CONSTRAINED NEURAL ORDINARY DIFFERENTIAL EQUATIONS WITH STABILITY GUARANTEES

Aaron R. Tuor, Ján Drgoňa & Draguna L. Vrabie
Pacific Northwest National Laboratory
Richland, Washington, USA
{aaron.tuor,jan.drgona,draguna.vrabie}@pnnl.gov

ABSTRACT

Differential equations are frequently used in engineering domains, such as modeling and control of industrial systems, where safety and performance guarantees are of paramount importance. Traditional physics-based modeling approaches require domain expertise and are often difficult to tune or adapt to new systems. In this paper, we show how to model discrete ordinary differential equations (ODE) with algebraic nonlinearities as deep neural networks with varying degrees of prior knowledge. We derive the stability guarantees of the network layers based on the implicit constraints imposed on the weight's eigenvalues. Moreover, we show how to use barrier methods to generically handle additional inequality constraints. We demonstrate the prediction accuracy of learned neural ODEs evaluated on open-loop simulations compared to ground truth dynamics with bi-linear terms.

1 INTRODUCTION

Ordinary differential equations (ODE) have numerous applications in various engineering domains such as thermodynamics, mechanics, chemical processes, circuit design, and optimal control. However, the solution of the ODEs often require sophisticated numerical methods [19]. Popular ODE methods are implemented in Dymola, OpenModelica, MapleSim, Simulink, C, FORTRAN, or Julia and are mostly restricted to users with expert knowledge.

On the other hand, approaches that bring together deep learning, scientific computing, and differential equations aim to provide this capability to a broader class of users [8; 5; 15; 2; 9; 17; 11]. Physics-informed neural networks [17] train fully connected deep neural nets while embedding physics knowledge in the loss function. In this work we take a different approach by directly specifying the structure of the neural network to capture the physics, given initial and boundary conditions. Studies on stability, applying the formal analysis from dynamical systems to deep learning models derive interesting implications [6; 10; 13; 18]. For instance, authors in [7] linked the vanishing and exploding gradient problems with the stability of the neural networks interpreted as ODEs and proposed restricted architectures with guaranteed stability. We integrate these findings into our neural ODE framework.

In this paper, we present a novel method for modeling discrete ODE systems as deep neural networks. We demonstrate the possibility of incorporating varying degrees of prior knowledge combining physics-based and purely data-driven modeling in a unified framework. Moreover, we show how to impose stability guarantees and inequality constraints on the layers of arbitrary neural architectures. We apply the proposed method to the identification of a ODE model with bi-linear terms simulating a thermal system. We show that embedding constraints and stability regularizations can provide advantages in sample efficiency, generalization, as well as physically consistent trajectories.

2 METHODS

Section 2.1 describes the architecture of discrete neural ODE with possible variations and extensions. In section 2.1, we derive the stability guarantees of generic neural architectures by constraining eigenvalues of the layer weights. Moreover, we introduce a generic method for imposing inequality constraints on hidden states of deep neural networks. The proposed method for modeling of neural ODE systems is demonstrated in section 3 on a case study from the energy domain.

2.1 Neural Ordinary Differential Equations

Ground Truth ODE: Our task is to model the dynamics of an unknown ground truth ODE system in discrete-time with linear dynamics and bi-linear algebraic form:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{E}\mathbf{d}_k, \tag{1a}$$

$$\mathbf{u}_k = \mathbf{a}_k\mathbf{H}\mathbf{b}_k + \mathbf{h}, \tag{1b}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ is the system state, $\mathbf{y}_k \in \mathbb{R}^{n_y}$ is the system output, $\mathbf{u_k} \in \mathbb{R}^{n_u}$ is the algebraic input, and $\mathbf{d}_k \in \mathbb{R}^{n_d}$ is measured disturbance at time k. The bi-linearity is defined via algebraic equation 1b with linear terms $\mathbf{H}$, $\mathbf{h}$ and inputs $\mathbf{a}_k \in \mathbb{R}^{n_a}$ and $\mathbf{b}_k \in \mathbb{R}^{n_b}$.

