Interactive Stress Analysis via Nonlinear Subspace Simulation

Lawson Fulton, David I.W. Levin, David Duvenaud, Alec Jacobson
University of Toronto
40 St. George Street
Toronto, ON M5S 2E4, Canada

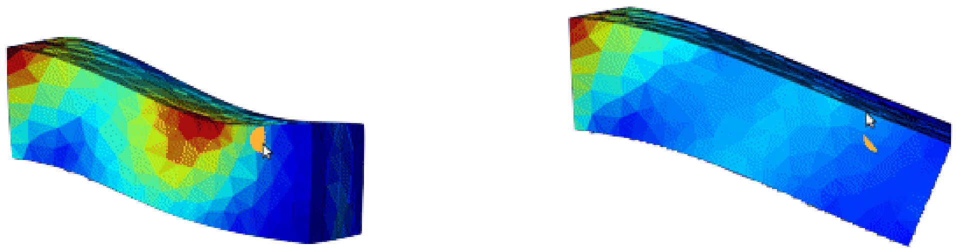
Figure 1: Real time stress field visualization of a 2k vertex model at 20ms/frame

CCS CONCEPTS

• **General and reference** ⟶ **Design**; • **Computing methodologies** ⟶ **Physical simulation**; *Mesh models*;

KEYWORDS

simulation, machine learning

ACM Reference format:
Lawson Fulton, David I.W. Levin, David Duvenaud, Alec Jacobson. 2018. Interactive Stress Analysis via Nonlinear Subspace Simulation. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 4 pages.
https://doi.org/0000001.0000001_2

1 INTRODUCTION

Predicting the stress experienced by physically simulated objects intended for manufacturing is an important step of the design process. However, the complex finite element simulations required to accurately perform this task are often too slow for real-time feedback.

Because of this, there has been decades of work attempting to accelerate solid mechanics simulations for the purposes of stress analysis. Such works take two broad forms, solver-based approaches (which attempt to improve the linear algebraic operations at the heart of a physics simulator) or discretization-based approaches (which try to reduce the degrees-of-freedom) in a computational mesh. One of the most successful techniques for speeding up simulations is linear modal analysis, a discretization-based method which performs simulation in a small linear reduced space constructed from the linear deformation modes of an object.

However, linear modal analysis has issues which prevent it from being a panacea to the performance problems of physics-based simulation. First, anything beyond the linear modes can be difficult to compute, relying on non-linear modal derivatives. Second, motions which span highly non-linear regions of the configuration space can require large linear spaces to accurately represent. This can have a dire impact on runtimes, since system matrices for small reduced spaces become dense and even a small increase in the number of modes can have a hampering effect on performance. Existing fast integration schemes for reduced models typically have $O(r^2)$ to $O(r^4)$ asymptotic behavior where r is the dimension of the reduced space as in [Barbič and James 2005] .

Ideally, one would generate as small a reduced space as necessary directly from simulation data. A strictly linear basis can make this difficult. In this paper we flaunt conventional wisdom and disregard linearity, in favor of non-liner dimensionality reduction techniques. In particular we draw from the field of machine learning and devise a new autoencoder architecture that can efficiently represent non-linear deformations with high fidelity using fewer degrees of freedom than a corresponding linear reduced representation. Furthermore we develop a minimization based time integrator that takes advantage of our reduced representation to produce dynamic simulations of production quality meshes at interactive rates.

2 RELATED WORK

There have been several attempts in recent literature to speed up stress analysis for shape design through pre-computation.

2.1 Sampling and Interpolation

The approach used in [Schulz et al. 2017] is to do a full stress analysis at multiple points within the parameter space, and use a clever interpolation scheme to produce intermediary results. This approach has the benefit of handling varying mesh topologies, but requires storing 2^k meshes where k is the number of model parameters, thus requiring large amounts of storage for complex designs.

