Data-driven discovery of coordinates and governing equations

Kathleen Champion[1*], Bethany Lusch[2], J. Nathan Kutz[1], Steven L. Brunton[3]

[1] Department of Applied Mathematics, University of Washington, Seattle, WA 98195, United States

[2] Argonne National Laboratory, Lemont, IL 60439, United States

[3] Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, United States

Abstract

The discovery of governing equations from scientific data has the potential to transform data-rich fields that lack well-characterized quantitative descriptions. Advances in sparse regression are currently enabling the tractable identification of both the structure and parameters of a nonlinear dynamical system from data. The resulting models have the fewest terms necessary to describe the dynamics, balancing model complexity with descriptive ability, and thus promoting interpretability and generalizability. This provides an algorithmic approach to Occam's razor for model discovery. However, this approach fundamentally relies on an effective coordinate system in which the dynamics have a simple representation. In this work, we design a custom autoencoder to discover a coordinate transformation into a reduced space where the dynamics may be sparsely represented. Thus, we simultaneously learn the governing equations and the associated coordinate system. We demonstrate this approach on several example high-dimensional dynamical systems with low-dimensional behavior. The resulting modeling framework combines the strengths of deep neural networks for flexible representation and sparse identification of nonlinear dynamics (SINDy) for parsimonious models. It is the first method of its kind to place the discovery of coordinates and models on an equal footing.

Keywords– model discovery, dynamical systems, machine learning, deep learning, autoencoder

1 Introduction

Governing equations are of fundamental importance across all scientific disciplines. Accurate models allow for fundamental understanding of physical processes, which in turn gives rise to an infrastructure for the development of technology. The traditional derivation of governing equations is based on underlying first principles, such as conservation laws and symmetries, or from universal laws, such as gravitation. However, in many modern systems, governing equations are unknown or only partially known, and recourse to first principles derivations is untenable. On the other hand, many of these modern systems have rich time-series data due to the emergence of sensor and measurement technologies (e.g. in biology and climate science). This has given rise to the new paradigm of data-driven model discovery. Indeed, the data-driven discovery of dynamical systems is the focus of intense research efforts across the physical and engineering sciences [1–15]. A central tension in model discovery is the balance between model efficiency and descriptive capabilities. Parsimonious models strike this balance, having the fewest terms required to capture essential interactions, and not more [1, 3, 9, 11]. Related to Occam's razor, parsimonious models tend to be more interpretable and generalizable. However, obtaining parsimonious models is fundamentally linked to the coordinate system in which the dynamics are measured. Without knowledge of the proper coordinates, standard approaches may fail to discover simple dynamical models. In this work, we develop a machine learning approach that simultaneously discovers effective coordinates via a custom autoencoder [16–18], along with the parsimonious dynamical system model via sparse regression in a library of candidate terms [9]. The joint discovery of models and coordinates is critical for understanding many modern complex systems.

* Corresponding author (kpchamp@uw.edu).

Numerous recent approaches leverage machine learning to identify dynamical systems models from data [19–26], with many using neural networks to model time-series data [18, 27–38]. When interpretability and generalizability are primary concerns, it is important to identify *parsimonious* models that have the fewest terms required to describe the dynamics, which is the antithesis of neural networks whose parametrizations are exceedingly large. A breakthrough approach used symbolic regression to learn the form of dynamical systems and governing laws from data [1, 3, 39]. Sparse identification of nonlinear dynamics (SINDy) [9] is a related approach that uses sparse regression to find the fewest terms in a library of candidate terms required to model a dynamical system. Because this approach is based on a sparsity-promoting linear regression, it is possible to incorporate partial knowledge of the physics, such as symmetries, constraints, and conservation laws [40]. Successful model identification relies on the assumption that the dynamics are measured in a coordinate system in which the dynamics may be sparsely represented. While simple models may exist in one coordinate system, a different coordinate system may obscure these parsimonious representations. For modern applications of data-driven discovery, there is no reason to believe that our sensors are measuring the correct variables to admit a parsimonious representation of the dynamics. This motivates the systematic and automated discovery of coordinate transformations to facilitate this sparse representation, which is the subject of this work.

See AMATH 563 build a collection of functions that you think might be in the time derivative.

The challenge of discovering an effective coordinate system is as fundamental and important as model discovery. Many key historical scientific breakthroughs were enabled by the discovery of appropriate coordinate systems. Celestial mechanics, for instance, was revolutionized by the heliocentric coordinate system of Copernicus, Galileo, and Kepler, thus displacing Ptolemy's *doctrine of the perfect circle*, which was dogma for more than a millennium. Fourier introduced his famous transform to simplify the representation of the heat equation, resulting in a *sparse*, diagonal, decoupled linear system. Eigen-coordinates have been used more broadly to enable simple and sparse decompositions, for example in quantum mechanics and electrodynamics, to characterize energy levels in atoms and propagating modes in waveguides, respectively. Principal component analysis (PCA) is one of the most prolific modern coordinate discovery methods, representing high-dimensional data in a low-dimensional linear subspace [41, 42]. Nonlinear extensions of PCA have been enabled by a neural network architecture, called an autoencoder [16, 17, 43]. However, PCA coordinates and autoencoders generally do not take dynamics into account and, thus, may not provide the right basis for parsimonious dynamical models. In a related vein, Koopman analysis seeks to discover coordinates that linearize nonlinear dynamics [44–47]; while linear models are useful for prediction and control, they cannot capture the full behavior of many nonlinear systems. Thus, it is important to develop methods that combine simplifying coordinate transformations and nonlinear dynamical models. We advocate for a balance between these approaches, identifying coordinate transformations where only a few nonlinear terms are present, as in the classic theory of near-identity transformations and normal forms [12, 48].

In this work we present a method for discovery of nonlinear coordinate transformations that enable associated parsimonious dynamics. Our method combines a custom autoencoder network with a SINDy model for parsimonious nonlinear dynamics. The autoencoder architecture enables the discovery of reduced coordinates from high-dimensional input data that can be used to reconstruct the full system. The reduced coordinates are found along with nonlinear governing equations for the dynamics in a joint optimization. We demonstrate the ability of our method to discover parsimonious dynamics on three examples: a high-dimensional spatial data set with dynamics governed by the chaotic Lorenz system, a spiral wave resulting from the reaction-diffusion equation, and the nonlinear pendulum. These results demonstrate how to focus neural networks to discover interpretable dynamical models. Critically, the proposed method is the first to provide a mathematical framework that places the discovery of coordinates and models on equal footing.

Here they are using the autoencoder to reduce the dimensionality of the input to a hopefully more meaningful coordinate system, then using that reduced coordinate system with SiNDy and other techniques to rediscover the dynamics?

