Discovering Sparse Interpretable Dynamics from Partial Observations

Peter Y. Lu,[1,*] Joan Ariño,[1,2] and Marin Soljačić[1]

[1] *Department of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[2] *Department of Physics, Universitat Politècnica de Catalunya, Barcelona, Spain*

(Dated: July 26, 2021)

Identifying the governing equations of a nonlinear dynamical system is key to both understanding the physical features of the system and constructing an accurate model of the dynamics that generalizes well beyond the available data. We propose a machine learning framework for discovering these governing equations using only partial observations, combining an encoder for state reconstruction with a sparse symbolic model. Our tests show that this method can successfully reconstruct the full system state and identify the underlying dynamics for a variety of ODE and PDE systems.

Introduction.—Analyzing data from a nonlinear dynamical system to understand its qualitative behavior and accurately predict future states is a ubiquitous problem in science and engineering. In many instances, this problem is further compounded by a lack of available data and only partial observations of the system state, e.g. forecasting fluid flow driven by unknown sources or predicting optical signal propagation without phase measurements. This means that, in addition to identifying and modeling the underlying dynamics, we must also reconstruct the hidden or unmeasured variables of the system state. While traditional approaches to system identification have had significant success with linear systems, nonlinear system identification and state reconstruction is a much more difficult and open problem [1]. Moreover, modeling nonlinear dynamics in a way that provides interpretability and physical insight is also a major challenge.

Modern machine learning approaches have made significant strides in black box predictive performance on many tasks [2], such as data-driven prediction of nonlinear dynamics [3–5] including methods that only use partial observations [6–9]. However, because deep learning models often fail to take into account known physics, they require vast quantities of data to train and tend to generalize poorly outside of their training distribution. Standard deep learning models also lack the interpretability necessary for developing a detailed physical understanding of the system, although recent unsupervised learning approaches can help mitigate this problem [10]. Introducing physical priors and building physics-informed inductive biases, such as symmetries, into neural network architectures can significantly improve the performance of deep learning models and provide a greater degree of interpretability [10–13].

Recent data-driven nonlinear system identification methods based on Koopman operator theory offer a compelling alternative to deep learning approaches as well as a theoretical framework for incorporating neural networks into system identification methods [14–17]. However, these approaches still run into barriers when dealing with chaotic dynamics and other forms of nonlinear dynamics that lead to a problematic continuous spectrum for the Koopman operator, although some progress has been made in addressing these limitations as well [17–19].

In this work, we focus on directly learning the governing equations of motion, which are often sparse and provide a highly interpretable representation of the dynamical system that also generalizes extremely well. Previous work has shown that, by imposing a sparsity prior on the governing equations, it is possible to obtain interpretable and parsimonious models of nonlinear dynamics [20–22]. This sparsity prior, in combination with an autoencoder architecture, can also aid in extracting interpretable coordinates or state variables from high dimensional data [23].

Rather than working in a generic high dimensional setting, we consider the common situation where a visible portion of the system state is known and observed but additional hidden states as well as the underlying dynamics are unknown. To deal with having only partial state information, we propose a machine learning method that combines an encoder, for reconstructing the full system state, and a sparse symbolic model, which learns the system dynamics, providing a flexible framework for both system identification and state reconstruction (Fig. 1). The full architecture is trained by matching the higher order time derivatives of the symbolic model with finite difference estimates from the data. As illustrated in our numerical experiments, this approach can be easily adapted for specific applications by incorporating known constraints into the architecture of the encoder and the design of the symbolic model.

Problem Formulation.—Consider a nonlinear dynamical system defined by the first order ODE

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}). \tag{1}$$

The visible or measured state is given by a known "projection" function $\mathbf{x}_v = \mathbf{g}(\mathbf{x})$ while the hidden states $\mathbf{x}_h$ must be reconstructed such that $\mathbf{a}(\mathbf{x}_v, \mathbf{x}_h) = \mathbf{x}$, where $\mathbf{a}$ is a known aggregation function. The goal is to determine the governing equations defined by $\mathbf{F}(\mathbf{x})$ while simultaneously reconstructing the hidden state $\mathbf{x}_h$.

