16th World Congress in Computational Mechanics (WCCM)
21-26 July 2024, Vancouver, Canada
A. Korobenko, M. Laforest, S. Prudhomme, R. Vaziri (Eds)

DATA-DRIVEN MODELING OF COMPLEX MECHANICAL COMPONENTS FOR INTEGRATION IN SYSTEM-LEVEL SIMULATIONS

Simon Vanpaemel[1,2,3], **J. Nathan Kutz**[2,4] **AND Steven L. Brunton**[2,5]

[1] Department of Mechanical Engineering, KU Leuven
Leuven, Belgium

[2] AI Institute in Dynamic Systems
Seattle, WA 98195, USA

[3] Flanders Make@KU Leuven
Leuven, Belgium

[4] Department of Applied Mathematics and Electrical Engineering, University of Washington
Seattle, WA 98105, USA

[5] Department of Mechanical Engineering, University of Washington
Seattle, WA 98105, USA

Key words: Data-driven modeling, nonlinear dynamics

Abstract. This contribution presents a data-driven approach featuring a physics-inspired neural network structure for modeling complex components in mecha(tro)nic systems. In the present approach, gated recurrent units (GRUs) are employed to approximate the ordinary differential equations (ODEs) describing the system's states over time, followed by a deep feedforward neural network (FFNN) mapping these states to a target variable. The networks are shown to predict a latent space capable of modeling the underlying dynamics, without the need for measuring the full state vector and only relying on input-output measurements. Subsequently it is shown that a nonlinear coordinate transformation exists between the latent space of the network and the states obtained from the reference ODE integration (simulation). To have a verification of the network's performance, it is applied to simulation-based data of an academic example for which the states and equations are known beforehand. Furthermore, the methodology is also applied to real measurement data from an INSTRON testing system capturing shock damper and bushing dynamic behaviour. Lastly, it is demonstrated that an ODE expression can be extracted from the trained network. This feature allows seamless integration of these networks into variable time-step, system-level simulation software.

1 INTRODUCTION

Mecha(tro)nic systems often contain complex nonlinear mechanical components and phenomena like bushings, shockdampers and friction contact. Although these elements are typically very locally introduced within the system, they have a large effect on the full system-behaviour and critical for performance, NVH, energy-efficiency and much more. Unfortunately, accurately and efficiently including these elements in system-level modeling approaches has proven to be challenging. High-fidelity modeling approaches, such as nonlinear finite element analysis, offer precise analysis of these components. However, this requires specific expert-knowledge to setup, and their computational cost often prohibits their inclusion in broader systems-level simulations, particularly for complex industrial systems featuring numerous such elements. Conversely, the approximation of these complex elements through idealized models and low-fidelity approaches proves challenging as these models often lack the required accuracy, leading to suboptimal analyses of the system's dynamic behavior.

The focus of this work lies on constructing accurate and computationally efficient models leveraging few input-output data signals. This type of data is often the most straightforward and least expensive to obtain in measurement campaigns, rather than measuring all the system's state variables which are also often not observable. Many of these approaches can be labeled as 'black-box' methods, which often lack robustness, generalizability (i.e. ability to predict outside of its training data) and interpretability (i.e. ability to provide (physical) insights in the data-driven model). Multiple methods exist that bring in first-principle modeling aspects into the network structure. For example, specific network architectures can be used that are suited to model dynamic systems, such as recurrent neural networks (RNNs) [1, 2, 3, 4, 5, 6, 7, 8]. RNNs are not directly inspired by physics per se, but they do share some conceptual similarities with dynamical systems, making them well-suited for modeling dynamic systems. RNNs, like dynamic systems, operate over time and maintain an internal state that evolves as new inputs are processed. This internal state can be thought of as capturing the system's memory or history of past inputs, similar to the state variables in dynamic systems. RNNs are inherently nonlinear systems, and their behavior can exhibit complex and nonlinear dynamics similar to those found in physical systems. This allows RNNs to model a wide range of phenomena and capture complex patterns in sequential data. In contrast, regular feedforward neural networks (FFNN) don't maintain an internal state and hence struggle to capture long-term dependencies or complex temporal patterns in time-series data. FFNNs are therefore less suited for multistep predictions [9]. Recurrent networks have been successfully used for applications where limited sensors are available [10, 11]. Additionally, many of these approaches can be combined with symbolic regression tools to discover the dynamic equations behind the data [12, 13]. This can further improve the interpretability of data-driven models.