2 Background

2.1 Sparse identification of nonlinear dynamics

We review the *sparse identification of nonlinear dynamics* (SINDy) [9] algorithm, which is a mathematical regression technique for extracting parsimonious dynamics from time-series data. The method takes snapshot data $\mathbf{x}(t) \in \mathbb{R}^n$ and attempts to discover a best-fit dynamical system with as few terms as possible:

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

The snapshots represent measurements of the state of the system in time t, and the function $\mathbf{f}$ constrains how the dynamics of the system evolve in time. We seek a parsimonious model for the dynamics, resulting in a function $\mathbf{f}$ that contains only a few active terms: it is sparse in a basis of possible functions. This is consistent with our extensive knowledge of a diverse set of evolution equations used throughout the physical, engineering and biological sciences. Thus, the functions that comprise $\mathbf{f}$ are typically known from our extensive modeling experience.

SINDy frames model discovery as a sparse regression problem. Assuming snapshot derivatives are available, or can be calculated from data, the snapshots are stacked to form data matrices

Here, rather than just a single variable we have several variables x1, x2 that vary over time.

$$\mathbf{X} = \begin{pmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{pmatrix}, \quad \dot{\mathbf{X}} = \begin{pmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{pmatrix}.$$

with $\mathbf{X}, \dot{\mathbf{X}} \in \mathbb{R}^{m \times n}$. Although $\mathbf{f}$ is unknown, we can construct an extensive library of p potential candidate basis functions $\boldsymbol{\Theta}(\mathbf{X}) = [\boldsymbol{\theta}_1(\mathbf{X}) \cdots \boldsymbol{\theta}_p(\mathbf{X})] \in \mathbb{R}^{m \times p}$, where each $\boldsymbol{\theta}_j$ is a candidate basis function or model term. We assume $m \gg p$ so that the number of data snapshots is much larger than the number of candidate library functions; it may be necessary to sample transient dynamics and multiple initial conditions to improve the condition number of $\boldsymbol{\Theta}$. The choice of basis functions typically reflects some knowledge about the system of interest: a common choice is polynomials in $\mathbf{x}$ as these are elements of many canonical models of dynamical systems [48]. The library is used to formulate a regression problem to approximately solve the overdetermined linear system of equations

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}$$

Find coefficients for each term and each variable. Note that different variables might have different terms in their dynamics naturally which is why we have a matrix of coefficients E

where the unknown matrix $\boldsymbol{\Xi} = (\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \cdots\ \boldsymbol{\xi}_n) \in \mathbb{R}^{p \times n}$ is the set of coefficients that determine the active terms from $\boldsymbol{\Theta}(\mathbf{X})$ in the dynamics $\mathbf{f}$. Sparsity-promoting regression is used to find parsimonious models, ensuring that $\boldsymbol{\Xi}$, or more precisely each $\boldsymbol{\xi}_j$, is sparse and only a few columns of $\boldsymbol{\Theta}(\mathbf{X})$ are selected. For high-dimensional systems, the goal is to identify a low-dimensional state $\mathbf{z} = \varphi(\mathbf{x})$ with dynamics $\dot{\mathbf{z}} = \mathbf{g}(\mathbf{z})$, as in Sec. 3. The standard SINDy approach uses a sequentially thresholded least squares algorithm to find the coefficients [9], which is a proxy for ℓ_0 optimization [49] and has convergence guarantees [50]. Yao and Bollt [2] previously formulated the dynamical system identification problem as a similar linear inverse problem, although sparsity-promoting regression was not used, so the resulting models generally included all terms in $\boldsymbol{\Theta}$. In either case, an appealing aspect of this model discovery formulation is that it results in an overdetermined linear system of equations for which many regularized solution techniques exist. Thus, it provides a computationally efficient counterpart to other model discovery frameworks [3, 6].

SINDy has been widely applied to identify models for fluid flows [40, 51], optical systems [52], chemical reaction dynamics [53], convection in a plasma [54], structural modeling [55], and for

model predictive control [56]. There are also a number of theoretical extensions to the SINDy framework, including for identifying partial differential equations [11, 57], multiscale physics [58], parametrically dependent dynamical models [59], hybrid (switching) dynamical systems [60], and models with rational function nonlinearities [61]. It can also incorporate partially known physics and constraints [40] and identify models based on physically realistic sensor measurements [51]. The algorithm can also be reformulated to include integral terms for noisy data [62] or handle incomplete or limited data [63, 64]. The selected modes can also be evaluated using information criteria for model selection [65]. These diverse mathematical developments provide a mature framework for broadening the applicability of the model discovery method.

2.2 Neural networks for dynamical systems

The success of neural networks (NNs) on problems such as image classification [66] and speech recognition [67] has led to the use of NNs to perform a wide range of tasks in science and engineering. One recent area of focus has been the use of NNs for studying dynamical systems, which has a surprisingly rich history [27]. In addition to improving solution techniques for systems with known equations [34–38], deep learning has been used for understanding and predicting dynamics for complex systems with potentially unknown equations [18, 28–33]. Several recent methods have trained NNs to predict dynamics, including a time-lagged autoencoder which takes the state at time t as input data and uses an autoencoder-like structure to predict the state at time $t + \tau$ [30]. Other approaches use a recurrent NN architecture, particularly long short-term memory (LSTM) networks, for applications involving sequential data [68–70]. LSTMs have recently been used to perform forecasting on chaotic dynamical systems [29]. Reservoir computing has also enabled impressive predictions [14]. Autoencoders are increasingly being leveraged for dynamical systems because of their close relationship to other dimensionality reduction techniques [43, 71–73].

Another class of NNs use deep learning to discover coordinates for Koopman analysis. Koopman theory seeks to discover coordinates that linearize nonlinear dynamics [44–47]. Methods such as *dynamic mode decomposition* (DMD) [4, 5, 10, 74], extended DMD [75], kernel DMD [76], and time-delay DMD [77, 78] build linear models for dynamics, but these methods rely on a proper set of coordinates for linearization. Several recent works have focused on the use of deep learning methods to discover the proper coordinates for DMD and extended DMD [31, 32, 79]. Other methods seek to learn Koopman eigenfunctions and the associated linear dynamics directly using autoencoders [18, 33].

Despite their widespread use, NNs face three major challenges: generalization, extrapolation, and interpretation. The hallmark success stories of NNs (computer vision and speech, for instance) have been on data sets that are fundamentally interpolatory in nature. The ability to extrapolate, and as a consequence generalize, is known to be an underlying weakness of NNs. This is especially relevant for dynamical systems and forecasting, which is typically an extrapolatory problem by nature. Thus models trained on historical data will generally fail to predict future events that are not represented in the training set. An additional limitation of deep learning is the lack of interpretability of the resulting models. While attempts have been made to interpret NN weights, network architectures are typically complicated with the number of parameters (or weights) far exceeding the original dimension of the dynamical system. The lack of interpretability also makes it difficult to generalize models to new data sets and parameter regimes. However, NN methods still have the potential to learn general, interpretable dynamical models if properly constrained or regularized. In addition to methods for discovering linear embeddings [18, 33], deep learning has also been used for parameter estimation of PDEs [36, 37].