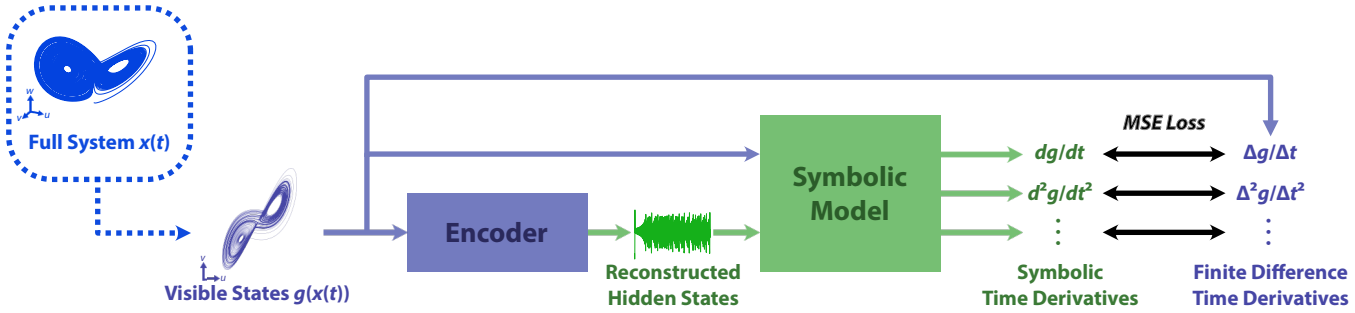

FIG. 1. A machine learning framework for simultaneous system identification and state reconstruction. With only a visible portion of the full state available $\mathbf{x}_v = \mathbf{g}(\mathbf{x})$, an encoder is first used to reconstruct the hidden states. The fully reconstructed state $\hat{\mathbf{x}}$, including the visible and hidden states, is then passed into a symbolic model of the governing equations. Using automatic differentiation, multiple symbolic time derivatives $d^p\mathbf{g}(\hat{\mathbf{x}})/dt^p$ of the visible states are generated from the symbolic model and compared with finite difference derivatives $\Delta^p\mathbf{g}(\mathbf{x})/\Delta t^p$ computed directly from the sequence of visible states. The entire architecture is trained end-to-end using the mean squared error (MSE) loss between the symbolic and finite difference derivatives.

Without prior knowledge detailing the structure of the dynamical system, we can generically choose the visible state $\mathbf{x}_v = (x_1, x_2, \ldots, x_k)$ to be a subset of the full state $\mathbf{x} = (x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n)$, i.e. $\mathbf{g}$ is a simple projection of $\mathbf{x}$ onto the subset $\mathbf{x}_v$. The remaining components would then form the hidden state $\mathbf{x}_h = (x_{k+1}, x_{k+2}, \ldots, x_n)$, and the aggregation function $\mathbf{a}$ just concatenates of the two states $\mathbf{x}_v, \mathbf{x}_h$. When additional information about the dynamical system is available, $\mathbf{g}$ and $\mathbf{a}$ can be chosen appropriately to reflect the structure of the dynamics (e.g. see our nonlinear Schrödinger phase reconstruction example).

Proposed Machine Learning Framework.—Our proposed framework consists of an encoder, which uses the visible states to reconstruct the corresponding hidden states, and an interpretable symbolic model, which represents the governing equations of the dynamical system. The encoder $\mathbf{e}_\eta$, typically a neural network architecture with learnable parameters η, takes as input the sequence of visible states $\{\mathbf{x}_v(t_0), \mathbf{x}_v(t_0 + \Delta t), \ldots, \mathbf{x}_v(t_N)\}$ and reconstructs the hidden states $\{\hat{\mathbf{x}}_h(t_0), \hat{\mathbf{x}}_h(t_0 + \Delta t), \ldots, \hat{\mathbf{x}}_h(t_N)\}$. We can then obtain a reconstruction of the full state by applying the aggregation function $\hat{\mathbf{x}} = \mathbf{a}(\mathbf{x}_v, \hat{\mathbf{x}}_h)$. The fully reconstructed state $\hat{\mathbf{x}}$ allows us to compute symbolic time derivatives defined by a symbolic model of the governing equations

$$\frac{d\hat{\mathbf{x}}}{dt} = \hat{\mathbf{F}}_\theta(\hat{\mathbf{x}}) \coloneqq \theta_1\mathbf{f}_1(\hat{\mathbf{x}}) + \theta_2\mathbf{f}_2(\hat{\mathbf{x}}) + \cdots + \theta_m\mathbf{f}_m(\hat{\mathbf{x}}), \quad (2)$$

where $\theta_1, \theta_2, \ldots, \theta_m$ are learnable coefficients and $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m$ are predefined terms, such as monomial expressions or linear combinations representing spatial derivatives (for PDE systems).

To jointly train the encoder and symbolic model using only partial observations, we match not only the first order time derivatives but also higher order time derivatives of the visible states with finite difference estimates from the data. These time derivatives are implicitly defined by the symbolic model (Eq. S1), so we develop and use an algorithmic trick that allows standard automatic differentiation methods [24] to automatically compute higher order symbolic time derivatives of the *reconstructed* visible states $\mathbf{g}(\hat{\mathbf{x}})$ (see Supplemental Materials for details). These symbolic derivatives can then be compared with finite difference time derivatives $\Delta^p\mathbf{g}(\mathbf{x})/\Delta t^p = \Delta^p\mathbf{x}_v/\Delta t^p$ computed directly from the visible states $\mathbf{x}_v$.

We train the entire architecture in an end-to-end fashion by optimizing the mean squared error (MSE) loss

$$\mathcal{L}(\eta, \theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{p=1}^{M}\alpha_p\left(\frac{d^p\mathbf{g}(\hat{\mathbf{x}}(t_i))}{dt^p} - \frac{\Delta^p\mathbf{x}_v(t_i)}{\Delta t^p}\right)^2, \quad (3)$$

where α_p are hyperparameters that determine the importance of each derivative order in the loss function. This loss implicitly depends on the encoder $\mathbf{e}_\eta$ through the reconstructed state $\hat{\mathbf{x}}$ and the symbolic model $\hat{\mathbf{F}}_\theta$ through the symbolic time derivatives. To achieve sparsity in the symbolic model, we use a simple thresholding approach—commonly used in sparse linear regression applications [20]—which sets a coefficient θ_i to zero if its absolute value falls below a chosen threshold θ_{thres}. We implement this sparsification at regular intervals during training. See the Supplemental Materials for additional model architecture and training details.