Discrete Neural ODE: Single time step of our neural ODE model has the following form:

$$f_{\mathtt{ODE}}(\mathbf{x}, \mathbf{a}, \mathbf{b}, \mathbf{d}) = f_{\mathtt{SSM}}(\tilde{\mathbf{A}}\mathbf{x} + \tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{E}}\mathbf{d}) \tag{2a}$$

$$\mathbf{u} = h_{\Theta}(\mathbf{a}, \mathbf{b}) \tag{2b}$$

By stacking multiple layers of equation 2 with shared weights, we can construct time-invariant ODE model with arbitrary depth N, where each layer corresponds to the one-time step defined by the sampling time of the training data. The main dynamics equation 2a is given as a state space model (SSM) with parameters $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{E}}$, and activation function $f_{\mathtt{SSM}}$. In this paper, the $f_{\mathtt{SSM}}$ is given as an identity operation to model linear dynamics. We further differentiate the baseline neural ODE model into three forms with varying degrees of prior knowledge about the algebraic equation 2b. In case of no prior knowledge, we use a *black-box* ODE ($\mathtt{ODE}_{\mathtt{B}}$), where equation 2b can be modeled by a standard multi-layer fully connected neural network. For the purposes of this paper, we have chosen two layers with `ReLU` activations and 8 hidden units. In practice, we may often know the structure of the underlying algebraic equation 2b, for instance, based on known physical laws governing the system dynamics. In our case, this equation is given as bi-linear term $\mathbf{u} = \mathbf{a}\tilde{\mathbf{H}}\mathbf{b} + \tilde{\mathbf{h}}$, with learnable parameters $\Theta = \{\tilde{\mathbf{H}}, \tilde{\mathbf{h}}\}$. We will refer to such a model as a *gray-box* ODE ($\mathtt{ODE}_{\mathtt{G}}$). When the structure, as well as the parameters of equation 2b are known, e.g., obtained from the engineering sheets, we use the *white-box* ODE ($\mathtt{ODE}_{\mathtt{W}}$) with given constants $\mathbf{H}$, $\mathbf{h}$ of its bi-linear term.

Variations and Extensions: To model the time-varying dynamics, we can stop sharing the weights in the successive layers to generate piecewise-linear approximations. Another extension is to use the nonlinear activation function $f_{\mathtt{SSM}}$ in equation 2a, and increasing the depth of a single time step model. Similarly, it is straightforward to extend the input space by state variables in equation 2b for approximating differential algebraic equations (DAE) [1]. Moreover, we can structurally prior arbitrary algebraic terms, given as polynomials constructed by stacking multiple bi-linear terms. An extensive list of structural priors and possible applications is beyond the scope of this paper.

2.2 Optimization with Stability Guarantees and Constraints Handling

Eigenvalues of the Layer Weights: The Perron–Frobenius theorem [12] states that the row-wise maximum and minimum of nonnegative square matrix $\mathbf{A}$ defines the upper and lower bound of its dominant eigenvalue. We use this to constrain the eigenvalues of the weights $\tilde{\mathbf{A}}$ to enforce the stability of the layer forward pass. This constraint is formulated as:

$$\mathbf{M} = 1 - 0.1\sigma(\mathbf{M}') \tag{3}$$

$$\tilde{\mathbf{A}}_{i,j} = \frac{\exp(\tilde{\mathbf{A}}'_{ij})\mathbf{M}_{i,j}}{\sum_{k=1}^{n_x} \exp(\tilde{\mathbf{A}}'_{ik})} \tag{4}$$

Where the matrix $\mathbf{M}$ is modeling damping given as a function of parameter $\mathbf{M}' \in \mathbb{R}^{n_x \times n_x}$. We use softmax regularized rows of the $\tilde{\mathbf{A}}'$ matrix in elementwise multiplication with $\mathbf{M}$ to generate the new weight matrix $\tilde{\mathbf{A}}$ of the state dynamics used in equation 2a. With dominant eigenvalue to be less or equal to one, the stability of the learned dynamics of the discrete system is guaranteed. Additionally, by having the eigenvalues of layer weights close to one for discrete time, or zero for continuous time systems, respectively, the well-posedness of the learning problem is estabilished by preventing exploding and vanishing gradients [7].

