Interpretable Physics-informed Machine Learning Methods for Scientific Modeling and Data Analysis

by

Peter Yucheng Lu

A.B., Harvard University (2016)

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author . . . .
Department of Physics
July 20, 2022

Certified by . . . .
Marin Soljačić
Professor of Physics
Thesis Supervisor

Accepted by . . . .
Deepto Chakrabarty
Associate Department Head of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

Interpretable Physics-informed Machine Learning Methods for Scientific Modeling and Data Analysis

by

Peter Yucheng Lu

Submitted to the Department of Physics
on July 20, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Physics

Abstract

With the recent advancement of modern machine learning methods, there are now many exciting opportunities to use machine learning in scientific research, including for modeling and data analysis. Machine learning has the potential to become an indispensable tool for scientific discovery, but it is often difficult to directly apply to scientific problems. Especially in the case of deep learning approaches, machine learning methods are often lacking in interpretability, robustness, out-of-distribution generalization, and data efficiency—all qualities that are necessary for many scientific and engineering applications. In this thesis, we will illustrate several approaches for addressing these issues using a variety of applications. First, we develop a physics-informed framework for partially observed system identification, showing how combining an encoder with a sparse symbolic model allows us to reconstruct unobserved hidden states as well as the exact governing equations. Then, we design a physics-informed deep representation learning architecture for analyzing spatiotemporal systems and demonstrate its ability to extract interpretable physical parameters, corresponding to uncontrolled variables, from time-series data. Finally, we use tools from optimal transport theory and manifold learning to develop a robust non-parametric method for discovering conservation laws, showing the advantage of using geometric machine learning methods to solve scientific problems. By designing physics-informed architectures and adapting representation learning methods for scientific applications, we can overcome many of the difficulties that are currently preventing machine learning from playing a more important role in scientific discovery and create more useful computational tools for scientists and engineers trying to analyze, understand, and model their data.

Thesis Supervisor: Marin Soljačić
Title: Professor of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I had a magnificent time at MIT over the past six years and would first and foremost like to thank my advisor Prof. Marin Soljačić, who invited me to join his group while I was still unsure of my own research interests, introduced me to the world of physics-informed machine learning, guided me through research projects and career choices, and gave me many opportunities to interact with the wider community of researchers interested in building new computational tools for science. I also sincerely thank Prof. John D. Joannopoulos and Prof. Phiala E. Shanahan for their helpful advice and unyielding support, as well as Prof. Jason W. Fleischer, Prof. J. Nathan Kutz, Prof. Justin Solomon, Prof. Max Tegmark, and Prof. Tess E. Smidt for providing feedback on my work, support for my research and career development, and inspiring discussions on the relationship between machine learning and physics.

I would like to thank Dr. Li Jing and Dr. Yichen Shen for welcoming me into the Soljačić group and giving me a chance to learn about machine learning, as well as Samuel Kim, Rumen Dangovski, Charlotte Loh, Andrew Ma, Ileana Rugina, Ziming Liu, Ali Ghorashi, Nicholas Rivera, Jamison Sloan, Dr. Thomas Christensen, Dr. Yannick Salamin, Dr. Yi Yang, Dr. Charles Roques-Carmes, and Dr. Josué López for being incredibly supportive collaborators, colleagues, and friends. In addition, I would like to thank my graduate school roommate and friend of 16 years Webster Guan for encouraging me to try machine learning through taking classes and joining hackathons. I am also grateful for all the amazing high school and undergraduate students—including Oreoluwa Alao, Joan Ariño Bernad, Sahil Pontula, Owen Dugan, Ruba Houssami, and Christine Yang—that I had the privilege of mentoring, often through the Research Science Institute (RSI) or the MIT UROP program. Working with excited and hardworking students is incredibly motivating and rewarding, and I hope to do more teaching and mentoring in the future.

Finally, I could not have made it through graduate school without the support of my wife Meng'ou (Mary), my mom and dad, my brother Albert, and many other family and friends.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Overview

In recent years, there has been a growing interest in the physics community for using modern machine learning approaches to tackle difficult problems in experimental data analysis as well as address important open questions in theoretical physics. However, despite the significant advances in machine learning over the past decade, we are still in the process of learning how to apply these methods to solve scientific problems. Deep learning, in particular, is often seen as a black box method that only provides predictions without interpretability, fails in unexpected ways, performs poorly outside of a training distribution, and requires vast amounts of available data. These issues are not disjoint and usually stem from the lack of a proper physics-informed prior to guide the design and training of machine learning models on physical systems. Thus, to solve hard problems in physics and other scientific fields, it is critical that we improve the interpretability, robustness, generalization performance, and data efficiency of current machine learning methods by creating new approaches that are specifically adapted for scientific applications.

Rather than simply providing black box predictions, interpretable physics-informed machine learning methods are designed to incorporate known physical laws, symmetries, and constraints, as well as provide interpretable results (e.g. learned interpretable representations) that characterize the underlying physical systems and lead to new scientific insights and discoveries. My current research [2, 72, 73, 89, 90] and this thesis focus on both:

1) designing physics-informed architectures to efficiently analyze, understand, and model complex physical systems—e.g. chaotic nonlinear dynamics—and

2) discovering interpretable physical features or representations in order to extract relevant information—e.g. uncontrolled variables or conserved quantities—from unknown or poorly understood physical systems.

In Chapter 2, we present a physics-informed machine learning framework for partially observed system identification (published as [90]). Determining 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. Achieving this kind of interpretable system identification is even more difficult for partially observed systems. We proposed a method that combines an encoder, for reconstructing hidden states from partial observations, with a sparse symbolic model, for learning the explicit governing equations. Unlike pure neural network architectures, using a sparse symbolic model provides a powerful inductive bias on the form of the governing equations and generalizes well on many physical systems [17]. Our tests show that this method can successfully reconstruct the full system state and identify the governing equations for a variety of ODE and PDE systems. This new approach not only helps us better understand incomplete real-world data, but also demonstrates the power of using an interpretable physics-informed symbolic model.

In Chapter 3, we design an unsupervised representation learning method for extracting interpretable physical parameters that correspond to uncontrolled variables (published as [89]). Experimental data is often affected by uncontrolled variables that make analysis and interpretation difficult. For spatiotemporal systems, this problem is further exacerbated by intricate and high dimensional dynamics. Using a variational autoencoder [53, 75] with a physics-informed architecture, our method is able to extract a latent space corresponding to the uncontrolled variables and, simultaneously, learn a tunable predictive model for the underlying system. We test our method on simulated data from a variety of spatiotemporal systems and show that we can accurately identify the relevant parameters and extract them from raw and even noisy spatiotemporal data. This work is an example of how physics-informed unsupervised representation learning can help analyze messy data from complex physical systems and identify new physics hiding within the data.

In Chapter 4, we introduce a robust non-parametric method for identifying conservation laws from trajectory data, using tools from optimal transport theory and manifold learning. Conservation laws are key theoretical and practical tools for understanding, characterizing, and modeling nonlinear dynamical systems. However, for many complex dynamical systems, the corresponding conserved quantities are difficult to identify, making it hard to analyze their dynamics and build efficient, stable predictive models. Current approaches for discovering conservation laws often depend on detailed dynamical information [66, 83], such as the equation of motion or fine-grained time measurements, with many recent proposals also relying on black box parametric deep learning methods [47, 83, 133]. We instead reformulate this task as a manifold learning problem and propose a non-parametric approach, combining the Wasserstein metric from optimal transport with diffusion maps, to discover conserved quantities that vary across trajectories sampled from a dynamical system. We test this new approach on a variety of physical systems—including conservative Hamiltonian systems, dissipative systems, and spatiotemporal systems—and demonstrate that our manifold learning method is able to both identify the number of conserved quantities and extract their values. Our proposed method provides a direct geometric approach to identifying conservation laws that is both robust and interpretable without requiring an explicit model of the system nor accurate time information.

In Chapter 5, we reflect on these proposed approaches and discuss future directions for building better physics-informed machine learning methods to aid in scientific modeling, data analysis, and discovery.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Discovering Sparse Interpretable Dynamics from Partial Observations

2.1 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 unobserved 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 [84]. 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 [45], such as data-driven prediction of nonlinear dynamics [12, 107, 108] including methods that only use partial observations [8, 24, 101, 115]. However, because deep learning models often fail to take into account

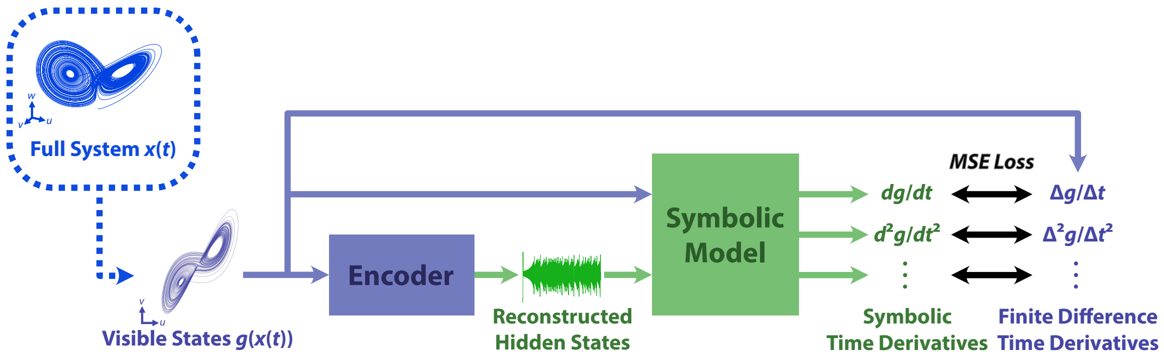


Figure 2.1: A machine learning framework for simultaneous system identification and state reconstruction. With only a visible portion of the full state available $\mathbf{x}_v = \mathbf{g}(\mathbf{x})$, an encoder is first used to reconstruct the hidden states. The fully reconstructed state $\hat{\mathbf{x}}$, 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.

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 new approaches incorporating intrinsic dimensionality estimation [23] and recent unsupervised learning methods [89] can help mitigate this issue. 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 [16, 89, 106, 140].

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 [19, 41, 95, 122]. However, these approaches still encounter barriers when dealing with certain types of nonlinear dynamics, such as chaos, which lead to a problematic continuous spectrum for the Koopman operator that cannot be modeled by a finite-dimensional linear system, although some progress has been made in addressing these limitations [18, 19, 91].

In this work, we choose to directly learn the symbolic governing equations of motion, which are often sparse and provide a highly interpretable representation of the dynamical system that also generalizes well. By fitting a symbolic model, we can capture the exact dynamics of the many physical systems in nature governed by symbolic equations. 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 [17, 28, 64]. This sparsity prior, in combination with an autoencoder architecture, can also aid in extracting interpretable state variables from high dimensional data [22].

We propose a machine learning framework for solving the common problem of partially observed system identification, where a portion of the system state is observed but the remaining hidden states as well as the underlying dynamics are unknown. Unlike in the generic high dimensional setting, this is a much more structured problem, and we take full advantage of this additional structure when designing our architecture. To deal with having only partial state information, our method 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. 2.1). 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.

2.2 Problem Formulation

Consider a nonlinear dynamical system defined by the first order ODE

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}). \tag{2.1}$$

The visible or observed 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 $\mathbf{x}_h$.

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).

2.3 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 $\{\hat{\mathbf{x}}_h(t_0), \hat{\mathbf{x}}_h(t_0 + \Delta t), \ldots, \hat{\mathbf{x}}_h(t_N)\}$. This should, in general, be possible for hidden states that are sufficiently coupled to the visible states due to Takens' embedding theorem [123]. 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

$$\frac{d\hat{\mathbf{x}}}{dt} = \hat{\mathbf{F}}_\theta(\hat{\mathbf{x}}) := \theta_1 \mathbf{f}_1(\hat{\mathbf{x}}) + \theta_2 \mathbf{f}_2(\hat{\mathbf{x}}) + \cdots + \theta_m \mathbf{f}_m(\hat{\mathbf{x}}), \tag{2.2}$$

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). If the dimensionality of the system state $\mathbf{x}$ is unknown, we can treat it as a hyperparameter, tuned to achieve an optimal trade off between high accuracy (e.g. low loss) and parsimony (e.g. fewer hidden states and model sparsity), or use recently suggested intrinsic dimensionality estimation methods [23].

To jointly train the encoder and symbolic model using only partial observations, we match 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. 2.2), so we develop and use an algorithmic trick that allows standard automatic differentiation methods [10] to compute higher order symbolic time derivatives of the *reconstructed* visible states $\mathbf{g}(\hat{\mathbf{x}})$ (see Appendix 2.A). 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

$$\mathcal{L}(\eta, \theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{p=1}^{M} \frac{\alpha_p}{\sigma_p^2} \left(\frac{d^p \mathbf{g}(\hat{\mathbf{x}}(t_i))}{dt^p} - \frac{\Delta^p \mathbf{x}_v(t_i)}{\Delta t^p} \right)^2, \tag{2.3}$$

where σ_p^2 is the empirical variance of the pth order finite difference derivative $\Delta^p \mathbf{x}_v(t_i)/\Delta t^p$, and the 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 [17]—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. While developing our approach, we also experimented with L_1 regularization but found that it tends to strongly degrade the performance of the learned model when applied with enough strength to achieve sparsity. See Appendix 2.B for additional architecture and training details.

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

2.4 Results

2.4.1 ODE Experiments

To demonstrate our method, we use data from two standard examples of chaotic nonlinear dynamics: the Rössler system (Fig. 2.2a) and the Lorenz system (Fig. 2.2b). 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—via the learned symbolic model—and reconstruct the hidden state w (Fig. 2.2). We achieve a hidden state 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.

Note that the hidden states discovered by our approach often differ from the ground truth hidden states by an affine transformation (e.g. $w' = aw + b$). In order to make a direct comparison, we fit the discovered hidden states to the true hidden states using linear regression and show the transformed result as the reconstructed governing equations and hidden states (Fig. 2.2).

2.4.2 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. 2.3a) and a 2D diffusive Lokta–Volterra predator–prey system (Fig. 2.3b)—commonly used for ecological modeling [33, 39, 76]. 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

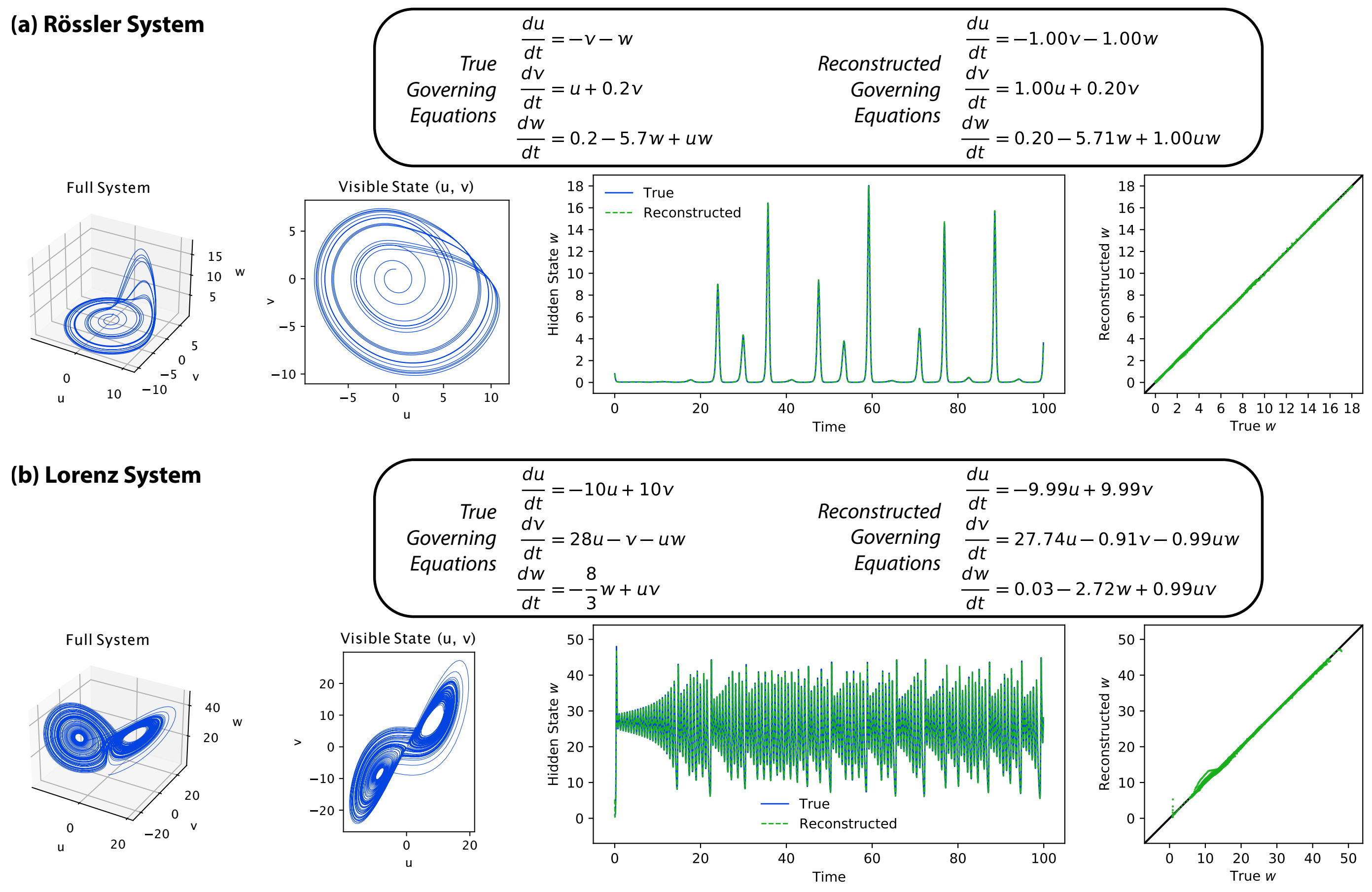


Figure 2.2: System identification and hidden state reconstruction for the (a) Rössler and (b) Lorenz systems. In both numerical experiments, the u and v components are visible while the w component is hidden. The true and reconstructed hidden states w are shown as a function of time and also plotted directly against each other for comparison. Note that the reconstructed hidden states shown here are directly reconstructed by the encoder from the corresponding visible states at nearby times and are not based on an autonomous prediction of the learned model.

equations and reconstruct the hidden component for both systems (Fig. 2.3), 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.

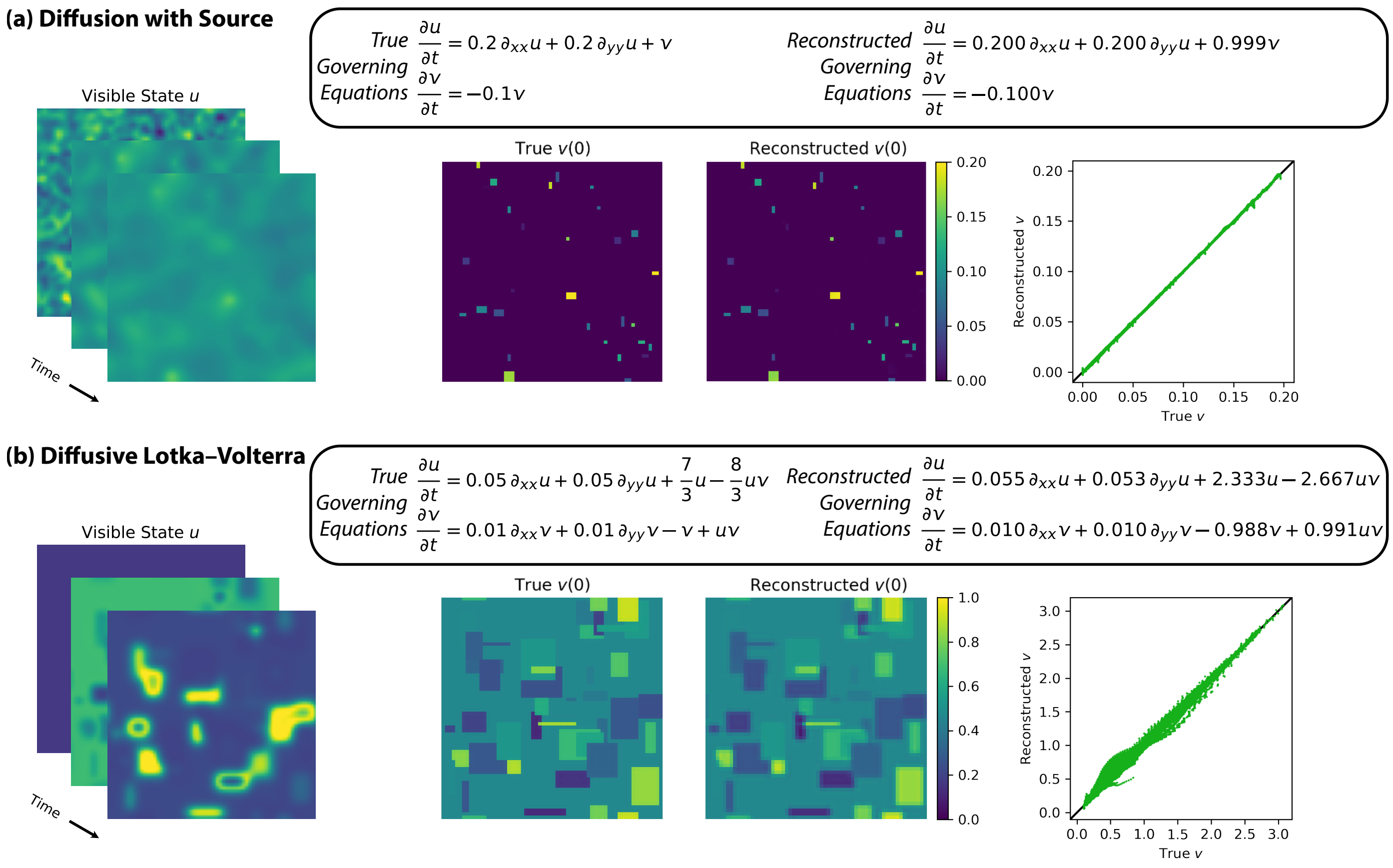


Figure 2.3: System identification and hidden state reconstruction for the (a) diffusion system with a decaying source term v and (b) diffusive Lokta–Volterra system. In both numerical experiments, the u component is visible while the v component is hidden. The true and reconstructed hidden states v are shown at time $t = 0$ and are also plotted directly against each other for comparison.

2.4.3 Phase Reconstruction

As a final example, we consider the phase reconstruction problem for the 1D nonlinear Schrödinger equation—a model for light propagation through a nonlinear fiber [1]—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 we have some prior knowledge about the structure of the dynamics: a complex wave equation with a global phase shift symmetry and only odd nonlinearities to model an optical material with inversion symmetry [1]. This allows us to limit the library of predefined terms used by our symbolic model. Our prior

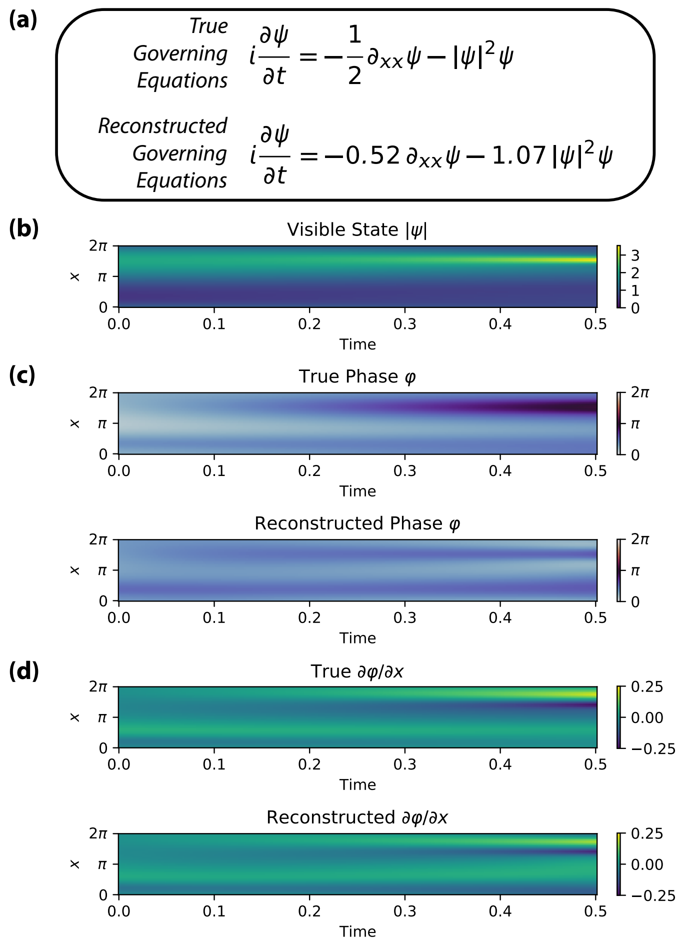


Figure 2.4: System identification (a) and phase reconstruction (c,d) for the nonlinear Schrödinger system. The magnitude $|\psi|$ of the wave is visible (b) while the phase $\varphi = \arg(\psi)$ is hidden (c) and must be reconstructed. The spatial derivative of the phase $\partial\varphi/\partial x$ (d) and its reconstruction are also shown.

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 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 [88].

2.5 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. By fitting symbolic models, we are able to discover the exact form of the symbolic equations governing the underlying physical systems, resulting in highly interpretable models and predictions (see Appendix 2.C). 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 [8, 24], our approach can be significantly more computationally 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 [106]. 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. Data smoothing techniques and careful tuning of our sparsity method help mitigate this to some extent (see Appendix 2.D.1) in a similar fashion to methods like SINDy [17, 64], and promising new methods for identifying the noise distribution alongside the dynamics [65] 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 (see Appendix 2.B) used in our final experiment on phase reconstruction has similarities with variational approaches used for equation discovery [28, 107, 109], and we believe that these variational methods can 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 [35, 73, 125]. These alternative symbolic architectures provide more powerful and flexible models with a sparse symbolic prior, potentially addressing some current limitations of our implementation (see Appendix 2.E) and allowing our framework to handle a wider range of governing equations—such as the Hill equations used in modeling gene expression [4]—without requiring large libraries of predefined terms.

2.A Automatic Computation of Symbolic Derivatives

The time derivatives can be derived by repeated differentiation of the symbolic model (Eq. 2.2) 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

$$\frac{dg_i}{dt} = \sum_j \frac{dg_i}{d\hat{x}_j}\frac{d\hat{x}_j}{dt} = \sum_j \frac{dg_i}{d\hat{x}_j}\hat{F}_{\theta j} \tag{2.4}$$

$$\frac{d^2 g_i}{dt^2} = \sum_{j,k} \frac{d^2 g_i}{d\hat{x}_j d\hat{x}_k}\hat{F}_{\theta j}\hat{F}_{\theta k} + \frac{dg_i}{d\hat{x}_j}\frac{d\hat{F}_{\theta j}}{d\hat{x}_k}\hat{F}_{\theta k}. \tag{2.5}$$

The expressions for the symbolic time derivatives (Eqs. 2.4 & 2.5) 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 [15]).