3 SINDy Autoencoders

We present a method for the simultaneous discovery of sparse dynamical models and coordinates that enable these simple representations. Our aim is to leverage the parsimony and interpretability of SINDy with the universal approximation capabilities of deep neural networks [80] in order to produce interpretable and generalizable models capable of extrapolation and forecasting. Our approach combines a SINDy model and a deep autoencoder network to perform a joint optimization that discovers intrinsic coordinates which have an associated parsimonious nonlinear dynamical model. The architecture is shown in Figure 1. We again consider dynamical systems of the form (1). While this dynamical model may be dense in terms of functions of the original measurement coordinates $\mathbf{x}$, our method seeks a set of reduced coordinates $\mathbf{z}(t) = \varphi(\mathbf{x}(t)) \in \mathbb{R}^d$ $(d \ll n)$ with an associated dynamical model

embed data into a lower dimensional space and learn the dynamics there

$$\frac{d}{dt}\mathbf{z}(t) = \mathbf{g}(\mathbf{z}(t)) \tag{2}$$

that provides a parsimonious description of the dynamics. This means that $\mathbf{g}$ contains only a few active terms. Along with the dynamical model, the method provides coordinate transforms φ, ψ that map the measurement coordinates to intrinsic coordinates via $\mathbf{z} = \varphi(\mathbf{x})$ (encoder) and back via $\mathbf{x} \approx \psi(\mathbf{z})$ (decoder). data —> low dimensional coordinate space.

The coordinate transformation is achieved using an autoencoder network architecture. The autoencoder is a feedforward neural network with a hidden layer that represents the intrinsic coordinates. Rather than performing a task such as prediction or classification, the network is trained to output an approximate reconstruction of its input, and the restrictions placed on the network architecture (e.g. the type, number, and size of the hidden layers) determine the properties of the intrinsic coordinates [17]; these networks are known to produce nonlinear generalizations of PCA [16]. A common choice is that the dimensionality of the intrinsic coordinates $\mathbf{z}$, determined by the number of units in the corresponding hidden layer, is much lower than that of the input data $\mathbf{x}$: in this case, the autoencoder learns a *nonlinear* embedding into a reduced latent space. Our network takes measurement data $\mathbf{x}(t) \in \mathbb{R}^n$ from a dynamical system as input and learns intrinsic coordinates $\mathbf{z}(t) \in \mathbb{R}^d$, where $d \ll n$ is chosen as a hyperparameter prior to training the network.

The low dimensional representation that the network learns might not be the best coordinate system to represent the dynamics of the system in.

While autoencoders can be trained in isolation to discover useful coordinate transformations and dimensionality reductions, there is no guarantee that the intrinsic coordinates learned will have associated sparse dynamical models. We require the network to learn coordinates associated with parsimonious dynamics by simultaneously learning a SINDy model for the dynamics of the intrinsic coordinates $\mathbf{z}$. This regularization is achieved by constructing a library $\boldsymbol{\Theta}(\mathbf{z}) = [\boldsymbol{\theta}_1(\mathbf{z}), \boldsymbol{\theta}_2(\mathbf{z}), \ldots, \boldsymbol{\theta}_p(\mathbf{z})]$ of candidate basis functions, e.g. polynomials, and learning a sparse set of coefficients $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d]$ that defines the dynamical system

Build a training procedure that penalizes solutions that do not generate sparse dynamics —> i.e. coordinate systems that are not good at representing the sparse dynamics of the system

$$\frac{d}{dt}\mathbf{z}(t) = \mathbf{g}(\mathbf{z}(t)) = \boldsymbol{\Theta}(\mathbf{z}(t))\boldsymbol{\Xi}. \tag{3}$$

While the functions in the library must be specified prior to training, the coefficients $\boldsymbol{\Xi}$ are learned along with the NN parameters as part of the training procedure. Assuming derivatives $\dot{\mathbf{x}}(t)$ of the original states are available or can be computed, one can calculate the derivative of the encoder variables as $\dot{\mathbf{z}}(t) = \nabla_{\mathbf{x}}\varphi(\mathbf{x}(t))\dot{\mathbf{x}}(t)$ and enforce accurate prediction of the dynamics by incorporating the following term into the loss function:

this alone does NOT generate sparse dynamics as there is no regularization on E. That comes later.

dot{ z } - predicted dot {z} note: z = phi(x)

$$\mathcal{L}_{d\mathbf{z}/dt} = \left\|\nabla_{\mathbf{x}}\varphi(\mathbf{x})\dot{\mathbf{x}} - \boldsymbol{\Theta}(\varphi(\mathbf{x})^T)\boldsymbol{\Xi}\right\|_2^2. \tag{4}$$

This term uses the SINDy model along with the gradient of the encoder to encourage the learned dynamical model to accurately predict the time derivatives of the encoder variables. We include