In this contribution, we present a data-driven approach featuring a physics-inspired network architecture. Although either (deep) recurrent or feedforward neural networks have successfully been used to model dynamic systems, the structure of the presented network combines both in order to follow the rationale of the classical first-principle based modeling approaches. Specifically, Gated Recurrent Units (GRU) are used to approximate the ODEs and deep feedforward networks (FFNN) are used for the measurement equations. The GRU layer is kept as simple as possible with only one layer in order to directly extract the ODE expression through the GRU-ODE approximation. This makes it straightforward to include in variable-timestep system-level simulation solvers. A complete symbolic expression is feasible by applying symbolic regression tools to the measurement equation. It is illustrated that these networks can be constructed with a certain latent space that is linked through a nonlinear coordinate transformation to the physics-based states which are obtained from a reference simulation. Because the GRU network is able to capture history information, it does not require the measurement of the full state vector for training, which is beneficial as in many cases this is not available from the standard input-output measurements that are typically available of these components. This approach can be applied to (high-fidelity) simulation-based data as well as real world measurement data. To have a verification of the network's performance, it is applied to simulation-based data of an academic example for which the states and equations are known beforehand. Furthermore, the methodology is also applied to real measurement data from an INSTRON testing system capturing bushing dynamic behavior.

This paper is organised as follows; Section 2 described the Methodology of the approaches. Section 2.2 describes the integration of this approach in system-level simulation approaches. Lastly, Section 3 verifies the proposed methodlogy's performance, it is applied to simulation-based data of an academic example for which the states and equations are known beforehand. Furthermore, the methodology is also applied to real measurement data from an INSTRON testing system capturing bushing dynamic behavior.

2 Methodology

The aim is to construct a universally applicable and relatively simple network architecture that follows the rationale of a first-principle modeling approach for dynamic systems. A well established method to describe dynamic elements is via a nonlinear state-space approach:

$$\dot{\mathbf{h}} = \mathbf{f}\left(\mathbf{h}, \mathbf{x}\right) \tag{1a}$$

$$\mathbf{y} = \mathbf{m}\left(\mathbf{h}, \mathbf{x}\right). \tag{1b}$$

In this context, $\mathbf{h}$ represents the state variables whose temporal evolution is governed by ordinary differential equations (ODEs) (1a). The system is initiated in a specific state $\mathbf{h_{init}}$ and can be excited by inputs $\mathbf{x}$ over time. Engineers are typically interested in specific quantities or target variables $\mathbf{y}$, which are obtained through measurement equations $\mathbf{m}$ that relate these quantities to the system's states $\mathbf{h}$ and inputs $\mathbf{x}$.

2.1 Network architecture

Having a structure that naturally follows first-principle models can increase the physical interpretability and generalizability of those networks. Therefore we propose a network architecture that, similarly to a state-space model of a dynamic system, can be devided in two main parts. The first part is a recurrent network that describes the evolution of the states

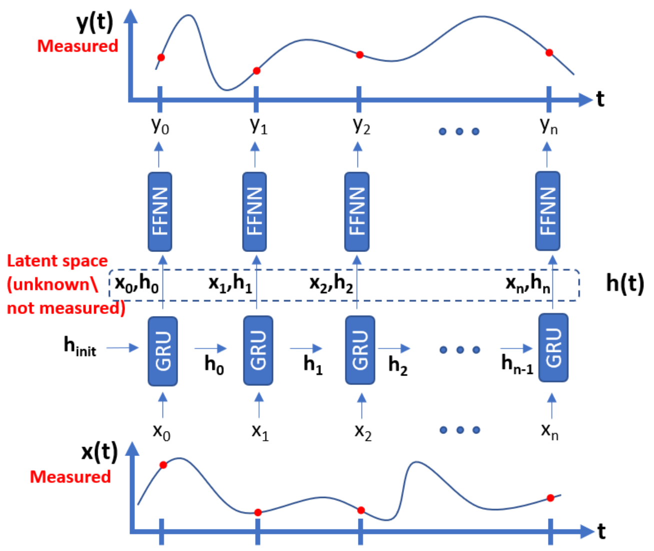


Figure 1: Scheme of GRU-FNN framwork

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{ir}\mathbf{x}_t + \mathbf{b}_{ir} + \mathbf{W}_{hr}\mathbf{h}_{(t-1)} + \mathbf{b}_{hr}\right) \tag{2a}$$