Automatic differentiation is the algorithmic backbone of modern deep learning [10], and a new generation of source-to-source automatic differentiation libraries are quickly becoming available [15, 57]. 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. 2.2) 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. 2.2), i.e.

$$\mathcal{I}(\hat{\mathbf{x}}(t), \epsilon) = \hat{\mathbf{x}}(t+\epsilon). \tag{2.6}$$

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

$$\begin{aligned} \frac{\partial \mathcal{I}(\hat{\mathbf{x}}(t), \epsilon)}{\partial \epsilon} &= \frac{d\hat{\mathbf{x}}(t+\epsilon)}{d\epsilon} \\ &= \hat{\mathbf{F}}_\theta(\hat{\mathbf{x}}(t+\epsilon)) \\ &= \hat{\mathbf{F}}_\theta(\mathcal{I}(\hat{\mathbf{x}}(t), \epsilon)), \end{aligned} \tag{2.7}$$

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

$$\frac{d^p \hat{\mathbf{x}}}{dt^p} = \left.\frac{\partial^p \mathcal{I}(\hat{\mathbf{x}}, \epsilon)}{\partial \epsilon^p}\right|_{\epsilon=0}. \tag{2.8}$$

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 $\mathcal{I}_0(\hat{\mathbf{x}}, \epsilon) := \hat{\mathbf{x}}$

that acts as the identity on the state $\hat{\mathbf{x}}$ but has a custom derivative rule

$$\frac{\partial \mathcal{I}_0(\hat{\mathbf{x}}, \epsilon)}{\partial \epsilon} := \hat{\mathbf{F}}_\theta(\mathcal{I}_0(\hat{\mathbf{x}}, \epsilon)). \tag{2.9}$$

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.

2.B Dataset, Architecture, and Training Details

The data and architecture requirements for using our approach are dependent on the properties of the dynamical system, the fraction of visible states, and the chosen symbolic model. For example, a more constrained symbolic model with a smaller library of terms will likely be more data efficient due to having a stronger inductive bias on the model. A smaller library also makes it easier for the sparse optimization to discover the correct sparsity pattern and thus accurate governing equations, so it is often better to start with a more constrained library of terms and then slowly expand it if model performance remains poor. In general, the trajectories from the data need to be long and varied enough in order to differentiate among the terms provided by the symbolic model, although we have found that a single trajectory is often sufficient for accurate state reconstruction and system identification. For the encoder architecture, we generally use small and relatively shallow neural networks, which already provide good hidden state reconstruction performance. This also means that our approach trains reasonably quickly, taking $\sim$ 2.5 minutes for the ODE systems on a single consumer GPU (GeForce RTX 2080 Ti) and $\sim$ 2 hours for the much larger PDE systems on four GPUs. However, it is certainly possible that more structured encoders may help in certain cases requiring more complex reconstructions. In addition, we are able to obtain accurate results in our tests by matching the first and second order time derivatives, although higher order derivatives will be necessary for datasets with a larger fraction of hidden states.

2.B.1 ODE Systems

Each system is sampled for 10000 time steps of size $\Delta t = 10^{-2}$, and the resulting time series data is 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 [145] 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 $\theta_{\text{thres}} = 10^{-3}$.

2.B.2 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 is 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 $\theta_{\text{thres}} = 2 \times 10^{-3}$.

2.B.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

$$\mathcal{R}_{\text{enc}} = \beta \left(\frac{\partial \hat{\psi}}{\partial t} - \frac{\Delta \hat{\psi}}{\Delta t} \right)^2 \tag{2.10}$$

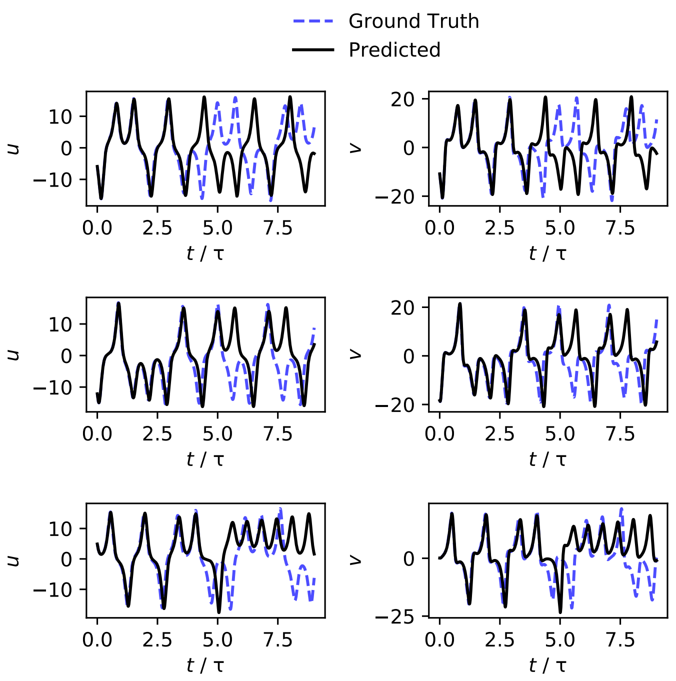


Figure 2.5: Three examples of prediction on test trajectories from the model trained on the Lorenz system with two visible states u, v. The time is given in units of the Lyapunov time τ, the characteristic time for two nearby trajectories to exponentially diverge in a chaotic system.

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.

2.C Prediction Examples

Because we are able to capture the true dynamics of each system using a symbolic model, our models exhibit excellent generalization performance beyond the training data. The expected accuracy of the predictions from the model is also highly interpretable, with prediction performance depending on the accuracy of the fitted coefficients of the symbolic model and the hidden state reconstruction of the initial state. Thus, for the purposes of method validation, it is often more instructive to directly

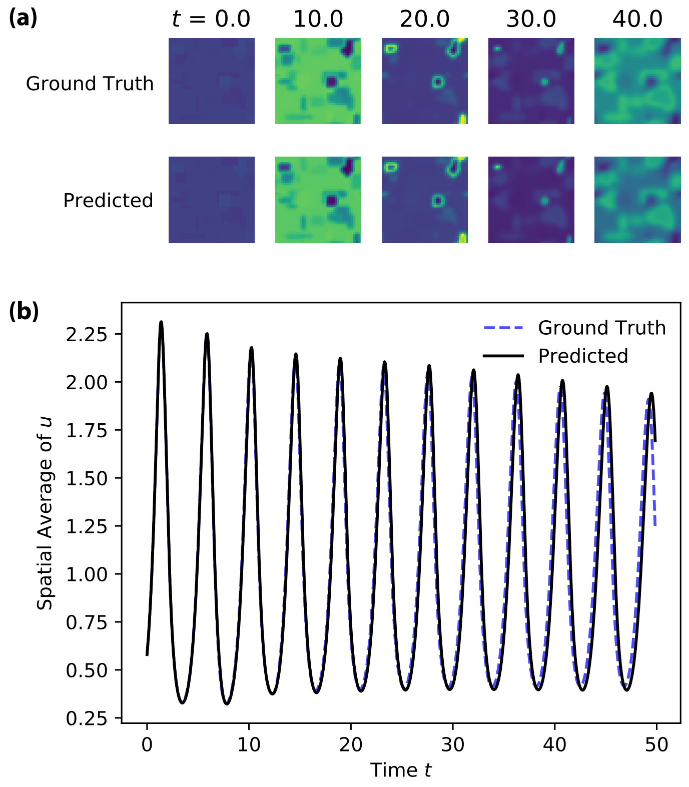


Figure 2.6: (a) Snapshots of the predicted vs. ground truth visible state $u(x, y, t)$ from the model trained on the diffusive Lokta–Volterra system with a single visible state. (b) Predicted vs. ground truth spatial average of the visible state u as a function of time.

examine the coefficients of the symbolic model and the quality of the hidden state reconstruction when judging the success of our method, both of which are discussed in the main text. However, prediction performance still represents an important metric to judge the quality of the resulting model when applying our approach to a new unknown dynamical system. Here, we show prediction examples from the Lorenz system (Fig. 2.5) and the diffusive Lokta–Volterra system (Fig. 2.6), demonstrating the quality and interpretability of the prediction performance that is expected from well-trained symbolic models.

Because the Lorenz system is chaotic, long-term prediction performance has a theoretically limit characterized by the Lyapunov time τ of the system. Despite this, we are still able to predict well up to $\sim 4\tau$ and capture the correct behavior of the system using our symbolic model (Fig. 2.5). Prediction performance on the diffusive Lokta–Volterra system is also very good (Fig. 2.6) with the slight deviations at long times resulting from imperfections in the initial hidden state reconstruction and small errors in the learned coefficients of the symbolic model. These are highly interpretable sources of error that are easy to analyze and iteratively refine.

2.D Additional Experiments: Lorenz System

2.D.1 Noise Robustness

Because our approach relies on empirically estimated time derivatives, it inherits the noise susceptibility of derivative estimation methods such as finite differences. To examine this issue, we perform a simple test of our method on the Lorenz system with added Gaussian noise ($\sigma = 10^{-3}$, 10^{-2}, 10^{-1}) using the same setup and hyperparameters as described in Section 2.4.1 and Appendix 2.B.1. Without any data smoothing, our method is able to discover the correct governing equations with added $\sigma = 10^{-3}$ noise, but is unable to do so for $\sigma = 10^{-2}$, 10^{-1}. With some simple data smoothing (quadratic Savitzky–Golay filter with window size 5), our method is also able to discover the correct governing equations for $\sigma = 10^{-2}$ but continues to fail for $\sigma = 10^{-1}$ (Table 2.1).

2.D.2 Alternative Visible Variables

For completeness, we also test variations on the chosen visible states for the Lorenz system, taking the visible states to be (v, w) and (u, w). We use the same hyperparameters as described in Appendix 2.B.1 with the exception of increasing the sparsity threshold to $\theta_{\text{thres}} = 5 \times 10^{-3}$. In both cases, we are able to identify the correct governing equations and reconstruct the missing hidden state (Table 2.1). One quirk of the (u, w) case is that the discovered hidden state is not simply an affine transformation of the true hidden state v but is instead a linear combination of u and v, i.e. $v' = av + bu + c$.

We also test using a single visible state u, which requires fitting third order time derivatives, and found that our current implementation is unsuccessful due to a failure

Table 2.1: Reconstructed governing equations (after an affine transformation of the hidden states) and relative hidden state reconstruction errors for the additional experiments using the Lorenz system. In order, we have: 3 experiments with added noise, 3 experiments with added noise and data smoothing, 2 experiments with alternative pairs of visible states, and 2 experiments with a single visible state u (with and without a fixed sparsity pattern). The added noise experiments use the default two visible states (u, v). "–" indicates an experiment that failed to recover the correct governing equations.

	Recon. Governing Equations	Hidden State Recon. Error
Ground Truth	$du/dt = -10u + 10v$ $dv/dt = 28u - v - uw$ $dw/dt = -8/3w + uv$	N/A
$\sigma = 10^{-3}$	$du/dt = -9.99u + 9.99v$ $dv/dt = 27.75u - 0.91v - 0.99uw$ $dw/dt = 0.03 - 2.73w + 0.99uv$	2.1×10^{-3}
$\sigma = 10^{-2}$	–	–
$\sigma = 10^{-1}$	–	–
$\sigma = 10^{-3}$ (smoothed)	$du/dt = -9.99u + 9.99v$ $dv/dt = 27.73u - 0.91v - 0.99uw$ $dw/dt = 0.03 - 2.72w + 0.99uv$	2.0×10^{-3}
$\sigma = 10^{-2}$ (smoothed)	$du/dt = -10.00u + 10.00v$ $dv/dt = 28.22u - 0.86v - 1.02uw$ $dw/dt = 0.05 - 2.75w + 0.97uv$	1.5×10^{-2}
$\sigma = 10^{-1}$ (smoothed)	–	–
Visible (v, w)	$du/dt = -9.88u + 9.88v$ $dv/dt = -0.22 + 27.67u - 0.95v - 0.99uw$ $dw/dt = -0.61 - 2.65w + 1.00uv$	1.5×10^{-3}
Visible (u, w)	$du/dt = -9.98u + 9.98v$ $dv/dt = 27.88u - 1.00v - 0.99uw$ $dw/dt = 0.88 - 2.71w + 0.99uv$	2.8×10^{-4}
Visible u	–	–
Visible u, fixed sparsity	$du/dt = -9.85u + 9.87v$ $dv/dt = 28.06u - 1.02v - 1.00uw$ $dw/dt = 1.30 - 2.70w + 1.00uv$	1.1×10^{-3} (v), 4.6×10^{-3} (w)

to identify the correct sparsity pattern. However, given the right sparsity pattern, our method quickly converges to the correct governing equations (Table 2.1). This result is discussed further in Appendix 2.E.1.

2.E Limitations

2.E.1 Reconstructing Hidden States

While we have demonstrated that our approach performs well over a wide range of tasks, there are still limitations to its ability to reconstruction hidden states and therefore identify the correct symbolic models. General theoretical limitations include the degree of interaction between the visible and hidden states as well as measurement noise. If a hidden state does not significantly affect the dynamics of the available visible states, it will be very difficult or impossible for any method to reconstruct. In fact, the degree to which different states are coupled can be tested using convergent cross mapping [120]. Furthermore, for data with a smaller fraction of visible states and thus more hidden states, the reconstruction task becomes harder. This manifests itself in our framework as requiring higher and higher order time derivative matching in order to fully reconstruct the hidden states, resulting in more sensitivity to noise. For example, if we want to use no more than second order time derivatives in our loss function, we require at least $N/2$ visible states for an N-dimensional system.

There are also more subtle issues with sparse optimization that causes problems for our current implementation. For example, if we attempt to reconstruct the Rossler or Lorenz system using only a single visible state, our symbolic model tends to get stuck in local minima and fails to find the correct sparsity pattern, i.e. the right sparse combination of terms. However, if we provide the correct sparsity pattern to the symbolic model, we obtain very accurate results on par with what we have shown for the case of two visible states. That is, given the right sparsity pattern, our method is able to learn an accurate reconstruction of the hidden states and also fit the coefficients of the symbolic model using only a single visible state. On the one hand, this highlights the advantages of having strong inductive biases, which can help avoid some of these bad local minima. On the other hand, there is a clear need for a more sophisticated approach to sparsity and symbolic regression [35, 73, 125] as well as potential for better physics-informed model initialization schemes, which we hope to explore further in the future.

2.E.2 Uncertainty Quantification

Like other sparse identification approaches [17], our method does not currently provide explicit uncertainty quantification. For uncertainty due to noisy data, one reasonable approach would be to assume a noise model for the data and use Bayesian inference methods to estimate uncertainty in the symbolic model coefficients [54]. However, because our framework also involves an encoder that is often a neural network, there is additional model uncertainty that is generally difficult to quantify. Estimating uncertainty for neural network models is an active area of research with promising options including ensemble methods [42] and other Bayesian neural network approaches [128].

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Extracting Interpretable Physical Parameters from Spatiotemporal Systems using Unsupervised Learning

3.1 Introduction

Physics has traditionally relied upon human ingenuity to identify key variables, discover physical laws, and model dynamical systems. With the recent explosion of available data coupled with advances in machine learning, fundamentally new methods of discovery are now possible. However, a major issue with applying these novel techniques to scientific and industrial applications is their interpretability: neural networks and deep learning are often seen as inherently black box methods. To make progress, we must incorporate scientific domain knowledge into our network architecture design and algorithms without sacrificing the flexibility provided by deep learning models [69, 70, 127]. In this work, we show that we can leverage unsupervised learning techniques in a physics-informed architecture to build models that learn to both identify relevant interpretable parameters and perform prediction. Because relevant parameters are necessary for predictive success, the two tasks of extracting parameters and creating a predictive model are closely linked, and we exploit this relationship to do both using a single architecture.

We focus our attention on spatiotemporal systems with dynamics governed by partial differential equations (PDEs). These systems are ubiquitous in nature and include physical phenomena in fluid dynamics and electromagnetism. Recently, there has been significant interest in the data-driven analysis and discovery of PDEs: e.g. explicitly identifying PDEs using sparse linear regression with a library of possible terms [113], using a convolutional architecture with symbolic regression to identify PDEs [86, 87], and representing PDE solutions as neural networks to solve and identify PDEs [12, 107, 108]. However, previous works on PDE discovery and parameter extraction often assume the entire dataset is governed by the same dynamics and also explicitly provide the key dynamical parameters (with potentially unknown values). In more complex scenarios, we may have limited control over the systems that we are studying and yet still want to model them and extract relevant physical features from their dynamics. If we attempt to study such systems by naïvely training a predictive model, we are likely to fail in one of two ways: first, a single explicit PDE model will be unable to capture the variations in the dynamics caused by uncontrolled variables in the data, and second, a generic deep learning method for time-series prediction such as long short-term memory (LSTM)-based models [44, 55] will not be interpretable or provide any physical insight and may also result in unphysical solutions at later times due to overfitting. To avoid these problems and gain a better understanding of the physical system, we must first identify important parameters or variables that are uncontrolled and that change in the raw data, producing varying dynamics. Recent work on learning parametric PDEs has taken steps toward addressing this issue [112]. We will use an unsupervised learning method to automate the process of determining the relevant parameters that control the system dynamics and constructing a predictive model—all without requiring information on the form of the governing PDE.

We propose a model architecture (Fig. 3.1(a)) based on variational autoencoders (VAEs) [75]. VAEs are widely used for dimensionality reduction and unsupervised learning tasks [45] and have been shown to be effective for studying a wide variety of physical phenomena, e.g. discovering simple representations of systems in classical and quantum mechanics [58], modeling protein folding and molecular dynamics [52, 130], and identifying condensed matter phase transitions [36, 132]. In terms of interpretability, the VAE architecture and its derivatives have also been shown to disentangle independent factors of variation [20, 25, 53]. The choice of a VAE-based architecture is motivated both by their prior success in extracting useful representations and by their strong theoretical foundation (Appendix 3.D). Prior works have also applied other methods of parameter identification, such as using principal component analysis (PCA) as a post-processing step with a standard autoencoder [144]. This, however, relies heavily on the poorly understood implicit regularization of the neural network architecture rather than the explicit regularization of a VAE, and, in our experience, VAEs produce more consistent and interpretable results.

Our architecture consists of an encoder (Fig. 3.1(b)) that extracts physical parameters characterizing the system dynamics and a decoder (Fig. 3.1(c)) that acts as a predictive model and propagates an initial condition forward in time given the extracted parameters. This differs from a traditional VAE due to the additional initial condition provided to the decoder, allowing the encoder to focus on extracting latent parameters that parameterize the dynamics of the system rather than the physical state. Our architecture can be thought of as a conditional VAE [119], although only the decoder is conditional. While similar architectures have been recently proposed for physical systems such as interacting particles [144] and moving objects [137], our model is specifically designed to study spatiotemporal phenomena, which have a continuous set of degrees of freedom.

To take advantage of the spatiotemporal structure of PDE-governed systems, we use convolutional layers—commonly employed in image recognition tasks [50, 78] to efficiently represent local features—in both the encoder and decoder portions of our architecture. The translation invariance of the convolutions allows us to train on small patches of data and then evaluate on larger systems with arbitrary boundary conditions. In the decoder, the convolutional layers are placed in a recurrent architecture to represent time propagation—analogous to a PDE solver with a finite difference approximation [86]. In addition, our architecture efficiently parameterizes PDE propagation by dynamically generating the convolutional kernels and biases of the decoder using the extracted latent parameters from the encoder. In this way, the latent parameters directly control the local propagation of the physical states in the decoder, resulting in more stable model predictions and a more physical encoding of the dynamics. These architecture choices provide the key physics-informed inductive biases that enhance the interpretability of the extracted parameters and ensure a physically reasonable predictive model.

To demonstrate the capabilities of this approach, we test our method on simulated data from PDE models for chaotic wave dynamics, optical nonlinearities, and convection and diffusion (Sec. 3.3). These numerical experiments show that our method can accurately identify and extract relevant physical parameters that characterize variations in the observed dynamics of a spatiotemporal system (Sec. 3.4.1), while at the same time construct a flexible and transferable predictive model (Sec. 3.4.2). We further show that the parameter extraction is robust to noisy data and can still be effective for chaotic systems where accurate prediction is difficult. Finally, we apply this method to nonlinear optical fiber propagation using data generated from an ab-initio electromagnetic simulation to test the model on a more realistic dataset (Sec. 3.5). The goal of our approach is to provide an additional tool for studying complex spatiotemporal systems when there are unknown and uncontrolled variables present.

3.2 Model Architecture

Our model (Fig. 3.1) has an encoder–decoder architecture based on a variational autoencoder (VAE) [53, 75]. Given a dataset of time-series from a spatiotemporal system, the *dynamics encoder* (DE) extracts latent parameters which parameterize the varying dynamics in the dataset. These latent parameters are then used by the *propagating decoder* (PD) to simulate the system given an initial condition and boundary conditions. During training, the model is optimized to match the output of the PD to a time-series example from the dataset. The goal of the VAE architecture is

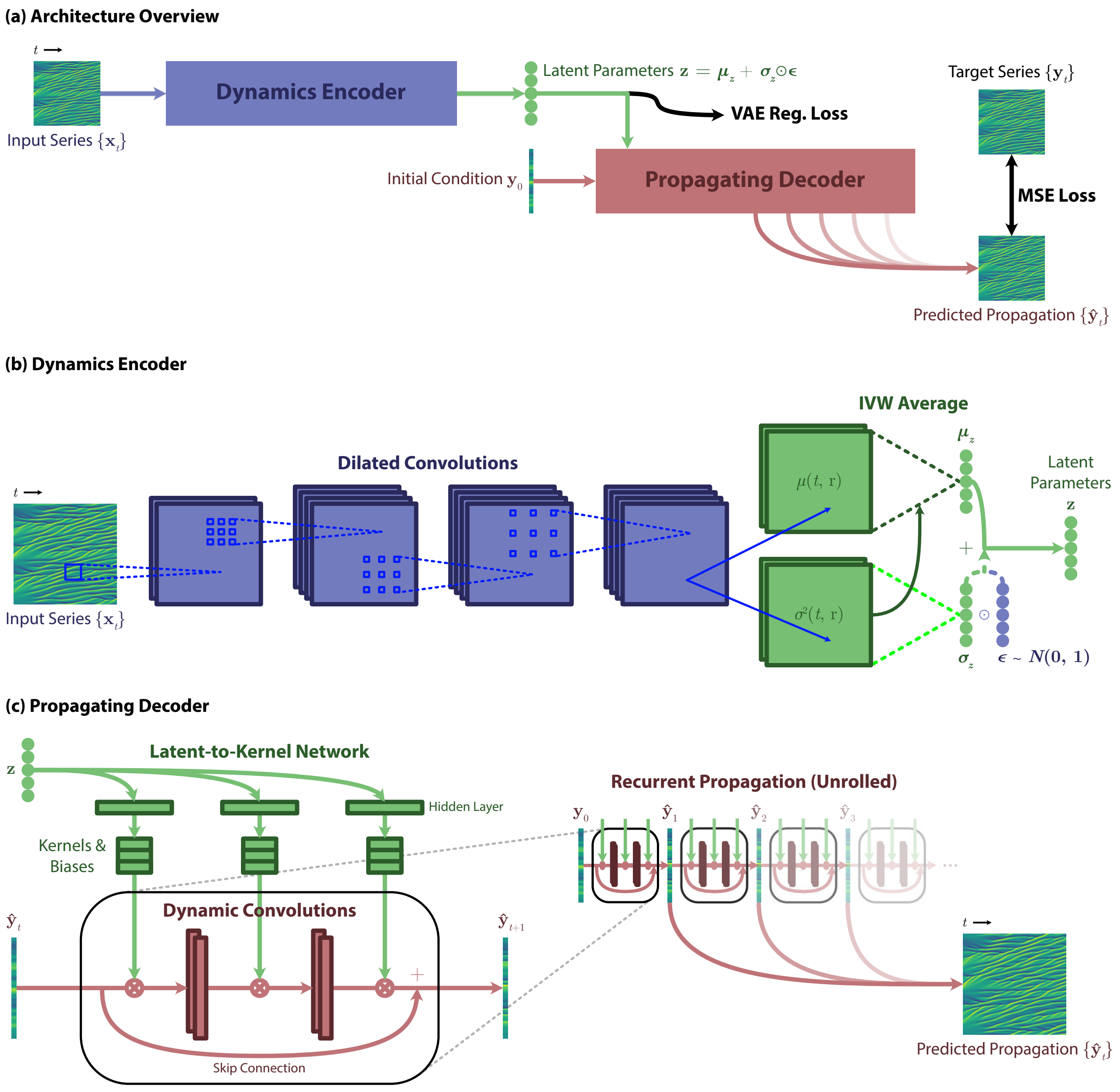


Figure 3.1: VAE-based model architecture. (a) The architecture consists of the dynamics encoder (DE) and the propagating decoder (PD) with kernels and biases given by a latent-to-kernel network. (b) The DE extracts the latent distribution parameters μ_z and σ_z from the input series $\{\mathbf{x}_t\}_{t=0}^{T_x}$ using dilated convolutions and inverse-variance weighted (IVW) averaging. During training, the latent parameters are sampled from the extracted distribution $z \sim \mathcal{N}(\mu_z, \sigma_z^2)$. (c) The PD then uses a fully-connected latent-to-kernel network to map the latent parameters z to the kernels and biases of the dynamic convolutional layers, which are used in a recurrent fashion to predict the propagation of the system $\{\hat{\mathbf{y}}_t\}_{t=1}^{T_y}$ starting from the initial condition $\hat{\mathbf{y}}_0 = \mathbf{y}_0$. The model is trained end-to-end using the mean squared error (MSE) loss between the predicted propagation series $\{\hat{\mathbf{y}}_t\}_{t=1}^{T_y}$ and target series $\{\mathbf{y}_t\}_{t=1}^{T_y}$ along with VAE regularization. Time-series limits are dropped in the figure labels for conciseness.

to allow the PD to push the DE to extract useful and informative latent parameters.

For training, the network requires time-series data that are grouped in pairs: the *input series* $\{\mathbf{x}_t\}_{t=0}^{T_x}$ is the input to the DE, and the *target series* $\{\mathbf{y}_t\}_{t=0}^{T_y}$ provides the initial condition $\mathbf{y}_0$ and the training targets $\{\mathbf{y}_t\}_{t=1}^{T_y}$ for the PD. Each pair of time-series $(\{\mathbf{x}_t\}_{t=0}^{T_x}, \{\mathbf{y}_t\}_{t=0}^{T_y})$ must follow the same dynamics and thus correspond to the same latent parameters. We can construct such a dataset from the raw data by cropping each original time-series to produce a pair of smaller time-series. This cropping can be performed randomly in both the time and space dimensions, which allows the network to train on a reduced system size while still making use of all the available data. In our examples, we also choose to crop dynamically during training, akin to data augmentation methods used in image recognition [50].

In detail, the DE network takes the full input series $\{\mathbf{x}_t\}_{t=0}^{T_x}$ and outputs a mean $\mu_{z|\mathbf{x}}$ and a variance $\sigma_{z|\mathbf{x}}^2$, which we will henceforth refer to as μ_z and σ_z^2 for compactness, representing a normal distribution for each latent parameter z. During training, each z is sampled from its corresponding distribution $\mathcal{N}(\mu_z, \sigma_z^2)$ using the VAE reparameterization trick: $z = \mu_z + \sigma_z \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$ is independently sampled for every training example during each training step. During evaluation, we simply take $z = \mu_z$. These parameters z along with an initial condition—the first state $\mathbf{y}_0$ in the target series—are then used by the PD network to predict the future propagation of the system. The predicted propagation series $\{\hat{\mathbf{y}}_t\}_{t=1}^{T}$ produced by the PD can be computed up to an arbitrary future time T, where $T = T_y$ during training to match the target series. By providing the PD with an initial condition, we allow the DE to focus on encoding parameters that characterize the dynamics of the data rather than encoding a particular state of the system. This is further reinforced by training on randomly cropped pairs of time-series as well as by the VAE regularization term (Appendix 3.D).