Inequality constraints via penalty method: For handling the inequality constraints we employ the penalty method for constrained optimization [3; 4]. The principle idea is based on penalizing the constraints in the objective of the unconstrained optimization problem. In particular, we use ReLU units to model the violations of inequality constraints:

$$\underline{\mathbf{x}}_k \leq \mathbf{x}_k + \mathbf{s}_k^{\underline{x}} \;\; \cong \;\; \mathbf{s}_k^{\underline{x}} = \texttt{ReLU}(-\mathbf{x}_k + \underline{\mathbf{x}}_k) \tag{5a}$$

$$\mathbf{x}_k - \mathbf{s}_k^{\overline{x}} \leq \overline{\mathbf{x}}_k \;\; \cong \;\; \mathbf{s}_k^{\overline{x}} = \texttt{ReLU}(\mathbf{x}_k - \overline{\mathbf{x}}_k) \tag{5b}$$

Here, $\mathbf{s}_k^x = \mathbf{s}_k^{\underline{x}} + \mathbf{s}_k^{\overline{x}}$ define joint slack variables representing the magnitude of the constraints violation, and $\underline{\mathbf{x}}_k$ and $\overline{\mathbf{x}}_k$ stand for time-varying lower and upper bound on the variable $\mathbf{x}_k$, respectively. Analogically the constraints can be defined for all model variables. In this paper, we impose the inequality constraints on states $\mathbf{x}_k$ and algebraic inputs $\mathbf{u}_k$, to keep their trajectories within physically realistic bounds. We refer to the constrained ODE models as `cODE`. The proposed method for constraints handling is generic and not limited to any specific neural architecture. The constraints can be imposed on model outputs, hidden states, or their derivatives.

Loss function: The objective penalizes the deviations of the model response defined by equation 2 from the training data obtained from simulating the ground truth system equation 1 over N time steps generating sequences of vectors, $\mathcal{X} = \mathbf{x}_0, ...\mathbf{x}_N$, $\mathcal{A} = \mathbf{a}_0, ...\mathbf{a}_N$, $\mathcal{B} = \mathbf{b}_0, ...\mathbf{b}_N$, $\mathcal{D} = \mathbf{d}_0, ...\mathbf{d}_N$. We assume that for optimization, we only have access to one observable variable $\mathbf{x}_{k,i}$ denoted by index i. The model is given only an initial state $\mathbf{x}_0$, together with system inputs $\mathcal{A}$ and $\mathcal{B}$, and disturbance $\mathcal{D}$ trajectories to produce sequences of state predictions, $\tilde{\mathcal{X}} = \tilde{\mathbf{x}}_0, ...\tilde{\mathbf{x}}_N$, as well as slack variables referring to the state and hidden inputs constraints violations $S^x = \mathbf{s}_0^x, ...\mathbf{s}_N^x$ and $\mathcal{S}^u = \mathbf{s}_0^u, ...\mathbf{s}_N^u$, respectively. Constraints violations are penalized in the objective with weighting factors λ and μ. The multi-objective Mean Squared Error (MSE) loss over given N-step prediction horizon is then:

$$\mathcal{L}_{\texttt{MSE}}(\tilde{\mathcal{X}}, \mathcal{X} | \tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{E}}, \Theta) = \frac{1}{N} \sum_{k=1}^{N} \left(||\tilde{\mathbf{x}}_{k,i} - \mathbf{x}_{k,i}||_2^2 + \lambda ||\mathbf{s}_k^x||_2^2 + \mu ||\mathbf{s}_k^u||_2^2 \right) \tag{6}$$

In the general neural ODE model given by equation 2 we optimize the $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{E}}$ parameter matrices of the linear dynamics. Moreover, in the case of $\texttt{ODE}_\texttt{B}$ and $\texttt{ODE}_\texttt{G}$, also the Θ parameters of the approximated algebraic equation 2b are optimized. The models are implemented[1] in Pytorch [16].