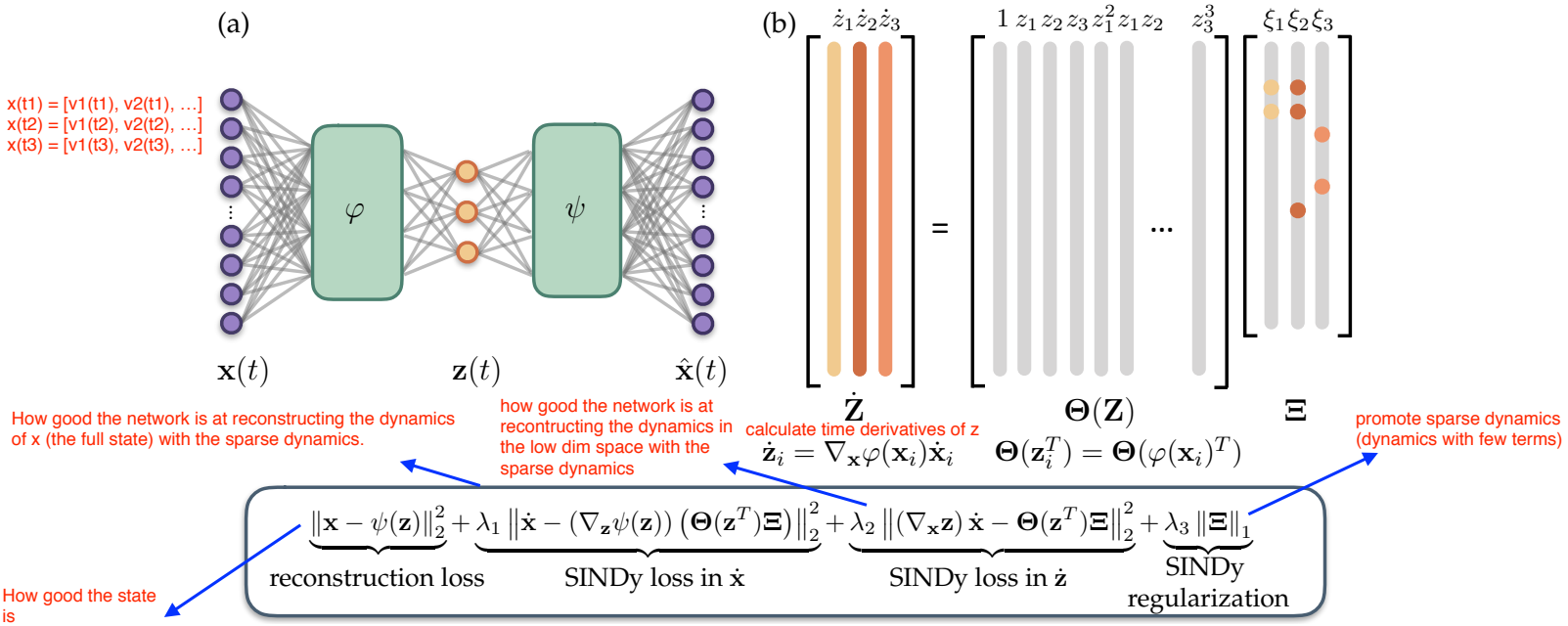

Figure 1: Schematic of the SINDy autoencoder method for simultaneous discovery of coordinates and parsimonious dynamics. (a) An autoencoder architecture is used to discover intrinsic coordinates $\mathbf{z}$ from high-dimensional input data $\mathbf{x}$. The network consists of two components: an encoder $\varphi(\mathbf{x})$, which maps the input data to the intrinsic coordinates $\mathbf{z}$, and a decoder $\psi(\mathbf{z})$, which reconstructs $\mathbf{x}$ from the intrinsic coordinates. (b) A SINDy model captures the dynamics of the intrinsic coordinates. The active terms in the dynamics are identified by the nonzero elements in $\boldsymbol{\Xi}$, which are learned as part of the NN training. The time derivatives of $\mathbf{z}$ are calculated using the derivatives of $\mathbf{x}$ and the gradient of the encoder φ. The inset shows the pointwise loss function used to train the network. The loss function encourages the network to minimize both the autoencoder reconstruction error and the SINDy loss in $\mathbf{z}$ and $\mathbf{x}$. L_1 regularization on $\boldsymbol{\Xi}$ is also included to encourage parsimonious dynamics. As a proxy for L0

an additional term in the loss function that ensures SINDy predictions can be used to reconstruct the time derivatives of the original data:

$$\mathcal{L}_{d\mathbf{x}/dt} = \left\|\dot{\mathbf{x}} - (\nabla_{\mathbf{z}}\psi(\varphi(\mathbf{x})))\left(\boldsymbol{\Theta}(\varphi(\mathbf{x})^T)\boldsymbol{\Xi}\right)\right\|_2^2. \tag{5}$$

We combine (4) and (5) with the standard autoencoder loss function

$$\mathcal{L}_{\text{recon}} = \|\mathbf{x} - \psi(\varphi(\mathbf{x}))\|_2^2, \tag{6}$$

which ensures that the autoencoder can accurately reconstruct the input data. We also include an L_1 regularization on the SINDy coefficients $\boldsymbol{\Xi}$, which promotes sparsity of the coefficients and therefore encourages a parsimonious model for the dynamics. The combination of the above four terms gives the following overall loss function:

$$\mathcal{L}_{\text{recon}} + \lambda_1\mathcal{L}_{d\mathbf{x}/dt} + \lambda_2\mathcal{L}_{d\mathbf{z}/dt} + \lambda_3\mathcal{L}_{\text{reg}}, \tag{7}$$

where the scalars $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters that determine the relative weighting of the three terms in the loss function.

In addition to the L_1 regularization, to obtain a model with only a few active terms we also incorporate sequential thresholding into the training procedure as a proxy for L_0 sparsity [49]. This

See explanation for this down below.

technique is inspired by the original algorithm used for SINDy [9], which combined least squares fitting with sequential thresholding to obtain a sparse dynamical model. To apply sequential thresholding during training, we specify a threshold that determines the minimum magnitude for coefficients in the SINDy model. At fixed intervals throughout the training, all coefficients below the threshold are set to zero and training resumes using only the terms left in the model. We train the network using the Adam optimizer [81]. In addition to the loss function weightings and SINDy coefficient threshold, training requires the choice of several other hyperparameters including learning rate, number of intrinsic coordinates d, network size, and activation functions. Full details of the training procedure are discussed in Section S1.4.

4 Results

We demonstrate the success of the proposed method on three example systems: a high-dimensional system with the underlying dynamics generated from the canonical chaotic Lorenz system, a two-dimensional reaction-diffusion system, and a two-dimensional spatial representation (synthetic video) of the nonlinear pendulum. Results are shown in Figure 2, and additional details for each example are provided in Section S2.

4.1 Chaotic Lorenz system

We first construct a high-dimensional example problem with dynamics based on the chaotic Lorenz system. The Lorenz system is a canonical model used as a test case for many dynamical systems methods, with dynamics given by the following equations

$$\dot{z}_1 = \sigma(z_2 - z_1) \tag{8a}$$
$$\dot{z}_2 = z_1(\rho - z_3) - z_2 \tag{8b}$$
$$\dot{z}_3 = z_1 z_2 - \beta z_3. \tag{8c}$$

The dynamics of the Lorenz system are chaotic and highly nonlinear, making it an ideal test problem for model discovery. To create a high-dimensional data set based on this system, we choose six fixed spatial modes $\mathbf{u}_1, \ldots, \mathbf{u}_6 \in \mathbb{R}^{128}$, given by Legendre polynomials, and define

$$\mathbf{x}(t) = \mathbf{u}_1 z_1(t) + \mathbf{u}_2 z_2(t) + \mathbf{u}_3 z_3(t) + \mathbf{u}_4 z_1(t)^3 + \mathbf{u}_5 z_2(t)^3 + \mathbf{u}_6 z_3(t)^3. \tag{9}$$

This results in a data set that is a nonlinear combination of the true Lorenz variables, shown in Figure 3a. The spatial and temporal modes that combine to give the full dynamics are shown in Figure 3b. Full details of how the data set is generated are given in Section S2.1.

