EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)

*Volume 38 (2019), Number 2*

# Latent-space Dynamics for Reduced Deformable Simulation

Lawson Fulton, Vismay Modi, David Duvenaud, David I. W. Levin and Alec Jacobson

University of Toronto, Canada

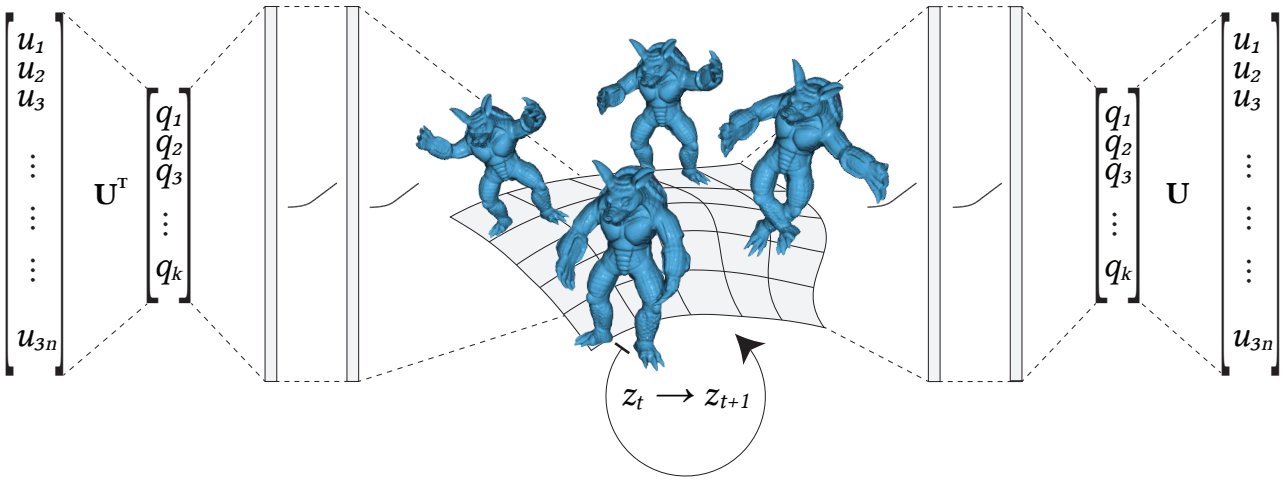


**Figure 1:** *We train an autoencoder on example deformations and perform fast simulation in the resulting low-dimensional nonlinear latent-space.*

**Abstract**

*We propose the first reduced model simulation framework for deformable solid dynamics using autoencoder neural networks. We provide a data-driven approach to generating nonlinear reduced spaces for deformation dynamics. In contrast to previous methods using machine learning which accelerate simulation by approximating the time-stepping function, we solve the true equations of motion in the latent-space using a variational formulation of implicit integration. Our approach produces drastically smaller reduced spaces than conventional linear model reduction, improving performance and robustness. Furthermore, our method works well with existing force-approximation cubature methods.*

**CCS Concepts**

•***Computing methodologies*** → ***Physical simulation;*** *Dimensionality reduction and manifold learning;*

## 1. Introduction

Computer graphics has long exploited the fact that the low spatial frequency deformations of discrete, three dimensional objects can be represented in a low dimensional space. This observation has been utilized by practitioners of physics-based animation to construct high-performance algorithms for the simulation of elastic materials. With few exceptions to date, reduced space methods have constructed *linear subspaces* for the low-dimensional description, that is displacements $\mathbf{u}$ are represented by $\mathbf{u} = \mathbf{U}\mathbf{q}$ for a matrix $\mathbf{U} \in \mathbb{R}^{n \times k}$. However, if the deformation being modeled is highly nonlinear, the number of basis vectors required to define the space may grow rapidly even if there is an underlying nonlinear parametrization with fewer degrees-of-freedom (DOFs), as exemplified in Figure 2. Unfortunately, save for a few cases (such as the humble rigid body), deriving well-posed non-linear reduced spaces is algorithmically challenging.

The advent of deep neural networks provides an apparent solution to this problem and for the past several years there has been an increasing interest in applying machine-learning approaches to applications in computer animation. Recent work has applied deep-

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

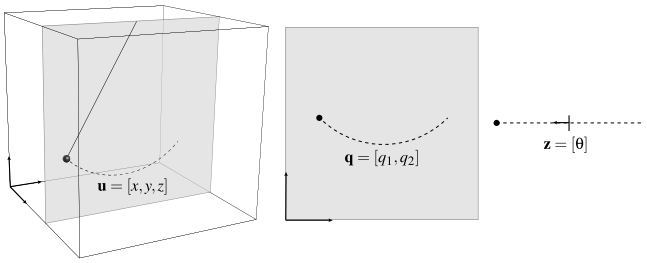


**Figure 2:** *In this simple example, a pendulum which swings on a single axis in 3-dimensional space can be reduced to 2 degrees of freedom with a linear subspace. However, the system can be further reduced to a single dimension by using a non-linear mapping.*

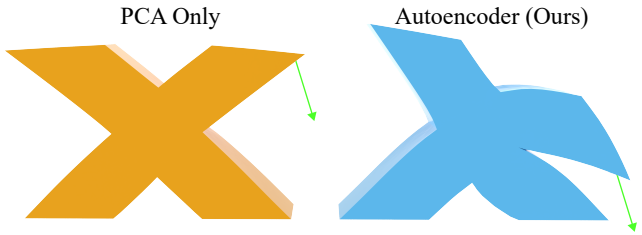


**Figure 3:** *Compared to linear PCA, the nonlinear autoencoder allows larger deformations to be captured using the same number of degrees of freedom.*

learning to accelerate the pressure projection in grid-based fluid simulation, add detail to smoke simulation and improve the performance of skinning transforms of complex, rigged characters. These methods have, so far, failed to crossover to the general simulation of elastic materials.

This lack of cross-pollination can be partly blamed on the nature of elastic simulation itself. Unlike inviscid fluid simulation, there is no pressure projection-like operator to approximate. It's closest analog, the linear system solve, present in implicit time integration schemes, is itself parameterized by the degrees-of-freedom of the physical system. This high-dimensionality makes it difficult to approximate. Worse still, errors in this approximation can have a pathological effect on the behavior of the non-linear solvers used to advance the simulation through time. Unlike smoke simulation, elastic simulation benefits less from the addition of high frequency details and unlike a skinned animation, one cannot assume the presence of an artist produced set of skinning handles.

The algorithm described in this paper tackles these problems by following in the footsteps of previous linear model-reduction approaches to elastic body simulation. Rather than attempting to replace the entire simulation pipeline with a learned analog, we instead focus on performing time integration of the elastodynamic system in a learned nonlinear latent-space. We represent this space using an artificial neural network of the autoencoder class, which is trained on simulation snapshots.

Our latent-space simulator is enabled by a variational formulation of the equations-of-motion, acting directly in our non-linear space. We solve these equations efficiently and stably using a quasi-newton optimization scheme. Furthermore, we show that other optimizations such as optimized cubature [AKJ08, vTSSH13, YLX*15, PBH15] are compatible with our method. Finally, we demonstrate that our approach produces deformations which are of equivalent or greater visual fidelity to those produced by other reduced space approaches while potentially improving performance and robustness.

## 2. Related Work

**Deformable Solid Simulation.** Simulation of elastic deformable materials has a long history in computer graphics, first introduced by the seminal work of Terzopoulos [TPBF87]. Since then, the finite element method (FEM) has been one of the favored tools for elastic simulation which solves the governing dynamic equations on a discrete volumetric mesh. Although there are other techniques for deformable simulation such as particle based methods [DG96, MC11], they are outside the scope of our present work. The reader is referred to the survey by Nealen [AMR*] for a comprehensive study of deformable simulation in computer graphics.

FEM is particularly well suited to creating accurate and realistic results. Unfortunately it is notoriously slow and often unsuitable for real-time applications, especially for highly detailed models. Fortunately, there has been significant work done to ameliorate this shortcoming. Broadly, one can divide such algorithms into two categories. The first category, Full Space Methods (i.e, [MZS*11, BML*14, LSW*18]), which gain performance by optimizing the solution of the full system of equations by various approximations and heuristics. The second approach, Dimensionality Reduction or Reduced Space Methods, gain performance by reducing the degrees-of-freedom (DOFs) of the physical system [PW89, OKHS03]. Our method falls squarely under the umbrella of dimensionality reduction, and we focus this discussion on works related to the latter category.

**Linear Dimensionality Reduction.** The most popular and prolific Reduced Space Methods are based on linear subspaces [PW89] where the basic approach is to build an orthonormal basis of small dimension which captures the relevant deformations and then solve the equations of motion in these reduced coordinates. In general, any subspace may be used and can be constructed by various means depending on what is known about the desired deformation space in advance. Linear modal analysis [Sha12, PW89, OKHS03] provides a basis for a general solution space by discarding high frequency vibration modes and can be augmented by additional basis vectors to accommodate larger deformations [BJ05, vTSSH13, YLX*15]. If one knows in advance the deformations that will be needed, as in our case, principal component analysis of simulation snapshots [KLM01, BJ05] have been used to construct the subspace.

Linear subspaces have found wide application outside of deformable solid simulation, including fluids [TLP06, SSW*13], shape deformation [vTSSH13, BvTH16], computational design [XLCB15, MHR*16, UMK17] and sound simulation [OSG02, JBP06]. However, very small linear subspaces can have difficulty representing even moderately non-linear deformations as seen in Figure 3 [KJ09, CLMK17]. The work on Hyper-Reduced Pro-

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

jective Dynamics [BEH18] attempts to overcome this limitation by combining model reduction with the very efficient full-space approach of Projective Dynamics [BML*14]. This allows the real-time use of large subspaces to capture significant deformations. However, it also inherits the limited range of possible constitutive materials from Projective Dynamics.