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{iz}\mathbf{x}_t + \mathbf{b}_{iz} + \mathbf{W}_{hz}\mathbf{h}_{(t-1)} + \mathbf{b}_{hz}\right) \tag{2b}$$

$$\mathbf{n}_t = tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{b}_{in} + \mathbf{r}_t \odot (\mathbf{W}_{hn}\mathbf{h}_{(t-1)} + \mathbf{b}_{hn})) \tag{2c}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{n}_t + \mathbf{z}_t \odot \mathbf{h}_{(t-1)} \tag{2d}$$

over time, similar to the ODEs (1a). More specifically, the GRU network [14] is chosen as recurrent network type, similar to LSTMs it has an improved performance for learning over longer time horizons [15]. Furthermore, for GRUs it is sufficient to keep track of the internal states over time, while the LSTM networks also keep track of a cell state over time. This aspect renders LSTMs less straightforward for ODE approximation compared to GRUs. The second part is a deep feedforward neural network that maps the states to the target variables, similar to the measurement equations (1b). The proposed network structure is visualized in Figure 1. The equations behind the GRU network are given in equations (2), where $\mathbf{W}_{ir}, \mathbf{b}_{ir}, \mathbf{W}_{hr}, \mathbf{b}_{hr}, \mathbf{W}_{iz}, \mathbf{b}_{iz}, \mathbf{W}_{hz}, \mathbf{b}_{hz}, \mathbf{W}_{hn}$ and $\mathbf{b}_{hn}$ denote the weight and bias matrices to be learned.

The measurement equation is a straightforward deep feedforward neural network with Rectified Linear Unit (ReLU) activation layer.

The dimension of the GRU network's internal state will depend on the complexity of the problem. Furthermore, it is assumed that the sampling timestepsize of the measured signals $x(t)$ and $y(t)$ between the observations are constant, which is typically the case for measurement data. If not, than one can easily (down)sample the respective signals to the appropriate timestep size.

Furthermore, the selection of the sampling timestep size Δt is also an important parameter, which should be chosen based on the temporal resolution of the time series data. If the data is sampled at high frequencies it may capture fine-grained temporal patterns. Conversely, if the data is sampled at lower frequencies, a larger timestep size may be more appropriate to capture broader trends. The most notable method for training recurrent networks is (truncated) backpropagation through time ((T)BPTT) [16], an additional parameter. In the 'truncated' case, one also needs to determine the length of the truncated sequence over which gradients are computed during training. The length of the sequence is a trade-off between the computational cost and the ability to learn more long-term dependencies more accurately in the data. In this work the states have been initialized to be zero. Nevertheless, interesting literature is available on initializing the state of the problem [6].

2.2 Integration in system-level simulation software

Complex mechanical components (e.g. bushings, shockdampers, etc.) are often part of a larger mecha(tro)nic system which can be modelled and simulated on a system-level through (flexible) multibody simulations. One of the main objectives of this work is to make sure that the proposed neural network, described in Section 2.1, can be easily integrated in system-level simulation environments. State-of-the-art simulation tools will employ variable-timestep size integrators. This means that additional interpolation techniques will need to be implemented such that the recurrent network receives inputs and generates outputs consistently for different timestep sizes. However, this comes with drawbacks as such interpolation schemes can be complex and can affect the accuracy of the predictions. Therefore an ODE approximation of GRU networks, denoted GRU-ODEs [17], are employed in this work (Section 2.2.1). The resulting ODEs that describe the evolution of the states of the nonlinear component can then be solved simultaneously with the differential equations of the complete mecha(tro)nic system. This simplifies the simulation workflow and there is no need to use any interpolation schemes.

The deep feedforward neural network that represents the measurement equation can directly be implemented using the layers' trained weights and ReLU functions.

2.2.1 ODE-approximation of the GRU network

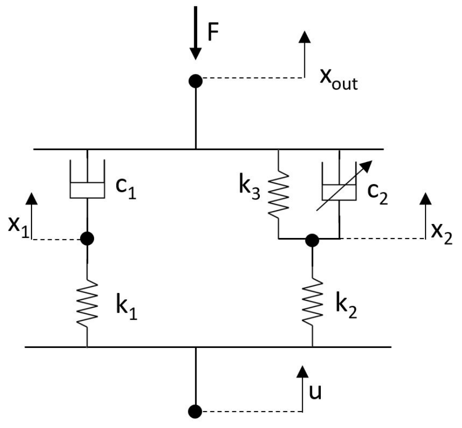
The GRU cell computes the current internal state $\mathbf{h}_t$ based on the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{(t-1)}$, and equations (2). This can be seen as the discretization using a forward Euler numerical integration scheme of the differential equations in a state-space description. Hence, the ODE description of the GRU cell can then be approximated as follows:

$$\dot{\mathbf{h}}_t \approx \frac{\mathbf{h}_t - \mathbf{h}_{(t-1)}}{\Delta t} = \frac{(1-\mathbf{z}_t) \odot \mathbf{n}_t + (\mathbf{z}_t - 1) \odot \mathbf{h}_{(t-1)}}{\Delta t} \tag{3}$$

which can be directly obtained and implemented from the trained weights of the GRU layer.