3 Numerical Case Study

We compare empirical results for identification of models with and without constraints (`cODE` vs `ODE`), and with three degrees of prior knowledge about the algebraic interaction between input variables ($\texttt{ODE}_\texttt{B}$, $\texttt{ODE}_\texttt{G}$, $\texttt{ODE}_\texttt{W}$). We simulated the true dynamics (equation 1) of a simple building thermal system with state $\mathbf{x} \in \mathbb{R}^4$ whose elements correspond to wall (x_1), ceiling (x_2), floor (x_3), and ambient room temperature (x_4, the observed state). The bi-linear term is a heat flow equation with constant parameter of specific heat capacity $\mathbf{H} = c_p, \mathbf{h} = 0$, and two variables, mass flow $\mathbf{a}_k = \dot{\mathbf{m}}_k$, and difference of supply and return temperature $\mathbf{b}_k = \Delta \mathbf{T}_k$. The corresponding control input signals $\mathcal{A}$, and $\mathcal{B}$ are generated as a sine and cosine wave, respectively, with the period of one day and amplitudes as nominal physical values of the true model. The disturbance signals $\mathcal{D}$ represent the historical environmental conditions. We use the 2^{nd}, 3^{rd}, and 4^{th} weeks of simulation as train, validation, and test sets (each containing 2016 contiguous time-steps). We chose this limited time period to demonstrate generalization capability with limited training samples.

For black, gray, and white-box ODE models with and without state and control restraints as described in section 2.1 we train models with N-step prediction objective, $N \in \{2^3, ..., 2^7\}$. For each of these 30 (model, N-step objective) pairs we train 30 models from random parameter initializations with full batch AdamW [14] updates (step-size ranging from 0.001 to 0.03) for 15,000 epochs. We evaluate the prediction performance of the learned models for both the N-step prediction training objective, and open-loop MSE for model simulation over the test set with $T = 2016$ time steps: $\frac{1}{T} \sum_{t=1}^{T} (\mathbf{x}_{k,4}^{\text{ID}} - \tilde{\mathbf{x}}_{k,4})^2$. Tables 1, and 2 show the N-step and open-loop MSE respectively on the test set for each N-step prediction horizon training objective. As N increases, models tend to higher

[1] Code is available for reproducing experiments at:
`https://github.com/pnnl/neural_ODE_ICLR2020`

N	8	16	32	64	128
$\mathtt{ODE_B}$	0.04	0.13	0.47	0.31	0.41
$\mathtt{ODE_G}$	0.08	0.33	0.92	0.82	0.65
$\mathtt{ODE_W}$	0.08	0.31	0.92	0.81	0.65
$\mathtt{cODE_B}$	0.03	0.13	0.46	0.30	0.35
$\mathtt{cODE_G}$	0.08	0.33	0.92	0.91	0.58
$\mathtt{cODE_W}$	0.08	0.33	0.92	0.82	0.59

Table 1: Best N-step prediction MSE.

N	8	16	32	64	128
$\mathtt{ODE_B}$	9.93	3.75	5.15	**2.24**	2.59
$\mathtt{ODE_G}$	19.0	23.0	4.19	**0.91**	2.56
$\mathtt{ODE_W}$	19.6	19.2	6.58	5.24	**3.81**
$\mathtt{cODE_B}$	3.44	3.48	4.94	2.47	**0.22**
$\mathtt{cODE_G}$	19.5	19.5	4.68	2.96	**0.56**
$\mathtt{cODE_W}$	19.9	19.6	6.91	7.60	**0.41**

Table 2: Best open-loop prediction MSE.