**Nonlinear Dimensionality Reduction.** This has prompted a number of works which attempt to reduce the simulation dimensionality through nonlinear means. Perhaps the most straight-forward approach is to coarsen the simulation mesh [KMOD09, NKJF09, CLSM15, CLMK17]. This allows the coarse mesh to serve as a non-linear reduced space and can yield excellent deformation fidelity. While coarsening does reduce the DOFs of the physical system, it is necessary to adjust the material parameters to overcome stiffening, and it has yet to be shown to obtain the total reduction that linear projection can achieve. Alternative numerical coarsening schemes exist which improve baseline agreement but come at the cost of greatly increased memory consumption [CBW*18].

An alternative approach to coarsening is to use animation rigs as a reduced space [HMT*12, WJBK15]. These methods do a good job of simulating motions that belong to the rig function space (colloquially called the rig-space). However, the physical response of an object is limited to the deformations created by the rig itself, not the likely behavior of the object in the world. More general frame-based simulation methods have also been proposed [GBFP11, FGBP11]. These use Lloyd relaxation-type methods to automatically position frames based on material stiffness. These frames then parameterize deformation via skinning transforms. While these methods remove the artist dependence of rig-space methods, they often still do not reduce the deformation space as much as modal methods since each frame can contain up to 12 scalar DOFs. The aforementioned techniques have even been combined [MGL*15], supplying a fix for being trapped in rig space using an overlapping, hierarchical description of dynamics. This comes at the cost of an additional kinetic filtering operation that glues the layers of the hierarchy together.

Sub-structuring is another promising attempt to capture more non-linear deformations. Sub-structuring methods represent a deformable body using a piecewise linear modal discretization [BZ11]. However this approach still requires the manual identification of individual components for best results and topology restrictions to prevent locking. Other algorithms attempt to gracefully revert back to full simulations when the modal deformation space becomes inaccurate [KJ09, TOK14]. However, reverting back to the full space comes with a heavy performance penalty. Time varying linear subspaces are yet another approach [HTC*14, XB16], however in this case the dynamics are still computed in a linear subspace corresponding to the current pose of an underlying non-dynamic pose space, making them specifically suitable to the secondary motions relevant to character animation.

Perhaps most similar in spirit to our approach is subspace simulation based on rotation-strain coordinates [HTZ*11, LHdG*14, PBH15]. In these works, the configuration of the simulation mesh is described by the rotation and strain of each constitutive element. In the rotation-strain model, reduced space simulation is performed using a purely linear modal approximation and the final mesh configuration is determined by finding the rotations and strains that best approximate the linear modal result by a least-squares projection. The major problem with such approaches (aside from the error induced by the projection step) is that the space spanned by per-element rotations and strains may not be embeddable in 3D as a continuous mesh. As a result, it is often necessary to tune the material parameters for the reduced simulation to match the full space result. As of yet there has been no proposed algorithm to automate this process. Furthermore, the nonlinear nature of this map is on a per-element basis, and therefore is not necessarily able to capture the nonlinearity present in the global deformation space.

A further class of recent work is dedicated to the reduction and manipulation of statically-deformable 3D mesh models using neural networks. A primary challenge to overcome is effectively training on the high-dimensional data of 3D meshes. By introducing rotation-invariant features [TGLX18], removing pose-dependent information [CO18] or using graph convolutions [TGL*18, LBBM18], effective training of high quality deformation spaces is possible. However, each of these operations complicates the formulation of deformation *dynamics*. Our approach most closely follows that of Fast and Deep Deformation Approximation [BODO18]. They learn the nonlinear relationship between rig parameters and vertex positions for characters. We do not require a rig, but we similarly reduce the input and output to our network by applying PCA to the training poses in order to simplify the learning task and reduce network complexity. This approach leads to a simple description of dynamics in the latent space.

**Machine Learning for Simulation.** In this paper we turn to machine learning in an attempt to build a fast reduced model simulation algorithm for high resolution elastic bodies. Machine learning algorithms for physics simulation are relatively new, but there have been recent efforts in fluid simulation to learn pressure projection operations [TSSP16], velocity updates [LJS*15], one-way coupled turbulence models [CT17] and interpolate between precomputed simulations [BPT17]. Until the recent appearance of DeepWarp [LSW*18] and [WKD*18], there had been no such work for deformable body simulation. We argue that this is because a coherent simulation of a deformable solid is much less tolerant to the approximation errors that machine learning algorithms tend to make. Thus, update learning approaches such as [LJS*15] are not likely to be able to, for instance, return to the reference configuration when forces are removed. DeepWarp is promising concurrent work that attempts to learn a mapping from a linear elasticity simulation to its nonlinear counterpart. However, being a Full Space method, it fails to decouple mesh resolution from simulation complexity.

The concurrent work by [WBT18] on fluid simulation takes a similar approach to ours in the sense that they learn a non-linear reduced space in which to perform time-integration. However, the time-stepping function is still approximated by a neural network and not solved exactly.

Our goal is to learn reduced spaces that require fewer degrees of freedom than other model reduction techniques while achieving equivalent or greater simulation fidelity, speed, and robustness. We target the use-case where examples of the desired deformation space are known in advance. However, we want the data to

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

be our guide and not require any additional input annotation such as rigs, weight painting, material tuning, etc. To do this we rely on non-linear dimensionality reduction via the autoencoder architecture [BGC15] to learn a small non-linear model of an object's configuration space based on provided simulation snapshots or deformation examples. We enhance our autoencoder by initializing the outer layers with a basis computed via PCA [LKA*17, BODO18] and show that training directly in this fixed PCA space is significantly faster with nearly identical results. This approach can quickly (sometimes in seconds) learn a small non-linear reduced space that can offer better performance than linear reduced models in cases where the desired deformations are large and nonlinear.

Further, we show that fast dynamics simulation can be performed using Quasi-Newton methods [LBK17] applied to an energy potential form of the elastodynamics equations [HLW06, MTGG11, HMT*12, PBH15]. Finally, we efficiently approximate strain-energy and reduced forces for arbitrary materials using optimized cubature [AKJ08] and show how our smaller reduced space allows the use of fewer cubature points while retaining stability.

## 3. Background: Reduced Model Simulation

In this section we introduce our notation and describe the general approach to linear model reduction for discrete deformable solids. For a more comprehensive review, we direct the reader to [SB12].

**Reduced Equations of Motion.** We start by considering a discretized tetrahedral or hexahedral volumetric mesh with $n$ nodes and $m$ elements where the configuration at time $t$ is given by the stacked per-node displacements $\mathbf{u}(t) \in \mathbb{R}^{3n}$ relative to some rest pose $\mathbf{x}_0 \in \mathbb{R}^{3n}$. The dynamics of this mesh are governed by Newton's second law, relating acceleration at the vertices to the internal and external forces.

$$\mathbf{M}\ddot{\mathbf{u}}(t) = \mathbf{f}_{\text{int}}(\mathbf{u}(t)) + \mathbf{f}_{\text{ext}}(t) \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{3n\times 3n}$ is a sparse mass matrix, $\ddot{\mathbf{u}}(t) \in \mathbb{R}^{3n}$ is a vector of the temporal second derivatives of the displacements $\mathbf{u}$, i.e., the accelerations, $\mathbf{f}_{\text{int}}$ which can also be defined as the negative gradient of an elastic potential $\mathbf{f}_{\text{int}}(\mathbf{u}) = -\nabla V(\mathbf{u})$ for $V : \mathbb{R}^{3n} \to \mathbb{R}$. This potential maps displacements to the resulting internal restorative forces of deformation, and $\mathbf{f}_{\text{ext}} \in \mathbb{R}^{3n}$ gives the possibly time-varying external forces (e.g., gravity), contact and other forces. In the following, we write $t$ implicitly for brevity.

Newton's second law yields a system of $3n$ equations that can be discretized in time and solved for the full space simulation, a very costly procedure for large meshes. Linear dimensionality reduction restricts the configuration space to a $k$-dimensional subspace $W$ of $\mathbb{R}^{3n}$ where $k << 3n$. This subspace can be parameterized by coefficients $\mathbf{q} \in \mathbb{R}^k$ by

$$\mathbf{u} = \mathbf{U}\mathbf{q}$$

Where $\mathbf{U} = [b_1, ..., b_k] \in \mathbb{R}^{3n\times k}$ is matrix defined by a set of basis vectors $\{b_1, ..., b_k\}$ which span $W$. Combining this with Equation (1) provides the reduced equations of motion

$$\tilde{\mathbf{M}}\ddot{\boldsymbol{q}}(t) = \tilde{\mathbf{f}}_{\text{int}}(\mathbf{q}(t)) + \mathbf{U}^T\mathbf{f}_{\text{ext}}(t) \tag{2}$$

Where $\tilde{\mathbf{M}} = \mathbf{U}^T\mathbf{M}\mathbf{U}$ is the $k \times k$ reduced mass matrix, and $\tilde{\mathbf{f}}_{\text{int}}(\mathbf{q}) = \mathbf{U}^T\mathbf{f}_{\text{int}}(\mathbf{U}\mathbf{q})$ are the reduced forces.