The full architecture is trained end-to-end using a mean-squared error loss between the predicted propagation series $\{\hat{\mathbf{y}}_t\}_{t=1}^{T_y}$ from the PD and the target series $\{\mathbf{y}_t\}_{t=1}^{T_y}$. We also add the VAE regularization loss—the KL divergence term $D_{\mathrm{KL}}(\mathcal{N}(\mu_z, \sigma_z^2) \parallel \mathcal{N}(0, 1))$—which encourages each latent parameter distribution $\mathcal{N}(\mu_z, \sigma_z^2)$ generated by the DE to approach the standard normal prior distribution $\mathcal{N}(0,1)$. The total loss function is given by

$$\mathcal{L} = \frac{1}{T_y} \sum_{t=1}^{T_y} (\mathbf{y}_t - \hat{\mathbf{y}}_t)^2 + \beta \sum_z D_{\mathrm{KL}}(\mathcal{N}(\mu_z, \sigma_z^2) \parallel \mathcal{N}(0,1)), \tag{3.1}$$

where T_y is the length of the target series without the initial $\mathbf{y}_0$, and β is a regularization hyperparameter which we tune for each dataset. The β parameter is key to learning disentangled representations [20, 53]. By using the VAE sampling method and regularizer, we compel the model to learn independent and interpretable latent parameters (Appendix 3.D). For additional training details, see Appendix 3.B.

The source code for our implementation is available at `https://github.com/peterparity/PDE-VAE-pytorch`.

3.2.1 Dynamics Encoder (DE)

The dynamics encoder (DE) network is designed to take advantage of existing symmetry or structure in the time-series data. We implement the DE as a deep convolutional network in both the time and space dimensions to allow the network to efficiently extract relevant features. To ensure the DE can handle arbitrary system sizes and time-series lengths, the architecture only contains convolutional layers with a weighted average applied at the output to obtain the latent parameters. The weights for the final averaging are also learned by the network and interpreted as variances so that the overall variance can also be computed. In this way, the network is able to focus on areas of the input series that are most important for estimating the latent parameters, akin to a visual attention mechanism [136].

Explicitly, we first compute local quantities

$$\mu(t, \mathbf{r}) = f_{\mathrm{DE},\mu}(\{\mathbf{x}_{t'}\}_{t'=0}^{T_x}) \tag{3.2}$$

$$\log \sigma^2(t, \mathbf{r}) = f_{\mathrm{DE},\sigma^2}(\{\mathbf{x}_{t'}\}_{t'=0}^{T_x}), \tag{3.3}$$

where $f_{\mathrm{DE},\mu}$ and f_{DE,σ^2} are multilayer convolutional networks in space and time (see

Appendix 3.A for details). Then, instead of using a fully-connected layer to compute the final mean μ_z and variance σ_z^2 for each latent parameter, we combine the local quantities by performing an inverse-variance weighted average using weights given by

$$w(t, \mathbf{r}) = \sigma^{-2}(t, \mathbf{r}) / \sum_{t,\mathbf{r}} \sigma^{-2}(t, \mathbf{r}) \tag{3.4}$$

to obtain

$$\mu_z = \sum_{t,\mathbf{r}} w(t, \mathbf{r})\, \mu(t, \mathbf{r}) \tag{3.5}$$

$$\sigma_z^2 = C^d / \sum_{t,\mathbf{r}} \sigma^{-2}(t, \mathbf{r}), \tag{3.6}$$

where C is a constant chosen to correct for the correlations between nearby points and d is the total number of time and space dimensions of the input. This averaging serves two purposes: it allows the DE to scale to arbitrary system sizes and geometries, and it improves the parameter extraction by placing greater emphasis on regions of high confidence. Assuming that non-overlapping patches should be treated as independent while overlapping patches are increasingly correlated, we take $C = 31$ to be the linear size of the receptive field of the convolutional networks $f_{\mathrm{DE},\mu}$ and f_{DE,σ^2}.

3.2.2 Propagating Decoder (PD)

The propagating decoder (PD) network is designed as a predictive model for spatiotemporal systems. We structure the PD as a multilayered convolutional network f_{PD} (see Appendix 3.A for details) with a residual skip connection that maps a state $\hat{\mathbf{y}}_t$ to the next state in the time-series $\hat{\mathbf{y}}_{t+1}$. Thus, each propagation step is given by

$$\hat{\mathbf{y}}_{t+1} = \hat{\mathbf{y}}_t + f_{\mathrm{PD}}(\hat{\mathbf{y}}_t). \tag{3.7}$$

To generate the predicted propagation series $\{\hat{\mathbf{y}}_t\}_{t=1}^{T_y}$ for comparison with the target series $\{\mathbf{y}_t\}_{t=1}^{T_y}$, we begin with the initial condition $\hat{\mathbf{y}}_0 = \mathbf{y}_0$ and then recursively apply the PD to propagate $\hat{\mathbf{y}}_t \to \hat{\mathbf{y}}_{t+1}$, forming a recurrent network. The PD acts as a physics simulator or, in this case, a PDE integrator with explicit time stepping. This architecture reflects both the spatial and temporal structure of PDE-governed systems and incorporates boundary conditions by properly padding $\hat{\mathbf{y}}_t$ at each time step before applying the convolutional layers. For example, to use periodic boundary conditions during evaluation, we apply periodic padding at each time step. During training, we treat the edges of the target series—cropped from a full training example—as a boundary condition by using the spatial boundary of each $\mathbf{y}_t$ in the target series to pad the corresponding state $\hat{\mathbf{y}}_t$ before propagation.

Unlike the convolutional layers of the DE, the kernels and biases for f_{PD} are not directly trained. Instead, the kernel weights and biases are a function of the latent parameters z. This type of layer is known as a dynamic convolution [60] or a cross-convolution [137]. Each convolutional kernel and corresponding bias is constructed by a separate fully-connected *latent-to-kernel* network that maps the latent parameters to each kernel or bias, forming a multiplicative connection in the PD. Thus, we can interpret the PD convolutional kernels and biases as encoding the dynamics of the system parameterized by z.

3.3 Simulated PDE Datasets

To study the ability of our architecture to perform parameter extraction, we generate simulated datasets of spatiotemporal systems that have spatially uniform, time-independent local dynamics in a box with periodic boundary conditions, i.e. we consider PDEs of the form

$$\frac{\partial \mathbf{u}(t, \mathbf{r})}{\partial t} = F(\mathbf{u}, \nabla \mathbf{u}, \mathbf{u}^2, (\mathbf{u} \cdot \nabla)\mathbf{u}, \ldots), \tag{3.8}$$

where F is a general space- and time-independent, nonlinear local operator acting on $\mathbf{u}$. This allows us to design an optimized, physics-informed model architecture. We test our model on a variety of spatiotemporal systems by creating the following three datasets that cover linear, nonlinear, and chaotic dynamics as well as giving both 1D and 2D examples. For details on the generation of the simulated datasets, see Appendix 3.C.

3.3.1 1D Kuramoto–Sivashinsky

The Kuramoto–Sivashinsky equation

$$\frac{\partial u}{\partial t} = -\gamma \, \partial_x^4 u - \partial_x^2 u - u \, \partial_x u \tag{3.9}$$

is nonlinear scalar wave equation with a viscosity damping parameter γ. This is a key example of a chaotic PDE [94] due to the instability caused by the negative second derivative term and was originally derived to model laminar flame fronts [79, 118]. The 1D Kuramoto–Sivashinsky dataset has a training set with 5,000 examples and a test set with 10,000 examples.

3.3.2 1D Nonlinear Schrödinger

The nonlinear Schrödinger equation

$$i\frac{\partial \psi}{\partial t} = -\frac{1}{2}\partial_x^2 \psi + \kappa \left|\psi\right|^2 \psi \tag{3.10}$$

is a complex scalar wave equation with a cubic nonlinearity controlled by the coefficient κ. In our data, we represent $\psi = u_1 + iu_2$ as a real two-component vector $\mathbf{u} = (u_1, u_2)$. This equation can be used to model the evolution of wave-packets in nonlinear optics and is known to exhibit soliton solutions [1]. The 1D nonlinear Schrödinger dataset has a training set with 5,000 examples and a test set with 10,000 examples.

3.3.3 2D Convection–Diffusion

The 2D convection–diffusion equation

$$\frac{\partial u}{\partial t} = D\nabla^2 u - \mathbf{v} \cdot \nabla u \tag{3.11}$$

Table 3.1: R^2 correlation coefficients from linear fits of the relevant latent parameters (Fig. 3.2) with the ground truth physical parameters for each dataset—both with and without added noise. For the three-parameter convection–diffusion dataset, the diffusion constant D is fit with a corresponding extracted latent parameter, while the drift velocity components v_x, v_y are fit with a corresponding two-dimensional subspace of the latent parameters due to the inherent rotational symmetry.

Dataset	Param.	No Noise	$\sigma = 0.1$ Noise
Kuramoto–Sivashinsky	γ	0.993	0.995
Nonlinear Schrödinger	κ	0.997	0.998
Convection–Diffusion	D	0.963	0.959
Convection–Diffusion	v_x	0.997	0.994
Convection–Diffusion	v_y	0.998	0.996

is a linear scalar wave equation consisting of a diffusion term with constant D and a velocity-dependent convection term with velocity field $\mathbf{v}$. The equation describes a diffusing quantity that is also affected by the flow or drift of the system, e.g. dye diffusing in a moving fluid. We consider the case of a constant velocity field. The 2D convection–diffusion dataset has a training set with 1,000 examples and a test set with 1,000 examples.

3.4 Numerical Experiments

We perform numerical experiments by training the model on both the original noiseless datasets and the datasets with added $\sigma = 0.1$ Gaussian noise—corresponding to 10% noise relative to the initial conditions. Then, we evaluate the trained models on the full size noiseless test set examples (no cropping). By also training on noisy datasets, we test the robustness of our method and show the effect of noise on the extracted parameters and prediction performance.

3.4.1 Parameter Extraction

During training, the model will only use a minimal set of latent parameters to encode the variation in the dynamics and will align each latent parameter in this subspace

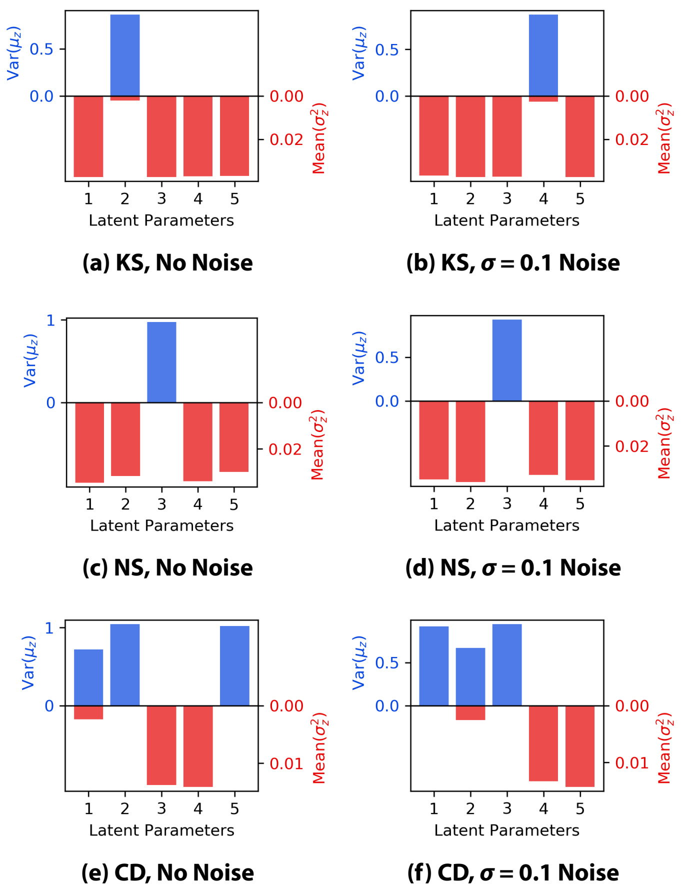


Figure 3.2: Identification of relevant latent parameters: the variance of μ_z (blue) and mean of σ_z^2 (red) for the five latent parameters in the models trained on the (a), (b) 1D Kuramoto–Sivashinsky (KS), (c), (d) 1D nonlinear Schrödinger (NS), and (e), (f) 2D convection–diffusion (CD) datasets both with and without added noise. In each case, the model has correctly identified the number of relevant parameters (one for the Kuramoto–Sivashinsky and nonlinear Schrödinger datasets, and three for the convection–diffusion dataset), which are characterized by high variance in μ_z and a low mean σ_z^2. These relevant latent parameters correspond to interpretable physical parameters that parameterize the dynamics of the system. The other latent parameters with near zero variance in μ_z and high mean σ_z^2 have collapsed to the prior and are non-informative. Note that while one would expect these collapsed parameters to have $\sigma_z^2 = 1$, the actual extracted σ_z^2 for the collapsed non-informative parameters is less than one. This is an artifact of evaluating the model on a larger system size and longer time-series than the cropped patches used during training (see Appendix 3.E for details).

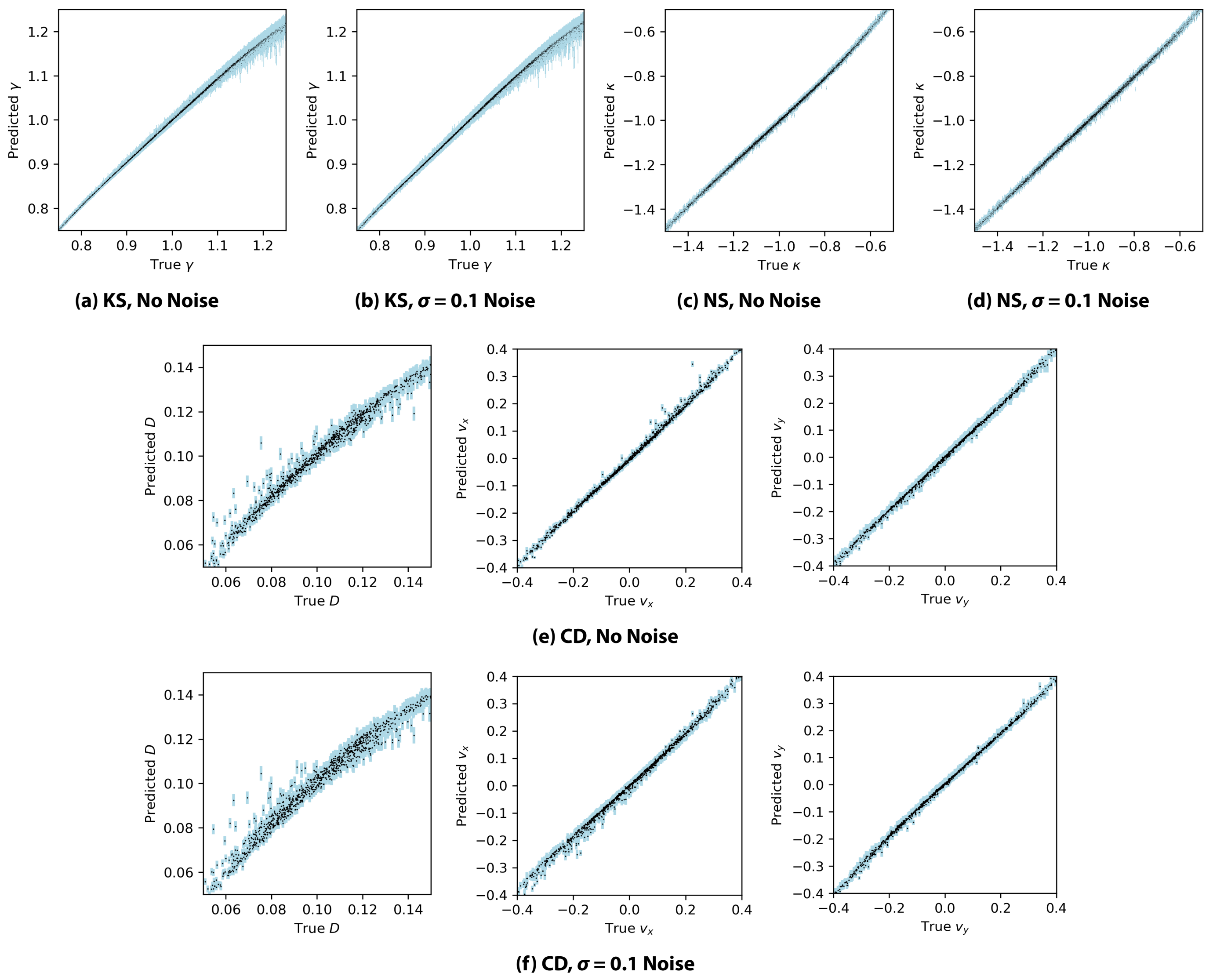


Figure 3.3: Predicted physical parameters from a linear fit with the relevant latent parameter (Fig. 3.2) vs. the ground truth physical parameters from the (a), (b) 1D Kuramoto–Sivashinsky (KS), (c), (d) 1D nonlinear Schrödinger (NS), and (e), (f) 2D convection–diffusion (CD) datasets. Because the drift velocity $\mathbf{v}$ from the CD dataset has an inherent rotational symmetry, they are encoded in a two-dimensional latent subspace (Table 3.2), so we instead show the predicted drift velocity components v_x, v_y from a multivariate linear regression in the subspace of two relevant latent parameters, which extracts the linear combination of latent parameters that correspond to v_x and v_y. The light blue shaded bars are the 95% confidence intervals produced by the models. Results are shown for models trained on (a), (c), (e) the original noiseless datasets as well as (b), (d), (f) the datasets with added $\sigma = 0.1$ Gaussian noise.

Table 3.2: R^2 correlation coefficients from individual linear fits of the 2D convection–diffusion dataset parameters with each relevant latent parameter (LP). High correlations are bolded, emphasizing the interpretability of the learned latent parameters as either corresponding to the diffusion constant D or the drift velocity components v_x, v_y. The drift velocity is matched with two latent parameters that form a two-dimensional latent subspace corresponding to the velocity vector.

	No Noise			$\sigma = 0.1$ Noise		
Param.	LP 1	LP 2	LP 5	LP 1	LP 2	LP 3
D	**0.963**	0.000	0.003	0.003	**0.959**	0.001
v_x	0.000	**0.205**	**0.766**	**0.395**	0.006	**0.554**
v_y	0.001	**0.818**	**0.205**	**0.568**	0.000	**0.473**

with an independent factor of variation due to the VAE regularization [20, 53]. Intuitively, the regularization encourages each latent parameter to independently collapse to a non-informative prior $\mathcal{N}(0, 1)$, and so the model prefers to minimize its use of the latent parameters and maintain their independence (Appendix 3.D). Therefore, the number of latent parameters provided to the model is not critical as long as it is greater than the number of independent factors of variation. In our experiments, we allow the model to use five latent parameters. Because the 1D datasets have only one varying physical parameter and the 2D dataset has three varying physical parameters, the trained model will only make use of one or three latent parameters, respectively, and the rest will collapse to the prior.

We can determine the number of relevant latent parameters and empirically verify this claim by examining the statistics of the extracted distribution parameters μ_z and σ_z^2 from the dynamics encoder (DE) for each dataset. A latent parameter that is useful to the propagating decoder (PD) for predicting the target series will have high variance in μ_z and a low mean σ_z^2, implying that the extracted parameter is precise and informative. A parameter which has collapsed to the prior and is non-informative will have low variance in μ_z and high mean σ_z^2. These statistics indeed show that our model can correctly determine the number of relevant parameters for each dataset (Fig. 3.2). In real applications, we will not have access to the ground truth physical parameters, so we must rely on these metrics to identify the relevant parameters extracted by the model.

To evaluate the performance of our parameter extraction method, we compare the extracted latent parameters from the model with the true physical parameters used to generate our simulated datasets: the viscosity damping parameter γ for the 1D Kuramoto–Sivashinsky dataset, the nonlinearity parameter κ for the 1D nonlinear Schrödinger dataset, and the diffusion constant D and drift velocity components v_x, v_y for the 2D convection–diffusion dataset. Because these simulated physical parameters are drawn from normal distributions (Appendix 3.C), we expect the relevant latent parameters—which have prior distribution $\mathcal{N}(0, 1)$—to be linearly related to the true parameters (Appendix 3.D). For real experimental systems, this is also a reasonable assumption for uncontrolled variables because natural parameters tend to be normally distributed due to the central limit theorem. We assess the quality of the extracted parameters by linearly fitting the relevant latent parameters with the ground truth physical parameters to obtain parameter predictions and R^2 correlation coefficients. Our numerical experiments show excellent parameter extraction on all three datasets (Fig. 3.3) with $R^2 > 0.95$ for all parameters (Table 3.1) and no degradation in performance with added Gaussian $\sigma = 0.1$ noise. However, we do observe some nonlinear behavior at the edges of the parameter range likely due to data sparsity in those regions.

Looking more closely at the results for the three-parameter convection–diffusion dataset, we see that the trained model correctly extracts three relevant latent parameters: one latent parameter corresponds to the diffusion constant and the remaining two-dimensional latent subspace corresponds to the drift velocity vector (Table 3.2). In particular, the model learns a rotated representation of the drift velocity vector as a two-dimensional latent subspace due to the inherent rotational symmetry of the dynamics (Appendix 3.E), so we can recover the v_x, v_y components of the drift velocity by performing a multivariate linear fit (Figs. 3.3(e), 3.3(f)). The successful separation of diffusion from drift velocity in the extracted parameters demonstrates our model's ability to distinguish distinct and interpretable factors of variation in the dynamics of the system.

3.4.2 Prediction Performance

In addition to testing parameter extraction, we evaluate the prediction performance of the trained models on their corresponding test sets. Due to training speed and stability considerations, our models are initially trained with a PD architecture containing only 16 hidden channels (Appendix 3.A). To show the potential for further model refinement, we fix the weights of the DEs trained on the original noiseless datasets and then train additional PDs, each with an expanded 64 hidden channels. These refined predictive models perform better than the original predictive models used during end-to-end training. For comparison, the datasets are also evaluated with a stiff exponential integrator for semilinear differential equations (ETDRK4 [71]) using a second order finite difference discretization on the same time and space meshes provided in the datasets. Although this integrator is the same one used during dataset generation, the time step and mesh size are set to match the available data to provide a reasonable baseline. During dataset generation, the solution is obtained starting with the exact form of the initial condition and converged using much finer mesh sizes (Appendix 3.C). Also, note that using a non-stiff integrator fails on many of the examples in the datasets, so a stiff integrator is required.

The models trained on the 1D Kuramoto–Sivashinsky and 1D nonlinear Schrödinger datasets both perform reasonably when compared with the traditional finite difference method (Figs. 3.5(a), 3.5(b)), with the model trained on the Kuramoto–Sivashinsky dataset maintaining a higher accuracy than its traditional counterpart. The prediction error of the 2D convection–diffusion model is dominated by the uncertainty in the parameter extraction, so the prediction performance is comparable to a finite difference exponential integrator with similar noise in the PDE parameters (Fig. 3.5(c)). For the models trained on datasets with added $\sigma = 0.1$ noise, we see some negative impact on prediction performance (Fig. 3.5) but no effect on the parameter extraction quality (Tables 3.1, 3.2).

The refined models, trained on the noiseless datasets, demonstrate that the PD—

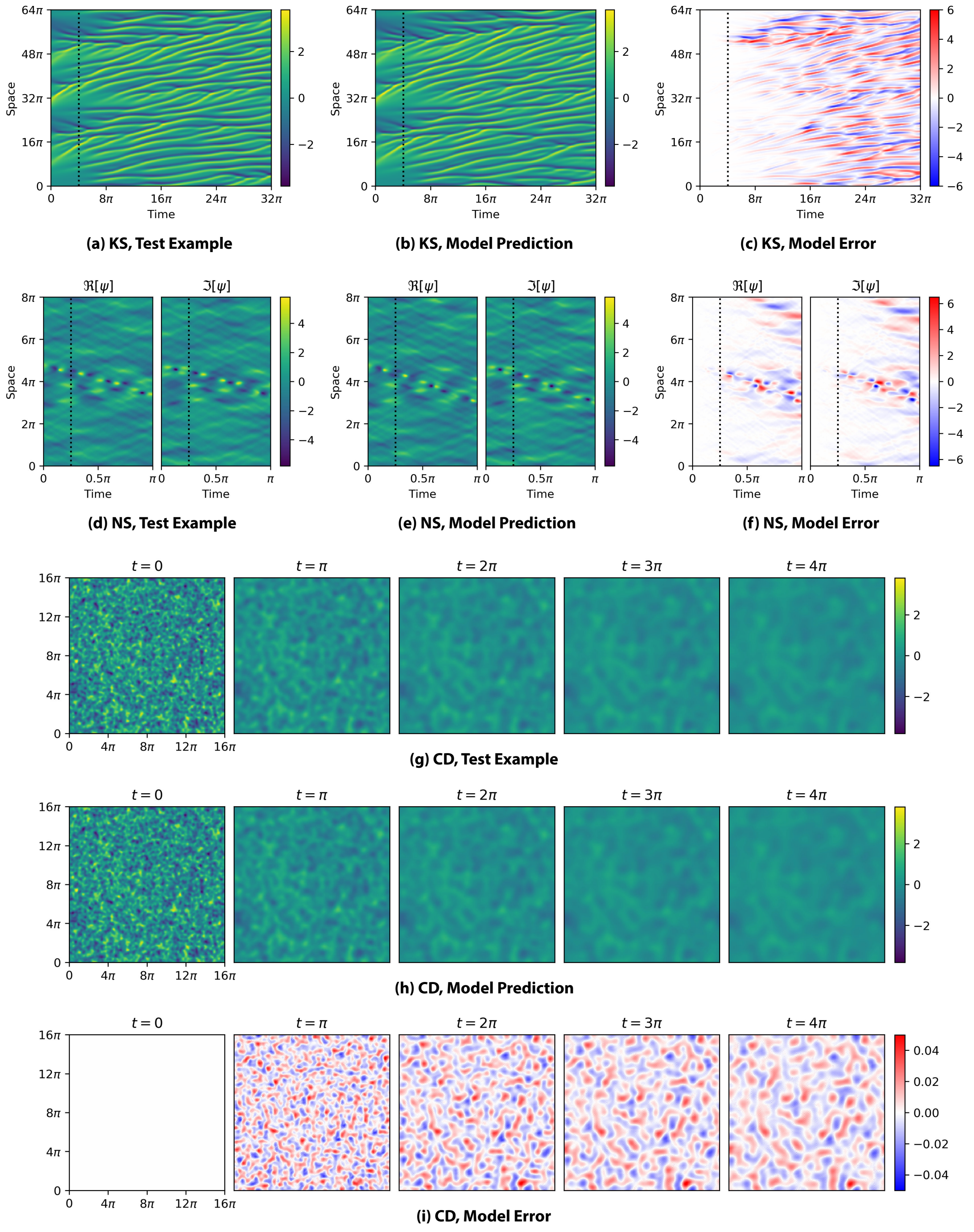


Figure 3.4: Time evolution of (a), (d), (g) examples from the 1D Kuramoto–Sivashinsky (KS), 1D nonlinear Schrödinger (NS), and 2D convection–diffusion (CD) test sets with periodic boundary conditions, (b), (e), (h) the predicted propagation of the refined model given the initial condition at time $t = 0$, and (c), (f), (i) the model prediction error. The black vertical dotted line denotes the maximum amount of time propagated by the model during training, corresponding to the length of the target series. For the CD dataset, the maximum amount of time propagated by model during training is $t = \pi$.