2.2 Reduced Space Computation

Another approach, as in [Chen et al. 2016], is to use a small linear basis to generate the stress field and do a complete solve within this reduced space. This approach avoids storage of meshes, but has the drawback of only working for fixed mesh topologies. Therefore it is unsuitable for typical CAD models which have varying mesh topologies for each choice of design parameters.

For large deformations, linear subspace integration techniques such as [Barbič and James 2005] and [von Tycowicz et al. 2013] can also be used.

Traditionally, subspace simulation proceeds by constructing an orthonormal basis matrix $U \in M^{r \times n}$ where n is the total number of degrees of freedom in the system, and r is the size of the reduced space. Then the mapping from the reduced to the full space can be written as

$$q^t = U^T z^t$$

Their are multiple approaches for constructing U, such as computing the linear deformation modes, or performing principle component analysis (PCA) on simulation snapshots. However, they all suffer from the same fundamental drawback of requiring a large number of bases to represent significant non-linear deformations.

2.3 Machine Learning Approaches

We are not the first to propose machine learning driven approach to physical simulation. In [Ladický et al. 2015] and [Tompson et al. 2016], they propose models to explicitly predict the solution of the systems.

2.4 Simulation as Optimization

Several authors have proposed minimization schemes for optimization such as [Liu et al. 2017] and [Wang and Yang 2016].

3 METHODS

Our approach is to create a reduced space representation of our system with as few degrees of freedom as possible while maintaining deformation fidelity. In [Musialski et al. 2016], they show that shrinking the size of the reduced space leads to far fewer gradient descent iterations during optimization.

3.1 Subspace Construction

Our goal is to create a reduced space mapping that can capture large non-linear deformations with few degrees of freedom. One way to represent such a mapping is to use an artificial neural network. In particular, to facilitate the training of our network, we use a fully connected autoencoder trained with a mean squared error (MSE) loss.

$$L_\theta(x) = ||D_\theta(E_\theta(x)) - x||^2$$

Where $D_\theta : \mathbb{R}^r \rightarrow \mathbb{R}^n$ is the decoder network, and $E_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^r$ is the encoder network and θ are the network parameters. Each network is defined by a sequence of fully connected layers of rectified linear units (RELU) varying widths. A single layer i has the form

$$D^i(x) = \max(W^i x - b^i, 0)$$

and the full decoder network of l layers is then

$$D(x) = D^l \circ D^{(l-1)} \circ \cdots \circ D^0(x)$$

We train this autoencoder using a form of stochastic gradient descent known as ADAM [Kingma and Ba 2014] on a collection of example displacements generated from a full space simulation for a particular model.

As a consequence of our MSE loss, we found that our network had difficulty converging to a smooth representation in its unmodified form. Our solution was to initialize the first and last layers of our autoencoder with weights obtained from performing PCA on the training data. The number of components used to perform the PCA was chosen such that the training data could be recovered with an MSE of less than 10^{-3}. Allowing the weights of the PCA layers to learn further did not improve convergence, and significantly slowed the rate of learning, hence we chose to fix them during training. A comparison of results from our PCA-autoencoder with a plain autoencoder can be seen in figure 2.

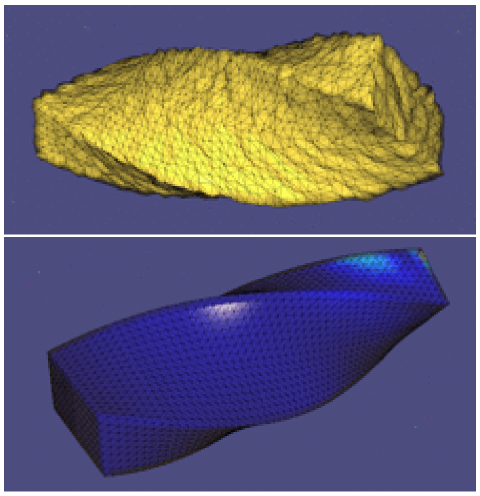

Figure 2: A comparison of the decoded representation without (top) and with (bottom) PCA initialized outer layers.