Furthermore, as indicated in [17], the advantages of using GRU-ODE: boundedness, as the internal state stays within $[-1, 1]$ range, making it robust against numerical errors. And it is Lipschitz continuous with constant $K = 2$, which has shown improved time series forecasting with low sample sizes. The continuity prior, designed to capture and enforce continuity properties in the latent state of the network, embedded in GRU-ODE is crucially important as it provides important prior information about the underlying process. This means that the GRU-ODE approach encourages the learned latent representations to exhibit smooth and continuous behavior.

The sampling timestep size Δt that is used during training of the network will have an effect on the accuracy of the ODE approximation. A too coarse timestep size can lead to poor



$$\begin{cases} \dot{x}_1 &= \frac{k_1}{c_1}\left(u - x_1\right) \\ \dot{x}_2 &= \left(\frac{k_2(u-x_2)-k_3x_2}{c_2}\right)^{\frac{1}{3}} \end{cases} \tag{4}$$

$$\begin{aligned} F &= c_1\dot{x}_1 + c_2\left(\dot{x}_2\right)^3 + k_3x_2 \\ &= k_1\left(u - x_1\right) + k_2\left(u - x_2\right) \end{aligned} \tag{5}$$

Figure 2: Spring-damper system

prediction of the ODE performance. Hence, after choosing a certain sampling timestep size, it is good to check both the predicion of the network, and compare the result with the GRU-ODE approximation.

3 Results

The methodology developed in Section 2 will be applied on two application cases. Firstly, in Section 3.1, it will be deployed on simulation data generated by a lumped parameter spring-damper model. This will serve as a verification of the methodologies' performance because the differential equations and states are known beforehand. Next, in Section 3.2, it will be applied to real measurement data of a rubber bushing that is captured using an *INSTRON E10000* test bench.

3.1 Lumped parameter spring-damper model

The schematic representation of the lumped parameter spring-damper system is given in Fig. 2. The system has four states denoted by u, x_1, x_2, x_3 and x_{out}. The state u represents the input-displacement that is applied to the system. For this case, the displacement x_{out} is set to be zero through the application of the 'reaction' force F. Hence, this reduces the remaining states to 2 because the constraints $x_{out} = 0$ and input $u(t)$ will be immediately applied in the system of differential equations that describe the system's dynamics. The spring elements are linear springs with stiffness k_1, k_2, k_3, the dashpot connected to the displacement x_1 is a linear damper. The nonlinear force through the dashpot connected to x_2 equals $c_2\left(x_2\right)^3$. This results in the system of differential equations, given by equations (4). Lastly, the constraint reaction force F which keeps $x_{out} = 0$ is given by equation 5.

In this example, the parameter values are set to $k_1 = 37.5$, $k_2 = 12.5$, $k_3 = 2.5$, $c_1 = 0.1$, $c_2 = 1e^{-6}$. The training data for this example has been generated by 20 simulation runs. Each of which is the initial condition $\begin{bmatrix} x_1 & x_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$. The input u for each simulation run is a summation of 10 sine waves with a random frequency component between 2 and $150\ Hz$, and a random amplitude between 0.1 and 1. All simulation runs simulate the behaviour for 1 second.

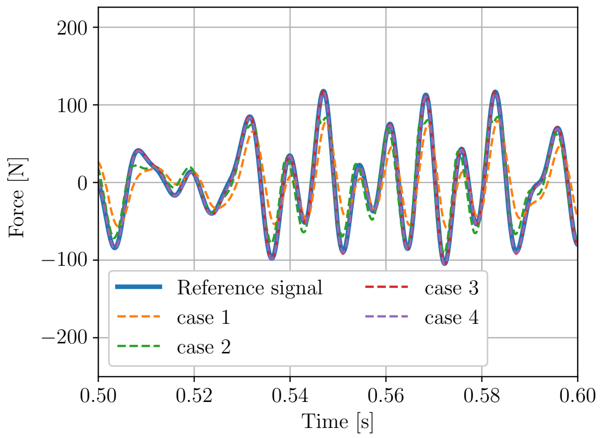


Figure 3: Prediction of the Neural network for varying latent space sizes

Case	Input + latent space	Loss	Loss/L_0
case 1	u	19418.48	$3.854e^{-1}$
case 2	u,h_1	8052.29	$1.598e^{-1}$
case 3	u,h_1,h_2	71.21	$1.414e^{-3}$
case 4	u,h_1,h_2,h_3	62.61	$1.243e^{-3}$

Table 1: Loss on training data for varying latent space size

The solution of the problem is evaluated at a timestep ΔT of $1e^{-4}$ seconds, and is chosen to be able to capture the highest dynamic content in the signal.