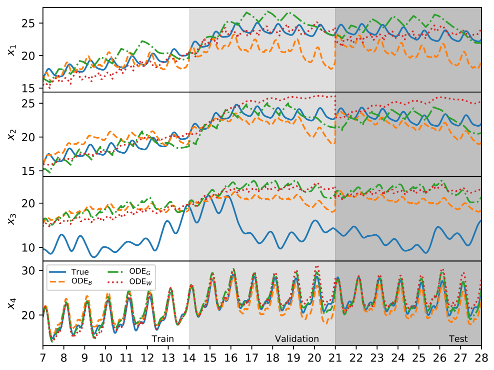

Figure 1: Non-constrained model trace.

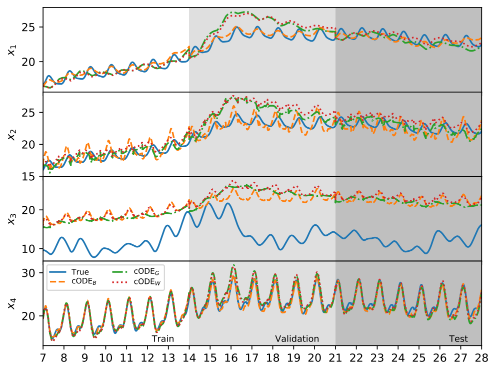

Figure 2: Constrained model trace.

N-step MSE, and lower open-loop MSE. This makes sense as, while training with a longer prediction horizon is a more difficult learning objective, the longer horizons provide closer approximations to the open-loop behavior of the learned models. Black and gray-box models without constraints fail to realize gains from the longest horizon, whereas the best performing models were constrained models with a 128-step prediction horizon objective. Interestingly, the best performing model was a $\mathtt{cODE_B}$, suggesting that given physically reasonable constraints, acceptable dynamics models can be learned given less prior knowledge of the true system. Figures 1 and 2 show open-loop simulations from non-constrained and constrained models, respectively. The solid blue line indicates the true system trajectory. Unconstrained models' trajectories drift more over time, especially for the unobserved variables. Notably, the $\mathtt{cODE_B}$ model does a remarkable job of tracking all state variables, excepting x_3 (floor temperature), which has the weakest connection with the observed x_4.

4 Conclusions and Future Work

This work presents novel methods for modeling discrete ordinary differential equations (ODE) as neural networks with i) stability guarantees based on eigenvalue regularization of the layer weights and ii) time-varying inequality constraints. Both i) and ii) are implemented using standard operations available in popular deep learning libraries. We demonstrate remarkable generalization and the ability to learn physically consistent ODE dynamics from a limited amount of training data. Empirical results underscore the advantages of using penalty methods for minimizing the constraints violation in the problem's loss function. Thus, enabling model safety assessment and certification.

The future work includes applying the neural ODEs and DAEs to model large-scale physical systems with various types of dynamic and algebraic nonlinearities. Computational efficiency and scalability of the proposed neural ODE can be further compared with classical ODE solution methods. For practical purposes, the authors intend to develop a library of physics-informed ODE and DAE priors commonly occurring in various engineering domains for user-friendly *gray-box* modeling. The authors also intend to explore the use of neural ODEs in model-based deep learning approaches to constrained optimal control for physical systems. The convergence guarantees can be obtained by means of Lyapunov stability analysis of the loss function. Another open research avenue is the development of customized optimizers for the solution of constrained optimization problems.

Finally, the generic nature of the proposed methods for enforcing stability and constraints handling can be explored on various neural architectures and learning tasks.

ACKNOWLEDGEMENTS

This work was funded by the Mathematics for Artificial Reasoning in Science investment at the Pacific Northwest National Laboratory (PNNL).

REFERENCES

[1] Pierluigi Amodio and Francesca Mazzia. Numerical solution of differential algebraic equations and computation of consistent initial/boundary conditions. *Journal of Computational and Applied Mathematics*, 87(1):135 – 146, 1997.

[2] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016.

[3] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization. SIAM, 2001.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

[6] Eldad Haber, Keegan Lensink, Eran Treister, and Lars Ruthotto. IMEXnet: A forward stable deep neural network. *CoRR*, abs/1903.02639, 2019.

