unconstrained coefficients per-body, interpreted as the entries of a \( 3 \times 4 \) transformation matrix \( [R|t] \). An additional potential term \( \| (R^T R - I) \|_F^2 \) 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. There are many challenges with simulating this setup in the traditional setting, even small-timestep classical dynamics with an implicit integrator easily gets stuck on the landscape. However, 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 are also able to generate subspace models for complex linkages. As before, these linkages are naively modeled as free-floating rigid objects, with strong penalty barriers holding each joint together; the system definition does not include any angular or relative parameterizations, etc. Learning a

### Table 1. Parameters and dimensions for the experiments appearing in this work.

<table>
<thead>
<tr>
<th>Name</th>
<th>Figure</th>
<th>Energy</th>
<th>Full dim</th>
<th>Reduced dim</th>
<th>Condition dim</th>
<th>MLP size</th>
<th>λ</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloth ball</td>
<td>Figure 1</td>
<td>cloth + penalty</td>
<td>6009</td>
<td>3</td>
<td>0</td>
<td>5x128</td>
<td>1.0</td>
<td>0.05</td>
</tr>
<tr>
<td>Klann linkage</td>
<td>Figure 2</td>
<td>rigid + penalty</td>
<td>84</td>
<td>1</td>
<td>0</td>
<td>5x64</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Stewart platform</td>
<td>Figure 2</td>
<td>rigid + penalty</td>
<td>168</td>
<td>6</td>
<td>0</td>
<td>5x128</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Bistable bar</td>
<td>Figures 3, 10</td>
<td>neohookean</td>
<td>830</td>
<td>8</td>
<td>0</td>
<td>5x128</td>
<td>0.5</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>Hanging sheet</td>
<td>Figure 4</td>
<td>cloth</td>
<td>6072</td>
<td>8</td>
<td>0</td>
<td>5x128</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Conditional bar</td>
<td>Figure 5</td>
<td>neohookean</td>
<td>942</td>
<td>8</td>
<td>2</td>
<td>5x128</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Chain</td>
<td>Figures 6, 8</td>
<td>rigid + collision</td>
<td>288</td>
<td>8</td>
<td>0</td>
<td>5x128</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Hanging cow</td>
<td>Figure 7</td>
<td>neohookean</td>
<td>14676</td>
<td>3</td>
<td>0</td>
<td>5x128</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>Cantilever</td>
<td>Figure 8</td>
<td>neohookean</td>
<td>1500</td>
<td>2</td>
<td>0</td>
<td>5x128</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Elephant</td>
<td>Figure 9</td>
<td>neohookean</td>
<td>1926</td>
<td>8</td>
<td>0</td>
<td>5x64</td>
<td>1.0</td>
<td>10^{-4}</td>
</tr>
</tbody>
</table>

![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, whereas our subspace easily fits it.](image-url)
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; this data collection often accounts for much of the effort in the process, 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 a supervised method trained on a sufficiently large and high-quality dataset; if a dataset is available it should certainly be used.

To demonstrate the complementary value of our low-dimensional data-free scheme, we present two preliminary applications which leverage our model’s particular structure. Because our subspaces densely map onto 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. Our model is agnostic to the details of the physical system and—as demonstrated in our experiments—yields high-quality results across a wide variety of settings.

**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 $d = 8$.**

**Fig. 9. Our approach can serve as an automatic sampler for 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.**

**Limitations.** Like most nonconvex models, our subspace training procedure may reach local minima; it inherits both the difficulties of optimizing deep neural networks and of numerically integrating stiff physical systems. We found the training procedure in Section 4 to be highly effective in practice, but future work could develop numerical methods tailored to this hybrid problem.

Local minima may also result from isolated, locally stable configurations in the physical system. For example, the inset figure shows a compressed elastic bar pinned at both ends. By chance, the training procedure identifies a space with a twist and could not escape it, resulting in a physically plausible—but unintentionally twisted—subspace. Note un-twisting the bar would require a 360° turn of one of the ends. We find that the training procedure in Section 4.1 generally avoids such artifacts, but it cannot guarantee to eliminate them.

More fundamentally, traditional methods like linear modal expansion are accompanied by theoretical analysis to understand which effects are truncated. 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 neural 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. We could also expand our learned models to target increased semantic control of subspaces, or to learn models that generalize over many physical systems at once.

**REFERENCES**