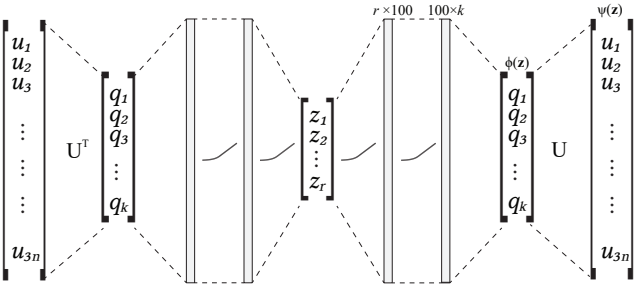


**Figure 4:** *An overview of our autoencoder network architecture.*

**Basis Construction.** As mentioned in the related work, there are two main approaches to construct the particular subspace $W$ and its associated basis $U$, either modal analysis and its extensions for a subspace that requires little prior knowledge, or constructing a basis from known deformation examples. Since the latter case is most relevant to our work, we focus on that here. The aforementioned review contains a detailed exposition of both cases [SB12].

Given a training set of $N$ example deformations as displacements $T = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N]$, we can employ PCA [SB12] to compute a basis $\mathbf{U}$ which spans approximately the same space as $T$, where the size of $\mathbf{U}$ can be chosen ahead of time, or determined based on some chosen tolerance.

**Reduced Force Approximation.** Equation (2) is now a system of $k$ equations that can be solved much more efficiently. However, the most costly component becomes evaluating the reduced internal forces $\tilde{\mathbf{f}}_{\text{int}}(\mathbf{q})$ which naively requires visiting every element in the mesh to compute $\mathbf{f}_{\text{int}}$ before projecting into $W$.

In some special cases $\tilde{\mathbf{f}}_{\text{int}}$ can be computed efficiently and exactly without reference to the full space, such as when using the StVK energy [BJ05]. Unfortunately, there is no general procedure to compute $\tilde{\mathbf{f}}_{\text{int}}$ exactly for an arbitrary material model.

The work on optimized cubature introduced by [AKJ08] provides a method for approximating the reduced forces by determining a small set of elements $C$ that, when given appropriate weights, can be summed to approximate the full reduced force for a particular configuration.

$$\tilde{\mathbf{f}}_{\text{int}}(\mathbf{q}) \approx \sum_{i=1}^{|C|} \mathbf{w}_i \tilde{\mathbf{f}}^i_{\text{int}}(\mathbf{q})$$

where $\mathbf{w} \in \mathbb{R}^{|C|}$ is a precomputed vector of weights and $\tilde{\mathbf{f}}^i_{\text{int}}$ is the reduced force resulting from element $i$ alone. The particular method for selecting the elements $C$ and weights $\mathbf{w}$ has been improved by subsequent authors [vTSSH13, YLX*15, PBH15]. In this paper, we chose to make use of the work by [AKJ08] because of their publicly available implementation. In principle though, any other method for cubature precomputation is compatible with our framework.

## 4. Nonlinear Reduced Model Learning

The linear model reduction approach can drastically reduce the size of the solution space of a simulation, but there is still room for

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

improvement. Consider the situation in Figure 2. Although a linear subspace can reduce the DOFs required, the realized configuration space actually lives on a smaller nonlinear manifold.

### 4.1. Dimensionality Reduction Using Autoencoders

We would like to address the problem of creating a nonlinear parametrization $\psi : \mathbb{R}^r \to \mathbb{R}^{3n}$ suitable for doing reduced model simulation. There are several properties we would like such a mapping to have. Our goal is to create a reduction that captures the nonlinearity specific to the data it is trained on, so we would like to *learn* the mapping from provided data. Second, we would like the mapping to be smooth, to avoid locking of the simulation. And finally, it should be easily differentiable to enable solving the equations of motion efficiently.

The successful use of PCA in linear model reduction hints towards using its nonlinear counterpart from the field of machine learning, the *autoencoder* [HS06]. An autoencoder is constructed by defining two parametric functions known as the *encoder* and the *decoder*, denoted here as $\overline{\psi} : \mathbb{R}^{3n} \to \mathbb{R}^r$ and $\psi : \mathbb{R}^r \to \mathbb{R}^{3n}$ respectively, with parameters chosen so that

$$\mathbf{u} \approx \psi(\overline{\psi}(\mathbf{u}))$$

for a given pose $\mathbf{u}$.

The functions are defined recursively with respect to $\ell$ *layers* of function composition:

$$\psi(\mathbf{z}) := \psi_\ell(\mathbf{z}) = f_\ell(\mathbf{W}_\ell \phi_{\ell-1}(\mathbf{z}) + \mathbf{b}_\ell) \tag{3}$$

where $f_j : \mathbb{R}^{r_{j-1}} \to \mathbb{R}^{r_j}$ is the component-wise (typically nonlinear) activation function mapping its $r_{j-1}$-long vector input to the next layer's $r_j$-long vector input (base cases: $r_0 = r$ and $r_\ell = 3n$), each $\mathbf{W}_j \in \mathbb{R}^{r_{j-1} \times r_j}$ is a matrix of weights and each $\mathbf{b}_j \in \mathbb{R}^{r_j}$ is a vector of offsets, also known as bias. The encoder has an analogous recursive definition and set of parameters:

$$\overline{\psi}(\mathbf{u}) := \overline{\psi}_0(\mathbf{u}) = \overline{f}_0(\overline{\mathbf{W}}_0 \overline{\psi}_1(\mathbf{q}) + \overline{\mathbf{b}}_0) \tag{4}$$

where each layer has reversed dimensions of the decoder function: $\overline{\psi}_j : \mathbb{R}^{r_j} \to \mathbb{R}^{r_{j-1}}$, $\overline{f}_j : \mathbb{R}^{r_{j-1}} \to \mathbb{R}^{r_{j-1}}$, $\overline{W}_j \in \mathbb{R}^{r_j \times r_{j-1}}$, and $\overline{\mathbf{b}}_j \in \mathbb{R}^{r_{j-1}}$.

Given this definition, one must choose layer sizes, network depth, input and output dimensions, and activation functions to determine the final structure of the network. These considerations will be discussed in subsection 4.3.

### 4.2. Training Method

Given as input a set of $N$ example displacements, $[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$, we determine the optimal parameters $\hat{\theta}$ of our encoder and decoder, by minimizing the summed round-trip *reconstruction loss* of mapping each example displacement $\mathbf{u}_i$ into the reduced (latent) space of $\mathbf{z}$ values and back up to the space of displacements in a $L_2$ sense:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{u}_i - \psi_\theta(\overline{\psi}_\theta(\mathbf{u}_i))\|_2^2 \tag{5}$$

Where instantiations of the encoder and decoder are written $\overline{\psi}_\theta$ and $\psi_\theta$, while $\theta$ refers to all of the *parameters*, $\{\mathbf{W}, \overline{\mathbf{W}}, \mathbf{b}, \overline{\mathbf{b}}\}$, taken together. We minimize our loss using ADAM [KB14], with gradients

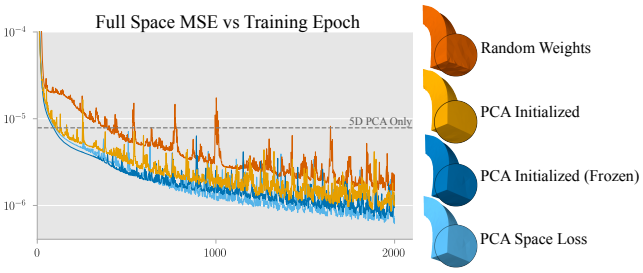


**Figure 5:** *We recorded the full-space MSE evaluated over the entire training set at the end of each epoch.* ***Orange:*** *Randomly initialized weights trained in full space.* ***Yellow:*** *PCA initialized weights allowed to change during training in full space.* ***Dark Blue:*** *PCA initialized weights frozen while training in full space.* ***Light Blue:*** *Loss computed in PCA coefficients.*

computed via the automatic differentiation (autodiff) framework TENSORFLOW [AAB*15] and KERAS [C*15]. We chose TENSORFLOW as the underlying autodiff framework due to it's support for high-performance C++ model deployment.

Unfortunately, training this network as it is with randomly initialized weights and an output variable for every single degree of freedom in the mesh yields unacceptable visual results and simulation behaviour, fraught with high-frequency artifacts (Figure 5). This is because our model must predict individual values for every single vertex displacement without any knowledge of their significant correlation. Also, the mean-squared error (MSE) alone is insensitive to these small yet salient perturbations in surface geometry. Although applying a regularizing term to the loss to inform the model of the correlation between vertices may be possible, we found that using PCA to reduce the size of the output for our network is highly effective. This technique has originated in the machine learning literature [KRMW14] and has also been used recently for other neural net models of mesh animation [LKA*17, BODO18].

It may seem counter-intuitive to perform PCA since this is precisely the technique we are trying to improve on. However, it is the final non-linear degrees of freedom that we are interested in. It makes no difference if one step in our pipeline is linear, as long as we choose a PCA basis large enough to capture all of the possible deformations we are interested in.

After performing PCA on the training data, we get a basis $\mathbf{U} \in \mathbb{R}^{3n \times k}$. The matrices $\mathbf{U}^T$ and $\mathbf{U}$ can be seen as initializing the first and last layers of our network respectively, so our decoder becomes

$$\psi(\mathbf{z}) = \mathbf{U}\phi(\mathbf{z}) \tag{6}$$

where $\phi : \mathbb{R}^k \to \mathbb{R}^{3n}$ is the strictly nonlinear portion of our network.