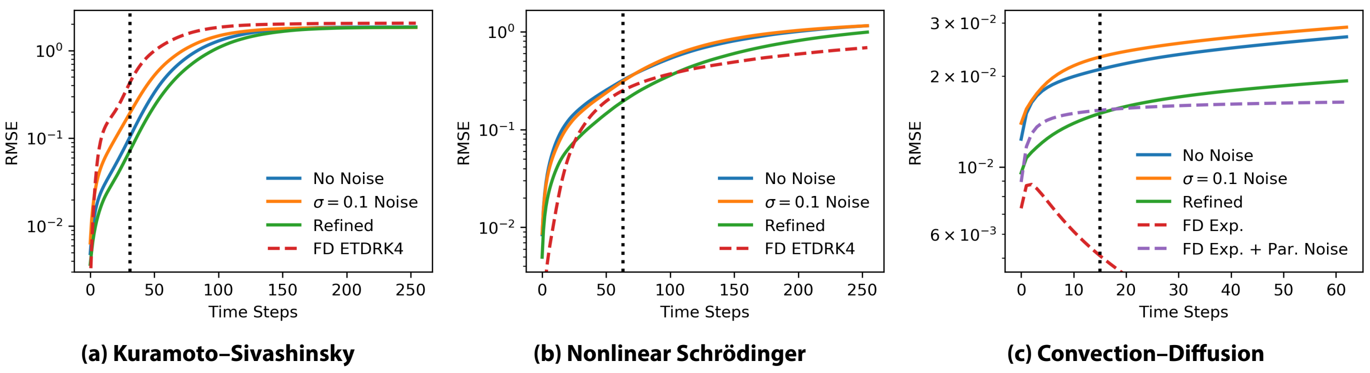


Figure 3.5: The root mean square prediction error (RMSE) during evaluation at each propagation time step, averaged over the corresponding test set, for models trained without noise (blue), trained with $\sigma = 0.1$ Gaussian noise (orange), and refined by fixing the DE and training with 64 hidden channels in the PD network on the noiseless datasets (green). Shown for comparison is an evaluation by a second order finite difference discretization integrated with the ETDRK4 method [71] (dashed red), which reduces to an exact exponential integrator for the 2D convection–diffusion system. Also, since the prediction accuracy on 2D convection–diffusion is dominated by parameter extraction uncertainty, we include for comparison the solver with additional parameter noise: $\sigma = 0.005$ for D, $\sigma = 0.01$ for v_x, v_y (dashed purple). The black vertical dotted line denotes the length of the target series T_y used for training each set of models, i.e. the maximum number of time steps propagated by the model during training.

the predictive network—can be improved independently of the DE—the parameter extraction network. Moreover, the solutions generated by these models remain stable and physically reasonable well beyond the number of time steps propagated during training, suggesting that the models have indeed learned physically meaningful propagators of the PDE-governed systems (Fig. 3.4).

3.5 Application to Nonlinear Fiber Propagation

To demonstrate our method applied on a more complex and realistic dataset, we use Meep [100], a finite difference time-domain electromagnetic simulator, to model pulse propagation through an optical fiber with a Kerr nonlinearity. These simulations model Maxwell's equations exactly, with no approximation apart from the discretization, and often reproduce real experiments point-by-point [121]. The simulated fiber consists of a two-layer core and the surrounding cladding with (relative) permittivity $\varepsilon = 1$ (cross section shown in Fig. 3.6a). We generate a dataset with 200 differ-

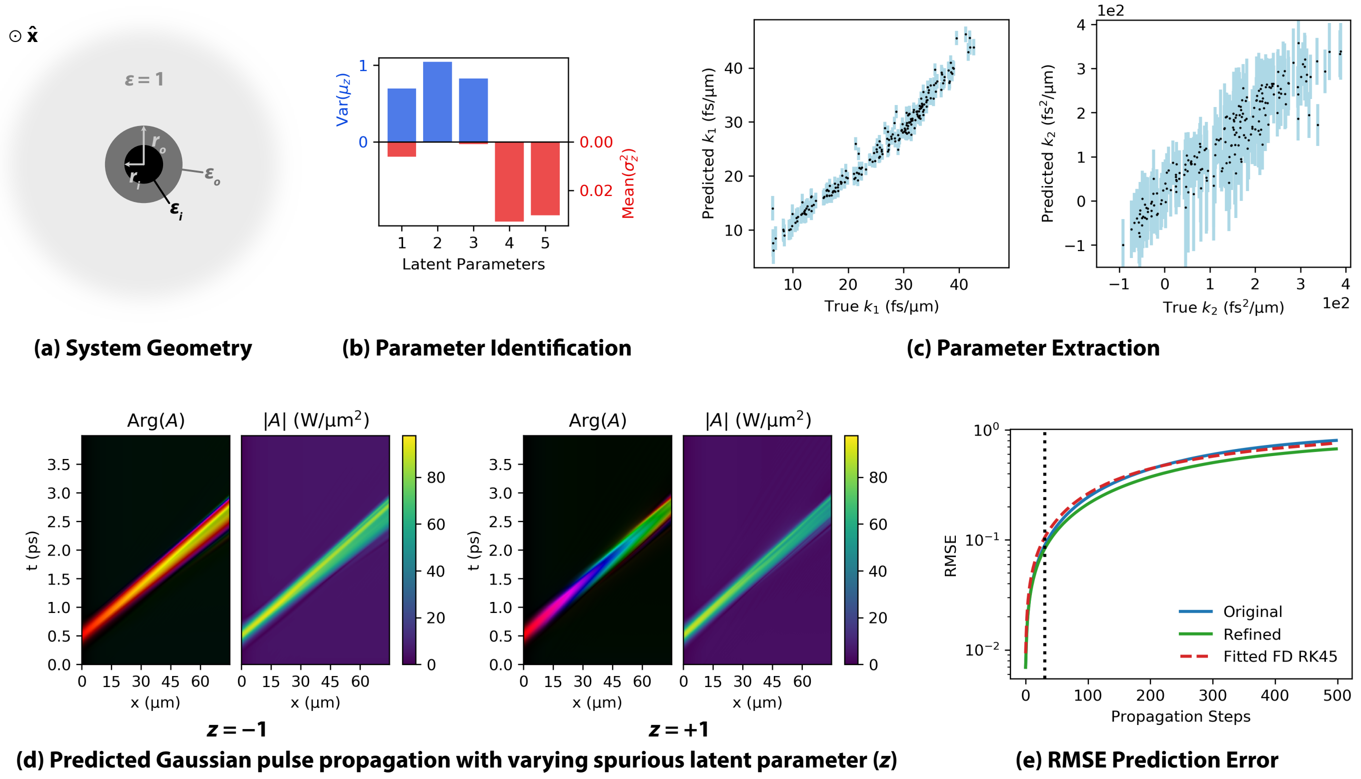


Figure 3.6: Analysis of the model trained on the nonlinear fiber propagation dataset. (a) The cross section of the fiber shows the cladding with (relative) permittivity $\varepsilon = 1$, the inner core with radius r_i and permittivity ε_i, and the outer core with radius r_o and permittivity ε_o. (b) Three relevant latent parameters are identified by the trained model. (c) A linear fit of the latent parameters predicts the true k_1 with $R^2 = 0.966$ and k_2 with $R^2 = 0.863$, corresponding to group velocity and second-order dispersion. (d) Varying the remaining spurious latent parameter z while propagating a Gaussian pulse using the trained propagating decoder (PD), we observe that the spurious parameter represents a phase velocity. The plot of Arg(A) shows the phase of the pulse mapped to the color hue while the amplitude is mapped to lightness. (e) The root mean squared prediction error (RMSE) at each propagation step of the originally trained PD (blue) and a further refined PD (green) are comparable to an explicit solution (second order finite difference, adaptive fourth order Runge–Kutta) of the effective equation (3.12) with fitted parameters (up to fourth-order dispersion) for each individual test set example (dashed red).

ent geometries by varying the size of the two-layer core through r_i and r_o as well as the corresponding permittivities ε_i and ε_o, representing uncontrolled experimental variables related to fabrication (see Appendix 3.C for details). Then, we excite a randomly generated pulse in each fiber and train our model on the flux-normalized amplitude $A(x,t)$ of the resulting pulse propagation.

There is no exact first-order PDE describing the evolution of this amplitude. However, in the slowly varying envelope approximation, $A(x,t)$ is governed by an effective nonlinear Schrödinger equation [1]

$$\frac{\partial A}{\partial x} = i \sum_{n=1}^{\infty} \frac{i^n k_n}{n!} \partial_t^n A + i\gamma \left|A\right|^2 A, \tag{3.12}$$

where the dispersion coefficients $k_n = \partial_\omega^n k(\omega)|_{\omega=\omega_0}$ can be computed from the dispersion relation $\omega(k)$ at a carrier frequency $2\pi\omega_0$, and the nonlinearity parameter γ is related to the Kerr nonlinearity (Appendix 3.C) and the shape of the propagating mode in the fiber [14]. Due to the form of this effective equation, we choose our propagation variable to be the distance x rather than the time t, i.e. our model will predict $A(x,t)$ given an initial pulse $A_0(x=0,t)$ by propagating forward in the x-direction. We obtain the ground truth dispersion coefficients using MPB [63], a frequency domain electromagnetic eigenmode solver, for comparison with the extracted parameters from the model.

The trained model identified and extracted three relevant latent parameters (Fig. 3.6b). Two independent directions in the latent parameter space (primarily, latent parameters 2 and 3) correspond to the group velocity $k_1 = 1/v_g$ with $R^2 = 0.966$ and second-order dispersion k_2 with $R^2 = 0.863$ (Fig. 3.6c). Note that the extracted parameters are not the geometry parameters used to generate the dataset, but rather parameters relevant to the effective propagation of the pulse. These two latent parameters also capture variations in higher order dispersion terms, which are correlated

[1] This effective equation is usually transformed to a co-moving time coordinate $\tau = t - x/v_g$, where v_g is the group velocity [14]. However, since the group velocity varies among of the dataset examples and because we want to use minimal preprocessing, we keep the original time coordinate t. We also include higher order dispersion terms which are relevant for our dataset.

with the group velocity and second-order dispersion in the dataset. As a result, although important and present in the effective equation (3.12), the higher order terms are already well correlated with the existing latent parameters, so no additional parameters are required to characterize the dynamics. See Appendix 3.H for more parameter analysis details.

In addition to the parameters corresponding to k_1 and k_2, the model extracted another relevant latent direction (primarily, latent parameter 1), which is independent and orthogonal to the previous two. This seemingly spurious latent parameter does not correspond to a term in the effective equation (3.12); instead, it represents a spurious phase velocity, which is the result of imperfect preprocessing (Appendix 3.C). We discovered this correspondence by varying the spurious parameter, while leaving all other latent parameters fixed, and observing the effect on the model predictions (Fig. 3.6d). This uncontrolled variable was successfully extracted by the model and subsequently identified, demonstrating the process by which unknown extracted parameters can be understood and interpreted.

We evaluate the prediction performance of the trained propagating decoder (PD) as well as a refined version of the PD (with an expanded 64 hidden channels) by comparing the prediction error of our models with an explicit solution to the effective equation (3.12) as a baseline. The effective equation parameters are determined by a linear fit of the finite difference derivatives computed from the dataset. For each example in the test set, we fit the dispersion parameters k_1, k_2, k_3, k_4 (i.e. up to fourth order dispersion) and the nonlinearity parameter γ. The effective equation is then integrated using a fourth order Runge–Kutta method with adaptive step size and second order finite difference discretization. This approach is a simplified version (with the terms pre-identified) of explicit PDE identification methods, such as SINDy [113]. Our predictive models achieve similar prediction performance when compared with this explicit effective equation baseline (Fig. 3.6e).

3.6 Discussion

We have developed a general unsupervised learning method for extracting unknown dynamical parameters from noisy spatiotemporal data *without* detailed knowledge of the underlying system or the exact form of the governing PDE. While we do not explicitly extract the governing PDE, our method provides a set of tunable relevant parameters, which characterize the system dynamics in independent and physically interpretable ways, coupled with a highly transferable predictive model. This is often enough to provide significant insight into the physics of the system: for example, by examining the effect of varying an unidentified relevant parameter on the predictions of the model, we can disentangle the parameter from other effects (parameterized by the other relevant parameters) and interpret it independently. This is precisely how we identified the spurious phase velocity parameter extracted by the model trained on nonlinear fiber propagation (Fig. 3.6d). One potential complication of interpreting the parameters extracted using our method is that each parameter must represent an independent factor of variation in the dynamics of the dataset. This means that if features of the dynamics are highly correlated in the underlying dataset, they will be parameterized by the same parameter, e.g. the parameters extracted from the nonlinear fiber propagation dataset corresponding to group velocity and second order dispersion also capture higher order dispersion terms (Appendix 3.H).

The flexibility and robustness of our model comes from using a generic physics-informed neural network model for nonlinear PDEs. The interpretability of the resulting extracted parameters is a result of the variational autoencoder (VAE) training and regularization (Appendix 3.D) as well as the inductive biases imposed by our physics-informed network design. By using appropriate spatial averaging in the dynamics encoder (DE) and dynamic convolutions in the propagating decoder (PD), we ensure that both the parameter extraction and the propagation prediction from our model are physically motivated and generalizable to arbitrary system sizes and geometries. The dynamic convolutions, in particular, are an important physical inductive bias for encouraging the model to learn latent parameters which govern the propagation dynamics. As a result, the learned parameter-to-kernel mappings in the trained predictive model are fully transferable, which we can demonstrate by evaluating the predictive model *without retraining* on a different set of boundary conditions (see Appendix 3.G).

Our strategy for modeling spatiotemporal systems is to retain the expressiveness of a neural network model while imposing general constraints—such as locality through using convolutional layers—to help the network learn more efficiently. For particular applications, this could also include spatial symmetries [29, 30, 131], e.g. properly transforming fluid flow vectors, as well as additional symmetries of the internal degrees of freedom, e.g. the global phase of the nonlinear Schrödinger equation. These architecture-based constraints encourage the model to learn physically relevant representations and can be tailored to individual applications, allowing us to incorporate domain knowledge into the model. This also lets us use datasets that are much smaller than is traditionally required by deep learning methods. The model trained on nonlinear fiber propagation used only 200 examples (Sec. 3.5), and we have been able to successfully train models on the Kuramoto–Sivashinsky dataset with as few as 10 examples (Appendix 3.F). Additionally, the model's robustness to noise is a powerful feature of deep learning methods and provides a promising avenue for studying dynamical systems in a data-driven fashion [114].

The primary challenges associated with applying our current implementation involve setting hyperparameters to improve training stability (see Appendix 3.B.1 for more details) and choosing the β regularization hyperparameter, which controls parameter extraction (Appendix 3.D). The choice of β can be somewhat ambiguous, with very high values resulting in no relevant parameters and very low values failing to enforce independence (for our choices, see Appendix 3.B). This trade-off and other related issues are a very active area of research, and parameter extraction methods will continue to improve following the rapid advances in unsupervised learning and disentangling representations, e.g. through a deeper theoretical understanding of the β-VAE [3, 20, 142] and alternative formulations [25, 143]. Physics-informed inductive biases, however, will remain the key ingredient for ensuring the representations are interpretable [85].

The predictive model will also likely achieve better accuracies by using more sophisticated architectures, such as echo state networks which have been shown to perform extremely well on even chaotic PDEs [103], or by explicitly incorporating differentiable PDE solvers with gradients computed by the adjoint method [26]. An echo state network or other alternative decoder architecture would have to be adapted to retain the transferability of our current PD. Using a differentiable PDE solver would allow the PD network to focus on encoding the PDE rather than also learning a stable integration method, and thus may improve the interpretability of the model.

Our unsupervised learning method is also highly complementary with the significant body of work applying machine learning methods for more accurate predictions of specific physical systems, such as multi-scale hydrodynamic systems [49] and turbulence modeling [81, 93, 135, 138]. These methods often combine a known physics model with a machine learned correction or parameterize an unknown part of the physics model using neural networks. Such machine learning–physics hybrid models can be adapted into decoders for our unsupervised learning method, with the extracted relevant parameters representing independent variations of and providing insight into the machine learned portions of the model. Our current method can also be adapted in the future for a more general class of spatiotemporal systems by incorporating spatially inhomogeneous latent parameters and will also be able to use data with incomplete physical state observations by inferring missing information.

The ultimate goal of this work is to provide additional insight into complex spatiotemporal dynamics using a data-driven approach. Our method is an example of a new machine learning tool for studying the physics of spatiotemporal systems with an emphasis on interpretability.

3.A Model Implementation Details

For the 1D datasets, the dynamics encoder (DE) uses 2D convolutions with output channel sizes $(4, 16, 64, 64, 5)$, linear kernel sizes $(3, 3, 3, 3, 1)$, and dilation factors

$(1, 2, 4, 8, 1)$. For the 2D dataset, the DE uses 3D convolutions with output channel sizes $(8, 64, 64, 64, 5)$ with the same kernel sizes and dilation factors. The $f_{\mathrm{DE},\mu}$ and f_{DE,σ^2} networks share the same convolution weights for the first four layers and have distinct final layers to produce μ and $\log \sigma^2$. The final output channel size is determined by the number of latent parameters; in our tests, we use five latent parameters.

For the 1D datasets, the propagating decoder (PD) architecture uses three 1D dynamic convolutional layers with output channel sizes $(16, 16, \text{data channel size})$, linear kernel size 5, and periodic padding. For the 2D datasets, the PD uses three 2D dynamic convolutional layers with the same output channel sizes, kernels, and padding. The refined models increase the number of hidden channels in the PD from 16 to 64, resulting in output channel sizes $(64, 64, \text{data channel size})$.

The latent-to-kernel network, which maps the latent parameters to each kernel or bias in the PD, consists of two fully-connected layers, i.e. one hidden layer. For the 1D datasets, the hidden layers have size $(4 \times \text{input channel size} \times \text{output channel size})$ for kernels and $(4 \times \text{output channel size})$ for biases, where the input and output channel sizes refer to the corresponding dynamic convolution in the PD. For the 2D dataset, the hidden layers have size $(16 \times \text{input channel size} \times \text{output channel size})$ for kernels and $(4 \times \text{output channel size})$ for biases.

Our architecture uses ReLU activations throughout except for the unactivated output layers of the DE, PD, and latent-to-kernel networks. The output of the PD convolutional network f_{PD} uses a tanh activation with a learnable scaling parameter ($x \mapsto \lambda \tanh(x/\lambda)$ with learnable parameter λ initialized to 1) to stabilize the recurrent architecture. The network f_{PD} also has a fixed multiplicative pre-factor set to 10^{-6} to improve the initial training stability. For the nonlinear fiber propagation dataset, we add Gaussian noise with $\sigma = 10^{-2}$ between propagation steps in the PD network for improved prediction stability.

Our model is implemented using PyTorch v1.1 [102], and the source code is available at `https://github.com/peterparity/PDE-VAE-pytorch`.

3.B Training Details

All models are trained using batch size 50 and the Adam optimizer [74] with learning rate 10^{-3}. Models for the 1D datasets and the noisy 2D convection–diffusion dataset were trained for 2,000 epochs; the model for the noiseless 2D convection–diffusion dataset was trained for 4,000 epochs; and the corresponding refined models were trained for 2,000 epochs. The VAE regularization hyperparameter is set to $\beta = 0.02$ for the 1D datasets and $\beta = 10^{-4}$ for the 2D convection–diffusion dataset. The model for the nonlinear fiber propagation application was trained for 40,000 epochs due to small size of the dataset and significant data augmentation; the corresponding refined model was trained for 20,000 epochs; and the VAE regularization hyperparameter was set to $\beta = 7 \times 10^{-4}$. During validation, we choose β such that we obtain the maximum number of relevant latent parameters while still maintaining statistical independence among the parameters as well as a clean separation between the relevant and irrelevant (i.e. collapsed to the prior) parameters. All hyperparameter tuning is done using the training set for validation.

For the 1D Kuramoto–Sivashinsky dataset, we train the model using a random 64×94 crop—in the time and space dimensions, respectively—for the input series and another random 64×76 crop for the target series. For the 1D nonlinear Schrödinger dataset, we train using a 64×94 crop for the input series and a 32×76 crop for the target series. For the 2D convection–diffusion dataset, we train using a $45 \times 62 \times 62$ crop—in the one time and two space dimensions, respectively—for the input series and a $16 \times 44 \times 44$ crop for the target series. For the nonlinear fiber propagation dataset, we train using a 128×158 crop for the input series and a 32×76 crop for the target series. During evaluation, we always use the full size time-series from the test set for both the input and target series.

3.B.1 Notes on Training Stability

We find our model to be particularly sensitive to the architecture of the propagating decoder (PD): with larger, more complex networks and more propagation steps during training resulting in increasing instability. The dynamics encoder (DE) does influence stability, but the effect is more indirect through its interaction with the PD and is not very architecture sensitive. This instability is likely related to the problem of vanishing and exploding gradients seen often in recurrent architectures, which is mitigated using gating mechanisms like in LSTM networks [44, 55] or by explicitly using unitary norm-preserving matrices [62]. Importantly, this is only a significant problem when training the model end-to-end using both the DE and PD; when we fix the DE weights, we are able to further refine a more complex PD model without instability (Sec. 3.4.2). This also does *not* affect the prediction performance and stability of the model during evaluation, which generalizes well past the number of time steps propagated during training (Fig. 3.5). We currently implement a learnable gating mechanism (Appendix 3.A) that significantly stabilizes the network but further work is required to fully address this issue.

3.C Dataset Generation Details

For each time-series example in the 1D Kuramoto–Sivashinsky dataset, we sample the viscosity damping parameter γ from a truncated normal distribution ($\mu = 1$, $\sigma = 0.125$, and truncation interval $[0.5, 1.5]$). We then use the ETDRK4 integrator [71] to generate each time-series to within a local relative error of 10^{-3}. Each time-series consists of a uniform time mesh with 256 points for a total time $T = 32\pi$ and a space mesh with $M = 256$ points for an $L = 64\pi$ unit cell. These are produced by solving on a finer time and space mesh to ensure convergence and then resampling to the dataset mesh sizes. Each initial state is generated from independently sampled, normally distributed Fourier components with a Gaussian band-limiting envelope of varying widths (uniformly sampled in the interval $[8\pi/L, M\pi/4L]$) and then normalized to unit variance.

For each time-series example in the 1D nonlinear Schrödinger, we sample the nonlinearity coefficient κ from a truncated normal distribution ($\mu = -1$, $\sigma = 0.25$, and truncation interval $[-2, 0]$). We then use the ETDRK4 integrator [71] to generate each time-series to within a local relative error of 10^{-3}. Each time-series consists of a uniform time mesh with 256 points for a total time $T = \pi$ and a space mesh with $M = 256$ points for an $L = 8\pi$ unit cell. These are produced by solving on a finer time and space mesh to ensure convergence and then resampling to the dataset mesh sizes. The initial states are generated in an analogous manner to the Kuramoto–Sivashinsky dataset.

For each time-series example in the 2D convection–diffusion dataset, we vary the parameters by sampling the diffusion constant D from a truncated normal distribution ($\mu = 0.1$, $\sigma = 0.025$, and truncation interval $[0, 0.2]$), and each velocity component v_x, v_y from a normal distribution ($\mu = 0$, $\sigma = 0.2$). Because the convection–diffusion equation is linear, we use the exact solution to generate the dataset. Each time-series consists of a uniform time mesh with 64 points for a total time $T = 4\pi$ and an $M \times M = 256 \times 256$ space mesh for an $L \times L = 16\pi \times 16\pi$ unit cell. Each initial state is generated from independently sampled, normally distributed Fourier components with a Gaussian band-limiting envelope of varying widths (uniformly sampled in the interval $[16\pi/L, M\pi/2L]$) and then normalized to unit variance.

The nonlinear fiber propagation dataset aims to roughly model a set of highly dispersive nonlinear optical fibers with variations in geometry due to fabrication. Each of the 200 examples in the nonlinear fiber propagation dataset corresponds to a randomly generated fiber geometry excited with a randomly generated pulse. Specifically, each fiber geometry (Fig. 3.6a) corresponds to a set of geometry parameters sampled from normal distributions: inner core radius r_i ($\mu = 75\,\text{nm}$, $\sigma = 3\,\text{nm}$), outer core radius r_o ($\mu = 150\,\text{nm}$, $\sigma = 7.5\,\text{nm}$), inner core (relative) permittivity ε_i ($\mu = 30$, $\sigma = 2$), and outer core permittivity ε_o ($\mu = 8$, $\sigma = 1$). There is an overall fixed Kerr nonlinearity that corresponds to a nonlinear refractive index $n_2 = (3.375 \times 10^{-4}\,\mu\text{m}^2\,\text{W}^{-1})/\varepsilon$, where ε is the material permittivity. The excited pulse is generated from independently sampled, normally distributed frequency components with a Gaussian band-limiting envelope centered on a carrier frequency $f = 200\,\text{THz}$ ($\lambda = 1.5\,\mu\text{m}$) and with a width of $f/20$. This pulse is slowly turned on with a sigmoid of width $20/f$, and, for the test set, the pulse is also turned off at half the total simulation time, allowing the pulse to fully propagate through the fiber. During the simulation, the amplitude $A(x,t)$ of the electric field at the center of fiber is recorded and later normalized to the total flux passing through the fiber. The space and time Fourier components of each resulting dataset example $A(x,t)$ are shifted to remove the carrier frequency and the peak wavenumber, resulting in a slowly varying envelope. Then, the final dataset is normalized again so that the amplitude at the initial point $x=0$ has, on average, unit variance over the whole dataset. Each example consists of an x-direction mesh with 500 points for a propagation length of 75 µm and a uniform t-direction mesh with 800 points for a total time 4.00 ps (training set) or 1000 points for a total time 5.00 ps (test set).

The dataset generation scripts are available at `https://github.com/peterparity/PDE-VAE-pytorch`.

3.D Understanding the Effects of VAE Regularization

Using VAE [75] or β-VAE [53] regularization in our model provides three main benefits for learning physically interpretable representations: the regularization encourages the model to minimize use of the latent parameters, enforces independence among the learned latent parameters, and matches the marginal latent distribution to a standard normal prior. We can explicitly see these effects by decomposing the data-averaged VAE regularization term in the following way [25, 56]:

$$\begin{aligned}
\mathbb{E}_{p_D(\mathbf{x})}[D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))] = \\
&D_{\mathrm{KL}}(q(\mathbf{z},\mathbf{x}) \parallel q(\mathbf{z})\, p_D(\mathbf{x})) && (3.13)\\
&+ D_{\mathrm{KL}}(q(\mathbf{z}) \parallel \prod_i q(z_i)) && (3.14)\\
&+ \sum_i D_{\mathrm{KL}}(q(z_i) \parallel p(z_i)), && (3.15)
\end{aligned}$$

where $p_D(\mathbf{x})$ is the data distribution, $p(\mathbf{z}) = \prod_i p(z_i)$ are the standard normal priors for the latent parameters $\mathbf{z} = (z_1, z_2, \ldots, z_i, \ldots)$, $q(\mathbf{z}|\mathbf{x})$ is the output distribution of the dynamics encoder, $q(\mathbf{z}, \mathbf{x}) = q(\mathbf{z}|\mathbf{x})\, p_D(\mathbf{x})$ is the joint distribution of the encoded latent parameters and the data, and $q(\mathbf{z}) = \int d\mathbf{x}\, q(\mathbf{z}, \mathbf{x})$ and $q(z_i) = \int d\mathbf{x} \prod_{j \neq i} dz_j\, q(\mathbf{z}, \mathbf{x})$ are the marginal distributions of the latent parameters $\mathbf{z}$ or a single latent parameter z_i, respectively. The three terms in this decomposition correspond directly to the three effects: the first term (3.13) represents the mutual information between the latent parameters and the data; the second term (3.14) represents the total correlation between the latent parameters; and the third term (3.15) consists of KL divergences between the marginal distribution for individual latent parameters and the standard normal prior.

