The Interface of Reduced Order Modeling and Deep Learning

Ziad Ghanem[1], S. Macrae Montgomery[2], Walker Powell[3]

The University of Texas at Dallas[1], Georgia Institute of Technology[2], North Carolina State University[3]

The Euler Bernouli Beam

Our system of interest is and Euler-Bernoulli beam fixed on the left and attached to an extensional spring on the right end. The beam has a length of 1 meter, with a harmonic forcing applied at 0.75 meters.

To mimic empirical data, the beam is simulated every .0002 seconds for a total of .2 seconds using an ABAQUS finite element model. Displacement data from 10 equidistant nodes along the beam is then gathered representing experimental sensors one might use in a field experiment.

Four datasets in total are generated, representing four distinct systems. The first dataset uses an extensional spring with a fixed linear stiffness varying the initial conditions. In the second dataset both initial conditions and the linear spring constant are varied. The third dataset again only varies the initial conditions but uses a fixed, cubic spring constant. In the final dataset both initial conditions and the cubic spring constant are varied.

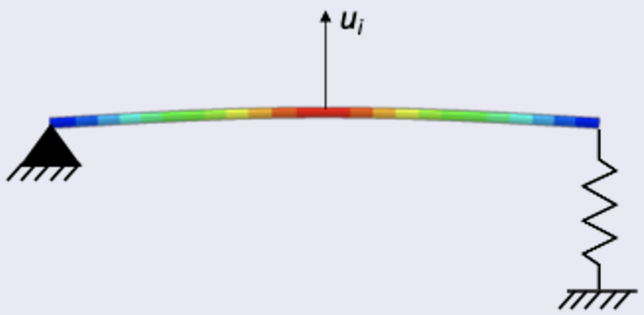

Figure: Forcing funtion $F = -\kappa u - \alpha u^3$

Motivation: Nonlinear Model Reduction

A major assumption of classical projection based model reduction techniques is that the dynamical system of interest evolves on some low dimension, linear subspace. This assumption raises a non-trivial problem for nonlinear systems for which a reduced model, as a linear approximation, is generally unable to achieve meaningful dimensionality reduction without sacrificing significant accuracy. If neural networks are able to learn the dynamics and predict the trajectory of ROM error they could prove useful to making nonlinear ROMs more feasible.

Acknowledgements

This research was conducted at the 2020 Nonlinear Mechanics and Dynamics Research Institute hosted by Sandia National Laboratories and the University of New Mexico.

Projection Based Model Order Reduction Framework

Suppose our dynamical problem can be expressed as an n^{th} order inhomogenous system of equations.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t). \end{aligned} \tag{1}$$

With state vector, $\mathbf{x} \in \mathbb{R}^n$, inputs, $\mathbf{u} \in \mathbb{R}^m$, outputs, $\mathbf{y} \in \mathbb{R}^q$, and state-space matrices, $\mathbf{A} \in \mathbb{R}^{nxn}$, $\mathbf{B} \in \mathbb{R}^{nxm}$, $\mathbf{C} \in \mathbb{R}^{qxn}$.

The objective of projection based model reduction is to project the governing equations of this dynamical system onto a low dimensional ($r \ll n$) subspace, obtaining a surrogate model which maps the same inputs, $\mathbf{u}$, to a set of outputs $\mathbf{y_s}$ which are close to $\mathbf{y}$ for all $\mathbf{u}$.

$$\begin{aligned} \dot{\mathbf{x}}_r(t) &= \mathbf{A}_r\mathbf{x}_r(t) + \mathbf{B}_r\mathbf{u}(t) \\ \mathbf{y}_r(t) &= \mathbf{C}_r\mathbf{x}_r(t). \end{aligned} \tag{2}$$

Now, with state vector, $\mathbf{x}_r \in \mathbb{R}^r$, and state-space matrices, $\mathbf{A}_r \in \mathbb{R}^{rxr}$, $\mathbf{B}_r \in \mathbb{R}^{rxm}$, $\mathbf{C}_r \in \mathbb{R}^{qxr}$.