The GRU-FFNN network 2.1 is trained for the problem above, with the discretized time signal u as input to the network, and the discretized force signal F as output of the model. It can be seen from equation 5 that the output, next to the input u, depends on the two states x_1 and x_2. Hence we would assume that if the latent space of the GRU-FFNN network resembles the states of the system, then selecting a space $\mathbf{h} \in \mathbb{R}^2$ with two latent states is the minimum dimension needed to accurately model this system. This aspect will be in more detail discussed in the next Section 3.1.1. The FFNN network consists of 6 layers, with subsequent dimensions: 256, 128, 64, 32, 16 and 1.

3.1.1 Effect of latent space size in the GRU-FFNN network

This section will analyse the prediction accuracy of the GRU-FFNN network for a change in dimension of the latent space $\mathbf{h}$. More specifically, the network will be trained for three different cases. The first case is with a latent space size of zero, hence the output is only dependent on the input u, basically only FFNN is available which maps the inputs to outputs. The second case is with a latent space size of one, $\mathbf{h} \in \mathbb{R}^1$. The third case is a latent space size of 2, $\mathbf{h} \in \mathbb{R}^2$. Lastly, a fourth case with a latent space size of 3, $\mathbf{h} \in \mathbb{R}^3$. For each case, the learning phase of the model is run for 500 epochs. The results are summarized in Table 1, where $L_0 = 50376.42$ represents the loss at the start of the learning process. It can be seen that the training and testing loss of the network continues to decrease significantly until a latent space size of 2. Hence, increasing the latent space beyond two does not significantly improve the prediction behaviour as is expected for this case. The prediction of the force signal given is given in Figure 3 for a randomly selected part of the training data.

3.1.2 Coordinate transformation of latent space towards physical ODEstates

The idea is that if the latent states of the network can represent the output sufficiently well, then it must contain information which can be traced back to the physical states of the first-principle based ODEs. Section 3.1.1 seems to indicate that the latent space contains the same information as the states which were obtained from the integration of the ODEs, as it can predict the system's output based on the states at each time instant. To assess this, an auto-encoder network is trained that resembles a nonlinear map between both coordinate systems.

A random test sequence has been generated with random amplitudes between [0.1-0.3] and random frequencies between [2-350] Hz. The mapping predicts the ODEstates x_1 and x_2 using the latent space of the network; and vice versa. The predictions are shown in Figure 4. This has also been analysed for other amplitudes and frequency ranges, and the general conclusion is that the prediction is better when predicting the latent space of the network using the ODE states, compared to the other way around. Furthermore, because the latent states of the GRU network are limited to be within $[-1, 1]$, the predictions are worse poorly when amplitudes of states and latent space exceed those of the training values, because the autoencoder is not trained for this regime and hence provides a poor map between the coordinate spaces. Nevertheless, a decent coordinate transformation can be found and applied between the two coordinate systems, as long as the data is within similar ranges of the training data. This indicates that the latent space contains similar information w.r.t. the physical states, but in another coordinate system.

3.2 Application to real measurement data on a rubber bushing

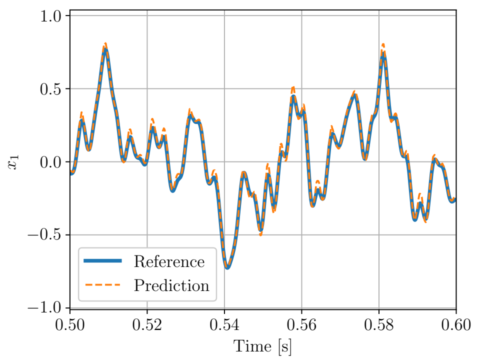
The methodology is also applied to measurement data captured from a rubber bushing. This rubber component was mounted in an *INSTRON E10000* test bench, as illustrated in Figure 5. During the test runs, the test bench applies an user-specified displacement signal using the linear actuator. The resulting force due to the applied displacement signal is measured through the setup's force cell.

