Orthogonal Laguerre Recurrent Neural Networks

Sergio A. Dorado-Rojas
Department of Electrical Engineering
Rensselaer Polytechnic Institute
Troy, NY
dorads@rpi.edu

Bhanukiran Vinzamuri
IBM Research
Yorktown Heights, NY
bhanu.vinzamuri@ibm.com

Luigi Vanfretti
Department of Electrical Engineering
Rensselaer Polytechnic Institute
Troy, NY
vanfrl@rpi.edu

Abstract

The inability of RNN architectures to incorporate stable discrete-time linear time-invariant dynamics has been a long-standing problem that has also affected their performance in practice. To mitigate this problem, in this paper, we propose a RNN architecture that embeds a linear time-invariant system basis comprising of Laguerre polynomials inside its structure. Laguerre functions are a family of Eigenfunctions arising from the Sturm-Liouville problem characterized by their orthonormality. Our RNN uses the embedded full state-space representation provided by such orthonormal functions in two different variants: a general orthogonal Laguerre network and a Ladder network. The state of such systems is used as a means to preserve input information for more extended periods by an orthogonal encoding. Both variants are enhanced by a non-linear static output layer that projects the state, the memory, and the past output for the hidden state. The proposed models are benchmarked exhaustively against other dynamical systems inspired-RNN's on two physics-based benchmarks that demonstrate their better performance.

1 Introduction.

The introduction of Legendre Memory Units (LMUs)[1] represents a deep connection between recurrent neural networks (RNNs) and dynamical systems from the *design* point-of-view, despite some previous efforts to *analyze* the structure from the dynamical systems perspective [2, 3]. The memory update policy of LMUs is the state equation of a Discrete-Time Linear Time-Invariant (DTLTI) system described by $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u_k$ where $\mathbf{x} \in \mathbb{R}^{n\times 1}$ is the memory (state of the cell), u_k is the input to the memory gate, and the pair $(\mathbf{A}, \mathbf{B})$ corresponds to a discretized state-space representation of shifted Legendre polynomials, portrayed by n coupled ordinary differential equations. The orthogonal time-domain projection inside LMUs improves memory capabilities and reduces the number of trainable parameters. Nevertheless, LMUs *only* make use of the state equation. The output equation (i.e., $\mathbf{y}_k = \mathbf{C}\mathbf{x}_k$) does not appear within their structure. The use of this equation inside a RNN becomes even more relevant if the $(\mathbf{A}, \mathbf{B})$ pair represents an orthogonal family of functions like shifted Legendre polynomials. With an embedded full state-space representation based on orthogonal functions, a RNN would be able to learn *any* stable DTLTI dynamics [4]. In other words, the use of the output equation would enable the RNN to steer the behavior of some internal signal towards *any* energy-limited signal, encoded inside the DTLTI system represented by the state equation and the output equation. Thus, the network's generalization capabilities would be increased by including an orthonormal basis in the state equation and the explicit use of the output equation. In this paper, we study how Laguerre polynomials can be used as an orthonormal basis and propose two variants, Laguerre and Ladder RNN, respectively.

Laguerre functions are a family of Eigenfunctions arising from the Sturm-Liouville problem characterized by their orthonormality. The set of Laguerre functions $\ell^{(i)}(t)$ $(i = 1, 2, 3, \dots)$ are defined $\ell_1(t) = \left(\sqrt{2p}\right) e^{-pt}, \ell_2(t) = \left(\sqrt{2p}\right)(1 - 2pt) e^{-pt}, \dots, \ell_i(t) = \left(\sqrt{2p}\right) \frac{e^{pt}}{(i-1)!} \frac{d^{i-1}}{dt^{i-1}} \left[t^{i-1} e^{-2pt}\right]$ where p is a parameter called time scaling factor. Laguerre functions are a complete set over $[0, \infty)$. Thus, they can be used to reconstruct any $\mathcal{L}_2$ function $f(t)$ by means of a Generalized Fourier Series (GFS) expansion $f(t) = \sum_{i=0}^{\infty} c_i \ell^{(i)}(t)$ with c_i being the ith coefficient of f on the ith element of the Laguerre basis. Moreover, under mild assumptions on the function $f \in \mathcal{L}_2$, a GFS expansion with N Laguerre functions can be used to define an arbitrarily close approximation to the function as $\int_0^{\infty} \left(f(t) - \sum_{i=1}^{N} c_i \ell^{(i)}(t)\right) dt < \varepsilon$ for any $\varepsilon > 0$ [5]. This characteristic means that it is possible to represent dynamics whose characteristics are conveniently captured by an absolutely integrable signal using a finite number of Laguerre functions [6]. In other words, it is possible to represent any stable LTI dynamics by Laguerre polynomials.