The code for implementing our framework and reproducing our results is available at `https://github.com/peterparity/symder`.

ODE Experiments.—To demonstrate our method, we use data from two standard examples of chaotic nonlinear dynamics: the Rössler system (Fig. 2a) and the Lorenz system (Fig. 2b). Both systems have a three-dimensional phase space (u, v, w), and we take the first two dimensions (u, v) to be the visible state with the remaining dimension w as the hidden state. In both cases, we are able to accurately identify the governing equations and

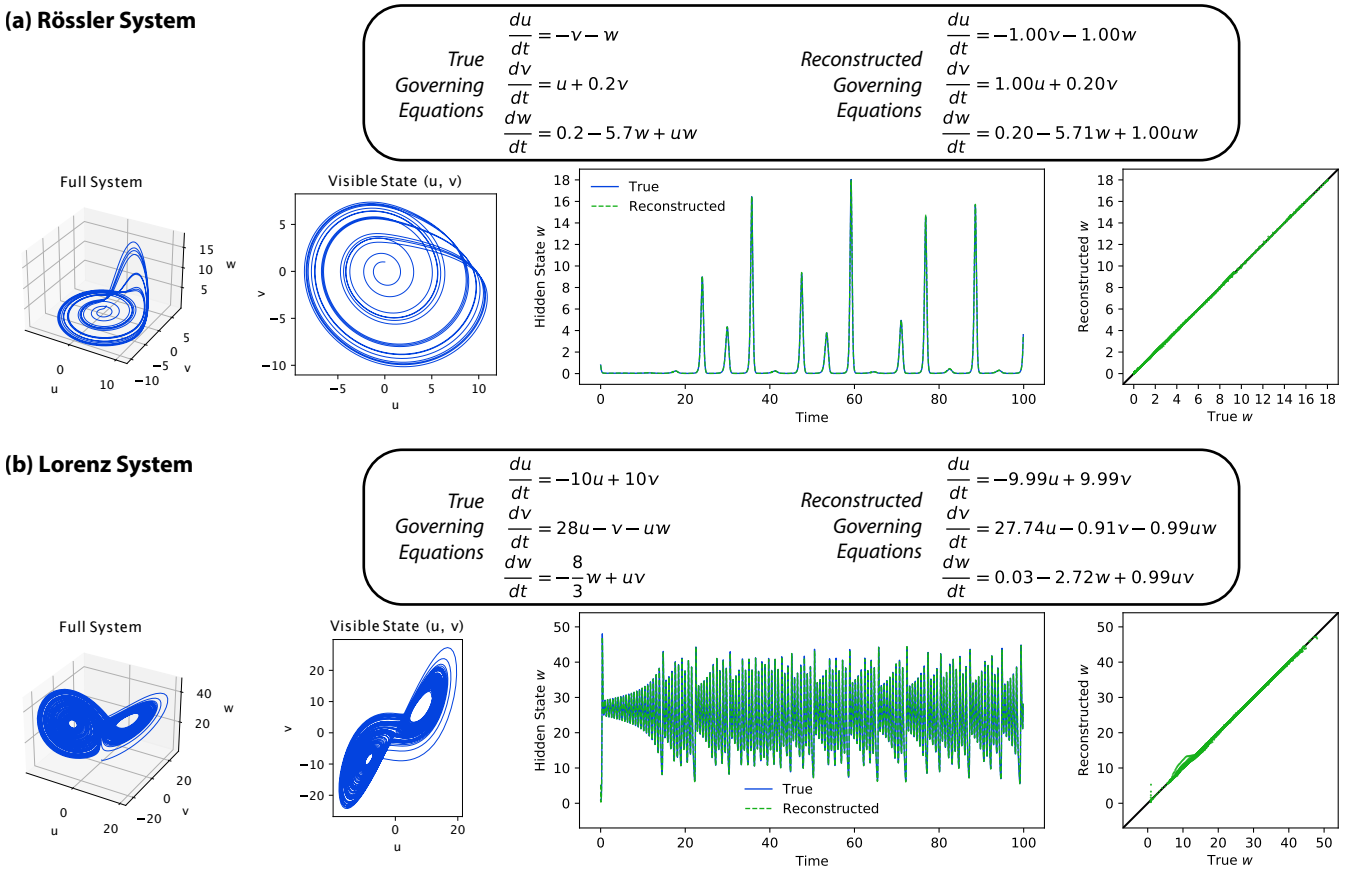

FIG. 2. System identification and hidden state reconstruction for the (a) Rössler and (b) Lorenz systems. In both numerical experiments, the u and v components are visible while the w component is hidden. The true and reconstructed hidden states w are shown as a function of time and also plotted directly against each other for comparison.

reconstruct the hidden state w (Fig. 2). We achieve a relative reconstruction error of 4.6×10^{-4} (relative to the range of the hidden state) for the Rössler system and 1.7×10^{-3} for the Lorenz system.