By minimizing the mutual information between the latent space and the data (3.13) as well as correlations among the latent parameters (3.14), the model is compelled to learn a latent space with minimal information and independent parameters, i.e. the model will use a minimal set of independent relevant latent parameters to capture only the necessary information for better prediction performance. The rest of the unused latent parameters will collapse to the prior. Furthermore, by matching the *marginal* latent parameter distributions $q(z_i)$ to the standard normal priors $p(z_i)$ (3.15), the VAE regularizer encourages a linear relationship between the relevant learned latent parameters and the true physical parameters *if* the physical parameters are normally distributed in the data. Even if a physical parameter z_{phys} is non-normally distributed, the VAE regularization will still compel the model to learn a monotonic relationship between z_{phys} and a corresponding latent parameter z given by

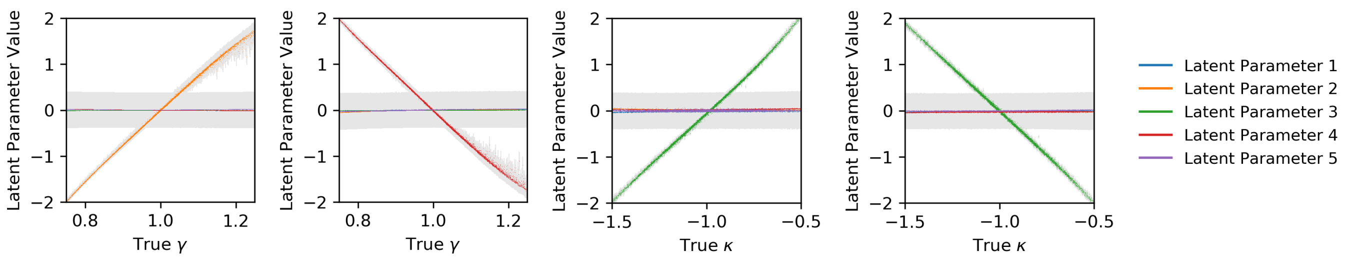
$$z_{\text{phys}} = \pm\, \text{CDF}_{p(z_{\text{phys}})} \circ \text{CDF}^{-1}_{p(z)}(z), \tag{3.16}$$

where $\text{CDF}_{p(\cdot)}$ is the cumulative distribution function for the probability distribution $p(\cdot)$. One caveat—in addition to ambiguities introduced by symmetries of the physical parameters—is that the relationship may not be monotonic for physical parameter distributions which have support on a topologically distinct space from the real line, e.g. a uniformly distributed periodic angle parameter. However, the result may still be interpretable, e.g. an angle parameter may be encoded as a ring in a two-dimensional latent subspace.

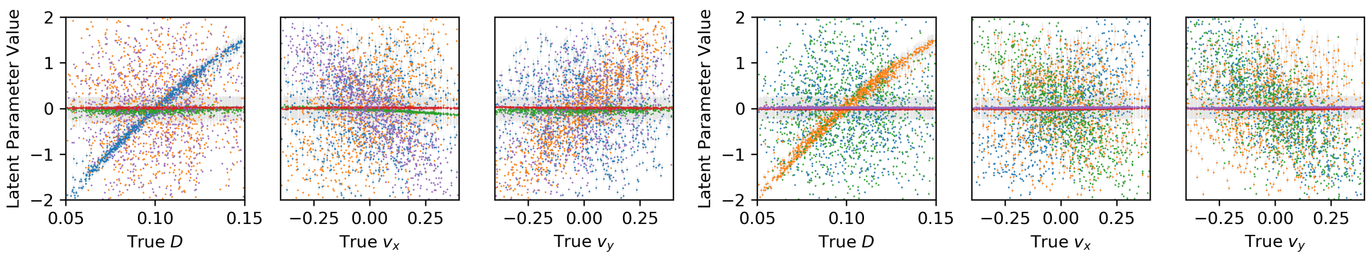
Although this decomposition is suggestive of the effects of VAE regularization, the study of the performance of VAE-based models and the relative importance and model dependence of each of these effects is still very much ongoing [3, 20, 25, 111, 142, 143]. While training our model, we empirically observe that the latent parameters retain their independence and that their marginal distributions match the standard normal priors, so only an increase in information stored in the latent space is traded for better prediction performance. We believe we can attribute this to the physics-informed inductive biases present in our architecture, which allows our model to achieve its best performance using a minimal set of independent latent parameters.

3.E Raw Parameter Extraction Results

We can explicitly see the relevant and collapsed latent parameters in the raw data by plotting the latent parameters versus the true physical parameters (Fig. 3.7). The latent parameters that show a correlation with the true physical parameters also have small variances σ_z^2 and correspond to the identified relevant latent parameters (Fig. 3.2), while the remaining latent parameters have collapsed to the prior. Note that for the collapsed parameters, we see variances σ_z^2 that are less than one—the value expected for parameters which have collapsed to the prior distribution $\mathcal{N}(0, 1)$. This is because, to average over systems of different sizes, the model makes the assumption that patches separated far in space or time provide independent estimates of the extracted parameters and computes the total variance accordingly. This assumption is reasonable for relevant parameters but will artificially lower the extracted variances for collapsed parameters. During testing, we choose to evaluate on the full system size resulting in this artifact. If we were to evaluate on smaller patches that match the size of the crops used during training, we would indeed see that the collapsed parameters have $\sigma_z^2 = 1$.



(a) KS, No Noise **(b) KS, $\sigma = 0.1$ Noise** **(c) NS, No Noise** **(d) NS, $\sigma = 0.1$ Noise**



(e) CD, No Noise **(f) CD, $\sigma = 0.1$ Noise**

Figure 3.7: The five latent parameters in the models trained on the (a), (b) 1D Kuramoto–Sivashinsky (KS), (c), (d) 1D nonlinear Schrödinger (NS), and (e), (f) 2D convection–diffusion (CD) datasets both with and without added noise vs. the ground truth physical parameters used to generate the datasets. In these plots of the raw extracted latent parameters, we see the direct correspondence between the identified relevant latent parameters in Fig. 3.2 and the true physical parameters as well as the collapse of the unused latent parameters. Note that the relevant latent parameters corresponding the drift velocity $\mathbf{v}$ in the convection–diffusion model are not precisely aligned with the two components v_x, v_y due to the inherent rotational symmetry. The gray shaded bars are the 95% confidence intervals ($\pm 1.96\sigma_z$) produced by the model.

We also note that, for the model trained on the 2D convection–diffusion dataset, the latent parameters associated with the drift velocity $\mathbf{v}$ are not aligned with the v_x, v_y velocity components. This is an expected result due to the inherent ambiguity of choosing a coordinate basis—introduced by the rotational symmetry of the velocity vector—and makes judging the extraction performance more difficult. Instead of examining one latent parameter at a time, we must consider the two-dimensional latent subspace associated with the velocity vector. Taking the two relevant latent parameters that are correlated with the drift velocity (Table 3.2), we can perform a multivariate linear regression of the velocity components v_x, v_y in this two-dimensional latent subspace to verify that the model has indeed learned a simple rotated representation of the velocity vector (Figs. 3.3(e), 3.3(f)).

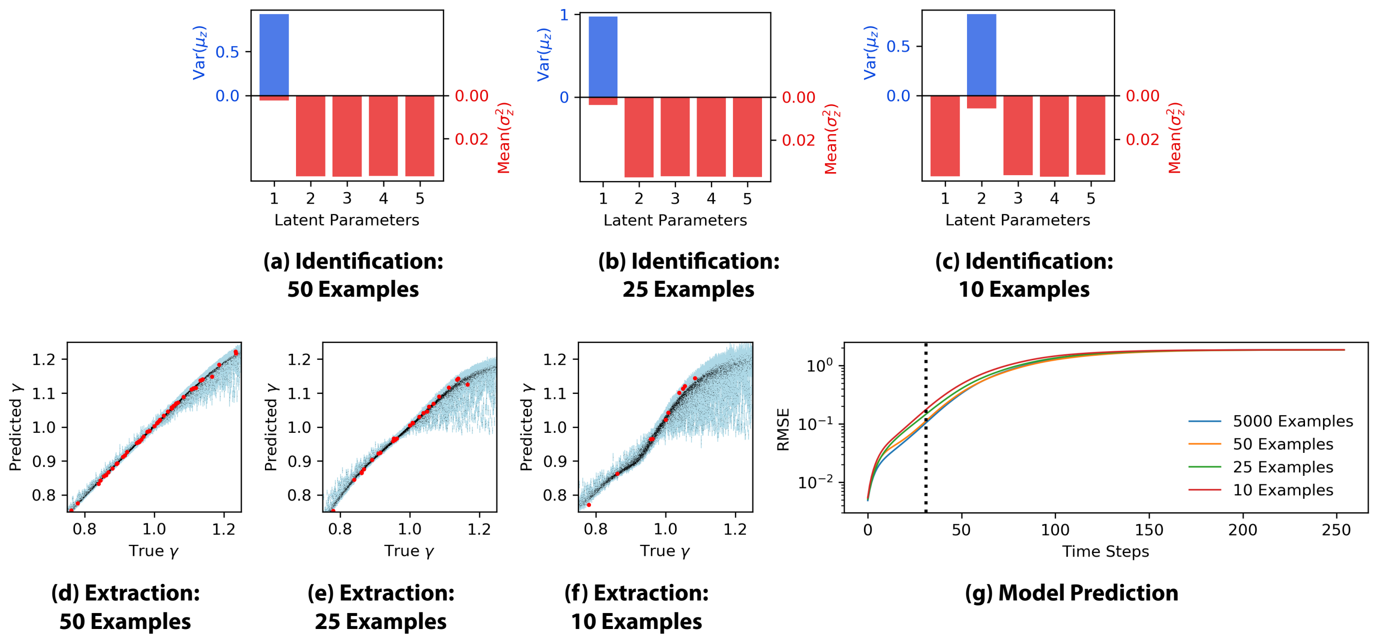
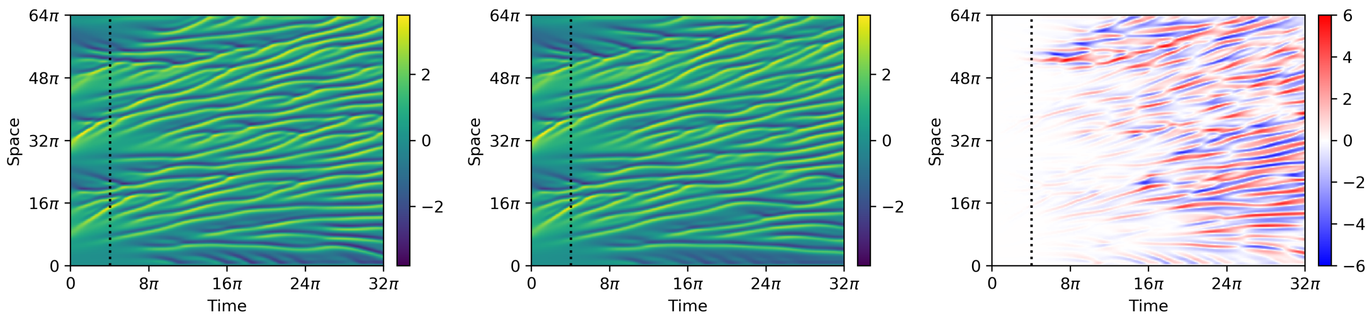


Figure 3.8: Parameter extraction and prediction performance for models trained on noiseless 1D Kuramoto–Sivashinsky datasets with 50, 25, and 10 examples to show the effect of dataset size. The (a), (b), (c) parameter identification and (d), (e), (f) extraction plots demonstrate the ability of the model to identify and extract a relevant latent parameter using very few examples. In the extraction plots, the small black points show the evaluation on the test set, the large red points show the training examples used for each model, and the light blue shaded bars are the 95% confidence intervals produced by the model. There is also a subtle decrease in prediction performance seen in (g) the root mean square prediction errors (RMSE) averaged over the 10,000 example test set. The parameter identification and extraction plots for the 5,000 example dataset are shown in Figs. 3.2(a) and 3.3(a), respectively.

3.F Performance Scaling with Dataset Size

Due to the significant physics-informed inductive biases in our architecture, our model still achieves usable results even when trained on very small datasets. We test the dataset size dependence of our method using the 1D Kuramoto–Sivashinsky system and find that the model is still able to identify the relevant latent parameter even with a dataset of just 10 examples (Fig. 3.8). The accuracy and precision of the extracted parameter and the prediction performance do begin to suffer when using such extremely small datasets, but the model is still able to provide some insight into the dynamics of the spatiotemporal system represented by the data.



(a) Test Example **(b) Model Prediction** **(c) Model Error**

Figure 3.9: Time evolution of (a) a 1D Kuramoto–Sivashinsky example with Dirichlet boundary conditions, (b) the predicted propagation using transferred kernels from the refined predictive model given the initial condition at time $t = 0$, and (c) the prediction error of the model. The refined predictive model for the 1D Kuramoto–Sivashinsky was originally trained on data with periodic boundaries and is adapted—without retraining—to use Dirichlet hard wall boundaries instead by adjusting the padding at each propagation time step. The black vertical dotted line denotes the maximum amount of time propagated by the model during the original training, corresponding to the length of the target series.

3.G Alternative Boundary Conditions

The fully convolutional structure of the propagating decoder (PD) means that we are able to evaluate our model on arbitrary geometries and boundary conditions. By training on small crops and evaluating on the full size examples in the test set (Sec. 3.4), we have already shown the trained model can be directly evaluated on larger system sizes. To show direct evaluation on an alternative boundary condition, we test the refined predictive model—originally trained on the 1D Kuramoto–Sivashinsky dataset with periodic boundaries—on a new test example generated with Dirichlet hard wall boundary conditions (Fig. 3.9). In general, we can apply alternative boundary conditions by adjusting the padding scheme of each propagation step in the PD. For Dirichlet boundaries, this corresponds to applying anti-reflection padding at each propagation step. This preliminary test suggests that we can achieve similar prediction performance using an alternative boundary condition, which the model has never previously seen, and demonstrates the transferability of the learned convolutional kernels.

3.H Nonlinear Fiber Parameter Analysis

In the three-dimensional relevant latent space (z_1, z_2, z_3) extracted by the model trained on the nonlinear fiber propagation dataset (Fig. 3.6b), we determine two independent and interpretable directions by a linear fit: $(-0.101, 0.971, 0.218)$ and $(0.477, -0.0303, -0.878)$, which correspond to the group velocity $k_1 = 1/v_g$ and second-order dispersion k_2, respectively (Fig. 3.6c). The final direction $(0.882, -0.107, 0.459)$ in the latent space—orthogonal to the previous two—is a seemingly spurious relevant parameter unrelated to the parameters of the effective equation (3.12) and represents a spurious phase velocity (Fig. 3.6d).

For the examples in the nonlinear fiber propagation dataset, higher order dispersion terms k_n are still significant. However, because these terms are correlated with k_1 and k_2 in the data, the model does not require additional latent parameters to capture their effect. Instead, the existing latent parameters also adjust the higher order dispersion terms; in other words, the latent parameters each correspond to a dispersion operator that includes higher order dispersion along with k_1 and k_2.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Discovering Conservation Laws using Optimal Transport and Manifold Learning

4.1 Introduction

Conservation laws are powerful constraints on the dynamics of many physical systems in nature, and the corresponding conserved quantities are essential features for characterizing the behavior of these systems. Through Noether's theorem, conservation laws are closely tied with the symmetries of a physical system and play a key role in our understanding of physics. Conservation laws also help stabilize and enhance the performance of predictive models for complex nonlinear dynamics, e.g. symplectic integrators for Hamiltonian systems [48] and pressure projection for incompressible fluid flow [46]. In fact, for chaotic dynamical systems, conserved quantities are often the only features that can be successfully predicted far into the future. Discovering conservation laws helps us characterize the long term behavior of complex dynamical systems and understand the underlying physics.

While the conservation laws of many physical systems are well-known and often derived from known symmetries, there are still many instances where it is difficult to even determine the number of conservation laws, let alone explicitly extract the conserved quantities. As a historical example, consider the Korteweg–De Vries (KdV) equation modeling shallow water waves. The KdV equation, despite its apparent complexity, has infinitely many conserved quantities [97] and is, in fact, fully solvable via an inverse scattering transform [43]—a discovery made after significant theoretical and computational effort. Developing better general methods for identifying conserved quantities will allow us to improve our understanding of new or understudied physical systems and build more efficient and stable predictive models.

In real-world applications, an accurate model for the underlying physical system is often unavailable, forcing us to identify conservation laws using only sample trajectories of the system dynamics. One broad approach is to use modern data-driven methods based on the Koopman operator formulation of dynamical systems, which lifts the dynamics into an infinite dimensional operator space [95]. In the Koopman formalism, conserved quantities are just one type of Koopman eigenfunction with eigenvalue zero. Thus, one approach is to first apply a system identification method, such as dynamic mode decomposition [116, 134], sparse identification with a library of basis functions [17], or even deep learning-based approaches [22, 90, 91], to model the system dynamics and then set up and solve the Koopman eigenvalue problem. Alternatively, previous work has also proposed directly setting up the eigenvalue problem by estimating time derivatives from data and then fitting the conserved quantities using a library of possible terms [66] or a neural network [83]. These methods can work quite well but require that the measured trajectories have sufficiently high time resolution in order to accurately estimate time derivatives.

Constructing a model for a dynamical system provides much more information than just the conservation laws. In fact, even estimating time derivatives is usually not necessary if we are only interested in identifying conserved quantities. In this work, we will instead focus on an alternative approach that does not require an explicit model or detailed time information but rather takes advantage of the geometric constraints imposed by conservation laws. Specifically, the presence of conservation laws restricts each trajectory in phase space to lie solely on a lower dimensional isosurface of the conserved quantities. The dimensionality of these isosurfaces can provide information about the number of conserved quantities or constraints [82]. Furthermore, since each isosurface corresponds to a particular set of conserved quantities, the variations in shape of the isosurfaces directly correspond to variations in the conserved quantities. In other words, we can identify and extract conserved quantities by examining the varying shapes of the isosurfaces sampled by the trajectories.

In contrast with recent work using black box deep learning methods to fit conserved quantities that are consistent with the sampled isosurfaces [47, 133], we propose and demonstrate a non-parametric manifold learning approach that directly characterizes the variations in the sampled isosurfaces, producing an embedding of the space of conserved quantities. Our method first uses the Wasserstein metric from optimal transport [126] to compute distances in shape space between pairs of sampled isosurfaces and then extracts a low dimensional embedding for the manifold of isosurfaces using diffusion maps [11, 32]. Each point in this embedding corresponds to a distinct isosurface and therefore to a distinct set of conserved quantities, i.e. the embedding explicitly parameterizes the space of varying conserved quantities. Related methods have been recently suggested for characterizing molecular conformations using the 1-Wasserstein distance together with diffusion maps [141] as well as performing system identification by comparing invariant measures using the 2-Wasserstein distance [139].

We provide an analytic analysis of our approach for a simple harmonic oscillator system and numerically test our method on several physical systems: the single and double pendulum, planar gravitational dynamics, the KdV equation for shallow water waves, and a nonlinear reaction–diffusion equation that generates an oscillating Turing pattern. We also investigate the robustness of our approach to noise in the measured trajectories as well as missing information in the form of a partially observed phase space.

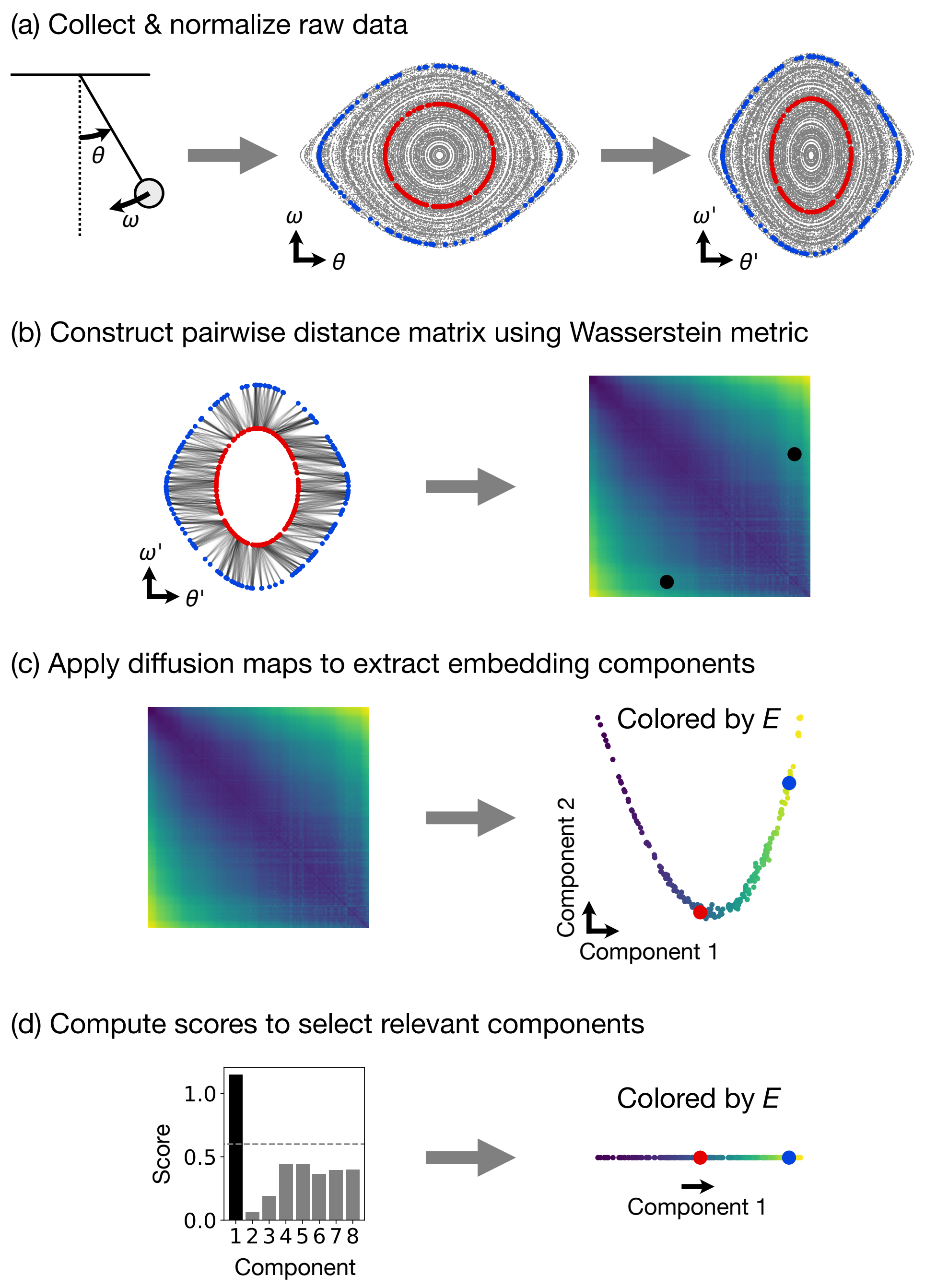

(a) Collect & normalize raw data
θ
ω
ω
θ
ω'
θ'
(b) Construct pairwise distance matrix using Wasserstein metric
ω'
θ'
(c) Apply diffusion maps to extract embedding components
Colored by E
Component 2
Component 1
(d) Compute scores to select relevant components
1.0
0.5
0.0
Score
1 2 3 4 5 6 7 8
Component
Colored by E
Component 1

Figure 4.1: Proposed non-parametric method for discovering conservation laws illustrated using a simple pendulum example (analyzed further in Sec. 4.4.2). (a) First, we collect and normalize the trajectory data from the dynamical system. Two example trajectories are highlighted in red and blue. (b) Then, we use the Wasserstein metric to compute the distance between each pair of trajectories and construct a distance matrix. For the two example trajectories, the optimal transport plan is shown, and the computed distance is marked on the distance matrix plot. (c) An embedding of the shape space manifold $\mathcal{C}$ is extracted from the distance matrix using diffusion maps. The embedding plot is colored by the energy of the pendulum E. The points corresponding to the two example trajectories are marked in red and blue. (d) Finally, a heuristic score (Appendix 4.B) is used to select relevant components. In this case, only component 1 is relevant, corresponding to a single conserved quantity—the energy E.

4.2 Proposed Manifold Learning Approach

Our proposed approach uses manifold learning to identify and embed the manifold of phase space isosurfaces sampled by the trajectories of a dynamical system. In particular, we compute a diffusion map (Fig. 4.1c) over a set of trajectories, each of which samples a particular phase space isosurface (Fig. 4.1a). The pairwise distances between these trajectories are given by the 2-Wasserstein distance (Fig. 4.1b), providing the metric structure necessary for applying diffusion maps. The manifold embedding extracted by the diffusion map corresponds directly to the space of conserved quantities (Fig. 4.1d). Note that this type of analysis does not require knowledge of the equations of motion (Eq. 4.1) and makes no direct reference to time.

4.2.1 Dynamical Systems

Consider a dynamical system with states $\mathbf{x} \in \mathcal{M}$ that live a in d-dimensional phase space $\mathcal{M}$ and evolve in time according to a system of first order ODEs

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}) \tag{4.1}$$

with n conserved quantities $G_1(\mathbf{x}), \ldots, G_n(\mathbf{x})$.

Conserved Quantities and Phase Space Isosurfaces

Along a particular trajectory $\mathbf{x}(t)$, the n conserved quantities form a set of constraints

$$G_i(\mathbf{x}) = c_i, \quad i \in \{1, 2, \ldots, n\} \tag{4.2}$$

which depend on the values of the conserved quantities $\mathbf{c} = \{c_1, c_2, \ldots, c_n\}$. This set of constraint equations restricts the trajectory to lie in a phase space isosurface $\mathcal{X}_\mathbf{c} \subseteq \mathcal{M}$ with dimension $d - n$. In fact, if any point of a trajectory lies on the isosurface $\mathcal{X}_\mathbf{c}$, then all other points from the trajectory will lie on the same isosurface.

By studying the variations in shape of these isosurfaces, we are able to directly characterize the space of conserved quantities. In particular, consider the manifold $\mathcal{C}$

formed by the isosurfaces $\mathcal{X}_{\mathbf{c}}$ in shape space. This manifold $\mathcal{C}$ is parameterized by the conserved quantities $\mathbf{c}$. Therefore, by analyzing $\mathcal{C}$ using manifold learning, we can extract the conservation laws of the underlying dynamical system.

Ergodicity and Physical Measures

To uniquely identify the isosurface associated with each trajectory, we must make several additional assumptions that will allow us to treat the set of points making up each trajectory as samples from an ergodic invariant measure on the corresponding isosurface. Specifically, we assume that, for each trajectory $\mathbf{x}(t)$ with conserved quantities $\mathbf{c}$, the dynamical system (Eq. 4.1) admits a physical measure [96] that is ergodic on the isosurface $\mathcal{X}_{\mathbf{c}}$ and is defined by

$$\mu_{\mathbf{c}} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \delta_{\mathbf{x}(t)}\, dt, \tag{4.3}$$

where $\delta_{\mathbf{x}(t)}$ is the Dirac measure centered on $\mathbf{x}(t)$. This ensures that trajectories with the same conserved quantities will sample the same distribution on the same isosurface, allowing us to use the distribution sampled by each trajectory as a proxy for the corresponding isosurface.

In practice, the sampled distribution may be lower dimensional than the corresponding isosurface if some of the conserved quantities do not vary in the dataset and instead correspond to fixed constraints, or if the dynamical system is dissipative. In the former case, this does not affect our ability to uniquely identify a distribution with an isosurface and its corresponding set of conserved quantities, meaning that we are able to apply this approach even if the provided phase space is much larger than the intrinsic phase space of the dynamical system. In the latter case, the dissipative nature of the system may cause information about conservation laws relevant during the transient portion of the dynamics to be lost, but we are still able to use our approach to identify conserved quantities relevant for the long term behavior of the system.