Laguerre polynomials have a convenient mathematical representation in state-space [5]. Let $\mathbf{l}$ be the vector containing the first N Laguerre polynomials as functions of time ($\mathbf{l} = \left[\begin{array}{cccc} \ell_1(t) & \ell_2(t) & \dots & \ell_N(t) \end{array}\right]^T$). In particular, we are interested in the values at $t = 0$, that is, $\mathbf{l}(0) = \left[\begin{array}{cccc} \ell_1(0) & \ell_2(0) & \dots & \ell_N(0) \end{array}\right]^T$. Then, the Laguerre polynomials can be generated by $\mathbf{l} = e^{\mathbf{A}_\ell t}\mathbf{l}(0)$ where the matrix $\mathbf{A}_\ell$, referred to as Laguerre matrix, is parametrized in terms of the scaling factor p and $\mathbf{A}_\ell$ is a lower triangular matrix. This mathematical convenience supports its application in areas such as Model Predictive Control [7, 5]. The vector of initial conditions together with the Laguerre matrix can be used to constitute a state-space representation for any stable continuous time (CTLTI) system [5]. For simplicity, consider the single-input single-output case $\dot{\mathbf{x}} = \mathbf{A}_\ell \mathbf{x} + \mathbf{B}_\ell u$, $y = \mathbf{C}_\ell \mathbf{x}$ with $\mathbf{A}_\ell \in \mathbb{R}^{n \times n}$, $\mathbf{B}_\ell = \mathbf{l}(0)$, and $\mathbf{C}_\ell = [c_1 \; c_2 \; \dots c_n]$. $\mathbf{C}_\ell$ is a coefficient matrix that depends on the system being considered. This equation can be extended to multiple inputs by concatenating the initial condition vector along the columns, and computing the coefficient vector for each of the output channels. Either way, the matrices $\mathbf{A}_\ell$ and $\mathbf{B}_\ell$ are fixed for all stable CTLTI systems. The coefficient matrix which changes from system to system, represents the coefficients of specific dynamics on the Laguerre basis.

A DT Laguerre network $\ell^{(i)}(z)$ is generated from the discretization of a CT Laguerre network, which in turn can be defined as the Laplace transform of the corresponding Laguerre polynomial $\ell^{(i)}(s) = \mathcal{L}\left\{\ell^{(i)}(t)\right\}$. Let $\ell_k^{(i)} = \mathcal{Z}^{-1}\left\{\ell^{(i)}(z)\right\}$ be the inverse Z-Transform of $\ell^{(i)}(z)$ (i.e., the time-domain representation of the ith Laguerre network). DT Laguerre sequences are orthonormal over $k \in [0, \infty)$, so they can be employed as a basis to reconstruct any absolutely summable sequence (i.e., any stable linear time-invariant system or finite-energy deterministic signal). The transfer functions of the DT Laguerre networks are given by $\ell^{(1)}(z) = \frac{\sqrt{1-a^2}}{1-az^{-1}}, \ell^{(2)}(z) = \frac{\sqrt{1-a^2}}{1-az^{-1}}\left(\frac{z^{-1}-a}{1-az^{-1}}\right) \dots, \ell^{(N)}(z) = \frac{\sqrt{1-a^2}}{1-az^{-1}}\left(\frac{z^{-1}-a}{1-az^{-1}}\right)^{N-1}$ where a is a parameter known as *scaling factor* that represents the location of the Laguerre network pole [5]. Consequently, stability and non-alternating behavior requires $0 \leq a < 1$.