Figure 3d shows the dynamical system discovered by the SINDy autoencoder. While the resulting model does not appear to match the original Lorenz dynamics, the discovered model is parsimonious, with only 7 active terms, and with dynamics that live on an attractor which has a two lobe structure similar to that of the original Lorenz attractor. Additionally, by choosing a suitable variable transformation the discovered model can be rewritten in the same form as the original Lorenz system. This demonstrates that the SINDy autoencoder is able to recover the correct sparsity pattern of the dynamics. The coefficients of the discovered model are close to the original parameters of the Lorenz system, up to an arbitrary scaling, which accounts for the difference in magnitude of the coefficients of $z_1 z_3$ in the second equation and $z_1 z_2$ in the third equation.

On a test data set of trajectories from 100 randomly chosen initial conditions, the mean square error of the decoder reconstruction is less than 3×10^{-5} of the fraction of the variance of the input

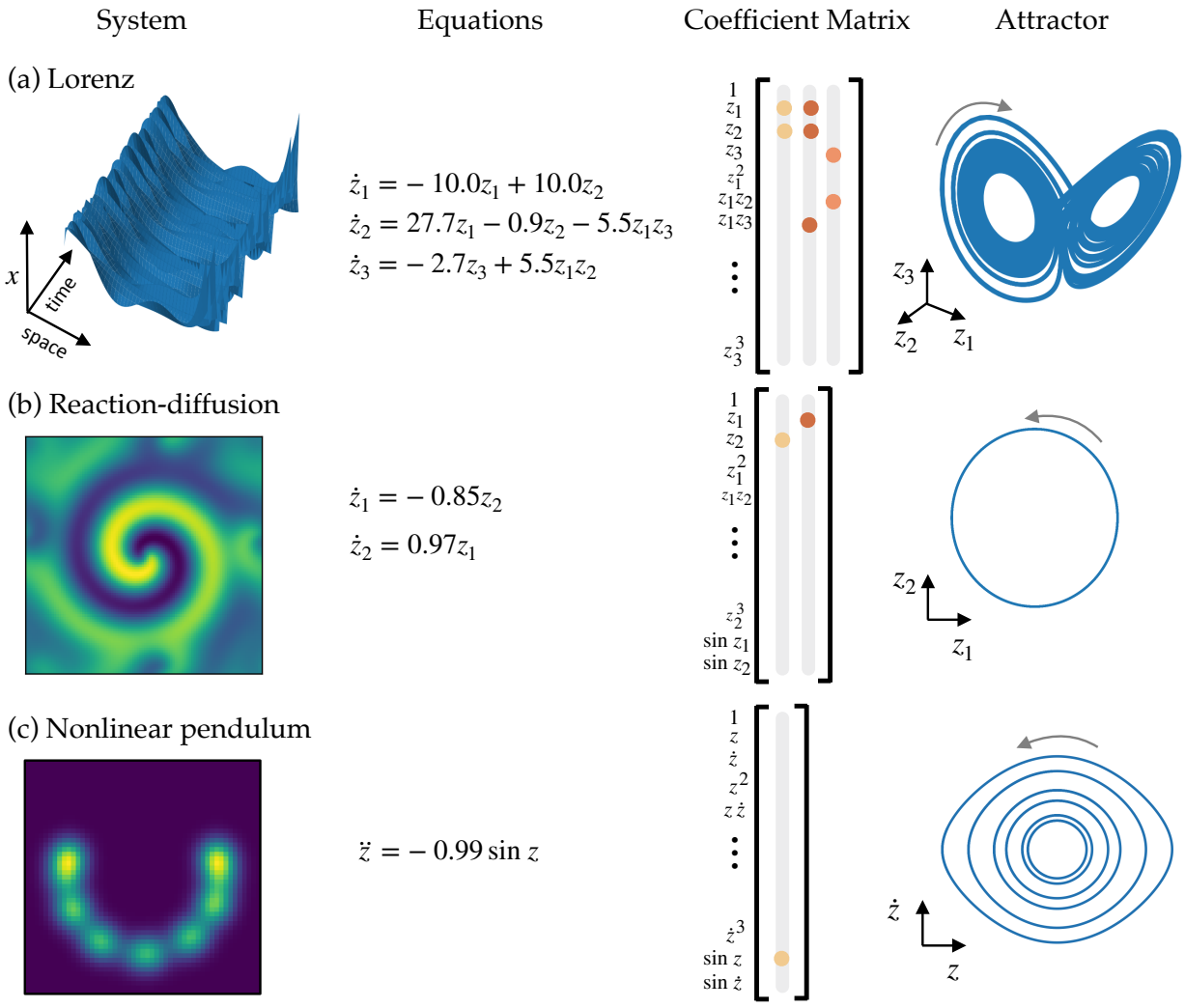

Figure 2: Discovered dynamical models for example systems. (a,b,c) Equations, SINDy coefficients $\boldsymbol{\Xi}$, and attractors for the Lorenz system, reaction-diffusion system, and nonlinear pendulum.

data. The fraction of the unexplained variance in predicting the derivatives $\dot{\mathbf{x}}$ and $\dot{\mathbf{z}}$ are 2×10^{-4} and 7×10^{-4}, respectively. Simulations of the resulting SINDy model are able to accurately reconstruct the dynamics of a single trajectory with less than 1% error over the duration of trajectories in the training data. Over longer durations, the trajectories start to diverge from the true trajectories. This result is not surprising due to the chaotic nature of the Lorenz system and its sensitivity to initial conditions. However, the dynamics of the discovered system match the sparsity pattern of the Lorenz system and the form of the attractor. Improved prediction over a longer duration could be achieved by increased parameter refinement or training the system with longer trajectories.

4.2 Reaction-diffusion

In practice, many high-dimensional data sets of interest come from dynamics governed by partial differential equations (PDEs) with more complicated interactions between spatial and temporal dynamics. To test the method on data generated by a PDE, we consider a lambda-omega reaction-