Although the weights computed to initialize $\mathbf{U}$ can be further modified during optimization, we found a significant training speedup by computing our loss directly in the reduced coordinates, without any reduction in quality( Figure 5). The form of our final

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

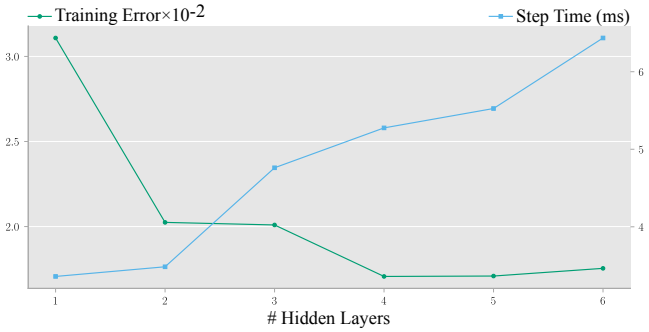


**Figure 6:** *Here we depict the trade-off between accuracy and speed with increasing model depth. Lower is better on both axes.*

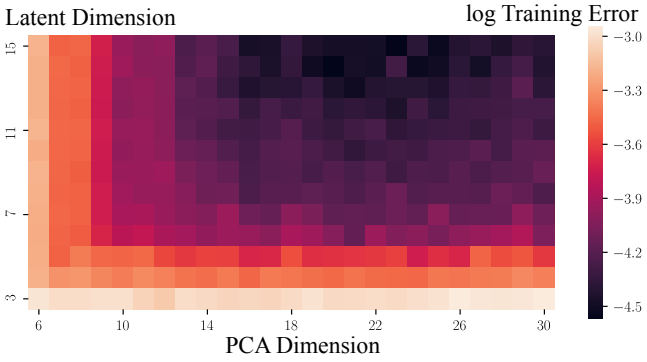


**Figure 7:** *Training error is visualized in relationship latent vector size, and number of PCA basis vectors. The noise present in the plot is a result of the gradient-descent training procedure.*

training objective is therefore written as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left\| \mathbf{U}^T \mathbf{u}_i - \phi_\theta(\bar{\phi}_\theta(\mathbf{U}^T \mathbf{u}_i)) \right\|_2^2 \tag{7}$$

where all of the $\mathbf{U}^T \mathbf{u}_i$ are precomputed. Our final model architecture can be seen in Figure 4.

### 4.3. Model Parameters

The choice of activation function has important implications to performance and training. In our case, the activations on the input and output layers were left as identity, since potential displacements are unbounded. For the rest of the network, we experimented with many of the most popular nonlinear activation functions such as sigmoid, tanh, ReLU, etc. We found that although ReLU is the simplest and most efficient, it produces simulation artifacts as can be seen in our video. The exponential linear unit (ELU) is essentially a smoothed ReLU function which makes it differentiable everywhere. We found ELU provided the best compromise between simulation and training performance.

$$\text{ELU}(x) = \begin{cases} x & \text{if } x >= 0 \\ e^x - 1 & \text{if } x < 0 \end{cases}$$

As for layer size and count, we found that two hidden layers of width 100 provided the best tradeoff between training accuracy and simulation speed as can be seen in Figure 6. This result mirrors that of other recent works such as [BODO18] which found only two layers to be sufficient after PCA reduction.

The final parameters to choose are the size of the encoded vector $r$, and the number of basis vectors $k$ we retain from PCA. These parameters determine to what level of accuracy the autoencoder can represent a given data set. Figure 7 shows that for a fixed latent space dimension $r$ , increasing PCA dimension $k$ decreases the overall error up to a certain "saturation" point, and vice versa. For our examples, we retain enough basis vectors $k$ such that the maximum per-vertex displacement reconstruction-error $\max_i \left\| \mathbf{u}_i - \mathbf{U}\mathbf{U}^T \mathbf{u}_i \right\|_2^2$ is under a user-determined error threshold.

Since the outer PCA layers limits the minimum possible achievable error, we choose a second larger tolerance level and then find the specific value of $r$ which meets the tolerance by starting with an appropriate guess, typically around $k/2$, and iteratively training our model with larger values of $r$ until the error is achieved.

In our case we opted for an absolute error tolerances of 3mm and 6mm respectively. We chose this error approach because it provides a more intuitive understanding of the degree to which the training set is captured. All of our meshes were on a length scale of about 15cm.

In the course of our experiments, we considered the use of more advanced autoencoder architectures such as the denoising autoencoder [VLBM08] and the variational autoencoder [KW13]. However, we found that although these alternatives are compatible with our method, the plain autoencoder architecture we described performs the best in terms of simulation quality and speed.

### 4.4. Training Data Generation

We are agnostic to the source of our training data, as long as the examples are representative of achievable configurations in the desired material model. Since our goal is to generate a non-linear reduced space that describes very specific scenarios, we use simulation snapshots generated by recording user interactions with a coarse mesh and replay the interaction forces onto the full resolution domain object (Figure 8) similar to the approach of [BJ05]. We use TetWild [HZG*18] to generate the volumetric meshes from surface meshes obtained from the Stanford 3D Scanning Repository or modeled from scratch.

Using this approach we generated high quality deformation spaces from just 500-1000 frames of simulation without subsampling or interpolating simulation frames. During training, convergence was typically achieved after 2000-3000 epochs with a batch size of 256 and a learning rate of 0.001. Other parameters of ADAM were left at the defaults described in the original paper [KB14]. Due to the small size of our networks, the gradient descent portion of our pipeline takes no longer than 10 minutes. All training and performance times are listed in Table 1.

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

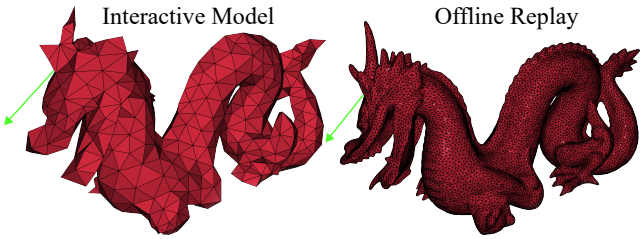


**Figure 8:** *Left: User interacts with a low-resolution (6k tetrahedra) dragon for real-time training pose generation. Right: Snapshot from high resolution (430k tetrahedra) simulation where external forces are replayed to generate final training poses.*

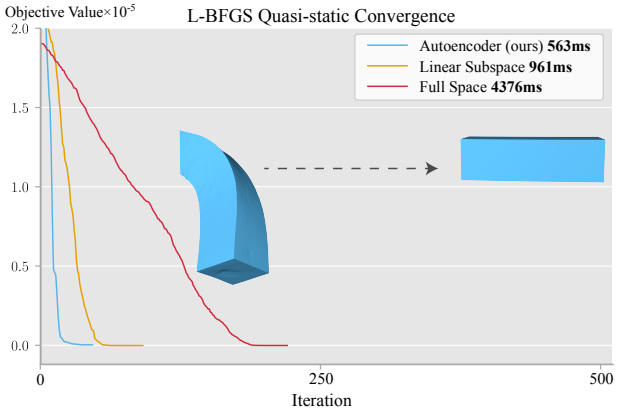


**Figure 9:** *A comparison of the L-BFGS convergence rate between the full space, a PCA only subspace, and our latent space where dimensions were chosen for approximately equal training error. Each solve was preconditioned with the rest-pose hessian, except for the Autoencoder which was preconditioned using Equation 15 at the starting pose.*

## 5. Latent-space Dynamics

In this section, we present the equations of motion in terms of the autoencoder latent variables and describe an approach to discretize and solve in time.

### 5.1. Implicit Integration in the Latent-space

Direct substitution of $\psi(\mathbf{z})$ into Equation 1 is problematic, since the nonlinear nature of our reduced space would require evaluating high order derivatives of our decoder network. This is a costly operation we would prefer to avoid.

Instead we build on the energy-diminishing integrator formalism [SH98] which has become popular in computer graphics [MTGG11, PBH15, LBK17] since it allows us to express stable implicit Euler integration as a minimization problem.

Following [MTGG11, PBH15] the Euclidean space timestepping equation is expressed as follows:

$$\mathbf{u}_{n+1} = \underset{\mathbf{u}}{\operatorname{argmin}} \frac{1}{2h^2} \|\mathbf{u} - \mathbf{u}_n - \dot{\mathbf{u}}_n h\|_{\mathbf{M}}^2 + V(\mathbf{u}) \tag{8}$$

Where $h$ is the chosen timestep and $V$ is the elastic potential energy leading to the internal restorative forces in the original formulation. We omit the external force term for clarity of exposition and since it can alternatively be defined as a component of the energy. In order to solve this problem using our reduced space, we reformulate the optimization at the position level using the substitution $\dot{\mathbf{u}}_n = \frac{1}{h}(\mathbf{u}_n - \mathbf{u}_{n-1})$ and optimize over $\mathbf{z}$ via the relationship $\mathbf{u} = \psi(\mathbf{z})$.

$$\mathbf{z}_{n+1} = \underset{\mathbf{z}}{\operatorname{argmin}} \frac{1}{2h^2} \|\psi(\mathbf{z}) - 2\mathbf{u}_n + \mathbf{u}_{n-1}\|_{\mathbf{M}}^2 + V(\psi(\mathbf{z})) \tag{9}$$

This allows us to find a solution in our reduced space, but still suffers from the requirement of evaluating a costly full-space mass-matrix product and maintaining the full space states $\mathbf{u}_n$ and $\mathbf{u}_n - 1$.