The Ladder network is founded upon a special case of Laguerre functions: discrete-time delays. The dynamic behavior of delays has received particular attention from the deep learning community, given their inherent long-term memory characteristics [8]. However, a practical implementation of a CT delay requires a rational function approximation, usually through Padé approximants as done in [8]. In contrast, DT delays have a simple transfer function representation (i.e., $G(z) = 1/z^m$ for an m-sample delay) and a finite-dimensional state-space representation which makes them suitable for an RNN implementation.

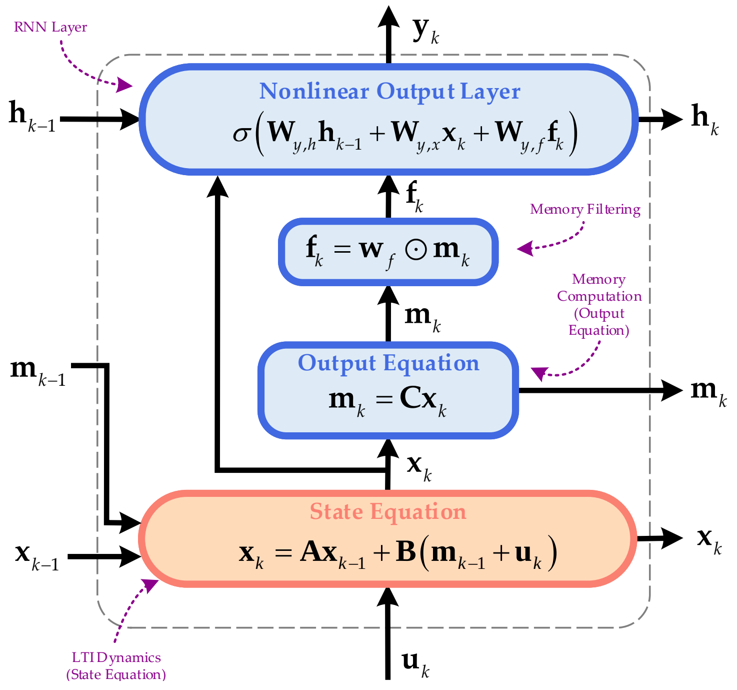
The proposed orthogonal architecture is detailed in Figure 1. The blocks with trainable weights are colored in blue (output equation, memory filtering, and nonlinear output layer). Note that the memory is constrained to have the same dimensions as the input so that it can be used to store input information in the output of the dynamical system. The design is interpreted as follows: consider an input with n_u features ($\mathbf{u}_k \in \mathbb{R}^{n_u \times 1}$). The state the output equation constitute a DTLTI system that processes the past memory $\mathbf{m}_{k-1}$, state $\mathbf{x}_{k-1}$, and the current input $\mathbf{u}_k$ to update the state $\mathbf{x}_k$ by $\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\left(\mathbf{u}_k + \mathbf{m}_{k-1}\right)$ where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times n_u}$ are discrete Laguerre matrices obtained after discretization of the continuous Laguerre matrices. n is the network order that corresponds to the number of Laguerre polynomials used to encode the DTLTI dynamics into the RNN. Once the state is updated, it is fed into the output equation to compute the memory update $\mathbf{m}_k$ through the learnable coordinate matrix $\mathbf{C}$. The memory is filtered elementwise by the local weight vector $\mathbf{w}_f$ to produce $\mathbf{f}_k$ which is fed into the nonlinear output layer.