3.2.1 Training

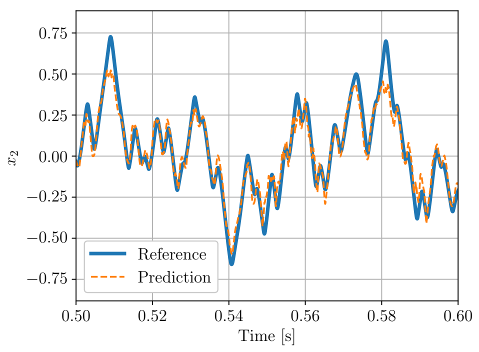
The training data is created by sequentially applying multiple sinusoidal input signals with a specified frequency $Freq$, each evaluated across varying amplitudes A. Each frequency and amplitude combination is also repeated for a certain amount of cycles. The specific combinations that are used in this work are given in Table2.

The training data was captured using different sampling frequencies, depending on the sinusoidal frequency of the input signal. Because the GRU-FFNN network requires a constant sampling timestep, and it was chosen to select the sampling timestepsize equal to $2e-3$ in order to capture sufficiently well the highest frequency content in the signal. Therefore, piecewise cubic hermite interpolating polynomials (PCHIP) were used to interpolate the measured input (displacement) and output (force) signals at the required sampling timestepsize.

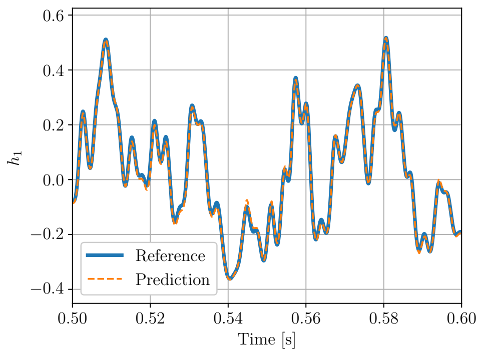
The size of the latent space is chosen to be 5, which was chosen by evaluating the prediction accuracy of the network. The size of the latent space was increased until the prediction was



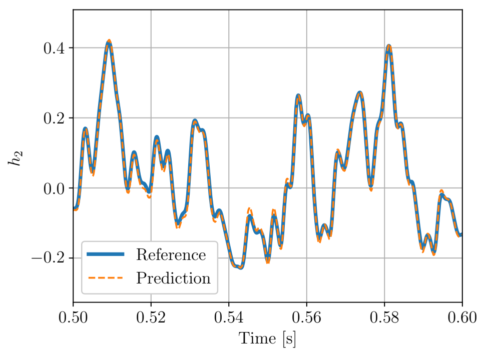
(a) x_1



(b) x_2



(c) h_1



(d) h_2

Figure 4: Nonlinear map between coordinate systems

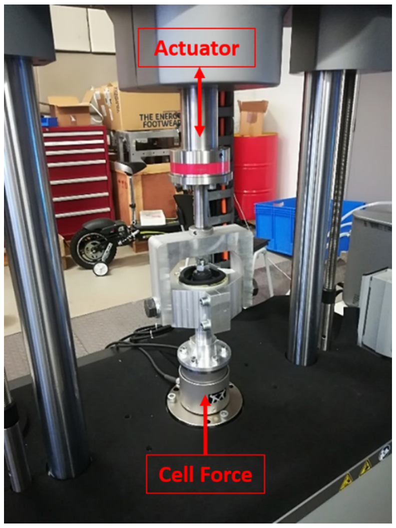
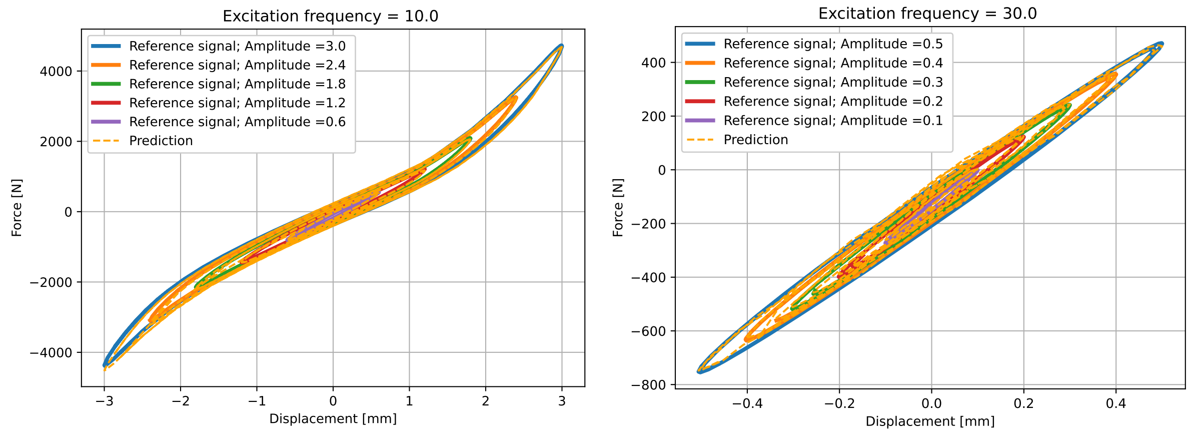