[7] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *CoRR*, abs/1705.03341, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[9] Mike Innes, Alan Edelman, Keno Fischer, Christopher Rackauckas, Elliot Saba, Viral B. Shah, and Will Tebbutt. A differentiable programming system to bridge machine learning and scientific computing. *CoRR*, abs/1907.07587, 2019.

[10] Dario Izzo, Dharmesh Tailor, and Thomas Vasileiou. On the stability analysis of optimal state feedbacks as represented by deep neural models. *CoRR*, abs/1812.02532, 2018.

[11] Junteng Jia and Austin R. Benson. Neural jump stochastic differential equations. *ArXiv*, abs/1905.10403, 2019.

[12] Oliver Knill. *Linear Algebra with Probability*. Harvard College Course Math 19b, 2011.

[13] J. Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11126–11134. Curran Associates, Inc., 2019.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[15] Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *CoRR*, abs/1907.04502, 2019.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[17] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686 – 707, 2019.

[18] Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamic systems. *CoRR*, abs/1808.00924, 2018.

[19] Endre Suli. *Numerical Solution of Ordinary Differential Equations*. Mathematical Institute, University of Oxford, 2017.

5 Appendix

In this appendix we present additional visualizations comparing model performance, eigenvalues and heatmaps for learned transition matrices, and a comparison with a preliminary Physics-Informed Recurrent Neural Network Model (PI-RNN) without constraints on the principal learned dynamics matrix $\tilde{\mathbf{A}}$ or its hidden states.

5.1 Additional performance visualizations

Figures 3 and 4 visualize the influence of the increasing prediction horizon N on the open loop MSE and N-step MSE loss, reported in Tables 1 and 2 , respectively. The increasing trend of the N-step MSE with larger prediction horizon N is given by the increasing complexity of the learning problem, which is correlated with the increased accuracy of the learned models in open-loop simulations.

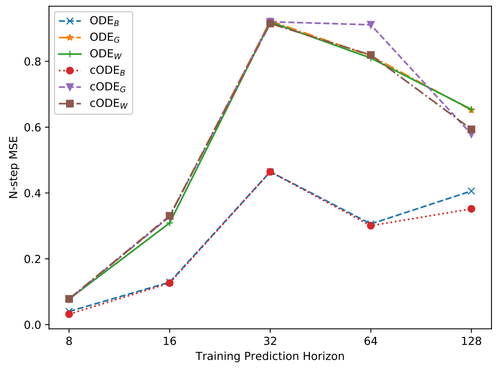

Figure 3: N-step best MSE

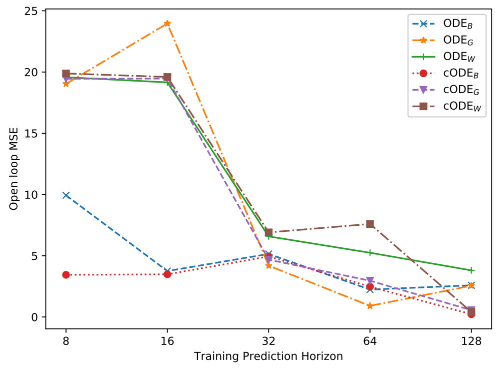

Figure 4: Open-loop best MSE

5.2 Ground Truth Physical Model

The system equation 1 represents a simple building thermal system with states $\mathbf{x}_k \in \mathbb{R}^4$ whose elements are wall ($\mathbf{x}_{k,1}$), ceiling ($\mathbf{x}_{k,2}$), floor ($\mathbf{x}_{k,3}$), and room temperature ($\mathbf{x}_{k,4}$, the observed state). Control inputs equation 1b represent heat flow equation of the building's radiator $\mathbf{u}_k = \dot{\mathbf{m}}_k c_p \Delta\mathbf{T}_k$, with mass flow $\dot{\mathbf{m}}_k$, specific heat capacity c_p, and temperature difference of the emission system $\Delta\mathbf{T}_k$. The disturbances $\mathbf{d}_k \in \mathbb{R}^3$ represent ambient temperature ($\mathbf{d}_{k,1}$), solar irradiation ($\mathbf{d}_{k,2}$), and internal heat gains ($\mathbf{d}_{k,3}$), respectively. We generate the state trajectories $\mathcal{X}$ for the system identification by simulating the model equation 1 with initial conditions of $\mathbf{x}_0 = 20\,°\mathrm{C}$, given the measured disturbance trajectories $\mathcal{D}$. In practice, $\mathcal{D}$ is obtained from weather forecast.