The nonlinear output layer has been adapted from the one proposed by [9]. It produces n_y hidden outputs through $\mathbf{h}_k = \mathbf{y}_k \in \mathbb{R}^{n_y\times 1}$. The logits are computed by the linear combination of the previous hidden output $\mathbf{h}_{k-1}$, the updated state $\mathbf{x}_k$ and memory $\mathbf{m}_k$ through the weights $\mathbf{W}_{y,h} \in \mathbb{R}^{n_y\times n_y}$, $\mathbf{W}_{y,x} \in \mathbb{R}^{n_y\times n}$ and $\mathbf{W}_{y,f} \in \mathbb{R}^{n_y\times n_u}$. The logits are passed through a nonlinear activation function σ that counts with a bias term not shown in the diagram.

Laguerre and Ladder Networks: The Laguerre network uses a fixed pair $(\mathbf{A}, \mathbf{B})$ where $\mathbf{A} = \mathbf{A}_\ell$ and $\mathbf{B} = \mathbf{B}_\ell$. In other words, the RNN has to determine during training the coordinate matrix $\mathbf{C}$ that results most optimal to the task under consideration. For this architecture, the order n (i.e., the number of Laguerre polynomials) is left as a design hyperparameter. Further hyperparameters are the scaling factor p (default, $p = 1$), the sampling time ΔT used for discretization (default, $\Delta T = 1$ s), and the sampling method (default, zero-order hold).

The Ladder network employs discrete Laguerre networks with $a = 0$ to introduce DT delay dynamics inside the RNN. For this reason, the matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ are non-trainable (i.e., the coefficient matrix $\mathbf{C}$ is fixed). In fact, $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ correspond to a state-space realization for the Ladder system $\mathbf{G}\left(z\right) = \text{diag}\left(1/z^{m_i}\right) \in \mathbb{C}^{n_u\times n_u}$ where $m_1,\ m_2,\ \ldots,\ m_{n_u-1},\ m_{n_u}$ are the delays of each channel, fulfilling $m_1 < m_2 < \cdots < m_{n_u}$. Note that $m_{n_u} = m_d$, the maximum delay, is specified by the user. Using this input, the algorithm to construct the Ladder network in transfer function form distributes the delays $m_1, m_2, \ldots, m_{n_u-1}$ equally among all the input channels. For example, for a system with 100 inputs and a maximum delay of 100, $\mathbf{G}(z)$ has the form of a diagonal matrix with the element in d^{th} row and column being $1/z^{m_d}$.

Once the matrix $\mathbf{G}\left(z\right)$ is constructed, it is transformed into controllable-canonical form. The resulting matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ are set inside the RNN architecture (Figure 1) to construct the Ladder variant.


RNN Layer
$\mathbf{y}_k$
$\mathbf{h}_{k-1}$
Nonlinear Output Layer
$\sigma\left(\mathbf{W}_{y,h}\mathbf{h}_{k-1} + \mathbf{W}_{y,x}\mathbf{x}_k + \mathbf{W}_{y,f}\mathbf{f}_k\right)$
$\mathbf{h}_k$
$\mathbf{f}_k$
Memory Filtering
$\mathbf{f}_k = \mathbf{w}_f \odot \mathbf{m}_k$
$\mathbf{m}_k$
Memory Computation (Output Equation)
$\mathbf{m}_{k-1}$
Output Equation
$\mathbf{m}_k = \mathbf{C}\mathbf{x}_k$
$\mathbf{m}_k$
$\mathbf{x}_k$
State Equation
$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\left(\mathbf{m}_{k-1} + \mathbf{u}_k\right)$
$\mathbf{x}_{k-1}$
$\mathbf{x}_k$
$\mathbf{u}_k$
LTI Dynamics (State Equation)
Figure 1: Detailed Orthogonal Laguerre Recurrent Neural Network Architecture.

2 Experiments

We validated the performance of our RNN architectures in two dynamical systems-based experiments. In the time-series prediction benchmark, the goal is to predict a nonlinear system's behavior from a set of available measurements [10]. For the system identification experiment, the learning problem aims at determining the state-space representation of a system from data as a regression task. Both experiments are performed for a pendulum with no friction and a fluid flow system.