Through experiment we found that two fully connected layers of 200 units between the PCA and encoded layer worked well. Increasing the number of layers or number of nodes past this point did not improve the converged error on our examples.

3.2 Equations of Motion

In order to take advantage of our reduced representation, it is useful to formulate our simulation as an implicit integration scheme based on minimization in the reduced space. We derive our update scheme from Gauss' principle of least constraint, which formulates the equations of motion as a minimization over the instantaneous acceleration a

$$a = \arg\min_a \frac{1}{2}(Ma - F)^T M^{-1}(Ma - F)$$

We proceed to discretize in time

$$a = \frac{v^{t+1} - v^t}{h} = \frac{q^{t+1} - 2q^t + q^{t-1}}{h^2}$$

and then computing the next configuration q^{t+1} is given by

$$q^{t+1} = \arg\min_{q^{t+1}} \frac{1}{2h^2}(q^{t+1} - 2q^t + q^{t-1})^T M(q^{t+1} - 2q^t + q^{t-1}) - \frac{1}{h}(v^{t+1} - v^t)^T F(q^{t+1}) \quad (1)$$

The force term in (1) can be seen as the work done by the body forces over the timestep and the external forces.

$$-\frac{1}{h}(v^{t+1} - v^t)^T F(q^{t+1}) = h\Psi(q^{t+1}) - h(v^{t+1} - v^t)^T F_{\text{ext}}$$

Where $\Psi(q^t)$ gives the deformation energy at the given configuration. We use a standard Neo-Hookean elasticity energy. The final update formula then becomes

$$q^{t+1} = \arg\min_{q^{t+1}} \frac{1}{2}(q^{t+1} - 2q^t + q^{t-1})^T M(q^{t+1} - 2q^t + q^{t-1}) + h^2[\Psi(q^{t+1}) - (q^{t+1} - 2q^t + q^{t-1})^T F_{\text{ext}}] \quad (2)$$

and the gradient is given by

$$\frac{\partial L}{\partial q^{t+1}} = M(q^{t+1} - 2q^t + q^{t-1}) - hF_{\text{int}}(q^{t+1}) - hF_{\text{ext}}$$

If we are performing the optimization within the constraints of a reduced space, then we let $q^t = D(z^t)$ where D is the mapping from our reduced coordinate configuration z to the full configuration q. And then our gradient simply becomes

$$\frac{\partial L}{\partial z^{t+1}} = J_{t+1}^T[M(q^{t+1} - 2q^t + q^{t-1}) - hF_{\text{int}}(q^{t+1}) - hF_{\text{ext}}]$$

Where J_{t+1} is the jacobian matrix of our reduced space.

$$J_{t+1} = \frac{\partial D}{\partial z}|_z^{t+1}$$

3.3 Optimization

Our formulation of the simulation update as a minimization problem is naturally suited to Quasi-Newton methods such as L-BFGS (Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm)[Byrd et al. 1994]. A similar approach is employed by [Liu et al. 2017] and [Wang and Yang 2016].

The main challenge when using this gradient descent approach with our neural network reduced space is the requirement for the network jacobian. In our examples we use finite differences to compute the jacobian because of technical limitations of our machine learning framework tensorflow. However, because our reduced space is small, this does not pose a significant performance disadvantage. In future work we hope to implement analytical jacobian calculation.

Another issue we encountered was lack of convergence in that autoencoder space. To combat this, we fixed the maximum L-BFGS iteration count to 5. This provided satisfactory performance.

3.4 Stress Analysis

For the stress analysis and visualization, we compute the von Mises stress from the Cauchy stress computed in the full space.

4 RESULTS AND DISCUSION

For performance comparison we ran our simulation on a 1908 vertex bar fixed at one end. The Young's modulus was set to 3×10^5 and the Poisson's ration to 0.45.