PDE Experiments.—To test our method in a more challenging setting, we use data from two PDE systems: a 2D diffusion system with an exponentially decaying source term (Fig. 3a) and a 2D diffusive Lokta–Volterra predator–prey system (Fig. 3b)—commonly used for ecological modeling [25–27]. For the diffusion system, we observe a diffusing visible state $u(x, y, t)$ and must reconstruct the hidden dynamic source term $v(x, y, t)$. Similarly, for the diffusive Lokta–Volterra system, one of the two components is visible $u(x, y, t)$ while the other is hidden $v(x, y, t)$. We accurately identify the governing equations and reconstruct the hidden component for both systems (Fig. 3), achieving a relative error of 1.4×10^{-4} for the diffusion system and 1.0×10^{-3} for the diffusive Lokta–Volterra system. The neural network encoder has more difficulty with the more complex and nonlinear diffusive Lokta–Volterra system, resulting in a slightly blurry reconstruction.

Phase Reconstruction.—As a final example, we consider the phase retrieval or reconstruction problem for the 1D nonlinear Schrödinger equation—a model for light propagation through a nonlinear fiber [28]—to demonstrate the breadth of our approach and its ability to handle a more difficult and structured problem. Using only visible amplitude data $|\psi(x, t)|$, we aim to identify the underlying dynamics and reconstruct the hidden phase $\varphi(x, t) = \arg(\psi(x, t))$. For this system, we also assume that we have some prior knowledge about the structure of the dynamics, namely that we have a complex wave equation with a global phase shift symmetry and only odd nonlinearities in order to model an optical material with inversion symmetry [28]. This allows us to limit the library of predefined terms used by our symbolic model. Our prior knowledge also informs our choice of projection $\mathbf{g}(\psi) = |\psi|$ and aggregation functions $\mathbf{a}(|\psi|, \varphi) = |\psi| e^{i\varphi}$.

Our method successfully identifies the governing equation for the nonlinear Schrödinger data and roughly captures the correct phase profile. Although the overall phase reconstruction seems somewhat poor, with a relative error of 0.35, this also includes an accumulated drift of the phase over time. The spatial derivative of the phase $\partial\varphi/\partial x$ has a much more reasonable relative

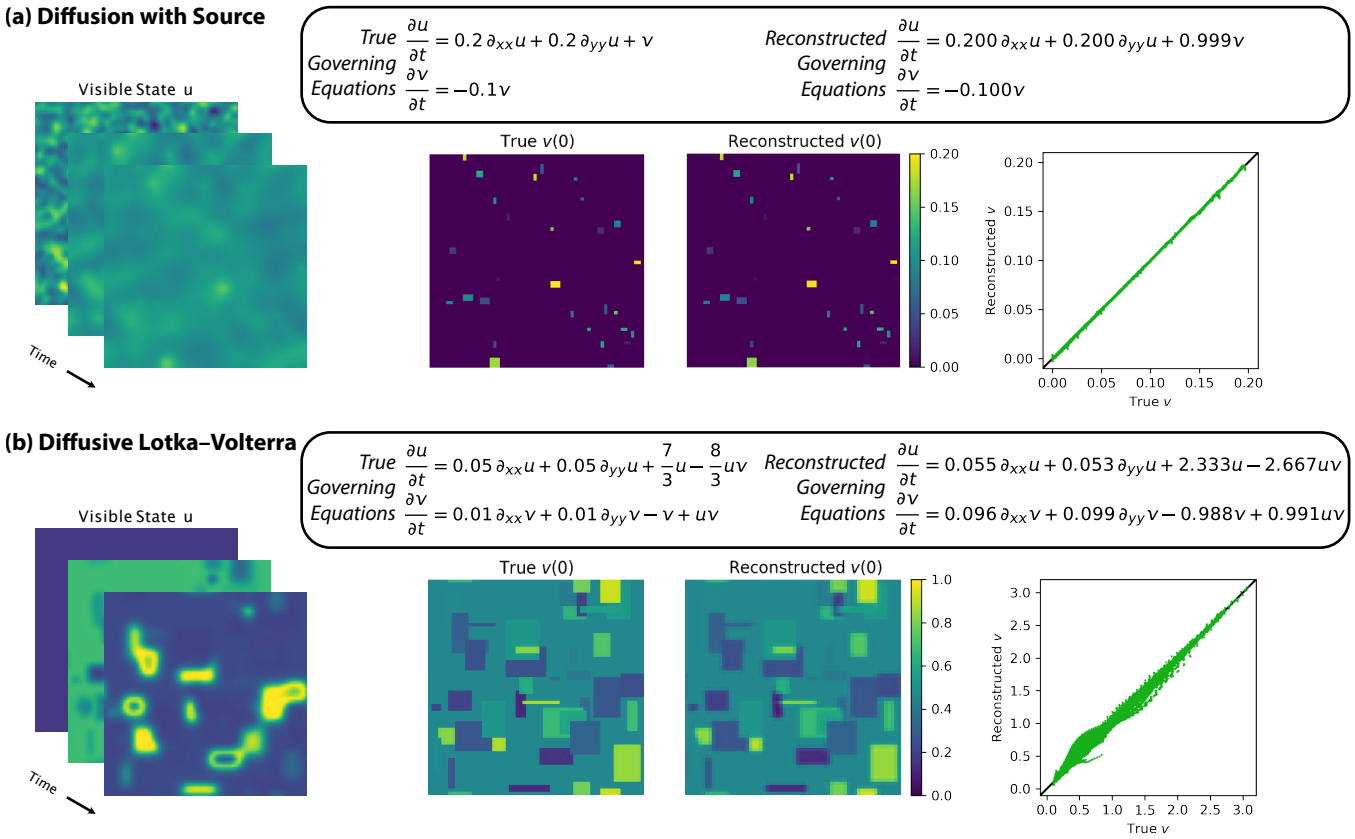

FIG. 3. System identification and hidden state reconstruction for the (a) diffusion system with a decaying source term v and (b) diffusive Lokta–Volterra system. In both numerical experiments, the u component is visible while the v component is hidden. The true and reconstructed hidden states v are shown at time $t = 0$ and are also plotted directly against each other for comparison.