Implementation

All models have been implemented in Tensorflow using the keras API. Training settings were adapted from those implemented by [1] and [11]. Experiments were executed on three different machines with GPU capabilities (1080 GTX, Titan RTX and Quadro RTX), all running on Ubuntu 18.04.3 LTS.

Pendulum

The pendulum dynamics are represented as $\frac{d^2\theta}{dt^2} + \frac{g}{\ell}\sin\theta = 0$ with $g = 9.8$ and $\ell = 1$ [10]. In this case, the initial condition was changed so that the system will switch between linear ($\theta_0 = 0.8$) and nonlinear operation ($\theta_0 = 2.4$). Notice that this pendulum equation considers no friction. A state-space representation with $x_1 := \theta$ and $x_2 := \dot{\theta}$ is given by $\dot{x}_1 = x_2,\ \dot{x}_2 = -\frac{g}{\ell}\sin x_1$. The data is generated by integrating the pendulum equations directly using a numerical method in the interval $[0, 170]$ s with $\Delta t = 0.1$ s. For system identification, an input (torque) is added to the $\dot{x}_2$ equation as an input together with a friction term.

Fluid Flow System

This benchmark represents a nonlinear fluid flow past a circular cylinder at Reynolds number 100 [12]. The system of ODEs employed to generate the data is expressed as $\dot{x}_1 = \mu x_1 - \omega x_2 + A x_1 x_3 + u, \dot{x}_2 = \omega x_1 + \mu x_2 + A x_2 x_3, \dot{x}_3 = -\lambda\left(x_3 - x_1^2 - x_2^2\right)$ with $\mu = 0.1, \omega = 1, A = -0.1$ and $\lambda = 10$. The initial condition is taken randomly. Data is generated for the range $[0, 2]$ s with $\Delta t = 0.001$ s. For system identification, the equation for $\dot{x}_1$ is perturbed by the input u.

In regards to the forecasting experiment, in contrast to the approach in [10], the problem is formulated in a supervised framework where the training data batches are generated by the keras functionality `TimeseriesGenerator`. Splitting between training and testing data is maintained (35.29% for training, and 64.71% for testing). Results are reported for five different seeds in Table 1. The Laguerre architecture accomodates to both the linear and the nonlinear environment despite its intrinsic linear construction, whereas the Ladder network shows strong performance in noisy conditions.

Forecasting error is defined as the relative difference between the ground truth and the prediction at the final step $e = \|\hat{y}_p - y\|_2 / \|y\|_2$ where $\hat{y}_p$ is the forecasted output and y is the ground truth. The experiments are performed in noiseless and noisy conditions (Gaussian noise). Results are shown in Figure 2. The Laguerre RNN exhibits a lower variance across folds than Ladder and LMU thanks to the adjustment of its internal dynamics during training.

Table 1: Prediction error at final step averaged over 5-fold for the prediction experiment

Model	Pendulum (no noise; linear)	Pendulum (noisy; linear)	Pendulum (no noise; nonlinear)	Pendulum (noisy; nonlinear)	Fluid Flow (noisy)
Ladder	0.0150 ± 0.0050	**0.0116 ± 0.0018**	0.1819 ± 0.1041	0.1929 ± 0.1261	**0.5135 ± 0.0626**
Laguerre	**0.0146 ± 0.0029**	0.0168 ± 0.0057	**0.1249 ± 0.0060**	**0.1257 ± 0.0059**	0.7973 ± 0.5440
LMU [8]	0.2261 ± 0.0986	0.1914 ± 0.0817	0.5132 ± 0.2465	0.5012 ± 0.1133	1.3942 ± 0.0025
BRC [13]	21.8499 ± 1.8473	23.1749 ± 1.8363	14.0479 ± 4.1699	18.0374 ± 1.7233	0.6973 ± 0.1684
nBRC [13]	2.1914 ± 0.9910	1.7015 ± 0.4043	2.6853 ± 1.5883	3.0102 ± 1.0030	0.5656 ± 0.0481

System Identification Experiment: The system identification problem[1] has been previously tackled with supervised [14] and unsupervised learning approaches [10]. Our benchmark is constrained to

[1]See this GitHub repository for more details regarding data generation and pre-processing.