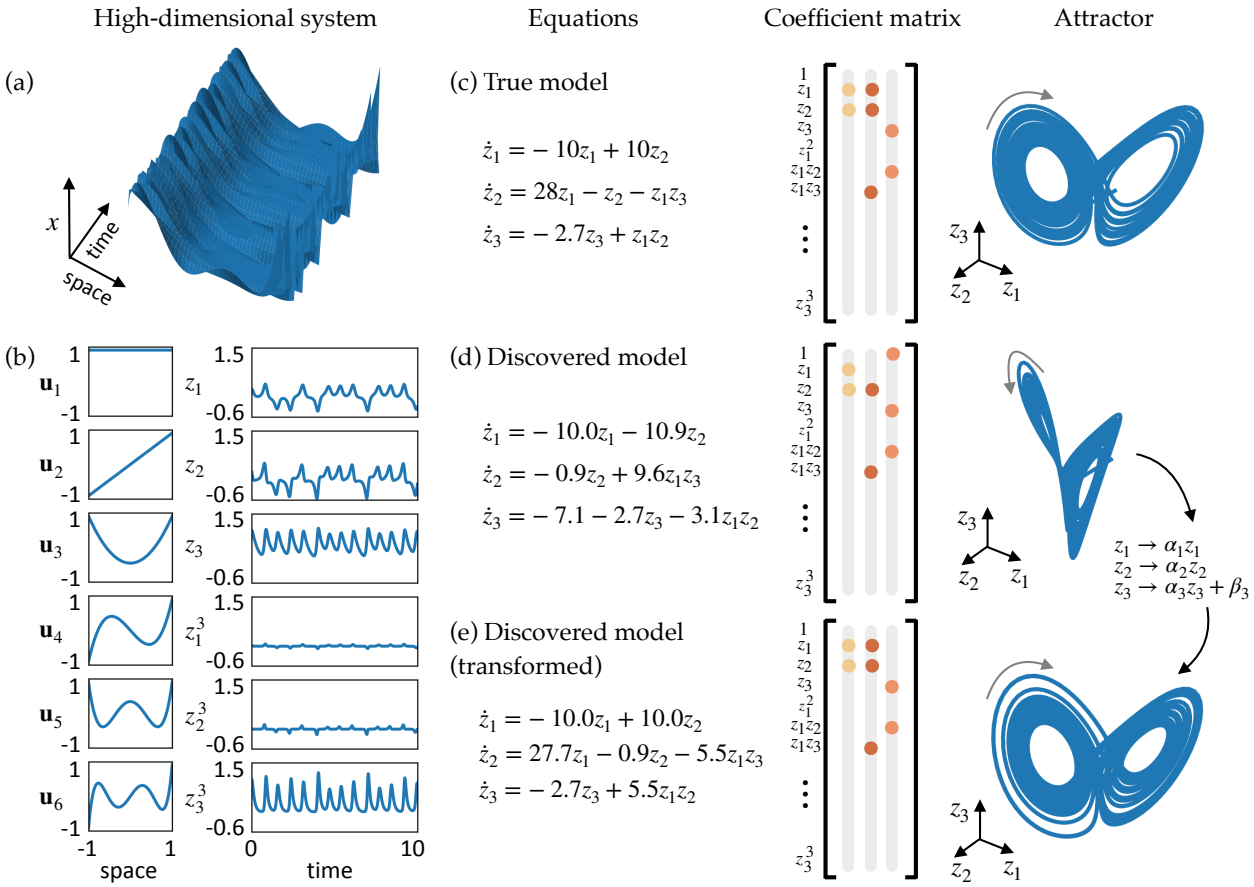

Figure 3: Model results on the high-dimensional Lorenz example. (a) Trajectories of the chaotic Lorenz system ($\mathbf{z}(t) \in \mathbb{R}^3$) are used to create a high-dimensional data set ($\mathbf{x}(t) \in \mathbb{R}^{128}$). (b) The spatial modes are created from the first six Legendre polynomials and the temporal modes are the variables in the Lorenz system and their cubes. The spatial and temporal modes are combined to create the high-dimensional data set via (9). (c,d) The equations, SINDy coefficients $\boldsymbol{\Xi}$, and attractors for the original Lorenz system and a dynamical system discovered by the SINDy autoencoder. The attractors are constructed by simulating the dynamical system forward in time from a single initial condition. (e) Applying a suitable variable transformation to the system in (d) reveals a model with the same sparsity pattern as the original Lorenz system. The parameters are close in value to the original system, with the exception of an arbitrary scaling, and the attractor has a similar structure to the original system.

diffusion system governed by

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d_1(u_{xx} + u_{yy}) \tag{10a}$$

$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d_2(v_{xx} + v_{yy}) \tag{10b}$$

with $d_1, d_2 = 0.1$ and $\beta = 1$. This set of equations generates a spiral wave formation, whose behavior can be approximately captured by two oscillating spatial modes. We apply our method to snapshots of $u(x, y, t)$ generated by the above equations. Snapshots are collected at discretized points of the xy-domain, resulting in a high-dimensional input data set with $n = 10^4$.

We train the SINDy autoencoder with $d = 2$. The resulting model is shown in Figure 2b. With two modes, the network discovers a model with nonlinear oscillatory dynamics. On test data,

the fraction of unexplained variance of both the input data $\mathbf{x}$ and the input derivatives $\dot{\mathbf{x}}$ is 0.016. The fraction of unexplained variance of $\dot{\mathbf{z}}$ is 0.002. Simulation of the dynamical model accurately captures the low dimensional dynamics, with the fraction of unexplained variance of $\mathbf{z}$ totaling 1×10^{-4}.

4.3 Nonlinear pendulum

As a final example, we consider simulated video of a nonlinear pendulum in two spatial dimensions. The nonlinear pendulum is governed by the following second order differential equation:

$$\ddot{z} = -\sin z. \tag{11}$$

We simulate the system from several initial conditions and generate a series of snapshot images with a two-dimensional Gaussian centered at the center of mass, determined by the pendulum's angle z. This series of images is the high-dimensional data input to the autoencoder. Despite the fact that the position of the pendulum can be represented by a simple one-dimensional variable, methods such as PCA are unable to obtain a low-dimensional representation of this data set. A nonlinear autoencoder, however, is able to discover a one-dimensional representation of the data set.

As PCA looks for the best LINEAR low dimensional representation

For this example, we use a second-order SINDy model: that is, we use a library of functions including the first derivatives $\dot{\mathbf{z}}$ to predict the second derivative $\ddot{\mathbf{z}}$. This approach is the same as with a first order SINDy model but requires estimates of the second derivatives in addition to estimates of the first derivatives. Second order gradients of the encoder and decoder are therefore also required. Computation of the derivatives is discussed in Section S1.3.

The SINDy autoencoder is trained with $d = 1$. Of the ten training instances, five correctly identify the nonlinear pendulum equation. We calculate test error on trajectories from 50 randomly chosen initial conditions. The best model has a fraction of unexplained variance of 8×10^{-4} for the decoder reconstruction of the input $\mathbf{x}$. The fraction of unexplained variance of the SINDy model predictions for $\ddot{\mathbf{x}}$ and $\ddot{\mathbf{z}}$ are 3×10^{-4} and 2×10^{-2}, respectively.

5 Discussion

We have presented a data-driven method for discovering interpretable, low-dimensional dynamical models and their associated coordinates for high-dimensional dynamical systems. The simultaneous discovery of both is critical for generating dynamical models that are sparse, and hence interpretable and generalizable. Our approach takes advantage of the power of NNs by using a flexible autoencoder architecture to discover nonlinear coordinate transformations that enable the discovery of parsimonious, nonlinear governing equations. This work addresses a major limitation of prior approaches for the discovery of governing equations, which is that the proper choice of measurement coordinates is often unknown. We demonstrate this method on three example systems, showing that it is able to identify coordinates associated with parsimonious dynamical equations. For the examples studied, the identified models are interpretable and can be used for forecasting (extrapolation) applications (see Figure S1).