Our key insight is that since the outer layer of our autoencoder is a linear transformation, we need only *partially* decode the current pose using the nonlinear portion of our network described in Equation 6 and assign it to $\mathbf{q} \in \mathbb{R}^k$ and multiply it with our precomputed partially reduced mass matrix $\tilde{\mathbf{M}} \in \mathbb{R}^{k \times k}$

$$\mathbf{q} = \phi(\mathbf{z}) \tag{10}$$

$$\tilde{\mathbf{M}} = \mathbf{U}^T \mathbf{M} \mathbf{U} \tag{11}$$

So our objective, written here as $E$, and timestepping equation finally becomes

$$E(\mathbf{z}, \mathbf{q}_n, \mathbf{q}_{n-1}) = \frac{1}{2h^2} \|\phi(\mathbf{z}) - 2\mathbf{q}_n + \mathbf{q}_{n-1}\|_{\tilde{\mathbf{M}}}^2 + V(\psi(\mathbf{z}))$$
$$\mathbf{z}_{n+1} = \underset{\mathbf{z}}{\operatorname{argmin}} E(\mathbf{z}, \mathbf{q}_n, \mathbf{q}_{n-1}) \tag{12}$$

### 5.2. Jacobian Evaluation and Solving with L-BFGS

One approach to minimizing Equation 12 would be to use a Newton's search [NW06]. However this again would require evaluating high-order derivatives of $\phi(\mathbf{z})$ which we are trying to avoid. Instead we rely on the popular Quasi-Newton method L-BFGS [BNS94] which has been successfully used to efficiently solve mechanics equations [MS79] and has recently shown promise for real-time deformable body simulation [LBK17].

We begin by evaluating the gradient of our objective in Equation 12 which is as follows:

$$\frac{\partial E}{\partial \mathbf{z}} = \frac{1}{h^2} \mathbf{J}_{\mathbf{z}}^T \tilde{\mathbf{M}} (\phi(\mathbf{z}) - 2\mathbf{q}_n + \mathbf{q}_{n-1}) - \mathbf{J}_{\mathbf{z}}^T U^T \mathbf{f}_{\text{int}}(\psi(\mathbf{z})) \tag{13}$$

Evaluating the Jacobian of our network $\mathbf{J} = \frac{\partial \phi}{\partial \mathbf{z}}$ is a potentially costly procedure. Naively, one could use the auto-differentiation framework to compute the Jacobian matrix one row at a time. Using the reverse-mode differentiation standard in TENSORFLOW, this would be an $\mathcal{O}(k^2)$ operation, since evaluation of the gradient for a single output variable is the same complexity as evaluating the original network $O(k)$.

However, we observe that constructing the Jacobian matrix explicitly is not necessary. We need only implement a function to directly compute the *action* of this matrix against a given vector $\mathbf{v} \in \mathbb{R}^k$

$$\text{vjp}(\mathbf{v}, \mathbf{z}) = \mathbf{J}_{\mathbf{z}}^T \mathbf{v} \tag{14}$$

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

also known as the vector-Jacobian product.

This is a standard operation in autodiff frameworks which support forward-mode, and has equivalent complexity to a single gradient evaluation with respect to an input variable $\mathcal{O}(k)$. Unfortunately our chosen framework does not directly support forward-mode autodiff, but we were able to employ the workaround described by [Tow17]. Evaluating the resulting Jacobian-vector product operator scales linearly with the cost of evaluating the feed-forward pass of the network itself.

L-BFGS uses gradient information to progressively refine an estimate of the Hessian of our cost function. With an initial guess of Identity, and warm starting with the previous timestep's solution, L-BFGS can solve our time step optimization problem reasonably quickly. However, we can increase its overall performance by warm staring the Hessian approximation. In our case we warm start each L-BFGS iteration with

$$\tilde{\mathbf{H}} = \mathbf{J}_{\mathbf{z}_n}^T \tilde{\mathbf{K}}_0 \mathbf{J}_{\mathbf{z}_n} \tag{15}$$

where $\tilde{\mathbf{K}}_0 = U^T \mathbf{K}_0 U$, $\mathbf{K}_0 = \frac{\partial^2 \mathbf{V}(\mathbf{0})}{\partial \mathbf{u}^2}$ is the reduced stiffness matrix at the objects rest state and $\mathbf{J}_{\mathbf{z}_n}$ is the Jacobian of $\phi(\mathbf{z})$ evaluated at the beginning of the current time step. Computing $\tilde{\mathbf{H}}$ requires constructing the full autoencoder Jacobian by making $r$ calls to our Jacobian-vector product function. However, this cost is minimal since we only construct it once at the begining of the optimization procedure. Furthermore, $\tilde{\mathbf{H}}$ is a matrix in our fully reduced space making it very fast to solve against. In our implementation we factorize the matrix at the beginning of the L-BFGS loop and then perform a single back solve per iteration.

We modified an open-source L-BFGS implementation for C++ [Qiu16] to enable our pre-conditioner. The included backtracking line-search satisfying the Wolfe conditions was used and the window size for the Hessian approximation was set to 8. Our criteria for convergence is given by $\|\nabla E\|_2 < \varepsilon \cdot \max(\|\mathbf{z}\|_2, 1.0)$ with $\varepsilon = 10^{-8}$. A convergence comparison without cubature between our space, PCA only, and the full space can be seen in Figure 9.

### 5.3. Discrete cubature acceleration

Now that we have an efficient way of minimizing Equation 12, the key performance bottleneck rests on the computation of the full space energy $\mathbf{V}(\psi(\mathbf{z}))$ and the projection of its negative gradient the internal forces $\mathbf{f}_{\text{int}}(\psi(\mathbf{z}))$. Direct computation of the full elastic potential gradient $\frac{\partial V}{\partial \mathbf{u}} \in \mathbb{R}^{3n}$ is by necessity at least $O(n)$. A typical finite-element elastic potential and its gradient are written as a summation over potential contributions from each of the $m$ elements of the mesh:

$$V(\mathbf{u}) = \sum_{i=1}^{m} V_i(\mathbf{u}) \qquad \text{and} \qquad \frac{\partial V}{\partial \mathbf{u}} = \sum_{i=1}^{m} \frac{\partial V_i}{\partial \mathbf{u}}, \tag{16}$$

where $V_i : \mathbb{R}^{3n} \to \mathbb{R}$ maps displacements to the contribution of the $i$th element.

Ideally, one could produce an analytical expression for the energy and forces directly in terms of the reduced coordinates. Even the most successful implementation of this approach in [BJ05] results in cubic time and space complexity, as well as being restricted to only the StVK material. This approach would be further complicated due to our coordinates being decoded through a complex and nonlinear function. Therefore, we turn to an approximate *cubature*-based approach pioneered by [AKJ08] and expanded upon by many others [vTSSH13, YLX*15, PBH15].

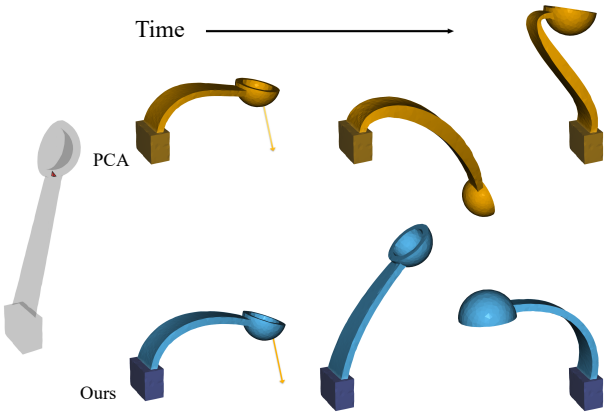


**Figure 10:** *Using only a single cubature point chosen by [AKJ08] (left) results in large deformation artifacts for the 6 dimensional linear subspace (top). Our autoencoder can operate with just 2 degrees of freedom and maintain high deformation fidelity.*

The summations over all elements may be approximated by a weighted, truncated summation, i.e., cubature method, over a subset $S \subseteq \{1, \ldots, m\}$ of $|S| \leq m$ elements:

$$V(\mathbf{u}) \approx \sum_{i \in S} w_i V_i(\mathbf{u}) \qquad \text{and} \qquad \frac{\partial V_i}{\partial \mathbf{u}} \approx \sum_{i \in S} w_i \frac{\partial e_i}{\partial \mathbf{u}}, \tag{17}$$

Where only the vertices in $\mathbf{u}$ corresponding to selected elements need be decoded, which we write as $\bar{\mathbf{u}} = \bar{\mathbf{U}}\mathbf{q}$. A successful cubature method will carefully select a fixed subset $S$ and corresponding weights $w$ so that approximation error is minimized for any *typical* displacement $\mathbf{u}$. In our case, we employ the algorithm described by [AKJ08] due to their freely available implementation. However, in principle our method is compatible with any cubature element selection algorithm and may benefit from more advanced selection techniques already mentioned.

The cubature optimization scheme we used requires the use of a linear subspace. For this, we used $\mathbf{U}$ on the outside layer of our network, with a relative error target of 0.05 or a maximum of 500 sample points. Although the optimized cubature method was designed specifically to approximate the reduced forces, we found that re-using the weights when doing a weighted sum of the energy (Equation 17) also provided a good approximation.

We found that when using models with very few latent degrees of freedom, far fewer cubature points than those chosen for the linear space can be used while retaining stability, as can be seen in Figure 10. This suggests a promising direction for future work on choosing cubature points based on the nonlinear degrees of freedom. However, for the sake of performance comparisons in this investigation, we use equal numbers of cubature points.