An interesting property of the thermal models of the buildings is that their transition matrix $\tilde{\mathbf{A}}$ is, in general, non-negative with stable eigenvalues. This feature motivates the use of the eigenvalue regularization given by equation 4. In this case, the damping factor $\mathbf{M}$ can be physically interpreted as heat losses of the building envelope. Hence physical insights can be used for tuning of the proposed model to different building types. The penalty constraints on the state trajectories are derived based on the physically meaningful values for the building.

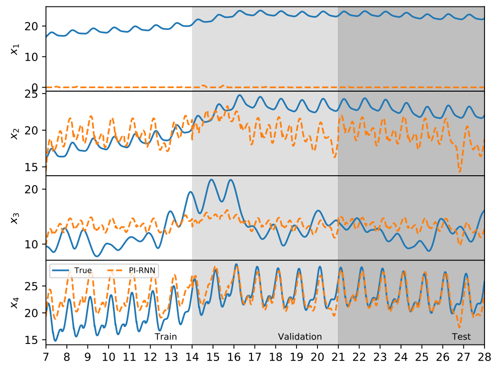

Figure 5: Open-loop trajectory for best performing S-RNN model.

	8	16	32	64	128
Open-loop	2.03	1.22	1.13	3.10	3.04
N-step	0.02	0.09	0.20	1.02	0.64

Table 3: Structured RNN MSE for N-step and open-loop prediction.

5.3 Structured RNN Model

Here we introduce a preliminary model with neither eigenvalue constraints on the learned $\tilde{\mathbf{A}}$ matrix nor barrier penalties in the learning objective. Considering the same limited knowledge of the underlying dynamics as in the $\texttt{ODE}_B$ model we introduce a preliminary model S-RNN. The discrete SSM model remains the same as equation 2a, but with a two layer neural network modeling the underlying bi-linear algebraic term as follows:

$$f_{\texttt{S-RNN}}(\mathbf{x}, \mathbf{a}, \mathbf{b}, \mathbf{d}) = f_{\texttt{SSM}}(\tilde{\mathbf{A}}\mathbf{x} + \tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{E}}\mathbf{d}) \tag{7a}$$

$$\mathbf{u} = \texttt{ReLU}\big(\tilde{\mathbf{W}}_2\mathbf{h}_1 + \tilde{\mathbf{W}}_3 \begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}\big) \tag{7b}$$

$$\mathbf{h}_1 = \texttt{ReLU}\big(\tilde{\mathbf{W}}_1 \begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}\big) \tag{7c}$$

where the matrices $\tilde{\mathbf{W}}$ are additional learned parameters for the algebraic equation approximation via neural network.

Figure 5 shows the open-loop trace of trained S-RNN on training, validation, and test set, respectively. The trajectories can be directly compared with Figures 1 and 2, at first glance it is visible that although capable of accurate prediction and generalization of the observed state, the S-RNN fails to capture the dynamics of the unobserved states, in contrast with ODE models. The corresponding open-loop MSE and N-step MSE with increasing values of the model prediction horizon N are given in Table 3. We observe that in contrast with ODE models S-RNN fails to leverage the advantage of the larger prediction horizons to improve its accuracy.