Proper Orthogonal Decomposition

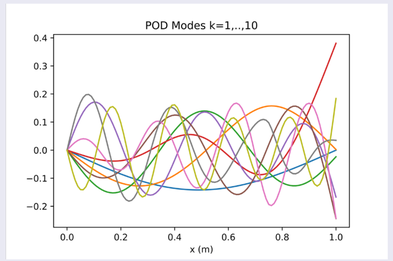

Figure: 10 POD Modes

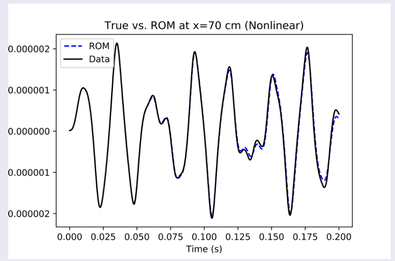

Figure: True Vs. ROM Displacement

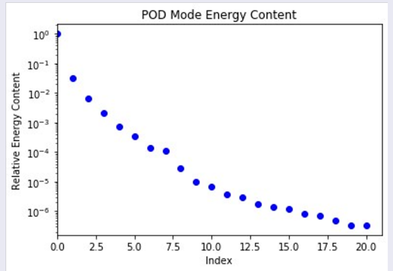

Figure: POD Mode Energy Content

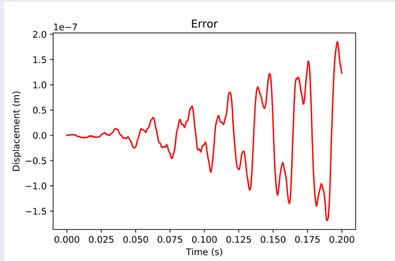

Figure: ROM Error

Forcasting ROM Error with Recurrent Neural Network

Recurrent Neural Networks (RNNs), like Feedforward Neural Networks (FFNs) are named after the manner in which information travels through the network. In a FNN, information travels straight through never returning to the same node twice. RNNs, on the other hand, are composed of cells rather than nodes and information recursively cycles through the same cell each time updating the so-called 'hidden state' of the cell.

The sequential architecture of RNNs allows knowledge of previous states to inform the likelihood of future states by archiving the weights and parameters of earlier cells to share with cells that come later in the sequence. This results in an emmergent property analagous to human memory so that an RNN is able to learn the transitions between states.

Instead of just predicting the error, we were able to have our network to generate a statistical model by using a negative log likelihood loss function and parameterizing a normal distribution, returning both a mean and variance.

Results

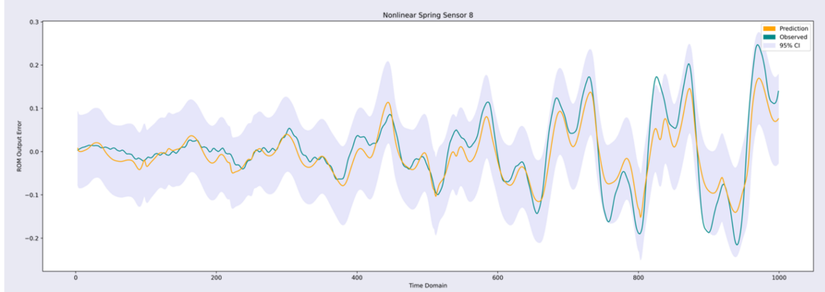

Figure: Nonlinear Spring Sensor 8, Predicted Error vs True Error with 95% CI

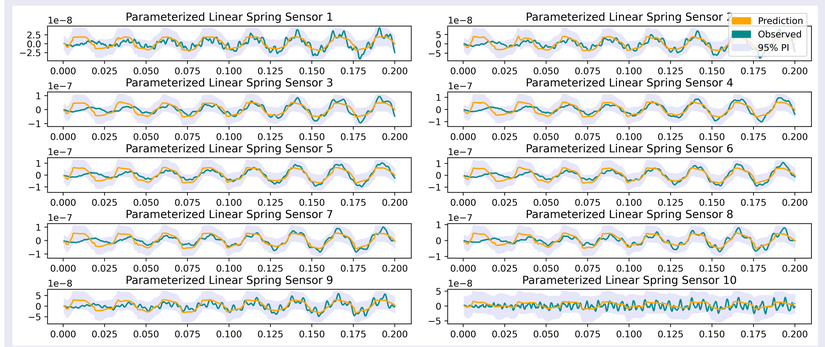

Figure: Linear Spring with varying κ, All Sensors, Predicted Error vs True Error with 95% CI