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

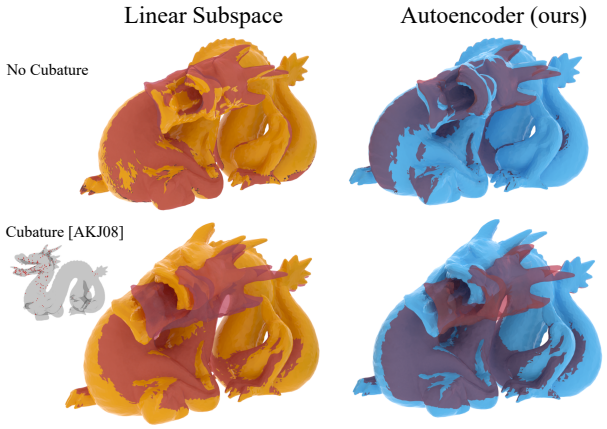


**Figure 11:** *Comparison of PCA subspace (orange #dof=29) and 10dof Autoencoder reduced space (blue #dof=10) with full-space simulation (transparent red #dof=83268, 430k tetrahedra) with and without cubature enabled and a Young's modulus of $10^5$ Pa. High quality correspondence is achieved in both cases, however our method achieves a $\sim$1.2$\times$ performance advantage over PCA alone (157Hz vs 185Hz).*

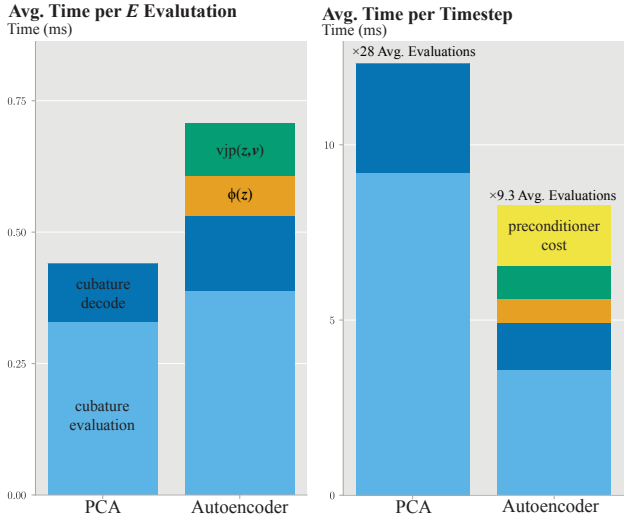


**Figure 12:** *This plot shows a breakdown of the primary time costs of our algorithm compared to PCA alone. Left: The average times for a single evaluation of the objective. Right: The average times across an entire timestep. The armadillo example was used to generate these plots.*

The un-optimized pseudo code for our final algorithm can be seen for the timestepper in algorithm 1 and for the objective function in algorithm 2.

## 6. Results & Discussion

In this section we demonstrate some of the advantages of performing reduced deformable simulation in our autoencoder latent-space.

### 6.1. Robustness and Visual Quality

During our experiments we found that we could choose a latent dimension $r$ approaching the true minimum DOFs in a system. Take for example the catapult in Figure 10. Our model is able to represent the large nonlinear bending deformation with only 2 DOFs. To achieve a similar level of accuracy with PCA alone, at least 6 DOFs are required. The surprising benefit of this highly-reduced space is that there is little room for non-physical configurations to be represented. As a consequence, we are able to get away with using only a single cubature point during simulation and still maintain believable results. In contrast, the 6 DOF linear subspace diverges wildly from any reasonable deformation.

When comparing against a full space simulation(Figure 11), we found that our nonlinear reduced space qualitatively performed at least as well as a linear space alone when $k_p$ was chosen to match for training error. In addition, our nonlinear space achieves an average 1.2$\times$ performance advantage when using the same number of cubature points. This is done while operating on an object that is an order-of-magnitude stiffer (in terms of Young's Modulus) than those reported in previous reduced simulation works for large deformation [vTSSH13]. However, as can be seen in the figure, when cubature is enabled, the accuracy of both the linear and autoencoder simulations suffer. We attribute this defect to the cubature method employed, and expect that this effect would vanish if one used a more advanced method such as [VTSSH15], or took into account information about the latent space while training.

Compared to previous reduced space methods such as rotation-strain coordinates [PBH15], our nonlinear formulation of the reduced space precludes the need to tune material parameters to achieve agreement with the full space as seen in Figure 14. Our method also allows material parameters to be changed after training, as long as the examples used to create the deformation space contain the poses desired.

### 6.2. Performance

When choosing a linear PCA-based subspace to compare against, it is important to consider what size basis $k_p$ to use. There are many factors to consider, such as qualitative appearance during simulation, performance, achieved training error etc. In our performance comparisons, we chose $k_p$ such that the maximum per-vertex reconstruction error falls within 0.1mm of the autoencoder reconstruction error. In some cases, this results in choosing $k_p$ that is equal to the outer dimension of the autoencoder $k$ given a large enough latent dimension $r$, such as in Figure 13. All of our models use the NeoHookean material model as implemented in GAUSS [Lev18], however this could easily be replaced with another energy.

Figure 13 shows a frame generated via interactive manipulation of our 164k element Armadillo mesh. Notice that even a small number of latent degrees-of-freedom can generate expressive animations as seen in our video, featuring independent manipulation of the armadillo's arms and legs. In this example the average performance gain is an average of 1.7$\times$ more timesteps per second.

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

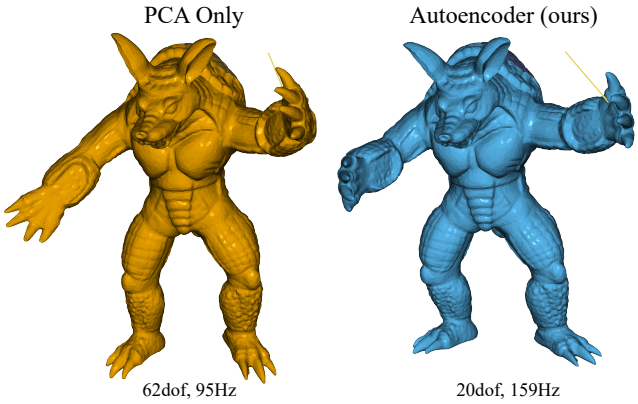


**Figure 13:** *A performance comparison for a 164k tetrahedra armadillo model. Our autoencoder space achieves equivalent training error with only 20 DOFs compared to the 62 needed for PCA alone.*

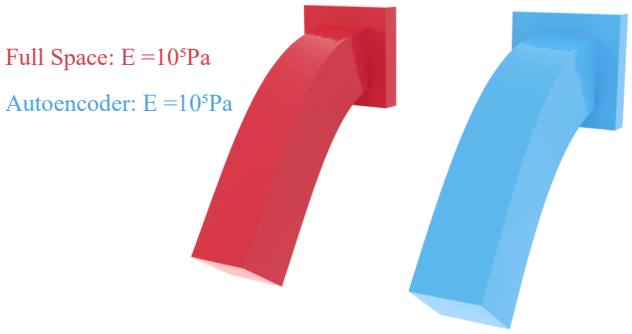


**Figure 14:** *A full space bar (left) with a Young's modulus of* $10^5 Pa$ *is compared to a bar simulated in our latent space. A nearly visually indistinguishable match is achieved without tuning any material parameters.*

As seen in Figure 12, a single objective evaluation in our latent-space is more costly than PCA alone due to the additional decode and vector-Jacobian product operations. However, the nonlinear nature of the latent-space drastically reduces the number of L-BFGS iterations needed to converge. The intuition being that a small step in $\mathbf{z}$ space can create a large and non-linear effect in displacements, where as small steps in $\mathbf{q}$ are still inherently linear and thus less dramatic, making walking in $\mathbf{z}$ space more efficient.

Performance numbers reported in these figures refer to the time required to compute a full timestep not including rendering. Visualizing a mesh requires fully decoding the displacements of all surface vertices, and if performed on the CPU can cause a significant slowdown for large meshes. Therefore we chose to decode the displacements, that is $\mathbf{u} = \mathbf{U}^T\mathbf{q}$, on the GPU using our vertex-shader. This easily gives us a fixed 60 frames per second on all studied examples. Further performance details for each model can be found in Table 1.

Training data was generated using an Intel Core i7-4770 CPU @ 3.40GHz with 16GB memory. The Autoencoder was trained using an Nvidia GeForce GTX 970 GPU and reduced space simulation was done entirely with CPU on a system with dual Intel Xeon E5-2637 v3 3.50GHz processors and 64GB RAM. Our code is implemented in C++ for the runtime portion and has not been optimized beyond multithreading using OpenMP and the Eigen linear algebra library. We did not implement GPU support for simulation computations other than performing $\mathbf{Uq}$ in the shader during rendering. We compare against our own implementations of linear reduced space and full simulation. All of our examples use the Neo-Hookean material model implemented in GAUSS [Lev18], however this could easily be replaced with another energy.

We also performed a comparison between the L-BFGS solver and a fully-implicit Newton solver for the PCA-only subspace, owing to the availability of the Hessian in this case. We found that fewer iterations were required for equivalent convergence criteria of $\varepsilon = 10^{-8}$. However, the additional cost of constructing the reduced stiffness matrix (including cubature) causes the Newton based approach to perform approximately 1.3 times slower than the L-BFGS solver. This reflects recent results showing that Quasi-Newton approaches generally provide equal or greater performance in real-time applications for unreduced dynamics [LBK17, WY16]. Other reduced methods have also favored quasi-Newton approaches to avoid expensive hessian construction and inversion [vTSSH13].