Figure 5: Bushing test bench

Freq $[Hz]$	A_1 $[mm]$	A_2 $[mm]$	A_3 $[mm]$	A_4 $[mm]$	A_5 $[mm]$	Cycles
1	0.7	1.4	2.1	2.8	3.5	50
2.5	0.7	1.4	2.1	2.8	3.5	125
5	0.7	1.4	2.1	2.8	3.5	250
7.5	0.6	1.2	1.8	2.4	3	375
10	0.6	1.2	1.8	2.4	3	500
12.5	0.5	1	1.5	2	2.5	625
15	0.5	1	1.5	2	2.5	750
17.5	0.4	0.8	1.2	1.6	2	875
20	0.4	0.8	1.2	1.6	2	1000
22.5	0.3	0.6	0.9	1.2	1.5	1125
25	0.2	0.4	0.6	0.8	1	1250
27.5	0.1	0.2	0.3	0.4	0.5	1375
30	0.1	0.2	0.3	0.4	0.5	1500

Table 2: Frequency and amplitude steps of measurement runs bushing



(a) 10 Hz (b) 30 Hz

Figure 6: Prediction on training data

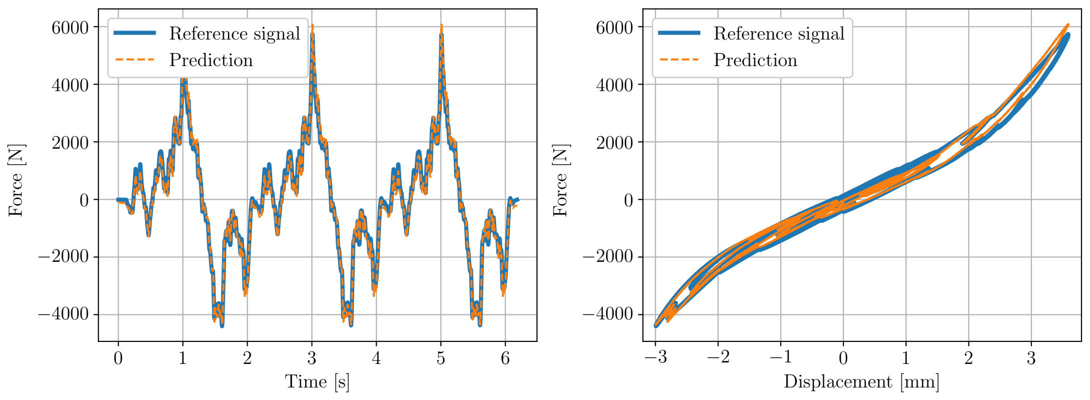
deemed to be sufficient to capture the most important dynamics of the larger part of the training data.

The network was trained using TBPTT with the length of the truncated sequence equal to 300, meaning that the gradients are computed over 300 timesteps, which spans 0.6 seconds. This was such that a large part of the lower frequencies could also be covered within this timespan.

3.2.2 Prediction on test and training data

The prediction of the network for a subset of the training data is given in Figure **??**. More specifically, the predictions for the frequencies 10 and 30 Hz are shown for their varying amplitudes that were given in Table 2. In each plot, the second to last cycle of the amplitude-frequency combination is visualized. It can be seen that the network can predict many of the amplitude-frequency combinations accurately. Nevertheless, the largest negative amplitudes at 1 Hz excitation signal have a relatively poor approximation. The latter could be improved by increasing the length of the truncated sequence, but this resulted in a worse prediction at other frequency-amplitude combinations. Increasing the latent space of the network will likely increase the accuracy of the prediction in general, but this was not done in order to prevent over-fitting. The objective of this Section is to illustrate the applicability of the network to real measurement data.

The test data is characterized by a randomly generated input displacement signal with a maximum amplitude of $3.6\ mm$ and frequency content up to $30\ Hz$ for ca 2 seconds. This cycle is repeated three times. The prediction on the test data is given in Figure 7. It can be seen that the GRU-FFNN network captures predicts well for multi-step time signals, more specifically the prediction was for ca 6 seconds which is approximately 3000 timesteps.