error of 0.057. Furthermore, given the governing equations extracted by our method, other more specialized algorithms for nonlinear phase retrieval can be used as a post-processing step to significantly improve the quality of the phase reconstruction [29].

Conclusion.—On a wide variety of dynamical systems, we have demonstrated that our proposed machine learning framework can successfully identify sparse interpretable dynamics and reconstruct hidden states using only partial observations. Our method is also straightforward to implement and use, easily adapting to differing levels of prior knowledge about the unknown hidden states and dynamics.

Compared with methods that require explicit integration [6, 8], our approach can be significantly more computational efficient since we only need to compute symbolic and finite difference derivatives. Methods that rely on explicit integration may also need to deal with stiffness and other issues that are relevant to choosing an appropriate integration scheme [11]. However, methods using explicit integration also have the advantage of being much more robust to noise. Because we require higher order finite difference time derivative estimates from data, our approach—like other derivative-based methods—is generally more susceptible to noise. Careful tuning of our sparsity method helps mitigate this to some extent in a similar fashion to methods like SINDy [20, 21], and promising new methods for identifying the noise distribution alongside the dynamics [30] could be incorporated into our framework in the future.

Our framework offers a strong foundation for designing interpretable machine learning methods to deal with partial observations and solve the combined system identification and state reconstruction task. We hope to continue developing more robust encoders and more flexible symbolic models that will work within our proposed framework. For example, the encoder (described in the Supplemental Materials) used in our final experiment on phase reconstruction has similarities with variational approaches used for PDE discovery [3, 22], and we believe that these variational methods could be incorporated into our framework to provide a smoother encoding and improve robustness to noise. In future work, we will also study symbolic models that have multiple layers of composable units designed for symbolic regression tasks [31–33]. These alternative symbolic architectures provide

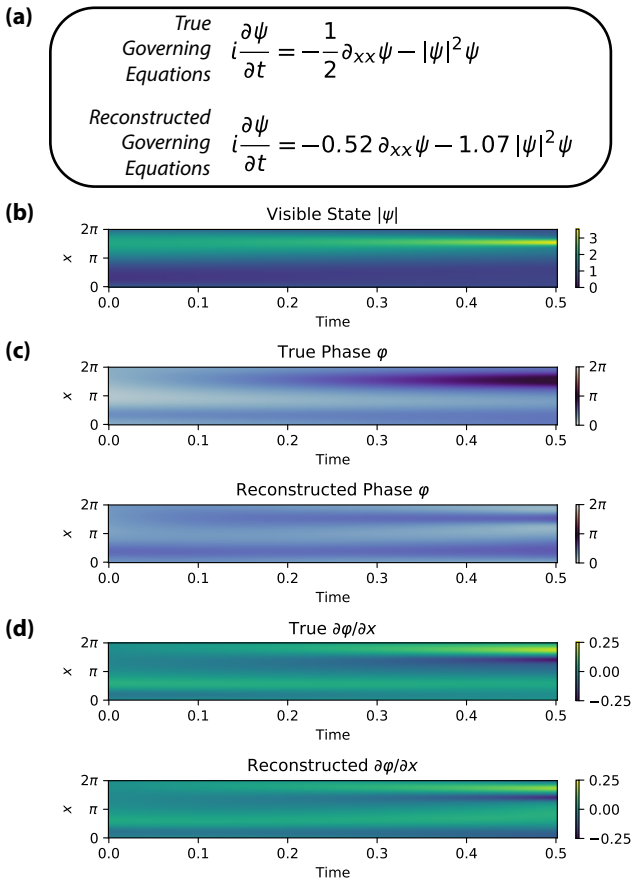

FIG. 4. System identification (a) and phase reconstruction (c,d) for the nonlinear Schrödinger system. The magnitude $|\psi|$ of the wave is visible (b) while the phase $\varphi = \arg(\psi)$ is hidden (c) and must be reconstructed. The spatial derivative of the phase $\partial\varphi/\partial x$ (d) and its reconstruction are also shown.

more powerful and flexible models with a sparse symbolic prior without requiring large libraries of predefined terms.

We would like to acknowledge useful discussions with Samuel Kim, Rumen Dangovski, Charlotte Loh, Andrew Ma, and Ileana Rugina. This research is supported in part by the U.S. Department of Defense through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. This work is further supported in part by the the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, `http://iaifi.org/`). It is also based upon work supported in part by the U.S. Army Research Office through the Institute for Soldier Nanotechnologies at MIT, under Collaborative Agreement Number W911NF-18-2-0048. Research was also sponsored in part by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

* lup@mit.edu

[1] L. Ljung, *System Identification: Theory for the User*, 2nd ed. (Pearson, 1999).