## 7. Limitations & Future Work

In this paper we have presented the first neural network-based, non-linear reduced space simulation of large-deformation elastic dynamics. One of the most surprising results of this work is how modest the performance improvement is over linear, reduced models Figure 12. However, being a new method, we feel that there are many opportunities to address this, each one being an exciting avenue of future work. As seen in our catapult example, even a single element can provide enough stability for simulation in our small latent-spaces. Therefore, developing a cubature approach which takes advantage of latent-space information, or loosens requirements such as weight non-negativity, when choosing weights and elements, could allow us to rely on much fewer cubature elements and consequently increase performance. Furthermore applying optimizations such as evaluating our network decode and Jacobian on the GPU during runtime could further close the per-evaluation performance gap.

As can be seen in Figure 15, we inherit some of the problems from traditional subspace simulation such as the lack of local deformations not included in the training data, or more generally being able to represent poses not included in the training data. Furthermore, we require physically plausible deformation examples to train our network. This restricts us to use cases where the desired deformation is known in advance and can be precomputed. It would be highly desirable to adapt the work on modal analysis and its extensions such as modal derivatives [BJ05] to our reduced spaces to enable simulation without specific knowledge about the desired deformations. There is a wealth of other existing work done to improve and extend linear subspace simulation such as rigid floating-

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

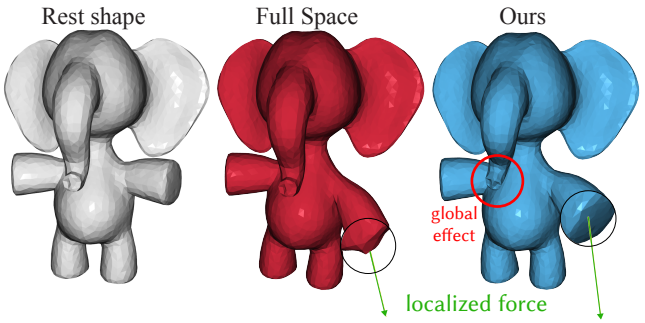


**Figure 15:** *A localized force is applied to the arm of the elephant. The full space simulation (left) shoes a corresponding local deformation, whereas our reduced space (right) cannot represent the produced bump, instead directing the energy into a small movement of the trunk.*

frames, collisions, sub-structuring, etc. that is in principle compatible with our method.

Also observed in Figure 15 is that of global motion resulting from locally applied forces. Pulling on the elephant's arm causes the trunk to wiggle slightly. This is because the encoded representation does not fully dis-entangle information about the trunk position from the arm position using the provided training data. Similarly, the space learned by our autoencoder is not always perfectly smooth. This is particularly apparent in the video corresponding to Figure 14. Exploring more complex neural network architectures such as the variational autoencoder (VAE) [KW13], or initializing outer layers with sparse modes rather than PCA is another potential direction of research, and may also prove useful in determining the ideal latent dimension size $r$ without expensive retraining of the model.

## 8. Acknowledgments

This work is funded in part by NSERC Discovery Grants (RGPIN–2017–05235, RGPIN-2017-05524, RGPAS–2017–507938, RGPAS-2017-507909), Connaught Funds (NR2016–17), the Canada Research Chairs Program, the Fields Institute, and gifts by Adobe Systems Inc, Autodesk Inc, and MESH Inc. We also thank Sarah Kushner for help with figure creation.

**Algorithm 1:** Simulation timestepping loop

Precompute $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{K}}_0$;
$\mathbf{z}_0, \mathbf{z}_1 \leftarrow \overline{\psi}(\mathbf{0})$;
$\mathbf{q}_0, \mathbf{q}_1 \leftarrow \mathbf{0}$;
$n \leftarrow 1$;
**while** $n \leq maxFrames$ **do**
  $\mathbf{J} \leftarrow$ Evaluate Autoencoder Jacobian at $\mathbf{z}_n$;
  $\mathbf{L} \leftarrow$ Cholesky prefactor $\tilde{\mathbf{H}} = \mathbf{J}^T \tilde{\mathbf{K}}_0 \mathbf{J}$;
  $\mathbf{z}_{n+1}, \mathbf{q}_{n+1} \leftarrow$ L-BFGSminimize$(E, \mathbf{z}_n, \mathbf{z}_{n-1}, \mathbf{L})$;
  $n \leftarrow n+1$;
**end**

**Algorithm 2:** Objective and gradient evaluation

$E(\mathbf{z}, \mathbf{q}_n, \mathbf{q}_{n-1})$;
**Input :** Previous two timestep values for $\mathbf{q}_i$, the current guess for $\mathbf{z}$ and cubature weights $\mathbf{w}$ and indices $S$;
**Output:** The objective value denoted by $x$ and the gradient $\mathbf{g}$;
$\mathbf{q} \leftarrow \phi(\mathbf{z})$;
$\bar{\mathbf{u}} \leftarrow \bar{\mathbf{U}}\mathbf{q}$ displacements for only cubature elements in $S$;
$V, \mathbf{f} \leftarrow \mathbf{0}$;
**for** $i, w_i$ ***in*** $S, \mathbf{w}$ **do**
  $V \leftarrow V + w_i V_i(\bar{\mathbf{u}})$;
  $\mathbf{f} \leftarrow \mathbf{f} + w_i \tilde{\mathbf{f}}^i_{\text{int}}(\bar{\mathbf{u}})$;
**end**
$x \leftarrow \frac{1}{2}\mathbf{q}^T\tilde{\mathbf{M}}\mathbf{q} + h^2 V$;
$\mathbf{g} \leftarrow \text{vjp}(\tilde{\mathbf{M}}\mathbf{q} - h^2\tilde{\mathbf{f}}_{\text{int}}, \mathbf{z})$;

## References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. 5

[AKJ08] AN S. S., KIM T., JAMES D. L.: Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 2, 4, 8

[AMR*] ANDREW N., MATTHIAS M., RICHARD K., EDDY B., MARK C.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4, 2

[BEH18] BRANDT C., EISEMANN E., HILDEBRANDT K.: Hyper-reduced projective dynamics. *ACM Trans. Graph. 37*, 4 (July 2018), 3

[BGC15] BENGIO Y., GOODFELLOW I. J., COURVILLE A.: Deep learning. *Nature 521* (2015), 4

[BJ05] BARBIČ J., JAMES D. L.: Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph. 24*, 3 (July 2005), 2, 4, 6, 8, 10

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 2, 3

[BNS94] BYRD R. H., NOCEDAL J., SCHNABEL R. B.: Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming 63*, 1 (Jan 1994), 7

[BODO18] BAILEY S. W., OTTE D., DILORENZO P., O'BRIEN J. F.: Fast and deep deformation approximations. *ACM Trans. Graph. 37*, 4 (July 2018), 3, 4, 5, 6

[BPT17] BONEV B., PRANTL L., THUEREY N.: Pre-computed Liquid Spaces with Generative Neural Networks. *arXiv to appear* (Apr 2017), 3

[BvTH16] BRANDT C., VON TYCOWICZ C., HILDEBRANDT K.: Geometric flows of curves in shape space for processing motion of deformable objects. *Computer Graphics Forum 35*, 2 (2016), 2

[BZ11] BARBIČ J., ZHAO Y.: Real-time large-deformation substructuring. *ACM Trans. Graph. 30*, 4 (July 2011), 3

[C*15] CHOLLET F., ET AL.: Keras. https://github.com/keras-team/keras, 5

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

| **Model** | $N_{tet}$ | AE(s) | PCA(s) | Cubature(s) | $\|S\|$ | $k_p$ | $r, (k)$ | PCA Step(ms) | AE Step(ms) |
|---|---|---|---|---|---|---|---|---|---|
| Beam | 45151 | 266 | 10 | 1493 | 503 | 23 | 10 (27) | 11.0 | 8.6 |
| X | 29686 | 290 | 34 | 9083 | 359 | 13 | 6 (19) | 5.2 | 4.6 |
| Armadillo | 164249 | 192 | 38 | 3360 | 401 | 62 | 20 (62) | 10.5 | 6.30 |
| Dragon | 429740 | 60 | 46 | 40493 | 197 | 25 | 11 (29) | 7.2 | 6.7 |
| Elephant | 61699 | 136 | 5 | 54 | 28 | 18 | 7 (28) | 4.2 | 4.2 |

**Table 1:** *From left to right: Model name, number of mesh elements, autoencoder training time, time taken to compute the PCA basis, cubature training time, # of cubature elements, size of PCA-only basis with equivalent error, autoencoder latent-space dimension (outer PCA space dimension), PCA only timestep time, autoencoder timestep time.*


[CBW*18] CHEN J., BAO H., WANG T., DESBRUN M., HUANG J.: Numerical coarsening using discontinuous shape functions. *ACM Trans. Graph. 37*, 4 (July 2018), 3

[CLMK17] CHEN D., LEVIN D. I. W., MATUSIK W., KAUFMAN D. M.: Dynamics-aware numerical coarsening for fabrication design. *ACM Trans. Graph. 36*, 4 (July 2017), 2, 3

[CLSM15] CHEN D., LEVIN D. I. W., SUEDA S., MATUSIK W.: Data-driven finite elements for geometry and material design. *ACM Trans. Graph. 34*, 4 (July 2015), 3

[CO18] CASAS D., OTADUY M. A.: Learning nonlinear soft-tissue dynamics for interactive avatars. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 1 (July 2018), 3

[CT17] CHU M., THUEREY N.: Data-Driven Synthesis of Smoke Flows with CNN-based Feature Descriptors. *Transaction on Graphics (SIGGRAPH) 36(4)* (Apr 2017), 3