4.2.2 Wasserstein Metric

To analyze the isosurface shape space manifold $\mathcal{C}$—i.e. the manifold of conserved quantities—using manifold learning methods, we need to place some structure on the points $\mathcal{X}_{\mathbf{c}} \in \mathcal{C}$. Having associated each isosurface $\mathcal{X}_{\mathbf{c}}$ with a corresponding distribution defined by an ergodic physical measure $\mu_{\mathbf{c}}$, we choose to lift the Euclidean metric on the phase space into the space of distributions using the 2-Wasserstein metric from optimal transport

$$W_2(\mu_{\mathbf{c}}, \mu_{\mathbf{c}'}) = \left(\inf_{\pi \in \Pi(\mu_{\mathbf{c}}, \mu_{\mathbf{c}'})} \int c(\mathbf{x}, \mathbf{y})\, d\pi(\mathbf{x}, \mathbf{y}) \right)^{1/2}, \tag{4.4}$$

where the cost function $c(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$ is the squared Euclidean distance, and $\pi \in \Pi(\mu_{\mathbf{c}}, \mu_{\mathbf{c}'})$ is a valid transport map between $\mu_{\mathbf{c}}$ and $\mu_{\mathbf{c}'}$ [126].

For discrete samples, the 2-Wasserstein distance between two sets of sample points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ and $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_l\}$ is defined as

$$W_2 = \left(\min_T \sum_{i,j} T_{ij} C_{ij} \right)^{1/2} \tag{4.5}$$

such that

$$\begin{aligned} &T \geq 0 \\ &\sum_j T_{ij} = 1 \\ &\sum_i T_{ij} = 1, \end{aligned} \tag{4.6}$$

where the cost matrix $C_{ij} = \|\mathbf{x}_i - \mathbf{y}_j\|^2$. To efficiently compute an entropy regularized form of this optimization problem, we use the Sinkhorn algorithm [37] and estimate the Wasserstein distance as a debiased Sinkhorn divergence [59].

One important subtlety in this construction is the choice of the ground metric for optimal transport. As previously mentioned, we use a Euclidean metric on the phase space, which implicitly imposes a choice of units to make the phase space dimensionless. In fact, there is no canonical choice for the ground metric, and different choices result in different Wasserstein metrics on the shape space. For example, when multiple conserved quantities are present, the relative effect of each on the computed Wasserstein distances will determine how prominent each conserved quantity is and how easily it is identified using manifold learning. To partially mitigate this issue and improve consistency, we normalize each component of our data to have a maximum absolute value of 1 before computing the pairwise Wasserstein distances.

4.2.3 Diffusion Maps

Using the structure provided by the Wasserstein metric, we then use diffusion maps to generate an embedding for $\mathcal{C}$. The diffusion map manifold learning method uses a spectral embedding algorithm applied to an affinity matrix to construct a low dimensional embedding of the data manifold [11, 32]. Using the computed pairwise Wasserstein distances, we first construct a kernel matrix using a Gaussian kernel

$$K_{ij} = \exp(-W_2(\mu_i, \mu_j)^2/\epsilon) \tag{4.7}$$

and then scale it to form an affinity matrix for our spectral embedding

$$M_{ij} = K_{ij}/(D_i D_j)^\alpha, \tag{4.8}$$

where $D_i = \sum_k K_{ik}$, and α is a hyperparameter. The spectral embedding algorithm then takes this affinity matrix and constructs a normalized graph Laplacian

$$L_{ij} = I_{ij} - M_{ij}/\sum_k M_{ik}, \tag{4.9}$$

where I is the identity matrix. The eigenvectors $\mathbf{v}_i$ corresponding to the smallest eigenvalues $\lambda_i \geq 0$ (excluding $\lambda_0 = 0$) of the Laplacian then provide an approximate low dimensional embedding of the manifold of conserved quantities $\mathcal{C}$. In our experiments, we set $\alpha = 1$ so that the Laplacian computed by the spectral embedding algorithm approximates the Laplace–Beltrami operator [11].

To estimate the dimensionality of $\mathcal{C}$ and choose the eigenvectors $\mathbf{v}_i$ to include in our embedding, we use a heuristic score that combines a measure of relevance, given by a length scale computed from the Laplacian eigenvalues, with a previously suggested measure of "unpredictability" for minimizing redundancy [104]. To construct our embedding, we only include the Laplacian eigenvectors with score above a chosen cutoff value and discard the rest as either noise or redundant embedding components. In all of our experiments, we take the cutoff to be 0.6 and find that this value works well across a wide variety of datasets and systems. See Appendix 4.B for more details.

4.3 Analytic Result for the Simple Harmonic Oscillator

In the case of a simple harmonic oscillator (SHO) without measurement noise and in the infinite sample limit, we are able to explicitly derive an analytic result for our proposed procedure. We first compute the pairwise distances provided by the Wasserstein metric and then derive the embedding produced by a diffusion map, which corresponds to the conserved energy of the SHO.

4.3.1 Wasserstein Metric: Constructing the Isosurface Shape Space

Consider a SHO with Hamiltonian

$$H(q,p) = \frac{1}{2m}p^2 + \frac{1}{2}m\omega^2 q^2 \tag{4.10}$$

given in terms of position q and momentum p. The SHO energy isosurfaces form concentric ellipses in a 2D phase space. Choosing units such that $m = 1$ and $\omega = 1$, we obtain concentric circles with uniformly distributed samples (assuming a uniform sampling in time). The 2-Wasserstein distance between a pair of uniformly distributed circular isosurfaces is simply given by the difference in radii $|r_1 - r_2|$. This is because, due to the rotational symmetry of the two distributions, the optimal transport plan for an isotropic cost function is to simply move each point on isosurface 1 radially outward (or inward) to the point on isosurface 2 with the same angle θ.

This result does not meaningfully change with a different choice of units, which is equivalent to rescaling the phase space coordinates q, p. If we rescale q, p by factors k_q, k_p, our cost function simply becomes

$$\begin{aligned} c(\theta_i, \theta_j) = k_q^2(r_1 \cos\theta_i - r_2 \cos\theta_j)^2 \\ + k_p^2(r_1 \sin\theta_i - r_2 \sin\theta_j)^2, \end{aligned} \tag{4.11}$$

where we label points on the isosurfaces by their angle θ on the original circular isosurfaces. The SHO optimal transport plan Π takes θ on isosurface 1 to the point with the same angle θ on isosurface 2, and Π for the SHO is invariant to coordinate rescaling (Appendix 4.F). Therefore, the total transport cost is

$$C = \frac{1}{2\pi} \int_0^{2\pi} c(\theta, \theta)\, d\theta = \frac{k_q^2 + k_p^2}{2}(r_1 - r_2)^2, \tag{4.12}$$

so the 2-Wasserstein distance is

$$\sqrt{C} = \sqrt{(k_q^2 + k_p^2)/2}\, |r_1 - r_2| \propto |r_1 - r_2|, \tag{4.13}$$

i.e. the same result modulo a constant factor. While this is not a general result, we find that our approach is often fairly robust to such changes, including the extreme case of scaling some phase space coordinates all the way down to zero resulting in a partially observed phase space (Appendix 4.C).

4.3.2 Diffusion Maps: Extracting the Conserved Energy

Once we have pairwise distances in the isosurface shape space, we can use diffusion maps to study the resulting manifold of isosurface shapes. With sufficient samples, the operator constructed by the diffusion map should converge to the Laplace–Beltrami operator on the manifold. For the SHO, the isosurface shape space is isomorphic to $\mathbb{R}^+$ with each circular isosurface mapped to its radius. If we sample trajectories with radii $r \in (0, \sqrt{2E_0})$ for some maximum energy E_0, then the manifold is a real line segment, and the resulting Laplacian operator (with open boundary conditions) has eigenvalues $\lambda_n = \pi^2 n^2 / 2E_0$ and corresponding eigenvectors $v_n(r) = \cos(\sqrt{\lambda_n}\, r)$. Therefore, the first eigenvector or embedding component

$$v_1(E) = \cos(\pi \sqrt{E/E_0}) \tag{4.14}$$

successfully encodes the conserved energy and is, in fact, a monotonic function of the energy.

4.4 Numerical Experiments

To demonstrate the ability of our approach to discover conservation laws, we generate and test our proposed method on datasets from wide range of dynamical systems, each consisting of randomly sampled trajectories with different initial conditions and corresponding conserved quantities. Note that we use the dimensionless form of each dynamical system to generate our datasets. All of the code necessary for reproducing our results is available at `https://github.com/peterparity/conservation-laws-manifold-learning`.

4.4.1 Simple Harmonic Oscillator

We first numerically test our analytic result for the SHO and obtain good agreement (Fig. 4.2) using both the default scaling $k_q = k_p = 1$ (Figs. 4.2a–d) as well as the position only scaling $k_q = 1$, $k_p = 0$ (Figs. 4.2e–h), which effectively reduces the dimension of the phase space. A linear fit of the first embedding component from the diffusion map with the analytically predicted component (Eq. 4.14) achieves a correlation coefficient of $R^2 = 0.9995$ for the default scaling and $R^2 = 0.9961$ for the position only scaling. We also verify that the heuristic score (Appendix 4.B)

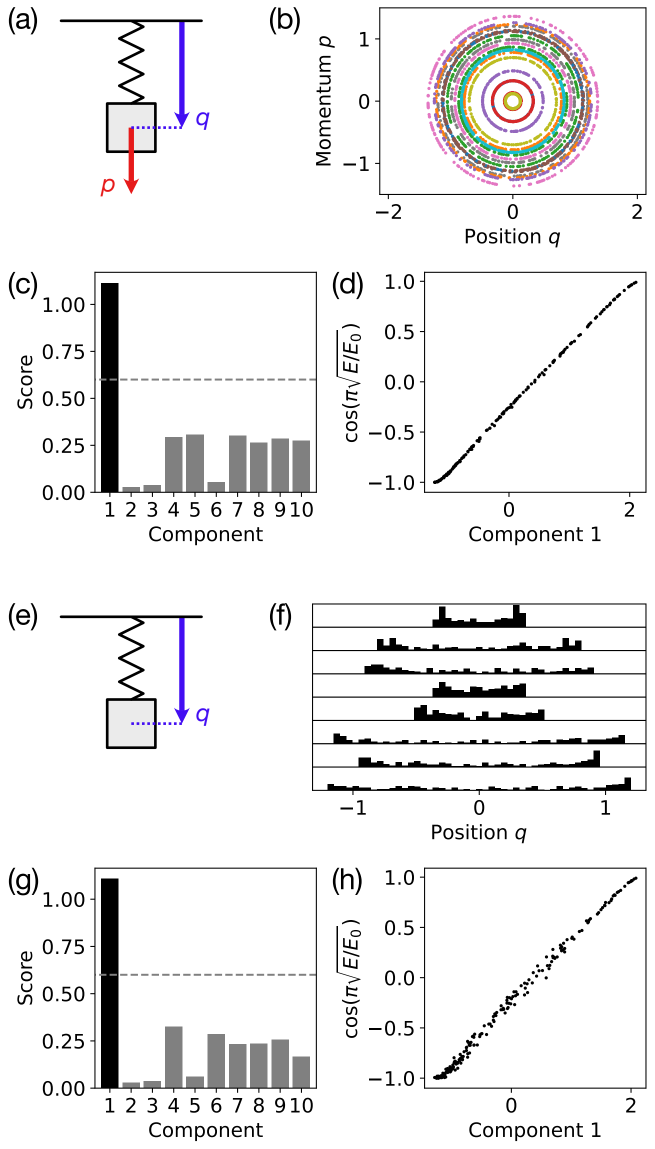


Figure 4.2: Identifying the conserved energy for (a) the simple harmonic oscillator (SHO). (b) Sample trajectories from the SHO dataset show sample points plotted in the 2D phase space (q, p). (c) The heuristic score (with cutoff 0.6) correctly identifies that the first embedding component extracted by the diffusion map is the only relevant component. (d) The extracted first component closely matches the analytically predicted first component (Eq. 4.14) for the SHO ($R^2 = 0.9995$). (e) Next, consider the SHO dataset with a partially observed phase space containing position only. (f) For each sample trajectory, the sample points are shown as a histogram. (g) The heuristic score is still able to identify the first component as relevant, and (h) this first component matches the analytic prediction ($R^2 = 0.9961$).

accurately determines that there is only one relevant embedding component (Figs. 4.2c, 4.2g), which corresponds to the conserved energy.

4.4.2 Simple Pendulum

To demonstrate our method on a simple nonlinear dynamical system, we analyze a simple pendulum that has a 2D phase space consisting of the angle θ and angular momentum ω (Fig. 4.3a). The equations of motion are

$$\begin{aligned} \frac{d\omega}{dt} &= -\sin\theta \\ \frac{d\theta}{dt} &= \omega. \end{aligned} \tag{4.15}$$

This system has a single scalar conserved quantity

$$E = \frac{1}{2}\omega^2 + (1 - \cos\theta) \tag{4.16}$$

corresponding to the total energy of the pendulum, so the trajectories form 1D orbits in phase space (Fig. 4.3b).

Our method is able to correctly determine that there is only a single conserved quantity (Fig. 4.3c) corresponding to the energy of the pendulum (Fig. 4.3d). The single extracted embedding component is monotonically related to the energy with Spearman's rank correlation coefficient $\rho = 0.9997$. We are also able to achieve similar results ($\rho = 0.9978$) with a high level ($\sigma = 0.5$) of added Gaussian noise (Figs. 4.3e–h), showing that our approach is quite robust to measurement noise.

4.4.3 Planar Gravitational Dynamics

To test our method on a system with multiple conserved quantities, we simulate the gravitational system of a planet orbiting a star (Fig. 4.4a). We fix the orbits to all lie in a 2D plane, giving us an effectively 4D phase space. The resulting equations of

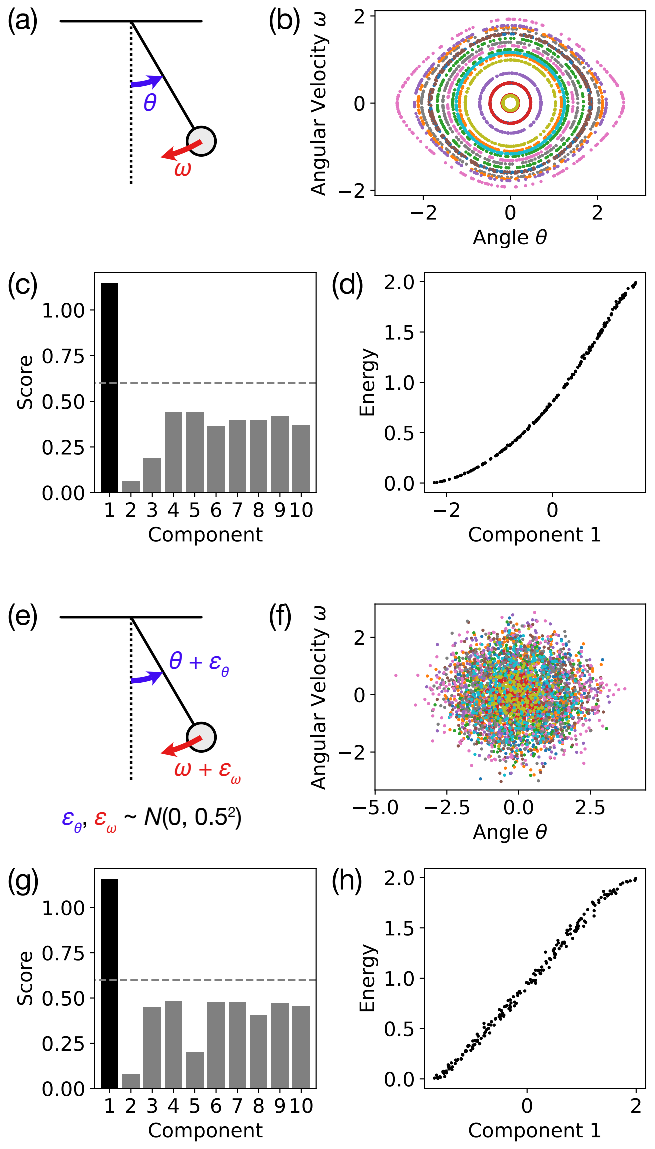


Figure 4.3: Identifying the conserved energy for (a) the simple pendulum. (b) Sample trajectories show sample points plotted in the 2D phase space (θ, ω). (c) The heuristic score (with cutoff 0.6) correctly identifies that the first embedding component extracted from by the diffusion map is the only relevant component, and (d) the extracted first component is monotonically related to the energy (rank correlation $\rho = 0.9997$). (e, f) With the addition of $\sigma = 0.5$ Gaussian noise to simulate measurement noise, (g) the heuristic score is still able to identify the first component as relevant, and (h) this first component corresponds well to the energy ($\rho = 0.9978$).

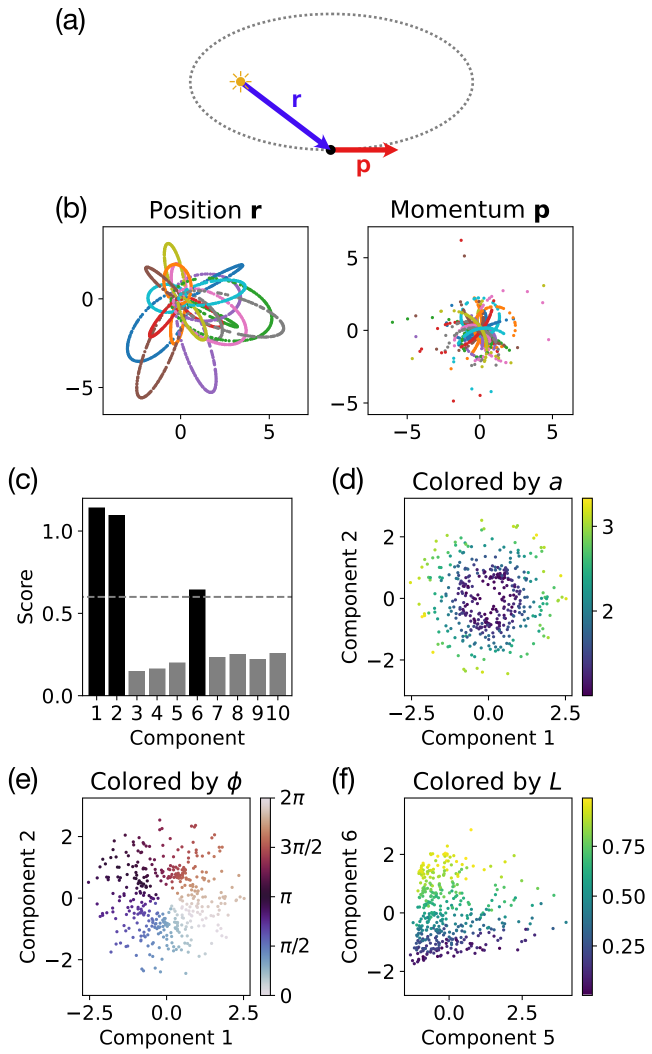


Figure 4.4: Identifying conserved quantities for (a) planar gravitational dynamics. (b) Sample trajectories show sample points plotted in 2D slices of the 4D phase space consisting of position $\mathbf{r}$ and momentum $\mathbf{p}$. (c) The heuristic score (with cutoff 0.6) identifies three relevant embedding components corresponding to the three independent conserved quantities. (d, e) Components 1 and 2 embed the the semi-major axis vector $\mathbf{a}$ with magnitude $a = -1/2E$ related to the energy and orientation given by the angle ϕ. (f) Component 6 corresponds to the angular momentum L.

motion are

$$\begin{aligned} \frac{d\mathbf{r}}{dt} &= \mathbf{p} \\ \frac{d\mathbf{p}}{dt} &= -\frac{\hat{\mathbf{r}}}{|\mathbf{r}|^2}. \end{aligned} \tag{4.17}$$

This system has one scalar and two vector conserved quantities

$$\begin{aligned} E &= \frac{\mathbf{p}^2}{2} - \frac{1}{|\mathbf{r}|} \\ \mathbf{L} &= \mathbf{r} \times \mathbf{p} \\ \mathbf{A} &= \mathbf{p} \times \mathbf{L} - \hat{\mathbf{r}}, \end{aligned} \tag{4.18}$$

which, in our 4D phase space, reduces to three scalar conserved quantities: the total energy E (or equivalently, the semi-major axis $a = -1/2E$), the angular momentum $L = |\mathbf{L}|$, and the orbital orientation angle ϕ, which is the angle of the LRL vector $\mathbf{A}$ relative to the x-axis. As a result, the trajectories also form 1D orbits in the phase space (Fig. 4.4b).

Our approach accurately identifies the three conserved quantities (Fig. 4.4c), and the extracted embedding corresponds most directly to the geometric features of the orbits (Figs. 4.4d–f). The first two components embed the semi-major axis vector $\mathbf{a} = (a \cos\phi, a \sin\phi)$ with magnitude given by the semi-major axis $a = -1/2E$, which is related to the energy E, and orientation given by the orientation angle ϕ of the elliptical orbit (Figs. 4.4d, 4.4e). The third relevant component (component 6) embeds the angular momentum L (Fig. 4.4f). A linear fit of the identified relevant embedding components with $a \cos\phi$ ($a \sin\phi$) has $R^2 = 0.987$ ($R^2 = 0.986$) and rank correlation $\rho = 0.994$ ($\rho = 0.992$). A similar linear fit with L has $R^2 = 0.927$ and $\rho = 0.970$.

This example demonstrates that, for a system with multiple conserved quantities, the ground metric for optimal transport controls the relative scale of each conserved quantity in the extracted embedding. In this case, the geometry of the shape space $\mathcal{C}$ is dominated by changes in the semi-major axis a and orientation angle ϕ, whereas changes in the angular momentum L, which controls the eccentricity of the orbit, play a more minor role and thus appear in a later embedding component with a lower score (Fig. 4.4c).

4.4.4 Double Pendulum

To test our approach on a non-integrable system with higher dimensional isosurfaces, we study the classic double pendulum system (Fig. 4.5a) with unit masses and unit length pendulum arms. This system has a 4D phase space, consisting of the angles θ_1, θ_2 and the angular velocities ω_1, ω_2 of the two pendulums (Fig. 4.5b), and only has a single scalar conserved quantity

$$E = \omega_1^2 + \frac{1}{2}\omega_2^2 + \omega_1\omega_2\cos(\theta_1 - \theta_2) - 2\cos\theta_1 - \cos\theta_2 \tag{4.19}$$

corresponding to the total energy. However, the double pendulum system has both chaotic and non-chaotic phases. In particular, at high energies, the system is chaotic and only conserves the total energy, while at low energies, the system behaves more like two coupled harmonic oscillators with two independent conserved energies

$$\begin{aligned} E_\pm = \frac{1}{8}\Big[&4\theta_1^2 + 2\theta_2^2 \pm \sqrt{2}\,\theta_1\theta_2 \\ &+ \left(2 \pm \sqrt{2}\right)\left(2\omega_1^2 + \omega_2^2\right) + 4\left(1 \pm \sqrt{2}\right)\omega_1\omega_2\Big] \end{aligned} \tag{4.20}$$

corresponding to the two modes of the coupled oscillator system. Therefore, we expect to see two distinct phases in our extracted embedding: one with a single conserved quantity E at high energy and another with two approximately conserved quantities $E_\pm$ at low energy, which approximately sum to $E \approx E_+ + E_-$.

At first glance, it appears as though our method has only identified a single relevant component corresponding to the conserved total energy E (Figs. 4.5c, 4.5e) with rank correlation $\rho = 0.996$. However, if we restrict ourselves to low energy trajectories with first embedding component $v_1 < -1$, we find that there is a region of the shape space which is two-dimensional, corresponding to the two independently conserved energies $E_\pm$ of the low energy non-chaotic phase where the double pendu-

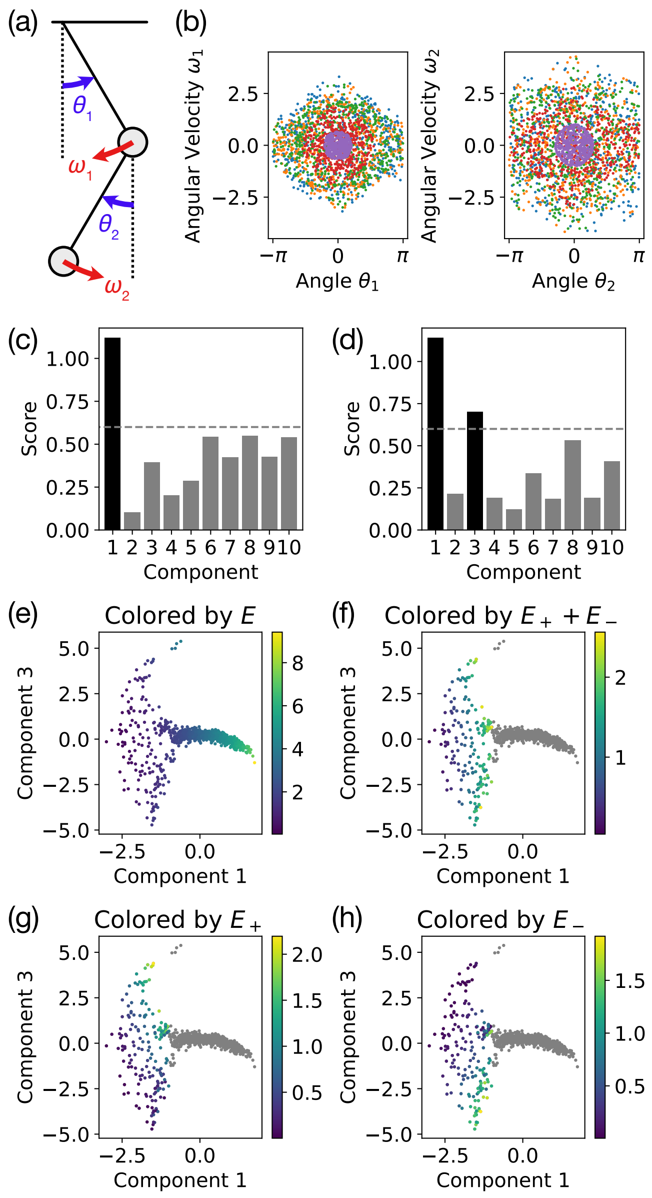


Figure 4.5: Identifying conserved quantities for (a) the double pendulum. (b) Sample trajectories show sample points plotted in 2D slices of the 4D phase space consisting of the pendulum angles θ_1, θ_2 and angular velocities ω_1, ω_2. (c) The heuristic score (with cutoff 0.6) identifies one relevant embedding component corresponding to (e) the total energy E. (d) However, if we restrict the embedding to trajectories with first component $v_1 > -1$ (i.e. low energy trajectories) and renormalize the embedding, we find (f–h) two conserved quantities corresponding to the energies $E_\pm$ of the two decoupled low energy modes. The gray points in Figures 4.5f–h correspond to the high energy trajectories (first component $v_1 < -1$) which are not relevant when considering the low energy non-chaotic phase of the double pendulum.