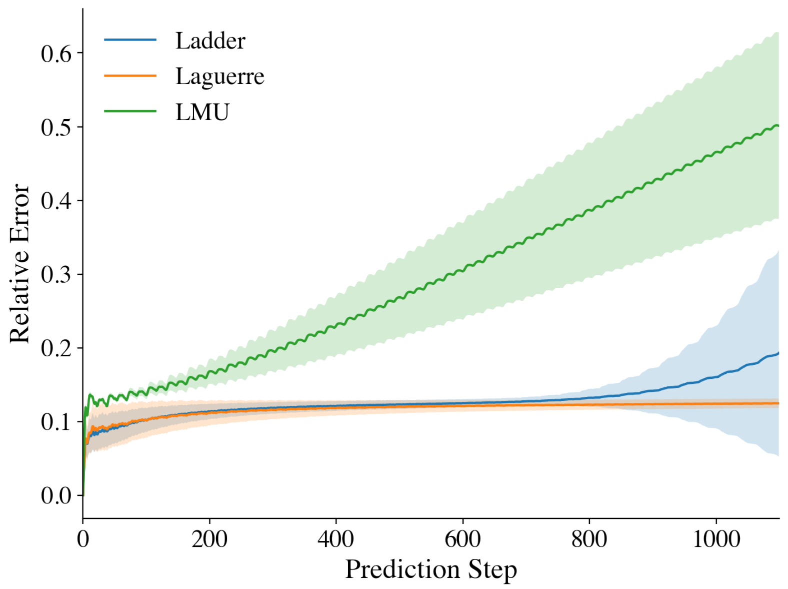


Figure 2: Relative prediction error for the noisy nonlinear pendulum experiment.

the dynamical system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u), \mathbf{y} = \mathbf{x}$ where u is the external input. Since we aim at estimating $\mathbf{y} = \mathbf{x}$, this can be considered a state estimation problem. The discrete version of the dynamics is given by $\mathbf{x}_{k+1} = \bar{\mathbf{f}}(\mathbf{x}_k, u_k), \mathbf{y}_k = \mathbf{x}_k$. A unit delay is applied to the state equation to bring it into RNN form ($\mathbf{h}_k = \sigma(\mathbf{u}_k, \mathbf{h}_{k-1})$). By doing so, we get $\mathbf{x}_k = \bar{\mathbf{f}}(\mathbf{x}_{k-1}, u_{k-1})$.

Considering sequences with n_{steps} time steps and a system with $n \in \mathbb{N}$ states, the state and the output of the discrete-time system are concatenated to construct the input $\mathbf{u}_{\text{RNN}} \in \mathbb{R}^{n_{\text{steps}} \times n}$. The first row of the input matrix is the transposed initial state vector $\mathbf{x}_0 = \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \dots x_0^{(n)} \end{bmatrix}^T$. The input values are placed in the first column since it is assumed that the RNN will deal only with a single output. Then, the RNN sees both the state and the exogenous signal u as inputs.

Training, testing and validation data (i.e., target output of the RNN $\mathbf{y}$) are generated by applying an input to the system and solving the underlying system of differential equations. For simplicity, we assume an arbitrary random initial state, although assuming the system to be initially relaxed (i.e., zero initial state) is a common practice. The input is taken to be a random signal so that most system modes are excited and can be detected in the output. RNN training is formulated as a regression problem using MSE as a loss function. For the pendulum (with friction) and the fluid flow systems, 500 instances are generated, 10% of which is used for validation. Performance is evaluated using averaged MAE with standard deviation over five-folds. Despite not being the best model for each experiment, our proposals show a balance between number of trainable parameters and performance.