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) `http://www.deeplearningbook.org`.

[3] M. Raissi, P. Perdikaris, and G. 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 (2019).

[4] J. Berg and K. Nyström, Data-driven discovery of PDEs in complex datasets, Journal of Computational Physics **384**, 239 (2019).

[5] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, Journal of Machine Learning Research **19**, 1 (2018).

[6] I. Ayed, E. d. Bézenac, A. Pajot, and P. Gallinari, Learning the spatio-temporal dynamics of physical processes from partial observations, in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020) pp. 3232–3236.

[7] S. Ouala, D. Nguyen, L. Drumetz, B. Chapron, A. Pascual, F. Collard, L. Gaultier, and R. Fablet, Learning latent dynamics for partially observed chaotic systems, Chaos **30**, 103121 (2020).

[8] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Neural ordinary differential equations, in *Advances in Neural Information Processing Systems*, Vol. 31, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018).

[9] P. Saha, S. Dash, and S. Mukhopadhyay, Physics-incorporated convolutional recurrent neural networks for source identification and forecasting of dynamical systems (2020), arXiv:2004.06243.

[10] P. Y. Lu, S. Kim, and M. Soljačić, Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning, Phys. Rev. X **10**, 031056 (2020).

[11] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, Universal differential equations for scientific machine learning (2020), arXiv:2001.04385.

[12] Y. Yin, V. L. Guen, J. Dona, E. de Bezenac, I. Ayed, N. Thome, and P. Gallinari, Augmenting physical models with deep networks for complex dynamics forecasting, in *International Conference on Learning Representations* (2021).

[13] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, Geometric deep learning: Grids, groups, graphs,

geodesics, and gauges (2021), arXiv:2104.13478.
[14] A. Mauroy, Y. Susuki, and I. Mezić, *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications* (Springer International Publishing, Cham, 2020).
[15] C. Folkestad, D. Pastor, I. Mezic, R. Mohr, M. Fonoberova, and J. Burdick, Extended dynamic mode decomposition with learned koopman eigenfunctions for prediction and control, in *2020 American Control Conference (ACC)* (2020) pp. 3906–3913.
[16] N. Takeishi, Y. Kawahara, and T. Yairi, Learning Koopman invariant subspaces for dynamic mode decomposition, in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).
[17] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, Modern koopman theory for dynamical systems (2021), arXiv:2102.12086.
[18] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz, Chaos as an intermittently forced linear system, Nature Communications **8**, 19 (2017).
[19] B. Lusch, J. N. Kutz, and S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature Communications **9**, 4950 (2018).
[20] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences **113**, 3932 (2016), https://www.pnas.org/content/113/15/3932.full.pdf.
[21] K. Kaheman, J. N. Kutz, and S. L. Brunton, SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **476**, 20200279 (2020).
[22] Z. Chen, Y. Liu, and H. Sun, Physics-informed learning of governing equations from scarce data (2020), arXiv:2005.03448.
[23] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, Data-driven discovery of coordinates and governing equations, Proceedings of the National Academy of Sciences **116**, 22445 (2019), https://www.pnas.org/content/116/45/22445.full.pdf.
[24] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: a survey, Journal of Machine Learning Research **18**, 1 (2018).
[25] D. M. Dubois, A model of patchiness for prey—predator plankton populations, Ecological Modelling **1**, 67 (1975).
[26] H. N. Comins and D. W. Blatt, Prey-predator models in spatially heterogeneous environments, Journal of Theoretical Biology **48**, 75 (1974).
[27] T. Kmet' and J. Holčík, The diffusive Lotka-Volterra model as applied to the population dynamics of the German carp and predator and prey species in the Danube River basin, Ecological Modelling **74**, 277 (1994).
[28] M. J. Ablowitz, *Nonlinear Dispersive Waves: Asymptotic Analysis and Solitons*, Cambridge Texts in Applied Mathematics (Cambridge University Press, 2011).
[29] C.-H. Lu, C. Barsi, M. O. Williams, J. N. Kutz, and J. W. Fleischer, Phase retrieval using nonlinear diversity, Appl. Opt. **52**, D92 (2013).
[30] K. Kaheman, S. L. Brunton, and J. N. Kutz, Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data (2020), arXiv:2009.08810.
[31] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Čeperić, and M. Soljačić, Integration of neural network-based symbolic regression in deep learning for scientific discovery, IEEE Transactions on Neural Networks and Learning Systems , 1 (2020).
[32] A. Costa, R. Dangovski, O. Dugan, S. Kim, P. Goyal, M. Soljačić, and J. Jacobson, Fast neural models for symbolic regression at scale (2020), arXiv:2007.10784.
[33] S.-M. Udrescu and M. Tegmark, Ai feynman: A physics-inspired method for symbolic regression, Science Advances **6**, 10.1126/sciadv.aay2631 (2020).
[34] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).
[35] M. Innes, Don't unroll adjoint: Differentiating SSA-form programs (2018), arXiv:1810.07951.
[36] J. Zhuang, T. Tang, Y. Ding, S. C. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan, Adabelief optimizer: Adapting stepsizes by the belief in observed gradients, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 18795–18806.