Method	# DOF	Step speed	Quality
Full space	1908	500ms	No distortion
Linear space	10	30-40ms	Minimal distortion
Linear space	3	15-25ms	Large distortion
Autoencoder	**3**	**20 − 30ms**	**Some distortion**

Figure 3: A comparison of the performance of various reduced spaces.

In figure 4 one can see the inaccurate stress field results produced by using too few degrees of freedom in a linear space.

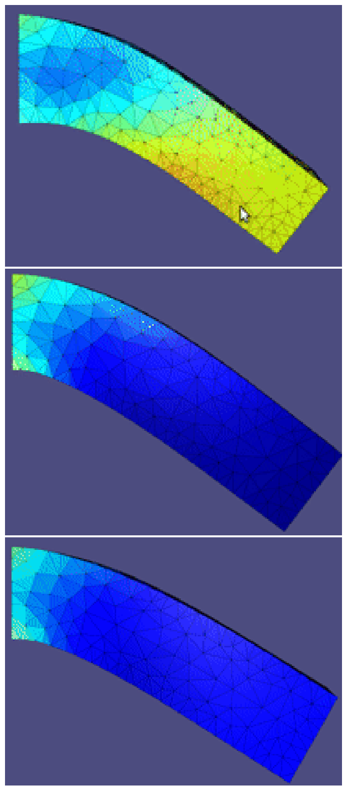

Figure 4: Top: 3dof PCA subspace, Middle: Full space, Bottom: 3dof Autoencoder space

5 CONCLUSION

We have demonstrated the effectiveness of non-linear subspace simulation for real time stress field analysis. With our unoptimized implementation, we are able to match state of the art results in linear subspace simulation using far fewer degrees of freedom.

5.1 Limitations and Future Work

The quality of the reduced space learned by our autoencoder is still one of the primary issues. We hope that by implementing dynamics aware regularization into the training process we will produce smoother mappings and improve the performance of the L-BFGS convergence.

We also hope to do away with finite differences and efficiently compute the analytical jacobian of our decoder.

REFERENCES

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

Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. 1994. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming* 63, 1 (01 Jan 1994), 129–156. https://doi.org/10.1007/BF01582063

Xiang Chen, Changxi Zheng, and Kun Zhou. 2016. Example-Based Subspace Stress Analysis for Interactive Shape Design. PP (10 2016), 1–1.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14

L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. 2015. Data-driven Fluid Simulations Using Regression Forests. *ACM Trans. Graph.* 34, 6, Article 199 (Oct. 2015), 9 pages. https://doi.org/10.1145/2816795.2818129

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Trans. Graph.* 36, 3, Article 23 (May 2017), 16 pages. https://doi.org/10.1145/2990496

Przemyslaw Musialski, Christian Hafner, Florian Rist, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2016. Non-linear Shape Optimization Using Local Subspace Projections. *ACM Trans. Graph.* 35, 4, Article 87 (July 2016), 13 pages. https://doi.org/10.1145/2897824.2925886

Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. 2017. Interactive Design Space Exploration and Optimization for CAD Models. *ACM Trans. Graph.* 36, 4, Article 157 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073688

J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. 2016. Accelerating Eulerian Fluid Simulation With Convolutional Networks. *ArXiv e-prints* (July 2016). arXiv:cs.CV/1607.03597

Christoph von Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. 2013. An Efficient Construction of Reduced Deformable Objects. *ACM Trans. Graph.* 32, 6, Article 213 (Nov. 2013), 10 pages. https://doi.org/10.1145/2508363.2508392

Huamin Wang and Yin Yang. 2016. Descent Methods for Elastic Body Simulation on the GPU. *ACM Trans. Graph.* 35, 6, Article 212 (Nov. 2016), 10 pages. https://doi.org/10.1145/2980179.2980236

Received February 2007; revised March 2009; final version June 2009; accepted July 2009