A current limitation of our approach is the requirement for clean measurement data that is approximately noise-free. Fitting a continuous-time dynamical system with SINDy requires reasonable estimates of the derivatives, which may be difficult to obtain from noisy data. While this represents a challenge, approaches for estimating derivatives from noisy data such as the total variation regularized derivative can prove useful in providing derivative estimates [82].

Moreover, there are emerging NN architectures explicitly constructed for separating signals from noise [83], which can be used as a pre-processing step in the data-driven discovery process advocated here. Alternatively our method can be used to fit a discrete-time dynamical system, in which case derivative estimates are not required. Many methods for modeling dynamical systems work in discrete time rather than continuous time, making this a reasonable alternative. It is also possible to use the integral formulation of SINDy to abate noise sensitivity [62].

A major problem with deep learning approaches is that models are typically neither interpretable nor generalizable. Specifically, NNs trained solely for prediction may fail to generalize to classes of behaviors not seen in the training set. We have demonstrated an approach for using NNs to obtain classically interpretable models through the discovery of low-dimensional dynamical systems, which are well-studied and often have physical interpretations. Once the proper terms in the governing equations are identified, the discovered model can be generalized to study other parameter regimes of the dynamics. While the coordinate transformation learned by the autoencoder may not generalize to data regimes far from the original training set, if the dynamics are known, the network can be retrained on new data with fixed terms in the latent dynamics space. The problem of relearning a coordinate transformation for a system with known dynamics is greatly simplified from the original challenge of learning the correct form of the underlying dynamics without knowledge of the proper coordinate transformation.

The challenge of utilizing NNs to answer scientific questions requires careful consideration of their strengths and limitations. While advances in deep learning and computing power present a tremendous opportunity for new scientific breakthroughs, care must be taken to ensure that valid conclusions are drawn from the results. One promising strategy is to combine machine learning approaches with well-established domain knowledge: for instance physics-informed learning leverages physical assumptions into NN architectures and training methods. Methods that provide interpretable models have the potential to enable new discoveries in data-rich fields. This work introduced a flexible framework for using NNs to discover models that are interpretable from a standard dynamical systems perspective. In the future, this approach could be adapted using domain knowledge to discover new models in specific fields.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1256082. The authors also acknowledge support from the Defense Advanced Research Projects Agency (DARPA PA-18-01-FP-125) and the Army Research Office (ARO W911NF-17-1-0306 and W911NF-17-1-0422). This work was facilitated through the use of advanced computational, storage, and networking infrastructure provided by AWS cloud computing credits funded by the STF at the University of Washington. This research was funded in part by the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We would also like to thank Jean-Christophe Loiseau and Karthik Duraisamy for valuable discussions about sparse dynamical systems and autoencoders.

References

[1] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

[2] Chen Yao and Erik M Bollt. Modeling and nonlinear parameter estimation with Kronecker product

representation for coupled oscillators and spatiotemporal systems. *Physica D: Nonlinear Phenomena*, 227(1):78–99, 2007.

[3] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[4] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D.S. Henningson. Spectral analysis of nonlinear flows. *J. Fluid Mech.*, 645:115–127, 2009.

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

[6] W. X. Wang, R. Yang, Y. C. Lai, V. Kovanis, and C. Grebogi. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Physical Review Letters*, 106:154101–1–154101–4, 2011.

[7] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

[8] Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

[9] 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, April 2016.

[10] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

[11] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(e1602614), 2017.

[12] Or Yair, Ronen Talmon, Ronald R Coifman, and Ioannis G Kevrekidis. Reconstruction of normal forms by learning informed observation geometries from data. *Proceedings of the National Academy of Sciences*, page 201620045, 2017.

[13] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *To appear in Annual Reviews of Fluid Mechanics (arXiv preprint arXiv:1804.00183)*, 2018.

[14] 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. *Physical review letters*, 120(2):024102, 2018.

[15] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[16] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Netw.*, 2(1):53–58, 1989.

[17] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[18] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

[19] James P Crutchfield and Bruce S McNamara. Equations of motion from a data series. *Complex systems*, 1:417–452, 1987.

[20] Ioannis G Kevrekidis, C William Gear, James M Hyman, Panagiotis G Kevrekidid, Olof Runborg, Constantinos Theodoropoulos, and others. Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. *Communications in Mathematical Sciences*, 1(4):715–762, 2003.

[21] Michael D Schmidt, Ravishankar R Vallabhajosyula, Jerry W Jenkins, Jonathan E Hood, Abhishek S Soni, John P Wikswo, and Hod Lipson. Automated refinement and inference of analytical models for metabolic networks. *Physical biology*, 8(5):055011, 2011.

[22] Bryan C Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical

models. *Nature communications*, 6, 2015.

[23] Bryan C Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PloS one*, 10(3):e0119821, 2015.

[24] 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.

[25] Anthony John Roberts. *Model emergent dynamics in complex systems*. SIAM, 2014.

[26] Hao Ye, Richard J Beamish, Sarah M Glaser, Sue CH Grant, Chih-hao Hsieh, Laura J Richards, Jon T Schnute, and George Sugihara. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proceedings of the National Academy of Sciences*, 112(13):E1569–E1576, 2015.

[27] R Gonzalez-Garcia, R Rico-Martinez, and IG Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & chemical engineering*, 22:S965–S968, 1998.

[28] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. VAMPnets: Deep learning of molecular kinetics. *Nature Communications*, 9(5), 2018.

[29] Pantelis R. Vlachas, Wonmin Byeon, Zhong Y. Wan, Themistoklis P. Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long-short term memory networks. *arXiv preprint arXiv:1802.07486*, 2018.

[30] 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, 2018.

[31] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. *arXiv preprint arXiv:1708.06850*, 2017.

[32] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pages 1130–1140, 2017.

[33] Samuel E Otto and Clarence W Rowley. Linearly-recurrent autoencoder networks for learning dynamics. *arXiv preprint arXiv:1712.01378*, 2017.

[34] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *arXiv preprint arXiv:1708.00588*, 2017.

[35] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[36] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[37] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[38] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Data-driven discretization: a method for systematic coarse graining of partial differential equations. *arXiv preprint arXiv:1808.04930*, 2018.

[39] Theodore Cornforth and Hod Lipson. Symbolic regression of multiple-time-scale dynamical systems. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 735–742. ACM, 2012.

[40] J.-C. Loiseau and S. L. Brunton. Constrained sparse Galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.

[41] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(7–12):559–572, 1901.

[42] P. J. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge Monographs in Mechanics. Cambridge University Press, Cambridge, England, 2nd edition, 2012.

[43] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.

[44] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[45] Igor Mezic. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, August 2005.

[46] Marko Budisic, Ryan Mohr, and Igor Mezic. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.

[47] Igor Mezic. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Review of Fluid Mechanics*, 45(1):357–378, 2013.