Table 2: MAE averaged over 5-fold for the system identification experiment and number of trainable parameters

Model	**Pendulum**	**Fluid Flow**	**Trainable Parameters**
Ladder	0.0133 ± 0.0012	0.0206 ± 0.0095	**59,788**
Laguerre	0.0185 ± 0.0024	0.0037 ± 0.0001	100,792
LMU	$\mathbf{0.0074 \pm 0.0020}$	0.0166 ± 0.0003	100,536
BRC	0.0891 ± 0.0005	0.0040 ± 0.0009	4,802
nBRC	0.0067 ± 0.0065	$\mathbf{0.0014 \pm 0.0005}$	324,002

Conclusion: In this paper, we have presented our preliminary work on designing RNN architectures inspired by discrete-time dynamical systems. Laguerre polynomials were chosen as the orthonormal basis to incorporate the dynamics. Our proposed method provides better performance and also reduces the number of trainable parameters compared to LMU's.

Broader Impact

Modeling non-linear dynamics from real-world time series data is a very important problem which is highly relevant while monitoring asset health (for large assets such as steam turbines, energy transformers) in Industry 4.0. Being able to predict and identify when the performance of an asset will deteriorate can have large implications in reducing its effective downtime, and it can also help in reducing the associated carbon footprint. Incorporating physics-based principles into modeling such assets can be a more effective way of prediction, but it is also a more challenging problem. The ability of Ladder and Laguerre RNN's to learn stable DTLTI dynamics due to the encoded orthonormal basis functions helps in improving the performance over other RNN's while modeling dynamical systems. Lower number of training parameters for Ladder RNN ensure that the models built are more tractable, efficient and can be trained with limited resources.

Acknowledgments and Disclosure of Funding

This work was supported by the Rensselaer-IBM AI Research Collaboration (http://airc.rpi.edu), part of the IBM AI Horizons Network (http://ibm.biz/AIHorizons).

References

[1] A. R. Voelker, I. Kajic, and C. Eliasmith, "Legendre Memory Units : Continuous-Time Representation in Recurrent Neural Networks," in *Advances in Neural Information Processing Systems*, no. December, 2019, pp. 15 544–15 553.

[2] D. Haviv, A. Rivkind, and O. Barak, "Understanding and Controlling Memory in Recurrent Neural Networks," feb 2019.

[3] D. Sussillo and O. Barak, "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks," *Neural Computation*, vol. 25, no. 3, pp. 626–649, mar 2013.

[4] C. Hwang and M.-Y. Chen, "Analysis and parameter identification of time-delay systems via shifted Legendre polynomials," *International Journal of Control*, vol. 41, no. 2, pp. 403–415, feb 1985.

[5] L. Wang, *Model Predictive Control System Design and Implementation using MATLAB*. Springer, 2009.

[6] A. D. Back and A. C. Tsoi, "Nonlinear system identification using discrete Laguerre functions," *Journal of Systems Engineering*, vol. 6, no. 6, 1996.

[7] L. Wang, "Discrete model predictive controller design using Laguerre functions," *Journal of Process Control*, vol. 14, no. 2, pp. 131–142, mar 2004.

[8] A. R. Voelker, "Dynamical Systems in Spiking Neuromorphic Hardware," Ph.D. dissertation, University of Waterloo, 2019.

[9] S. Chandar, C. Sankar, E. Vorontsov, S. E. Kahou, and Y. Bengio, "Towards Non-Saturating Recurrent Units for Modelling Long-Term Dependencies," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3280–3287, jul 2019.

[10] O. Azencot, N. B. Erichson, V. Lin, and M. M. Mahoney, "Forecasting Sequential Data using Consistent Koopman Autoencoders," in *37th International Conference on Machine Learning*, 2020.

[11] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, jul 2019. [Online]. Available: http://link.springer.com/10.1007/s10618-019-00619-1

[12] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, no. 1, p. 4950, dec 2018.

[13] N. Vecoven, D. Ernst, and G. Drion, “A bio-inspired bistable recurrent cell allows for long-lasting memory,” jun 2020.

[14] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, “Deep State Space Models for Nonlinear System Identification,” mar 2020.