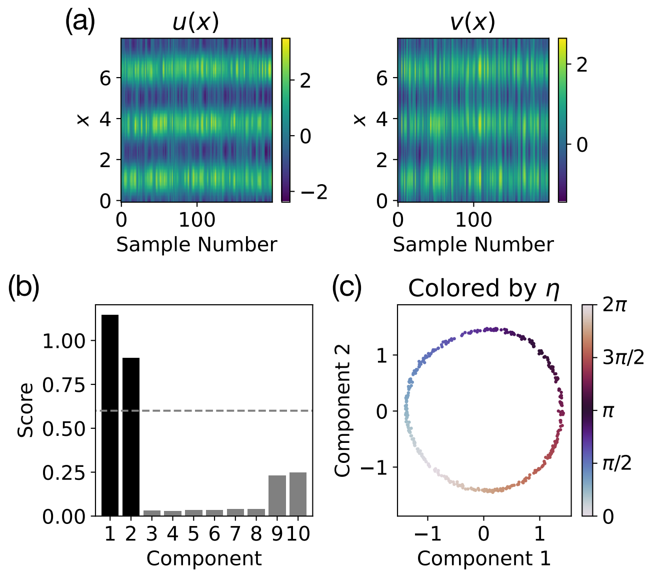


Figure 4.6: Identifying the conserved spatial phase for the oscillating Turing pattern system. (a) An example trajectory, with randomly sampled states $u(x)$ and $v(x)$ plotted, illustrates the high dimensional nature of the problem. (b) The heuristic score (with cutoff 0.6) identifies two relevant components, but on further examination, (c) we see that there is just a single conserved angle, corresponding to the spatial phase η of the Turing pattern, that needs to be embedded in two dimensions due to its topology.

lum behaves like a coupled oscillator system with two distinct modes. For the low energy trajectories, a linear fit of the now two relevant components with E_+ (E_-) has rank correlation $\rho = 0.836$ ($\rho = 0.937$). If we restrict ourselves to even lower energy trajectories with $v_1 < -2$, a similar linear fit for E_+ (E_-) has rank correlation $\rho = 0.934$ ($\rho = 0.989$).

This analysis of the double pendulum shows that our method can still provide significant insight into complex dynamical systems with multiple phases involving varying numbers of conserved quantities. This manifests itself as manifolds of different dimensions in shape space that are stitched together at phase transitions, presenting a significant challenge for most manifold learning methods. In this example, this difficulty is reflected in the performance of the heuristic score, which is designed to identify relevant embedding components for a single manifold.

4.4.5 Oscillating Turing Patterns

Next, we consider an oscillating Turing pattern system that is both dissipative and has a much higher dimensional phase space than our previous examples. In particular, we study the Barrio–Varea–Aragón–Maini (BVAM) model [7, 9]

$$\begin{aligned} \frac{\partial u}{\partial t} &= D\frac{\partial^2 u}{\partial x^2} + u - v - Cuv - uv^2 \\ \frac{\partial v}{\partial t} &= \frac{\partial^2 v}{\partial x^2} - \frac{3}{2}v + Hu + Cuv + uv^2 \end{aligned} \tag{4.21}$$

with $D = 0.08$, $C = -1.5$, and $H = 3$, following Aragón et al. [7] who showed that this set of parameters results in a spatial Turing pattern that also exhibits chaotic oscillating temporal dynamics, on a periodic domain with size 8. The phase space of the BVAM system consists of the two functions $u(x)$ and $v(x)$ which we discretize on a mesh of size 50, giving us an effective phase space dimension of 100. Because this system is dissipative, we will focus on characterizing the long term behavior of the dynamics, i.e. the oscillating Turing pattern, which appears to have a conserved spatial phase η for our chosen set of parameters corresponding to the spatial position of the Turing pattern.

Our method successfully identifies the spatial phase η but embeds the angle as a circle in a 2D embedding space (Fig. 4.6)—a result of the periodic topology of η. While this shows that the number of relevant components determined by our heuristic score may not always match the true manifold dimensionality, such cases are often easily identified by examining the components directly (Fig. 4.6c) or by cross checking with an intrinsic dimensionality estimator [13]. A linear fit of the two relevant components with $\cos\eta$ ($\sin\eta$) has $R^2 = 0.9991$ ($R^2 = 0.9997$) and $\rho = 0.9993$ ($\rho = 0.9992$). This example both tests our method on a high dimensional phase space and demonstrates how our approach can be applied to dissipative systems to study long term behavior.

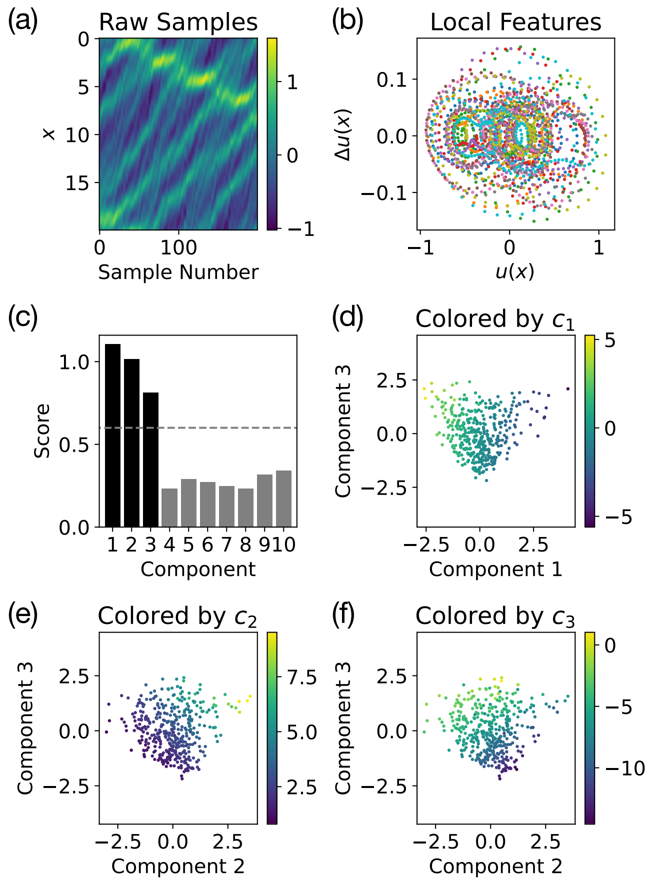


Figure 4.7: Identifying three local conserved quantities of the Korteweg–De Vries (KdV) equation. (a) An example trajectory from the KdV dataset shows the high dimensional raw sampled states $u(x)$. (b) To focus on local conserved quantities, we extract a distribution of the local features $u(x), \Delta u(x)$ from the raw states, removing the explicit spatial label. The plot shows the local feature distributions for a few sample states. (c) The heuristic score (with cutoff 0.6) correctly identifies three relevant components corresponding to (d–f) the three local conserved quantities (Eq. 4.23).

4.4.6 Korteweg–De Vries Equation

For many spatiotemporal dynamical systems, the conservation laws are local in nature. Locality can significantly simplify the analysis of the conserved quantities and suggests a way to restrict the type of conserved quantities identified by our method. Specifically, we can adapt our approach to focus on local conserved quantities by replacing the raw states (Fig. 4.7a) by a distribution of local features (Fig. 4.7b), removing the explicit spatial label and providing a fully translation invariant representation of the state. Then, instead of using the Euclidean metric in the original phase space, we use the energy distance [40, 110] between the distributions of local features as the ground metric for optimal transport.

To demonstrate this method for identifying local conserved quantities, we consider the Korteweg–De Vries (KdV) equation

$$\frac{\partial u}{\partial t} = -\frac{\partial^3 u}{\partial x^3} - 6u\frac{\partial u}{\partial x}. \tag{4.22}$$

The KdV equation is fully integrable [43] and has infinitely many conserved quantities [97], the most robust of which are the most local conserved quantities expressible in terms of low order spatial derivatives. To focus on these robust local conserved quantities, we use finite differences (i.e. $u(x), \Delta u(x) = u(x+\Delta x) - u(x), \Delta^2 u(x), \ldots$) as our local features, allowing us to restrict the spatial derivative order of the identified conserved quantities. In this experiment, we only take $u(x)$ and $\Delta u(x)$, meaning that the identified local conserved quantities will only contain up to first order spatial derivatives. For the KdV equation, there are three such local conserved quantities:

$$\begin{aligned} c_1 &= \int_0^l u\,dx \\ c_2 &= \int_0^l u^2\,dx \\ c_3 &= \int_0^l \left[u^3 - \frac{1}{2}\left(\frac{\partial u}{\partial t}\right)^2\right] dx, \end{aligned} \tag{4.23}$$

which also have direct analogues in generalized KdV-type equations [5].

Our method successfully identifies three relevant components (Fig. 4.7c) corresponding to (d–f) the three local conserved quantities (Eq. 4.23). Linear fits of these components to c_1, c_2, and c_3 have rank correlations $\rho = 0.995$, 0.994, and 0.985, respectively. This result shows how our approach can be adapted to incorporate known structure, such as locality and translation symmetry, in applications to complex high dimensional dynamical systems.

4.5 Conclusion

We have proposed a non-parametric manifold learning method for discovering conservation laws, tested our method on a wide variety of dynamical systems—including complex chaotic systems with multiple phases and high dimensional spatiotemporal dynamics—and also shown how to adapt our approach to incorporate additional structure such as locality and translation symmetry. Our method does not assume or construct an explicit model for the system nor require accurate time information like previous approaches [66, 83], only relying on the ergodicity of the dynamics modulo the conservation laws (Sec. 4.2.1). As a result, our method is also quite robust to measurement noise and can often deal with missing information such as a partially observed phase space (Figs. 4.2e–h, Figs. 4.3e–h, Appendix 4.C).

Compared with recently proposed deep learning-based methods [47, 133], our approach is much more interpretable since it relies on explicit geometric constructions and well-studied manifold learning methods that directly determine the geometry of the shape space $\mathcal{C}$ and, therefore, the identified conserved quantities. This is reflected in our ability to explicitly derive the expected result for the simple harmonic oscillator (Sec. 4.3), as well as in the identified conserved quantities in many of our experiments. For example, the embedding of the semi-major axis vector in the planar gravitational dynamics experiment (Sec. 4.4.3) stems directly from the elliptical geometry of the orbits and their orientation in phase space, which is captured by the Euclidean ground metric and lifted into shape space by optimal transport. Our method also correctly captures the subtleties of the double pendulum system (Sec. 4.4.4) by providing an embedding that shows both a 1D manifold at high energies and a 2D manifold at low energies—a difficult prospect for deep learning approaches that try to explicitly fit conserved quantities.

Our manifold learning approach to identifying conserved quantities provides a new way to analyze data from complex dynamical systems and uncover useful conservation laws that will ultimately improve our understanding of these systems as well as aid in developing predictive models that accurately capture long term behavior. Our method also serves as a strong non-parametric baseline for future methods that aim to discover conservation laws from data. We also believe that similar combinations of optimal transport and manifold learning have the potential to be applied to a wide variety of other problems that also rely on geometrically characterizing families of distributions and hope to investigate such applications in the near future.

4.A Dataset Details

The SHO dataset contains 200 sample trajectories, each with 200 uniformly sampled states in time.

The simple pendulum dataset contains 200 trajectories with uniformly sampled energies $E \in [0, 2]$. Each trajectory has 200 sampled states at uniformly sampled times $t \in [0, 2000]$.

The planar gravitational dynamics dataset contains 400 trajectories with uniformly sampled energies $E \in [-0.15, -0.5]$, angular momenta $L \in [0, 1]$, and orbital orientation angles $\phi \in [0, 2\pi)$. Each trajectory has 200 sampled states at uniformly sampled times $t \in [0, 2000]$.

The double pendulum dataset contains 1000 trajectories with initial angles $\theta_1, \theta_2 \sim \mathrm{Unif}(-0.75\pi, 0.75\pi)$ and initial angular velocities $\omega_1, \omega_2 \sim N(0, 0.5^2)$. Each trajectory contains 500 points uniformly sampled in time $t \in [0, 50000]$. One additional subtlety of applying our approach to the double pendulum comes from the periodicity of the angles θ_1, θ_2 describing the positions of the two pendulums. The Euclidean ground metric used for optimal transport must take into account this periodicity, so we choose to leave the data unnormalized and use the shortest Euclidean distance between pairs of points in the periodic phase space.

The oscillating Turing pattern dataset contains 400 trajectories, where we initialize our states $u(x)$ and $v(x)$ with unit Gaussian noise in Fourier space and take 200 states with uniformly sampled times $t \in [300, 1300]$. By allowing for a transient time of 300, we focus our study on the long term behavior of the oscillating Turing pattern.

Finally, we study the KdV equation on a periodic domain of size $l = 20$ and with mesh size 200 (downsampled from a mesh size of 1000 used during data generation). The dataset contains 400 trajectories each with 200 states at uniformly sampled times $t \in [0, 10]$. To produce a reasonable variety of initial conditions, each trajectory is initialized with normally distributed Fourier components scaled by a Gaussian band-limiting envelope with width uniformly sampled in the interval $[10\pi/l, 20\pi/l]$.

4.B Heuristic Score for a Minimal Diffusion Maps Embedding

Traditionally, diffusion maps [32] and Laplacian eigenmaps [11] leave the embedding dimension n as a hyperparameter and simply use the eigenvectors corresponding to the n smallest eigenvalues to construct the embedding. In practice, the embedding dimension n is often chosen for convenience (e.g. in visualization applications) or by examining the eigenvalues λ_i and looking for a sharp increase in the magnitude of the eigenvalues that would separate the signal from the noise. Because identifying the number of conservation laws is an important step in our approach, we refine this heuristic by directly computing an approximate length scale

$$l_i = \sqrt{-\epsilon/\log(1-\lambda_i)}, \tag{4.24}$$

where ϵ is the scale factor from the Gaussian kernel used to construct the Laplacian matrix L (Eq. 4.7). We derive this length scale by considering the normalized kernel $I - L$ to be an approximation of the heat kernel $\exp(\epsilon\Delta)$, implying that the length scales l_i associated with the Laplace–Beltrami operator Δ are given by

$$\exp(\epsilon\Delta) = I - L \implies \exp(-\epsilon/l_i^2) = 1 - \lambda_i. \tag{4.25}$$

We then divide by l_1 to obtain the relative length scale

$$\frac{l_i}{l_1} = \sqrt{\frac{\log(1-\lambda_1)}{\log(1-\lambda_i)}}, \tag{4.26}$$

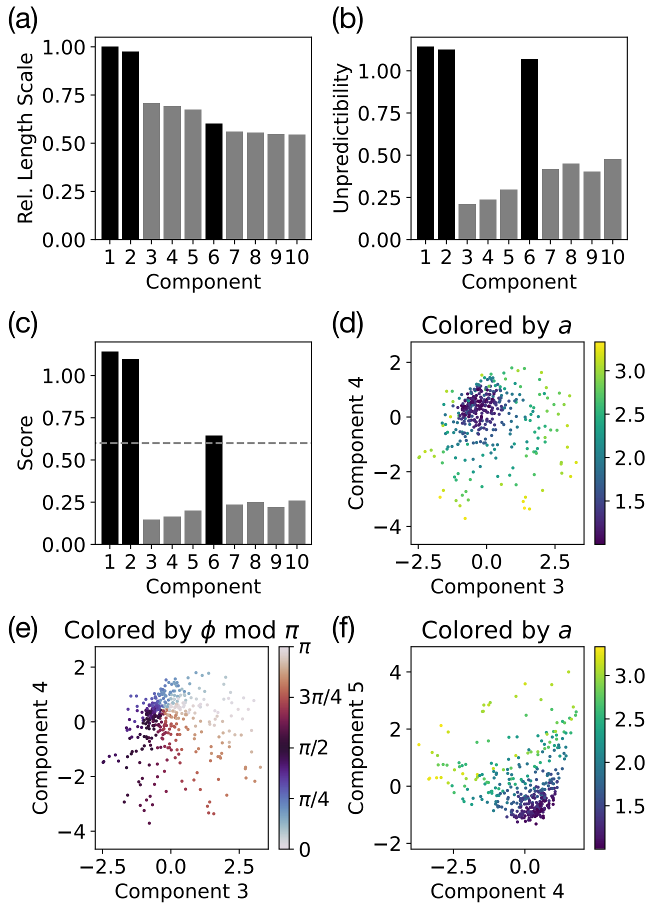


Figure 4.8: Breakdown of the heuristic score and illustration of redundant embedding components from the planar gravitational dynamics experiment. (a) The relative length scale l_i/l_1 for each embedding component is computed from the corresponding eigenvalue λ_i of the Laplacian matrix (Eq. 4.24). (b) The unpredictability measure m_i for each component is computing using a nearest neighbor estimator [104]. (c) The combined score $m_i l_i/l_1$ is the product of the relative length scale and the unpredictability measure. (d–f) The components 3, 4, and 5 are identified by the unpredictability measure as redundant. If we examine these three components, we find that they together embed a second order angular mode of components 1 and 2 (Figs. 4.4d, 4.4e). In particular, the embedding is shaped like the surface of a cone with the height (or radial distance) roughly corresponding to the semi-major axis a and the angle around the cone corresponding to ϕ mod π, a second order mode of the orientation angle ϕ.

which can be used as a heuristic measure of relevance—components with a small relative length scale are more likely to be noise. Compared with directly using the eigenvalues λ_i, we find this heuristic to be less sensitive to the choice of ϵ in the kernel.

In addition to noise, there is the common problem of redundant embedding components that stem from the structure of the Laplacian operator: higher order modes of previous eigenvectors often appear before more informative eigenvectors corresponding to new manifold directions. This problem is clearly illustrated in the planar gravitational dynamics experiment (Sec. 4.4.3), where components 3, 4, and 5 are all redundant with components 1 and 2 but component 6 is a new and relevant conserved quantity (Figs. 4.8d–f). To address this issue, the key observation is that, while all components of the diffusion map are linearly independent, redundant components are still predictable (via a nonlinear function) from previous components. Therefore, we require a measure of "unpredictablility" that allows us to identify redundancies. We choose the heuristic m_i proposed by Pfau and Burgess [104] that uses a nearest neighbor estimator (using 5 nearest neighbors) to determine whether a new embedding component is too predictable and therefore redundant.

To compute our final heurstic score (Fig. 4.8c), we take the product of the relative length scale l_i/l_1 (Fig. 4.8a) with the unpredictability measure m_i (Fig. 4.8b). We find this simple combined score performs well for identifying relevant embedding components by removing both noise components as well as redundant components.

4.B.1 Choosing a Score Cutoff

To use the heuristic score to identify the number of conserved quantities and construct a minimal embedding, we require a score cutoff to separate relevant components that we keep in our embedding from irrelevant components that we discard. To choose this cutoff, we sweep cutoff values in the interval $[0, 1]$, compute the embedding size (i.e. the number of relevant components) based on the chosen cutoff, and then examine the result to identify a robust value for the cutoff (Fig. 4.9).

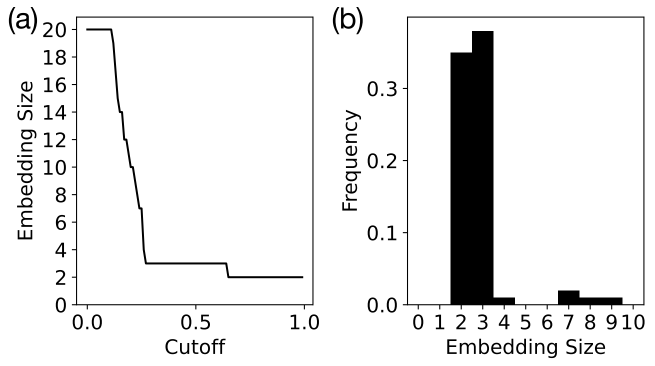


Figure 4.9: Example from the planar gravitational dynamics experiment of identifying the number of conserved quantities (i.e. the embedding size). (a) Sweeping the cutoff value from 0 to 1, we find plateaus indicating robustness at embedding size 2 and 3. Note that there is a spurious plateau at the maximum embedding size 20. (b) A histogram of the embedding sizes confirms that the number of conserved quantities is likely to be 3.

Table 4.1: Rank correlations ρ of linear fits with ground truth conserved quantities for the additional experiments.

Dataset	Conserved Quantity	ρ
Simple Pendulum: *Position Only*	E	0.998
Simple Pendulum: *Position Only + Noise*	E	0.996
Planar Gravitational Dynamics: *Position Only*	$a\cos\phi$	0.994
	$a\sin\phi$	0.993
	L	0.968
Planar Gravitational Dynamics: *Noise*	$a\cos\phi$	0.994
	$a\sin\phi$	0.992
	L	0.945

4.C Additional Experiments

To further demonstrate the robustness of our approach, we show several additional experiments on the simple pendulum and planar gravitational dynamics datasets. For the simple pendulum, our method still performs well when using only angle θ measurements, i.e. a partially observed phase space (Fig. 4.10a). In fact, even if we add Gaussian noise ($\sigma = 0.5$) on top of the partially observed phase space, we still obtain a similar result (Fig. 4.10b). Similarly, for planar gravitational dynamics with only position $\mathbf{r}$ data or with added Gaussian noise ($\sigma = 0.5$), our method is still able

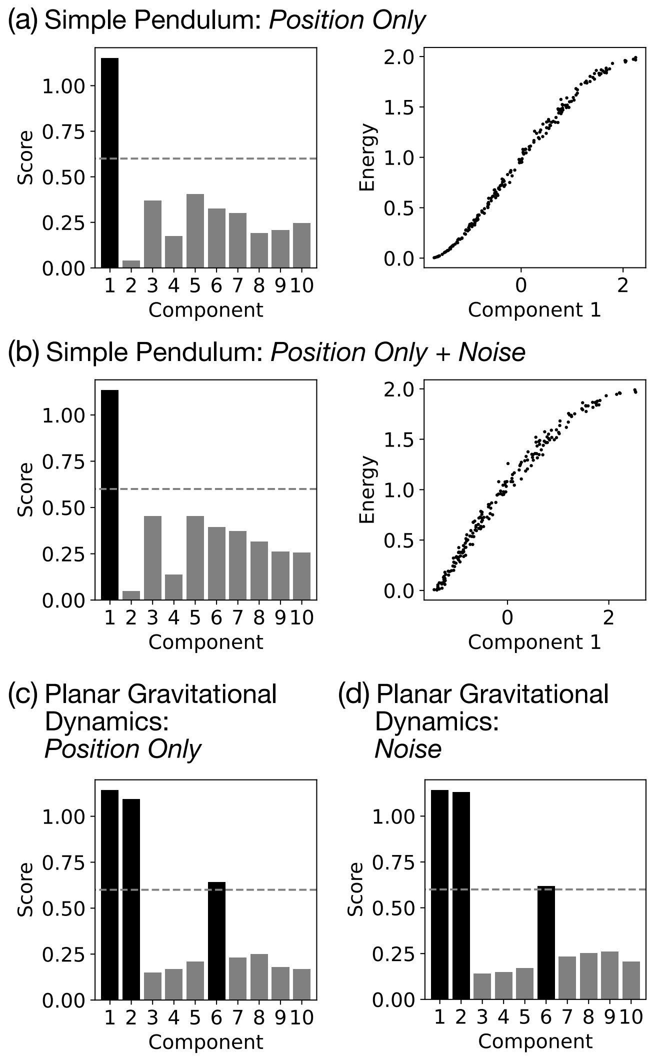


Figure 4.10: Additional experiments illustrating the robustness of our approach. (a) For the simple pendulum system, even when provided only angle θ measurement data (without angular velocity ω), our method is able to identify a single relevant component corresponding to the energy the pendulum ($\rho = 0.998$). (b) If we then also add $\sigma = 0.5$ Gaussian noise, we can still achieve a similar result ($\rho = 0.996$). For planar gravitational dynamics, our method also performs well given (c) only position $\mathbf{r}$ data or (d) with $\sigma = 0.5$ Gaussian noise, correctly identifying the three conserved quantities.

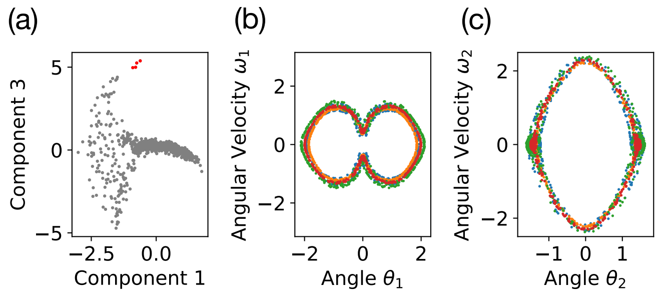


Figure 4.11: Nonlinear periodic orbit of the double pendulum. (a) The four red highlighted points in the extracted embedding correspond to (b, c) a periodic orbit of the double pendulum that is connected to but well outside of the linear coupled oscillator regime.

to identify the three conserved quantities (Figs. 4.10c, 4.10d). The corresponding rank correlations with the ground truth conserved quantities are given in Table 4.1.

4.D Nonlinear Periodic Orbit of the Double Pendulum

In addition to the chaotic and linear non-chaotic phases, the double pendulum can also exhibit other kinds of complex behavior, including highly nonlinear periodic orbits. In our extracted embedding (Fig. 4.11a), we see an example of such a nonlinear periodic orbit (Figs. 4.11b, 4.11c). The placement of this periodic orbit in the embedding also meaningfully connects it with the low energy in-phase mode from the linear coupled oscillator regime (Fig. 4.5g), i.e. this periodic orbit can be thought of as a nonlinear high energy extension of the low energy in-phase mode.

4.E Additional Method Details

All of the code necessary for generating our datasets, applying our method, and reproducing our results is available at `https://github.com/peterparity/conservation-laws-manifold-learning`.

4.E.1 Sinkhorn Algorithm

To estimate the 2-Wasserstein distance using the Sinkhorn algorithm [37], we use a convergence threshold of 0.01 and a decaying entropy regularization parameter that starts at 10.0 and decays by 0.995 at each step until it reaches a target of 0.1. This computation of pairwise Wasserstein distances between the trajectories is currently the performance bottleneck of our approach but is easily parallelized over multiple GPUs using the OTT-JAX library [38]. One option to speed up convergence, which we hope to investigate in the future, is to allow for a significantly larger target regularization parameter. The result would remain a valid distance metric that, in fact, interpolates between the Wasserstein metric and a maximum mean discrepancy (MMD) metric [40].

4.E.2 Diffusion Maps

To improve the noise robustness of our diffusion map, we follow Karoui and Wu [68] and replace the diagonal of the affinity matrix M (Eq. 4.8) with zeros, i.e.

$$M^*_{ij} = M_{ij} - M_{ii}I_{ij}, \tag{4.27}$$

before constructing the Laplacian matrix L. Because this induces an overall shift in the eigenvalues of the Laplacian that interacts poorly with our length scale heuristic (Eq. 4.24), we correct for this by subtracting off the normalized mean shift

$$s = \frac{1}{N}\sum_{i=1}^{N}\left(M_{ii}/\sum_{j} M_{ij}\right) \tag{4.28}$$

from the Laplacian matrix L to obtain the corrected Laplacian

$$L^*_{ij} = L_{ij} - sI_{ij}, \tag{4.29}$$

which we use to generate our embeddings.

4.F Proof of Optimal Transport for the Simple Harmonic Oscillator

Let the transport cost between a pair of points $(\theta_i, \theta_j) \in S^1 \times S^1$ be

$$c(\theta_i, \theta_j) = k_q^2 (r_1 \cos\theta_i - r_2 \cos\theta_j)^2 + k_p^2 (r_1 \sin\theta_i - r_2 \sin\theta_j)^2. \tag{4.30}$$

Then, for the proposed optimal transport plan Π with support Γ containing all points $(\theta, \theta) \in S^1 \times S^1$, we will show that Γ is c-cyclically monotone, and therefore Π is optimal. See Medio and Lines [96] for further details.