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96* (Berlin, Heidelberg, 1996), Springer-Verlag, 2

[FGBP11] FAURE F., GILLES B., BOUSQUET G., PAI D. K.: Sparse Meshless Models of Complex Deformable Solids. 3

[GBFP11] GILLES B., BOUSQUET G., FAURE F., PAI D. K.: Frame-based elastic models. *ACM Trans. Graph. 30*, 2 (Apr. 2011), 3

[HLW06] HAIRER E., LUBICH C., WANNER G.: *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, vol. 31. Springer Science & Business Media, 4

[HMT*12] HAHN F., MARTIN S., THOMASZEWSKI B., SUMNER R., COROS S., GROSS M.: Rig-space physics. *ACM Trans. Graph. 31*, 4 (July 2012), 3, 4

[HS06] HINTON G. E., SALAKHUTDINOV R. R.: Reducing the dimensionality of data with neural networks. *Science 313*, 5786 (2006), 5

[HTC*14] HAHN F., THOMASZEWSKI B., COROS S., SUMNER R. W., COLE F., MEYER M., DEROSE T., GROSS M.: Subspace clothing simulation using adaptive bases. *ACM Trans. Graph. 33*, 4 (July 2014), 3

[HTZ*11] HUANG J., TONG Y., ZHOU K., BAO H., DESBRUN M.: Interactive shape interpolation through controllable dynamic deformation. *IEEE Transactions on Visualization and Computer Graphics 17*, 7 (July 2011), 3

[HZG*18] HU Y., ZHOU Q., GAO X., JACOBSON A., ZORIN D., PANOZZO D.: Tetrahedral meshing in the wild. *ACM Trans. Graph. 37* (2018), 6

[JBP06] JAMES D. L., BARBIČ J., PAI D. K.: Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph. 25*, 3 (July 2006), 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. 5, 6

[KJ09] KIM T., JAMES D. L.: Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 2, 3

[KLM01] KRYSL P., LALL S., MARSDEN J. E.: Dimensional model reduction in non-linear finite element dynamics of solids and structures. *INT J NUMER METH ENG 51*, 4 (2001), 2

[KMOD09] KHAREVYCH L., MULLEN P., OWHADI H., DESBRUN M.: Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph. 28*, 3 (July 2009), 3

[KRMW14] KINGMA D. P., REZENDE D. J., MOHAMED S., WELLING M.: Semi-supervised learning with deep generative models. *CoRR abs/1406.5298* (2014), 5

[KW13] KINGMA D. P., WELLING M.: Auto-encoding variational bayes. 6, 11

[LBBM18] LITANY O., BRONSTEIN A. M., BRONSTEIN M. M., MAKADIA A.: Deformable shape completion with graph convolutional autoencoders. 3

[LBK17] LIU T., BOUAZIZ S., KAVAN L.: Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Trans. Graph. 36*, 3 (May 2017), 4, 7, 10

[Lev18] LEVIN D. I. W.: Gauss: Gaggle of algorithms and utilities for simulating stuff. https://github.com/dilevin/GAUSS, 9, 10

[LHdG*14] LI S., HUANG J., DE GOES F., JIN X., BAO H., DESBRUN M.: Space-time editing of elastic motion through material optimization and reduction. *ACM Trans. Graph. 33*, 4 (July 2014), 3

[LJS*15] LADICKÝ L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 3

[LKA*17] LAINE S., KARRAS T., AILA T., HERVA A., SAITO S., YU R., LI H., LEHTINEN J.: Production-level facial performance capture using deep convolutional neural networks. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2017), SCA '17, ACM, 4, 5

[LSW*18] LUO R., SHAO T., WANG H., XU W., ZHOU K., YANG Y.: Deepwarp: Dnn-based nonlinear deformation. *CoRR abs/1803.09109* (2018), 2, 3

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (July 2011), 2

[MGL*15] MALGAT R., GILLES B., LEVIN D. I. W., NESME M., FAURE F.: Multifarious hierarchies of mechanical models for artist assigned levels-of-detail. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2015), SCA '15, ACM, 3

[MHR*16] MUSIALSKI P., HAFNER C., RIST F., BIRSAK M., WIMMER M., KOBBELT L.: Non-linear shape optimization using local subspace projections. *ACM Trans. Graph. 35*, 4 (July 2016), 2

[MS79] MATTHIES H., STRANG G.: The solution of nonlinear finite


© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.

element equations. *International Journal for Numerical Methods in Engineering 14*, 11 (1979), 7

[MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. *ACM Trans. Graph. 30*, 4 (July 2011), 4, 7

[MZS*11] MCADAMS A., ZHU Y., SELLE A., EMPEY M., TAMSTORF R., TERAN J., SIFAKIS E.: Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (July 2011), 2

[NKJF09] NESME M., KRY P. G., JEŘÁBKOVÁ L., FAURE F.: Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. 28*, 3 (July 2009), 3

[NW06] NOCEDAL J., WRIGHT S.: *Numerical optimization*. Springer Science & Business Media, 7

[OKHS03] O'BRIEN J., K. HAUSER K., SHEN C.: 2

[OSG02] O'BRIEN J. F., SHEN C., GATCHALIAN C. M.: Synthesizing sounds from rigid-body simulations. In *The ACM SIGGRAPH 2002 Symposium on Computer Animation* (July 2002), ACM Press, 2

[PBH15] PAN Z., BAO H., HUANG J.: Subspace dynamic simulation using rotation-strain coordinates. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 2, 3, 4, 7, 8, 9

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: Modal dynamics for graphics and animation. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 2

[Qiu16] QIU Y.: L-bfgs++. http://yixuan.cos.name/LBFGSpp/, 8

[SB12] SIFAKIS E., BARBIC J.: Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses* (New York, NY, USA, 2012), SIGGRAPH '12, ACM, 4

[SH98] STUART A., HUMPHRIES A. R.: *Dynamical systems and numerical analysis*, vol. 2. Cambridge University Press, 7

[Sha12] SHABANA A. A.: *Theory of vibration: Volume II: discrete and continuous systems*. Springer Science & Business Media, 2

[SSW*13] STANTON M., SHENG Y., WICKE M., PERAZZI F., YUEN A., NARASIMHAN S., TREUILLE A.: Non-polynomial galerkin projection on deforming meshes. *ACM Trans. Graph. 32*, 4 (July 2013), 2

[TGL*18] TAN Q., GAO L., LAI Y.-K., YANG J., XIA S.: Mesh-based autoencoders for localized deformation component analysis. 3

[TGLX18] TAN Q., GAO L., LAI Y.-K., XIA S.: Variational autoencoders for deforming 3d mesh models. 3

[TLP06] TREUILLE A., LEWIS A., POPOVIĆ Z.: Model reduction for real-time fluids. *ACM Trans. Graph. 25*, 3 (July 2006), 2

[TOK14] TENG Y., OTADUY M. A., KIM T.: Simulating articulated subspace self-contact. *ACM Trans. Graph. 33*, 4 (July 2014), 3

[Tow17] TOWNSEND J.: Blog post: A new trick for calculating jacobian vector products. https://j-towns.github.io/2017/06/12/A-new-trick.html, 8

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), SIGGRAPH '87, ACM, 2

[TSSP16] TOMPSON J., SCHLACHTER K., SPRECHMANN P., PERLIN K.: Accelerating Eulerian Fluid Simulation With Convolutional Networks. *ArXiv e-prints* (July 2016), 3

[UMK17] ULU E., MCCANN J., KARA L. B.: Lightweight structure design under force location uncertainty. *ACM Trans. Graph. 36*, 4 (July 2017), 2

[VLBM08] VINCENT P., LAROCHELLE H., BENGIO Y., MANZAGOL P.-A.: Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (New York, NY, USA, 2008), ICML '08, ACM, 6

[vTSSH13] VON TYCOWICZ C., SCHULZ C., SEIDEL H.-P., HILDEBRANDT K.: An efficient construction of reduced deformable objects. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 2, 4, 8, 9, 10

[VTSSH15] VON-TYCOWICZ C., SCHULZ C., SEIDEL H.-P., HILDEBRANDT K.: Real-time nonlinear shape interpolation. *ACM Trans. Graph. 34*, 3 (May 2015), 9

[WBT18] WIEWEL S., BECHER M., THUEREY N.: Latent-space physics: Towards learning the temporal evolution of fluid flow. *CoRR abs/1802.10123* (2018), 3

[WJBK15] WANG Y., JACOBSON A., BARBIČ J., KAVAN L.: Linear subspace design for real-time shape deformation. *ACM Trans. Graph. 34*, 4 (July 2015), 3

[WKD*18] WANG B., KRY P. G., DENG Y., ASCHER U. M., HUANG H., CHEN B.: Neural material: Learning elastic constitutive material and damping models from sparse data. *CoRR abs/1808.04931* (2018), 3

[WY16] WANG H., YANG Y.: Descent methods for elastic body simulation on the gpu. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 10

[XB16] XU H., BARBIČ J.: Pose-space subspace dynamics. *ACM Trans. Graph. 35*, 4 (July 2016), 3

[XLCB15] XU H., LI Y., CHEN Y., BARBIČ J.: Interactive material design using model reduction. *ACM Trans. Graph. 34*, 2 (Mar. 2015), 2

[YLX*15] YANG Y., LI D., XU W., TIAN Y., ZHENG C.: Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 2, 4, 8

© 2019 The Author(s)
Computer Graphics Forum © 2019 The Eurographics Association and John Wiley & Sons Ltd.