Supplemental Materials: Discovering Sparse Interpretable Dynamics from Partial Observations

AUTOMATIC COMPUTATION OF SYMBOLIC DERIVATIVES

The time derivatives can be derived by repeated differentiation of the symbolic model

$$\frac{d\hat{\mathbf{x}}}{dt} = \hat{\mathbf{F}}_\theta(\hat{\mathbf{x}}), \tag{S1}$$

substituting back in previously computed derivatives to obtain expressions only in terms of the reconstructed state $\hat{\mathbf{x}}$. For example, the first and second time derivatives can be written in index notation as

$$\frac{dg_i}{dt} = \sum_j \frac{dg_i}{d\hat{x}_j}\frac{d\hat{x}_j}{dt} = \sum_j \frac{dg_i}{d\hat{x}_j}\hat{F}_{\theta j} \tag{S2}$$

$$\frac{d^2 g_i}{dt^2} = \sum_{j,k} \frac{d^2 g_i}{d\hat{x}_j d\hat{x}_k}\hat{F}_{\theta j}\hat{F}_{\theta k} + \frac{dg_i}{d\hat{x}_j}\frac{d\hat{F}_{\theta j}}{d\hat{x}_k}\hat{F}_{\theta k}. \tag{S3}$$

The expressions for the symbolic time derivatives (Eqs. S2 & S3) quickly grow more and more unwieldy for higher order derivatives. Implementing these expressions by hand is likely to be both time-consuming and error-prone, especially for more complex symbolic models such as those used in our PDE experiments. To address this issue, we develop an automated approach that takes advantage of powerful modern automatic differentiation software (in our case, the JAX library [34]).

Automatic differentiation is the algorithmic backbone of modern deep learning [24], and a new generation of source-to-source automatic differentiation libraries are quickly becoming available [34, 35]. Automatic differentiation uses a library of custom derivative rules defined on a set of primitive functions which can then be arbitrarily composed to form more complex expressions. The algorithm normally requires a forward evaluation of a function that sets up a backward pass which computes the gradient of the function. In our case, the appropriate forward step is integrating the symbolic model (Eq. S1) using an ODE solver, which makes the time variable and its derivatives explicit rather than being implicitly defined by the governing equations. This, however, would introduce significant overhead and would not produce the exact expressions that we derived earlier. In fact, integration should not be necessary at all for efficiently implementing symbolic differentiation. Instead, we propose a simple algorithmic trick that allows standard automatic differentiation to compute symbolic time derivatives without explicit integration.

Consider a function $\mathcal{I}(\hat{\mathbf{x}}, \epsilon)$ that propagates the state $\hat{\mathbf{x}}$ forward by a time ϵ according to the governing equations (Eq. S1), i.e.

$$\mathcal{I}(\hat{\mathbf{x}}(t), \epsilon) = \hat{\mathbf{x}}(t+\epsilon). \tag{S4}$$

As $\epsilon \to 0$, $\mathcal{I}(\hat{\mathbf{x}}(t), 0) = \hat{\mathbf{x}}(t)$ reduces to the identity. Taking a derivative with respect to ϵ, we find that

$$\begin{aligned}\frac{\partial \mathcal{I}(\hat{\mathbf{x}}(t), \epsilon)}{\partial \epsilon} &= \frac{d\hat{\mathbf{x}}(t+\epsilon)}{d\epsilon} \\ &= \hat{\mathbf{F}}_\theta(\hat{\mathbf{x}}(t+\epsilon)) \\ &= \hat{\mathbf{F}}_\theta(\mathcal{I}(\hat{\mathbf{x}}(t), \epsilon)),\end{aligned} \tag{S5}$$

which reduces to $\partial \mathcal{I}(\hat{\mathbf{x}}, \epsilon)/\partial \epsilon|_{\epsilon=0} = d\hat{\mathbf{x}}/dt = \hat{\mathbf{F}}_\theta(\hat{\mathbf{x}})$ as $\epsilon \to 0$. This generalizes to higher order derivatives, allowing us to compute time derivatives of $\hat{\mathbf{x}}$ as

$$\frac{d^p \hat{\mathbf{x}}}{dt^p} = \left.\frac{\partial^p \mathcal{I}(\hat{\mathbf{x}}, \epsilon)}{\partial \epsilon^p}\right|_{\epsilon=0}. \tag{S6}$$

Since we only ever evaluate at $\epsilon = 0$, this formulation makes the time variable explicit without having to integrate the governing equations. To implement this trick using an automatic differentiation algorithm, we define a wrapper function $\mathcal{I}_0(\hat{\mathbf{x}}, \epsilon) \coloneqq \hat{\mathbf{x}}$ that acts as the identity on the state $\hat{\mathbf{x}}$ but has a custom derivative rule

$$\frac{\partial \mathcal{I}_0(\hat{\mathbf{x}}, \epsilon)}{\partial \epsilon} \coloneqq \hat{\mathbf{F}}_\theta(\mathcal{I}_0(\hat{\mathbf{x}}, \epsilon)). \tag{S7}$$

This allows standard automatic differentiation to correctly compute exact symbolic time derivatives of our governing equations, including higher order derivatives. Our code for implementing this algorithmic trick and for reproducing the rest of our results is available at `https://github.com/peterparity/symder`.