To demonstrate this fact, consider a finite set of pairs $\{(\theta_1, \theta_1), (\theta_2, \theta_2), \ldots, (\theta_n, \theta_n)\} \subset \Gamma$. Restricted to this finite set, the total cost given the transport plan Π is

$$C = \frac{1}{n} \sum_{i=1}^{n} c(\theta_i, \theta_i) \tag{4.31}$$

$$= \frac{r_1^2 + r_2^2}{n} \sum_{i=1}^{n} (k_q^2 \cos^2\theta_i + k_p^2 \sin^2\theta_i) - \frac{2 r_1 r_2}{n} \sum_{i=1}^{n} (k_q^2 \cos^2\theta_i + k_p^2 \sin^2\theta_i). \tag{4.32}$$

Now, consider an alternative transport plan Π' with support $\{(\theta_1, \theta_2), (\theta_2, \theta_3), \ldots, (\theta_n, \theta_1)\}$ forming a cycle. The total cost is given by

$$C' = \frac{1}{n} \sum_{i=1}^{n} c(\theta_i, \theta_{i+1}) \tag{4.33}$$

$$= \frac{r_1^2 + r_2^2}{n} \sum_{i=1}^{n} (k_q^2 \cos^2\theta_i + k_p^2 \sin^2\theta_i) - \frac{2 r_1 r_2}{n} \sum_{i=1}^{n} (k_q^2 \cos\theta_i \cos\theta_{i+1} + k_p^2 \sin\theta_i \sin\theta_{i+1}), \tag{4.34}$$

where we let $\theta_{n+1} = \theta_1$. Then, the difference

$$\begin{aligned} C' - C &= \frac{2r_1 r_2 k_q^2}{n} \sum_{i=1}^{n} \left[\frac{\cos^2\theta_i + \cos^2\theta_{i+1}}{2} - \cos\theta_i \cos\theta_{i+1} \right] \\ &\quad + \frac{2r_1 r_2 k_p^2}{n} \sum_{i=1}^{n} \left[\frac{\sin^2\theta_i + \sin^2\theta_{i+1}}{2} - \sin\theta_i \sin\theta_{i+1} \right] \qquad (4.35) \\ &\geq 0, \end{aligned}$$

since

$$\frac{\cos^2\theta_i + \cos^2\theta_{i+1}}{2} \geq \cos\theta_i \cos\theta_{i+1} \qquad (4.36)$$

$$\frac{\sin^2\theta_i + \sin^2\theta_{i+1}}{2} \geq \sin\theta_i \sin\theta_{i+1} \qquad (4.37)$$

by the AM–GM inequality (and trivially true if the right hand side is negative). Therefore, any such cycle will result in an equal or higher transport cost (strictly higher if at least one pair θ_i, θ_{i+1} are distinct), implying that Γ is c-cyclically monotone.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Conclusion and Future Work

In this thesis, we have proposed and demonstrated three machine learning approaches designed specifically for scientific applications. While much of this current work is aimed at analyzing data from nonlinear dynamical systems, the same concepts and methods can be applied across a broad range of problems in physics as well as other scientific or engineering disciplines. The underlying barriers to using machine learning in these scientific domains are the same: namely, the lack of interpretability and insight into the learned model and the need for physical priors or inductive biases in the design of machine learning architectures and algorithms. To address these issues moving forward, we propose the following broad research directions:

1) design and optimize physics-informed machine learning architectures that incorporate physical laws, symmetries, and constraints to take advantage of the inductive biases provided by known physical structures;

2) integrate these physics-informed architectures with existing computational methods to combine the flexibility of neural networks with the theoretical guarantees of traditional algorithms; and

3) develop new methods for learning interpretable physical representations from raw data in order to provide new insights into the underlying physics and aid in the process of scientific discovery.

5.1 Physics-Informed Architectures

In Chapter 2, we demonstrated the power of symbolic models to improve generalization performance and impose physics priors, but we often give up expressivity in the process due to the need to select a predefined library of terms. My collaborators and I have also examined how to enhance the expressivity of differentiable symbolic models to allow arbitrary function compositions [73]. Our proposed symbolic model is designed for sparse symbolic regression tasks but is also easily incorporated into machine learning architectures, combining interpretable symbolic physics with the flexibility and power of deep learning. One of the primary difficulties with training such models is the question of how to effectively impose sparsity while retaining differentiability. Standard L_1 regularization often results in degraded model performance, so recent works that take a probabilistic sampling approach to selecting sparse symbolic expressions offer an intriguing alternative [34].

In Chapter 3, our use of an equivariant physics-informed architecture allowed us to extract relevant parameters even with very small datasets (Appendix 3.F) and to adapt, without retraining, our learned predictive model to work with a different set of boundary conditions (Appendix 3.G). The significant body of work on equivariant architectures [6, 31, 67, 77, 124] and other physics-informed neural network architectures—e.g. for solving PDEs [80] or for simulating quantum chemistry [51, 105]—promises to make deep learning methods more data efficient and better suited for learning from structured scientific data. Optimizing expressive symbolic models and equivariant architectures as well as developing new physics-informed architectures will be important steps toward improving interpretability and creating effective machine learning methods for scientific applications.

5.2 Enhancing Existing Computational Methods

In addition to designing more efficient and expressive physics-informed architectures, great gains in performance and interpretability can come from combining such architectures with traditional theoretically-motivated models as well as integrating them into existing computational methods. Our framework in Chapter 2 augments a symbolic model with an encoder, enabling partially observed system identification. In Chapter 3, we showed how a predictive model that is structured like a physics simulator imposes a strong inductive bias on the learned latent space, resulting in interpretable latent parameters and high quality predictions. My collaborators and I have also demonstrated this concept when we examined the performance of using physics-informed Bayesian neural network architectures as surrogate models for Bayesian optimization on scientific problems [72]. We found that using appropriately structured architectures alongside relevant auxiliary information about the physical system often significantly outperforms alternative methods, such as Gaussian processes with handcrafted kernels.

The same theme is playing out across many research areas, including recent work on phase retrieval [129], differentiable physics simulators [117], inverse design for photonics [61], Koopman operator-based methods for modeling dynamical systems [91], flow-based sampling for statistical many-body systems [99] and lattice gauge theory [67], and neural network quantum states for variational Monte Carlo [21]. By combining machine learning architectures with existing models and computational methods, it is possible to obtain interpretable and highly generalizable models, and, in some cases, theoretical bounds or guarantees of correctness that would otherwise be absent when naively applying deep learning to scientific problems.

5.3 Learning Interpretable Representations

As we demonstrated in Chapter 3, unsupervised representation learning methods can provide a novel way to gain additional insight into unknown physical systems and discover new physics. In addition to autoencoder-based methods, we introduced an alternative geometric approach in Chapter 4 that relies on manifold learning to uncover the underlying structure of an existing high-dimensional representation. My collaborators and I have also used manifold learning in combination with reservoir computing methods to study variations in chaotic dynamical systems [2]. Manifold learning is currently being used in a variety of scientific applications, from identifying phase transitions in condensed matter physics [92] to visualizing complex high-dimensional data in computational biology [98]. The related field of contrastive representation learning is also becoming increasingly popular, especially for computer vision tasks [27], and we believe there are significant opportunities to apply these new methods to scientific domains.

We plan to further investigate how unsupervised representation learning, including manifold learning and contrastive learning methods, can be used to discover new physical laws and constraints as well as extract interpretable physical features—such as low-dimensional intrinsic coordinates and order parameters. Combined with efficient and expressive physics-informed architectures, these machine learning methods may soon become essential tools for scientists looking to understand and discover something new in their complex datasets.

5.4 Final Thoughts

Interpretable physics-informed machine learning methods should be designed to extract key factors of variation, intrinsic coordinates, and physically-meaningful representations from experimental data that not only help us make better predictions but also enhance our theoretical understanding of complex physical systems. To achieve this, machine learning algorithms and neural network architectures originally developed for other data science applications must be adapted to take advantage of known physical laws, symmetries, and constraints. These physics-informed architectures can then be combined with known physical models and incorporated into existing computational methods to further bridge the gap between powerful black box models like neural networks and well-understood theoretical physics.

As a physicist who sees machine learning as a serious tool for doing science, I am excited by the new insights that can be extracted from experimental data and theoretical models by carefully applying these new techniques. We are quickly learning that interpretable physics-informed machine learning is both a necessary development to make machine learning useful for science and an incredibly powerful tool that will undoubtedly lead to new discoveries. As a machine learning researcher, I also expect that the interpretable machine learning methods and design principles that our community is developing will have applications far beyond problems in physics.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Mark J. Ablowitz. *Nonlinear Dispersive Waves: Asymptotic Analysis and Solitons*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2011. doi: 10.1017/CBO9780511998324.

[2] Oreoluwa Alao*, Peter Y. Lu*, and Marin Soljačić. Discovering dynamical parameters by interpreting echo state networks. In *NeurIPS 2021 AI for Science Workshop*, 2021. URL `https://openreview.net/forum?id=coaSxusdBLX`.

[3] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a broken ELBO. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 159–168, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/alemi18a.html`.

[4] U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall/CRC Computational Biology Series. CRC Press, New York, 2019. ISBN 9781000001327.

[5] Stephen Anco, Maria Rosa, and Maria Luz Gandarias. Conservation laws and symmetries of time-dependent generalized kdv equations. *Discrete and Continuous Dynamical Systems - S*, 11(4):607–615, 2018.

[6] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[7] J. L. Aragón, R. A. Barrio, T. E. Woolley, R. E. Baker, and P. K. Maini. Nonlinear effects on turing patterns: Time oscillations and chaos. *Phys. Rev. E*, 86:026201, Aug 2012. doi: 10.1103/PhysRevE.86.026201.

[8] Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, and Patrick 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)*, pages 3232–3236, 2020. doi: 10.1109/ICASSP40776.2020.9053035.

[9] R.A. Barrio, C. Varea, J.L. Aragón, and P.K. Maini. A two-dimensional numerical study of spatial pattern formation in interacting turing systems. *Bulletin of Mathematical Biology*, 61(3):483–505, 1999. ISSN 0092-8240. doi: https://doi.org/10.1006/bulm.1998.0093.

[10] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.

[11] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. doi: 10.1162/089976603321780317.

[12] Jens Berg and Kaj Nyström. Data-driven discovery of PDEs in complex datasets. *Journal of Computational Physics*, 384:239–252, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.01.036.

[13] Adam Block, Zeyu Jia, Yury Polyanskiy, and Alexander Rakhlin. Intrinsic dimension estimation using wasserstein distances, 2021.

[14] Robert W. Boyd. *Nonlinear Optics*. Academic Press, Burlington, 3 edition, 2008. ISBN 978-0-12-369470-6.

[15] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[16] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

[17] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1517384113.

[18] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(1):19, 2017.

[19] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern koopman theory for dynamical systems, 2021.

[20] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE, 2018.

[21] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017. doi: 10.1126/science.aag2302.

[22] 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*, 116(45):22445–22451, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1906995116.

[23] Boyuan Chen, Kuang Huang, Sunand Raghupathi, Ishaan Chandratreya, Qiang Du, and Hod Lipson. Discovering state variables hidden in experimental data, 2021.

[24] Ricky T. Q. 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*, volume 31, pages 6572–6583, Red Hook, NY, 2018. Curran Associates, Inc.

[25] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2610–2620. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7527-isolating-sources-of-disentanglement-in-variational-autoencoders.pdf`.

[26] 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. URL `http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf`.

[27] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607. PMLR, 13–18 Jul 2020.

[28] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature Communications*, 12(1):6136, Oct 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-26434-1.

[29] Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/cohenc16.html`.

[30] Taco S. Cohen and Max Welling. Steerable CNNs. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rJQKYt5ll`.

[31] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant CNNs on homogeneous spaces. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[32] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426–7431, May 2005. doi: 10.1073/pnas.0500334102.

[33] Hugh N. Comins and David W.E. Blatt. Prey-predator models in spatially heterogeneous environments. *Journal of Theoretical Biology*, 48(1):75–83, 1974. ISSN 0022-5193. doi: https://doi.org/10.1016/0022-5193(74)90180-5.

[34] Allan Costa, Rumen Dangovski, Owen Dugan, Samuel Kim, Pawan Goyal, Marin Soljačić, and Joseph Jacobson. Fast neural models for symbolic regression at scale. July 2020.

[35] Allan Costa, Rumen Dangovski, Owen Dugan, Samuel Kim, Pawan Goyal, Marin Soljačić, and Joseph Jacobson. Fast neural models for symbolic regression at scale, 2020.

[36] Marco Cristoforetti, Giuseppe Jurman, Andrea I. Nardelli, and Cesare Furlanello. Towards meaningful physics from generative models, 2017.

[37] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf`.

[38] Marco Cuturi, Laetitia Meng-Papaxanthos, Yingtao Tian, Charlotte Bunne, Geoff Davis, and Olivier Teboul. Optimal transport tools (ott): A jax toolbox for all things wasserstein, 2022.

[39] Daniel M. Dubois. A model of patchiness for prey—predator plankton populations. *Ecological Modelling*, 1(1):67–80, 1975. ISSN 0304-3800. doi: https://doi.org/10.1016/0304-3800(75)90006-X.

[40] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In Kamalika Chaudhuri and Masashi

Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2681–2690. PMLR, 16–18 Apr 2019. URL `https://proceedings.mlr.press/v89/feydy19a.html`.

[41] Carl Folkestad, Daniel Pastor, Igor Mezic, Ryan Mohr, Maria Fonoberova, and Joel Burdick. Extended dynamic mode decomposition with learned koopman eigenfunctions for prediction and control. In *2020 American Control Conference (ACC)*, pages 3906–3913, 2020. doi: 10.23919/ACC45564.2020.9147729.

[42] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2019.

[43] Clifford S. Gardner, John M. Greene, Martin D. Kruskal, and Robert M. Miura. Method for solving the korteweg–de vries equation. *Phys. Rev. Lett.*, 19:1095–1097, Nov 1967. doi: 10.1103/PhysRevLett.19.1095.

[44] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000. doi: 10.1162/089976600300015015.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[46] J.L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6011–6045, 2006. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2005.10.010.

[47] Seungwoong Ha and Hawoong Jeong. Discovering conservation laws from trajectories via machine learning, 2021.

[48] Ernst Hairer, Gerhard Wanner, and Christian Lubich. *Symplectic Integration of Hamiltonian Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-30666-5. doi: 10.1007/3-540-30666-8_6.

[49] Jiequn Han, Chao Ma, Zheng Ma, and Weinan E. Uniformly accurate machine learning-based hydrodynamic models for kinetic equations. *Proceedings of the National Academy of Sciences*, 116(44):21983–21991, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1909854116.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[51] Jan Hermann, Zeno Schätzle, and Frank Noé. Deep-neural-network solution of the electronic schrödinger equation. *Nature Chemistry*, 12(10):891–897, Oct 2020. ISSN 1755-4349. doi: 10.1038/s41557-020-0544-y.

[52] Carlos X. Hernández, Hannah K. Wayment-Steele, Mohammad M. Sultan, Brooke E. Husic, and Vijay S. Pande. Variational encoding of complex dynamics. *Phys. Rev. E*, 97:062412, Jun 2018. doi: 10.1103/PhysRevE.97.062412.

[53] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy2fzU9gl`.

[54] Seth M. Hirsh, David A. Barajas-Solano, and J. Nathan Kutz. Sparsifying priors for bayesian uncertainty quantification in model discovery. *Royal Society Open Science*, 9(2):211823, 2022. doi: 10.1098/rsos.211823.

[55] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[56] Matthew D. Hoffman and Matthew J. Johnson. ELBO surgery: yet another way to carve up the variational evidence lower bound. In *Neural Information Processing Systems Workshop on Advances in Approximate Bayesian Inference*, 2016.

[57] Michael Innes. Don't unroll adjoint: Differentiating ssa-form programs, 2018.

[58] Raban Iten, Tony Metger, Henrik Wilming, Lídia del Rio, and Renato Renner. Discovering physical concepts with neural networks. *Phys. Rev. Lett.*, 124: 010508, Jan 2020. doi: 10.1103/PhysRevLett.124.010508.

[59] Hicham Janati, Marco Cuturi, and Alexandre Gramfort. Debiased Sinkhorn barycenters. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4692–4701. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/janati20a.html`.

[60] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic Filter Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 667–675. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6578-dynamic-filter-networks.pdf`.

[61] Jiaqi Jiang and Jonathan A. Fan. Global optimization of dielectric metasurfaces using a physics-driven neural network. *Nano Letters*, 19(8):5366–5372, 2019. doi: 10.1021/acs.nanolett.9b01857.

[62] Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In Doina Precup and Yee Whye Teh,

editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1733–1741, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/jing17a.html`.

[63] Steven G. Johnson and J. D. Joannopoulos. Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis. *Opt. Express*, 8(3): 173–190, Jan 2001. doi: 10.1364/OE.8.000173.

[64] Kadierdan Kaheman, J. Nathan Kutz, and Steven 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(2242):20200279, 2020. doi: 10.1098/rspa.2020.0279.

[65] Kadierdan Kaheman, Steven L Brunton, and J Nathan Kutz. Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data. *Machine Learning: Science and Technology*, 3(1):015031, mar 2022. doi: 10.1088/2632-2153/ac567a.

[66] Eurika Kaiser, J. Nathan Kutz, and Steven L. Brunton. Discovering conservation laws from data for control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6415–6421, 2018. doi: 10.1109/CDC.2018.8618963.

[67] Gurtej Kanwar, Michael S. Albergo, Denis Boyda, Kyle Cranmer, Daniel C. Hackett, Sébastien Racanière, Danilo Jimenez Rezende, and Phiala E. Shanahan. Equivariant flow-based sampling for lattice gauge theory. *Phys. Rev. Lett.*, 125:121601, Sep 2020. doi: 10.1103/PhysRevLett.125.121601.

[68] Noureddine El Karoui and Hau-Tieng Wu. Graph connection Laplacian methods can be made robust to noise. *The Annals of Statistics*, 44(1):346 – 372, 2016. doi: 10.1214/14-AOS1275.

[69] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, October 2017.

[70] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling, 2017.

[71] Aly-Khan Kassam and Lloyd N Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

[72] Samuel Kim, Peter Y. Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Scalable and flexible deep bayesian optimization with auxiliary information for scientific problems. April 2021.

[73] Samuel Kim, Peter Y. Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(9):4166–4177, 2021. doi: 10.1109/TNNLS.2020.3017010.

[74] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[75] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2013.

[76] 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(3):277–285, 1994. ISSN 0304-3800. doi: https://doi.org/10.1016/0304-3800(94)90123-6.

[77] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–Gordan nets: a fully fourier space spherical convolutional neural network. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[79] Yoshiki Kuramoto. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 1978.

[80] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[81] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016. doi: 10.1017/jfm.2016.615.

[82] Ziming Liu and Max Tegmark. Machine learning conservation laws from trajectories. *Phys. Rev. Lett.*, 126:180604, May 2021. doi: 10.1103/PhysRevLett.126.180604.

[83] Ziming Liu, Varun Madhavan, and Max Tegmark. AI Poincaré 2.0: Machine learning conservation laws from differential equations, 2022.

[84] Lennart Ljung. *System Identification: Theory for the User*. Pearson, Upper Saddle River, NJ, 2nd edition, 1999. ISBN 9780136566953.

[85] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4114–4124, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/locatello19a.html`.

[86] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from data. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3208–3216, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/long18a.html`.

[87] Zichao Long, Yiping Lu, and Bin Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.108925.

[88] Chien-Hung Lu, Christopher Barsi, Matthew O. Williams, J. Nathan Kutz, and Jason W. Fleischer. Phase retrieval using nonlinear diversity. *Appl. Opt.*, 52 (10):D92–D96, Apr 2013. doi: 10.1364/AO.52.000D92.

[89] Peter Y. Lu, Samuel Kim, and Marin Soljačić. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *Phys. Rev. X*, 10:031056, Sep 2020. doi: 10.1103/PhysRevX.10.031056.

[90] Peter Y. Lu, Joan Ariño, and Marin Soljačić. Discovering sparse interpretable dynamics from partial observations. To appear in *Commun. Phys.*, July 2021.

[91] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9 (1):4950, 2018. doi: 10.1038/s41467-018-07210-0.

[92] Eran Lustig, Or Yair, Ronen Talmon, and Mordechai Segev. Identifying topological phase transitions in experiments using manifold learning. *Phys. Rev. Lett.*, 125:127401, Sep 2020. doi: 10.1103/PhysRevLett.125.127401.

[93] Chao Ma, Jianchun Wang, and Weinan E. Model reduction with memory and the machine learning of dynamical systems. *Communications in Computational Physics*, 25(4):947–962, 2018. ISSN 1991-7120. doi: https://doi.org/10.4208/cicp.OA-2018-0269.

[94] Paul Manneville. Liapounov exponents for the Kuramoto-Sivashinsky model. In *Macroscopic Modelling of Turbulent Flows*, pages 319–326. Springer, 1985.

[95] Alexandre Mauroy, Yoshihiko Susuki, and Igor Mezić. *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*. Springer International Publishing, Cham, 2020. ISBN 978-3-030-35713-9. doi: 10.1007/978-3-030-35713-9_1.

[96] Alfredo Medio and Marji Lines. *Nonlinear Dynamics: A Primer*. Cambridge University Press, 2001. doi: 10.1017/CBO9780511754050.

[97] Robert M. Miura, Clifford S. Gardner, and Martin D. Kruskal. Korteweg-de vries equation and generalizations. ii. existence of conservation laws and constants of motion. *Journal of Mathematical Physics*, 9(8):1204–1209, 1968. doi: 10.1063/1.1664701.

[98] Kevin R. Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B. Burkhardt, William S. Chen, Kristina Yim, Antonia van den Elzen, Matthew J. Hirn, Ronald R. Coifman, Natalia B. Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37(12):1482–1492, Dec 2019. ISSN 1546-1696. doi: 10.1038/s41587-019-0336-3.

[99] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019. doi: 10.1126/science.aaw1147.

[100] Ardavan F. Oskooi, David Roundy, Mihai Ibanescu, Peter Bermel, J.D. Joannopoulos, and Steven G. Johnson. Meep: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications*, 181(3):687 – 702, 2010. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2009.11.008.

[101] 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(10):103121, 2020. doi: 10.1063/5.0019309.

[102] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[103] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A

reservoir computing approach. *Phys. Rev. Lett.*, 120:024102, Jan 2018. doi: 10.1103/PhysRevLett.120.024102.

[104] David Pfau and Christopher P. Burgess. Minimally redundant laplacian eigenmaps, 2018. URL `https://openreview.net/forum?id=rkmf_v1vf`.

[105] David Pfau, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Phys. Rev. Research*, 2:033429, Sep 2020. doi: 10.1103/PhysRevResearch.2.033429.

[106] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning, 2020.

[107] 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. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045.

[108] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19(25):1–24, 2018. URL `http://jmlr.org/papers/v19/18-046.html`.

[109] H. Ribera, S. Shirman, A. V. Nguyen, and N. M. Mangan. Model selection of chaotic systems from data with hidden variables using sparse data assimilation, 2021.

[110] Maria L. Rizzo and Gábor J. Székely. Energy distance. *WIREs Computational Statistics*, 8(1):27–38, 2016. doi: https://doi.org/10.1002/wics.1375.

[111] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference, 2018.

[112] S. Rudy, A. Alla, S. Brunton, and J. Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019. doi: 10.1137/18M1191944.

[113] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4): e1602614, April 2017.

[114] Samuel H. Rudy, J. Nathan Kutz, and Steven L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.06.056.

[115] Priyabrata Saha, Saurabh Dash, and Saibal Mukhopadhyay. Physics-incorporated convolutional recurrent neural networks for source identification and forecasting of dynamical systems. *Neural Netw.*, 144(C):359–371, dec 2021. ISSN 0893-6080. doi: 10.1016/j.neunet.2021.08.033.

[116] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. doi: 10.1017/S0022112010001217.

[117] Samuel Schoenholz and Ekin Dogus Cubuk. JAX MD: A framework for differentiable physics. In *Advances in Neural Information Processing Systems*, volume 33, pages 11428–11441. Curran Associates, Inc., 2020.

[118] GI Sivashinsky. On flame propagation under conditions of stoichiometry. *SIAM Journal on Applied Mathematics*, 39(1):67–82, 1980.

[119] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditional-generative-models.pdf`.

[120] George Sugihara, Robert May, Hao Ye, Chih hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch. Detecting causality in complex ecosystems. *Science*, 338(6106):496–500, 2012. doi: 10.1126/science.1227079.

[121] Allen Taflove and Susan C Hagness. *Computational electrodynamics: the finite-difference time-domain method.* Artech House, 2005.

[122] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1130–1140, Red Hook, NY, 2017. Curran Associates, Inc.

[123] Floris Takens. Detecting strange attractors in turbulence. In David Rand and Lai-Sang Young, editors, *Dynamical Systems and Turbulence, Warwick 1980*, pages 366–381, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg. ISBN 978-3-540-38945-3.

[124] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. February 2018.

[125] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16), 2020. doi: 10.1126/sciadv.aay2631.

[126] Cédric Villani. *Optimal Transport: Old and New.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9_6.

[127] Ce Wang, Hui Zhai, and Yi-Zhuang You. Uncover the black box of machine learning applied to quantum problem by an introspective learning architecture, 2019.

[128] Hao Wang and Dit-Yan Yeung. A survey on Bayesian deep learning. *ACM Comput. Surv.*, 53(5), sep 2020. ISSN 0360-0300. doi: 10.1145/3409383.

[129] Yaotian Wang, Xiaohang Sun, and Jason Fleischer. When deep denoising meets iterative phase retrieval. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 10007–10017. PMLR, 13–18 Jul 2020.

[130] Christoph Wehmeyer and Frank Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, 148(24):241703, June 2018.

[131] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3D Steerable CNNs: Learning rotationally equivariant features in volumetric data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10381–10392. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/8239-3d-steerable-cnns-learning-rotationally-equivariant-features-in-volumetric-data.pdf`.

[132] Sebastian J. Wetzel. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Phys. Rev. E*, 96:022140, Aug 2017. doi: 10.1103/PhysRevE.96.022140.

[133] Sebastian J. Wetzel, Roger G. Melko, Joseph Scott, Maysum Panju, and Vijay Ganesh. Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks. *Phys. Rev. Research*, 2:033499, Sep 2020. doi: 10.1103/PhysRevResearch.2.033499.

[134] M. Williams, I. Kevrekidis, and C. Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015.

[135] Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Phys. Rev. Fluids*, 3:074602, Jul 2018. doi: 10.1103/PhysRevFluids.3.074602.

[136] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*,

volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/xuc15.html`.

[137] Tianfan Xue, Jiajun Wu, Katherine Bouman, and William Freeman. Visual Dynamics: Stochastic Future Generation via Layered Cross Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. doi: 10.1109/TPAMI.2018.2854726.

[138] X. I. A. Yang, S. Zafar, J.-X. Wang, and H. Xiao. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Phys. Rev. Fluids*, 4:034602, Mar 2019. doi: 10.1103/PhysRevFluids.4.034602.

[139] Yunan Yang, Levon Nurbekyan, Elisa Negrini, Robert Martin, and Mirjeta Pasha. Optimal transport for parameter identification of chaotic dynamics via invariant measures, 2021.

[140] Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel de Bezenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=kmG8vRXTFv`.

[141] N. Zelesko, A. Moscovich, J. Kileel, and A. Singer. Earthmover-based manifold learning for analyzing molecular conformation spaces. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 1715–1719, 2020. doi: 10.1109/ISBI45749.2020.9098723.

[142] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models, 2017.

[143] Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Balancing learning and inference in variational autoencoders. In *AAAI Conference on Artificial Intelligence*, 2019.

[144] David Zheng, Vinson Luo, Jiajun Wu, and Joshua B. Tenenbaum. Unsupervised Learning of Latent Physical Properties Using Perception-Prediction Networks. In *Conference on Uncertainty in Artificial Intelligence*, 2018. URL `http://auai.org/uai2018/proceedings/papers/191.pdf`.

[145] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18795–18806, Red Hook, NY, 2020. Curran Associates, Inc.