(a) Force (b) Force-Displacement

Figure 7: Prediction on test data

4 Conclusion and future work

This work presents physics-inspired network structured, denoted GRU-FFNN, to model complex (nonlinear) components that are typically part of a larger mecha(tro)nic system. The Gated Recurrent Units (GRU) are used to approximate the ODEs, and deep feedforward layers (FFNN) are used to mimick the measurement equations. Furthermore, the trained weights of the GRU layer can directly be used to extract an ODE-description. These differential equations can typically be included in many system-level simulation environments, and can be solved with state-of-the-art variable timestep simulation solvers. The main advantages of this approach is that it is a generic and straightforward method, to model a wide range of complex dynamic components and integrate them in system-level simulation. Relying on typically available input-output measurements. The proposed network architecture has been trained on simulation-based data force-displacement data of a nonlinear spring-damper system, where it is shown that the size of the latent space of the GRU network is a hyperparameter which should be tuned to the complexity of the problem. For this problem, the size of the latent space is chosen to be equal to the amount of state variables used to describe the ODE equations. Subsequently, an auto-encoder was trained to find the nonlinear map between both coordinate spaces. However, the prediction between the coordinate systems was relatively successful if the data was within the range of the training-data. The network is also trained on real measurement data from an INSTRON testing system capturing bushing dynamic behavior, where it is shown that a limited amount of latent states are sufficient for good prediction accuracy. This is especially beneficial in system-level simulation environments.

In future work we target to use regression tools for nonlinear dynamics such as SINDy [12], to retrieve symbolic equations that describe the evolution of the latent space over time. Future work on this method will focus on embedding physical constraints and methodologies in order to make sure that the latent states $\mathbf{h}$ can be represented as close as possible to the phyiscal ODE states, by incorporating any first-principle or physics based knowledge.

5 Acknowledgements

Internal Funds KU Leuven, the R. Snoeys Foundation and Research Foundation - Flanders (FWO-V404023N) are gratefully acknowledged for their support. This research was partially supported by Flanders Make, the strategic research center for the manufacturing industry. The authors also acknowledge support in part by the US National Science Foundation (NSF) AI Institute for Dynamical Systems (dynamicsai.org), grant 2112085. The authors would also like to thank Dr. Rocco Adduci for providing the measurement data on the rubber bushings.

References

[1] Ken ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993.

[2] Liang Jin, P. N. Nikiforuk, and Madan M. Gupta. Approximation of Discrete-Time State-Space Trajectories Using Dynamic Recurrent Neural Networks. *IEEE Transactions on Automatic Control*, 40(7):1266–1270, 1995.

[3] A. Delgado, C. Kambhampati, and K. Warwick. Dynamic recurrent neural network for system identification and control. *IEE Proc.-Control Theory*, 142(4):307–314, 1995.

[4] Coryn Bailer-Jones, David MacKay, and Philip Withers. A recurrent neural network for modelling dynamical systems. *Network: Computation in Neural Systems*, 9(4):531–547, 1998.

[5] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling q. *Pattern Recognition Letters*, 42:11–24, 2014.

[6] Nima Mohajerin and Steven L. Waslander. Multistep Prediction of Dynamic Systems with Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3370–3383, 2019.

[7] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[8] Bryan Lim, Stefan Zohren, and Bryan Lim. Time-series forecasting with deep learning : a survey. *Philosophical Transactions Royal Society A*, 379(2194), 2021.

[9] Simon Haykin. *Neural Networks - A Comprehensive Foundation*. Pearson Education, 2nd edition, 1999.

[10] Jan P. Williams, Olivia Zahn, and J. Nathan Kutz. Sensing with shallow recurrent decoder networks. *arXiv preprint arXiv:2301.12011*, 2023.

[11] Megan R. Ebers, Jan P. Williams, Katherine M. Steele, and J. Nathan Kutz. Leveraging arbitrary mobile sensor trajectories with shallow recurrent decoder networks for full-state reconstruction. *arXiv preprint arXiv:2307.11793*, 2023.

[12] Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, and William Bialek. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 113(15):3932–3937, 2016.

[13] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences of the United States of America*, 116(45):22445–22451, 2019.

[14] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *Proceedings of SSST 2014 - 8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] Ronald J. Williams and Jing Peng. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, 2(4):490–501, 1990.

[17] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. *arXiv preprint arXiv:1905.12374*, 2020.