5.4 Effect of the Learned Eigenvalues

Table 4 compares the eigenvalues of the learned transition matrix $\tilde{\mathbf{A}}$ with the eigenvalues of the ground truth model. All physics-informed models accurately learned the stable dominant eigenvalue of the primary dynamics. However, differences arise when comparing the rest of the eigenvalue spectrum. The eigenvalues of the trained ODE models have, in general, shorter Euclidean distance from the ground truth values compared to S-RNN model. However, the eigenvalues of the constrained SSM have the shortest Euclidean distance from those of the ground truth system. Moreover, the better estimate of the eigenvalues of the system dynamics can be correlated with better open-loop performance, as reported in Table 4. Additionally, S-RNN is the only model learning complex eigenvalues. This can be further examined through the physical interpretation of the eigenvalues given as follows: real parts represent gains of the system, while the imaginary parts define the frequencies of the dynamics signals. Hence, S-RNN model learned to associate the periodic behavior of the training data with the main system dynamics given by $\tilde{\mathbf{A}}$ transition matrix. However, this is not correct association because the periodicity of the training data is the consequence of the periodic nature of the control inputs $\mathcal{A}$, $\mathcal{B}$ and disturbance trajectories $\mathcal{D}$ (day and night patterns). This may provide an explanation of why ODE models outperform S-RNN in the open-loop prediction. Moreover, it can explain a remarkable capability of the ODE models in also predicting the unobserved states trajectories, despite not being explicitly trained to do so. In contrast, S-RNN fails to get even close to true trajectories of the unobserved states, as shown in Figure 5.

Table 4: Comparison of Eigenvalues for $\tilde{\mathbf{A}}$ transition matrix

	λ_1	λ_2	λ_3	λ_4
True	1.0	0.99	0.98	0.25
S-RNN	0.99	0.11+0.11i	0.11-0.11i	-0.05
ODE_B	1.0	0.88	0.21	0.02
ODE_G	1.0	0.62	0.15	-0.01
ODE_W	1.0	0.76	0.47	0.07
cODE_B	1.0	0.89	0.15	-0.03
cODE_G	1.0	0.60	0.21	0.02
cODE_W	1.0	0.65	0.25	0.03

5.5 Comparison of the Learned State Transition Parameters

Figure 6 compares the heat maps of the state transition matrix parameters of the learned models $\tilde{\mathbf{A}}$ with ground truth values $\mathbf{A}$. The comparison of the learned model parameters of S-RNN with ODE models is less clear than in the case of eigenvalues. Nevertheless, we can spot that ODE models are slightly sparser than S-RNN. Especially, ODE_W models learn the most similar sparsity patterns in visual comparison with the diagonal structure of the ground truth model. However, the ODE_W is outperformed by ODE_B model in the open-loop prediction task, indicating that not learning the true parameters but having the eigenvalues correct matters the most. Without any sparsity regularizations or structural priors on the $\tilde{\mathbf{A}}$ matrix, no model can exactly identify the ground truth model parameters. However, this does not prevent the models to learn physically consistent open-loop dynamical trajectories with large time horizons. This might suggest that the solution to this system identification problem is not unique. In conclusion, it is important to say that a more rigorous analysis of the learned system parameters and eigenvalues, as well as the model structure, needs to be made to verify or falsify the qualitative statements in this appendix.

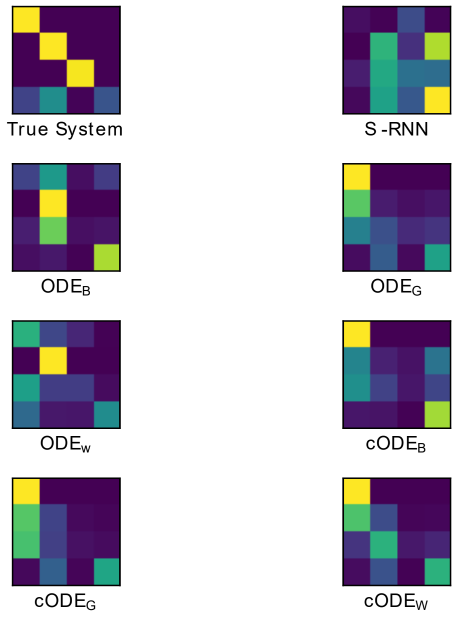

Figure 6: Heat maps of learned $\tilde{\mathbf{A}}$ matrices