The proposed algorithmic trick for computing higher order time derivatives, which exploits modern automatic differentiation, further simplifies the implementation of our method and allows the user to focus on designing an appropriate encoder and choosing a reasonable library of predefined terms for the sparse symbolic model.

DATASET, ARCHITECTURE, AND TRAINING DETAILS

ODE Systems

Each system is sampled for 10000 time steps of size $\Delta t = 10^{-2}$, and the resulting time series data and computed finite difference derivatives are normalized to unit variance.

The encoder takes a set of nine visible states $\{\mathbf{x}_v(t - 4\Delta t), \mathbf{x}_v(t - 3\Delta t), \ldots, \mathbf{x}_v(t + 4\Delta t)\}$ as input to reconstruct each hidden state $\hat{\mathbf{x}}_h(t)$ and is implemented as a sequence of three 1D time-wise convolutional layers with kernel sizes 9–1–1 and layer sizes 128–128–1. This architecture enforces locality in time, allowing the neural

network to learn a simpler and more interpretable mapping. The predefined terms of the symbolic model consist of constant, linear, and quadratic monomial terms, i.e. 1, u, v, w, u^2, v^2, w^2, uv, uw, and vw, for each governing equation.

We also scale the effective time step of the symbolic model by a factor of 10 to improve training by preconditioning the model coefficients. We then train for 50000 steps using the AdaBelief optimizer [36] with learning rate 10^{-3} and with hyperparameters $\alpha_1 = \alpha_2 = 1$ to equally weight the first two time derivative terms in the loss function ($\alpha_p = 0$ for $p > 2$). Every 5000 training steps, we sparsify the symbolic model, setting coefficients to zero if their absolute value is below $\theta_{\text{thres}} = 10^{-3}$.

One additional caveat is that the equation and hidden state obtained by our approach is not exactly the same as the original and instead corresponds to the correct governing equations for an affine transformation of the hidden state $w' = aw + b$. In order to make a direct comparison, we use linear regression to fit the reconstructed hidden states to the original hidden states and show the resulting transformed equations.

PDE Systems

Each system is sampled on a 64×64 spatial mesh with grid spacing $\Delta x = \Delta y = 1$ for 1000 time steps of size $\Delta t = 5 \times 10^{-2}$, and the resulting data and estimated derivatives are normalized to unit variance.

The encoder is a sequence of three 3D spatiotemporal convolutional layers with kernel sizes 5–1–1 and layer sizes 64–64–1, which enforces locality in both time and space. The predefined terms of the symbolic model consist of constant, linear, and quadratic terms as well as up to second order spatial derivative terms, e.g. $\partial_x u$, $\partial_y u$, $\partial_{xx} u$, $\partial_{yy} u$, $\partial_{xy} u$, and similarly for v.

We scale the effective time step and spatial grid spacing of the symbolic model by a factor of 10 and $\sqrt{10}$, respectively, to precondition the model coefficients. For the diffusion system, we train for 50000 steps with learning rate 10^{-4} and hyperparameters $\alpha_1 = 1$ and $\alpha_2 = 10$, and we sparsify the symbolic model every 1000 training steps with $\theta_{\text{thres}} = 5 \times 10^{-3}$. For the diffusive Lokta–Volterra system, we train for 100000 steps with learning rate 10^{-3} and hyperparameters $\alpha_1 = \alpha_2 = 1$, and we sparsify the symbolic model every 1000 training steps with $\theta_{\text{thres}} = 2 \times 10^{-3}$.

Phase Reconstruction

The system is sampled on a size 64 mesh with spacing $\Delta x = 2\pi/64$ for 500 time steps of size $\Delta t = 10^{-3}$.

Using the available prior knowledge, we allow the symbolic model to use spatial derivative terms $\partial_x^p \psi$ for $p \in \{1, 2, 3, 4\}$ and nonlinearity terms $|\psi|^q \psi$ for $q = \{2, 4, 6, 8\}$. We scale the effective time step by a factor of 10 to precondition the model coefficients and train for 100000 time steps with learning rate 10^{-4} and hyperparameters $\alpha_1 = \alpha_2 = 1$ and $\beta = 10^3$. We sparsify the symbolic model every 10000 training steps with $\theta_{\text{thres}} = 10^{-3}$.

Unlike the previous examples, reconstructing the phase is a much trickier problem that cannot be done using a local spatiotemporal encoder. Instead of using a neural network mapping, we use a direct embedding of the phase as function of time, i.e. for each point in the spatiotemporal grid of the original data, we learn a parameter for the phase $\hat{\varphi}(x, t)$. This simple approach has the advantage of being incredibly flexible but also more difficult to train, requiring an additional encoder regularization term

$$\mathcal{R}_{\text{enc}} = \beta \left(\frac{\partial \hat{\psi}}{\partial t} - \frac{\Delta \hat{\psi}}{\Delta t} \right)^2 \tag{S8}$$

that ensures the symbolic time derivatives match the finite difference time derivatives of the reconstructed state $\hat{\psi} = \mathbf{a}(|\psi|, \hat{\varphi})$. Unlike the compact neural network encoders from the previous experiments, this encoder also must scale with the dataset size and does not provide a useful mapping that can be used for future hidden state reconstruction.