[48] Philip Holmes and John Guckenheimer. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, volume 42 of *Applied Mathematical Sciences*. Springer-Verlag, Berlin, Heidelberg, 1983.

[49] Peng Zheng, Travis Askham, Steven L Brunton, J Nathan Kutz, and Aleksandr Y Aravkin. A unified framework for sparse relaxed regularized regression: Sr3. *IEEE Access*, 7:1404–1423, 2019.

[50] Linan Zhang and Hayden Schaeffer. On the convergence of the SINDy algorithm. *arXiv preprint arXiv:1805.06445*, 2018.

[51] J.-C. Loiseau, B. R. Noack, and S. L. Brunton. Sparse reduced-order modeling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018.

[52] Mariia Sorokina, Stylianos Sygletos, and Sergei Turitsyn. Sparse identification for nonlinear optical communication systems: SINO method. *Optics express*, 24(26):30433–30443, 2016.

[53] Moritz Hoffmann, Christoph Fröhner, and Frank Noé. Reactive SINDy: Discovering governing reactions from concentration data. *Journal of Chemical Physics*, 150(025101), 2019.

[54] Magnus Dam, Morten Brøns, Jens Juul Rasmussen, Volker Naulin, and Jan S Hesthaven. Sparse identification of a predator-prey system from simulation data of a convection model. *Physics of Plasmas*, 24(2):022310, 2017.

[55] Zhilu Lai and Satish Nagarajaiah. Sparse structural system identification method for nonlinear dynamic systems with hysteresis/inelastic behavior. *Mechanical Systems and Signal Processing*, 117:813–842, 2019.

[56] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society of London A*, 474(2219), 2018.

[57] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. In *Proc. R. Soc. A*, volume 473, page 20160446. The Royal Society, 2017.

[58] Kathleen P Champion, Steven L Brunton, and J Nathan Kutz. Discovery of nonlinear multiscale systems: Sampling strategies and embeddings. *SIAM Journal on Applied Dynamical Systems*, 18(1):312–333, 2019.

[59] Samuel Rudy, Alessandro Alla, Steven L Brunton, and J Nathan Kutz. Data-driven identification of parametric partial differential equations. *arXiv preprint arXiv:1806.00732*, 2018.

[60] Niall M Mangan, Travis Askham, Steven L Brunton, J Nathan Kutz, and Joshua L Proctor. Model selection for hybrid dynamical systems via sparse regression. *Proceedings of the Royal Society A*, 475(2223):20180534, 2019.

[61] Niall M Mangan, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, 2(1):52–63, 2016.

[62] Hayden Schaeffer and Scott G McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, 2017.

[63] Giang Tran and Rachel Ward. Exact recovery of chaotic systems from highly corrupted data. *arXiv preprint arXiv:1607.01067*, 2016.

[64] Hayden Schaeffer, Giang Tran, and Rachel Ward. Extracting sparse high-dimensional dynamics from limited data. *SIAM Journal on Applied Mathematics*, 78(6):3279–3295, 2018.

[65] Niall M Mangan, J Nathan Kutz, Steven L Brunton, and Joshua L Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A*, 473(2204):1–16, 2017.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[67] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

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

[69] Sepp Hochreiter and Jürgen Schmidhuber. LSTM can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[70] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[71] Kevin T Carlberg, Antony Jameson, Mykel J Kochenderfer, Jeremy Morton, Liqian Peng, and Freddie D Witherden. Recovering missing cfd data for high-order discretizations using deep neural networks and dynamics learning. *arXiv preprint arXiv:1812.01177*, 2018.

[72] Francisco J Gonzalez and Maciej Balajewicz. Learning low-dimensional feature dynamics using deep convolutional recurrent autoencoders. *arXiv preprint arXiv:1808.01346*, 2018.

[73] Kookjin Lee and Kevin Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *arXiv preprint arXiv:1812.08373*, 2018.

[74] PJ Schmid, L Li, MP Juniper, and O Pust. Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics*, 25(1-4):249–259, 2011.

[75] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, December 2015.

[76] Matthew O Williams, Clarence W Rowley, and Ioannis G Kevrekidis. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.

[77] 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), December 2017.

[78] Hassan Arbabi and Igor Mezić. Ergodic theory, dynamic mode decomposition and computation of spectral properties of the Koopman operator. *SIAM J. Appl. Dyn. Syst.*, 16(4):2096–2126, 2017.

[79] Qianxiao Li, Felix Dietrich, Erik M. Bollt, and Ioannis G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[80] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, 1989.

[81] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[82] Rick Chartrand. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics*, 2011, 2011.

[83] Samuel H Rudy, J Nathan Kutz, and Steven L Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *arXiv preprint arXiv:1808.02578*, 2018.

[84] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[85] Robert Tibshirani, Martin Wainwright, and Trevor Hastie. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

S1 Network Architecture and Training

S1.1 Network architecture

The autoencoder network consists of a series of fully-connected layers. Each layer has an associated weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$. We use sigmoid activation functions $f(x) = 1/(1 + \exp(-x))$, which are applied at all layers of the network, except for the last layer of the encoder and the last layer of the decoder. Other choices of activation function, such as rectified linear units and exponential linear units, may also be used and appear to achieve similar results.

S1.2 Loss function

The loss function used in training is a weighted sum of four terms: autoencoder reconstruction $\mathcal{L}_{\text{recon}}$, SINDy prediction on the input variables $\mathcal{L}_{d\mathbf{x}/dt}$, SINDy prediction on the encoder variables $\mathcal{L}_{d\mathbf{z}/dt}$, and SINDy coefficient regularization $\mathcal{L}_{\text{reg}}$. For a data set with m input samples, each loss is explicitly defined as follows:

$$\mathcal{L}_{\text{recon}} = \frac{1}{m}\sum_{i=1}^{m} \|\mathbf{x}_i - \psi(\varphi(\mathbf{x}_i))\|_2^2 \tag{12a}$$

$$\mathcal{L}_{d\mathbf{x}/dt} = \frac{1}{m}\sum_{i=1}^{m} \left\|\dot{\mathbf{x}}_i - (\nabla_{\mathbf{z}}\psi(\varphi(\mathbf{x}_i)))\left(\boldsymbol{\Theta}(\varphi(\mathbf{x}_i)^T)\boldsymbol{\Xi}\right)\right\|_2^2 \tag{12b}$$

$$\mathcal{L}_{d\mathbf{z}/dt} = \frac{1}{m}\sum_{i=1}^{m} \left\|\nabla_{\mathbf{x}}\varphi(\mathbf{x}_i)\dot{\mathbf{x}}_i - \boldsymbol{\Theta}(\varphi(\mathbf{x}_i)^T)\boldsymbol{\Xi}\right\|_2^2 \tag{12c}$$

$$\mathcal{L}_{\text{reg}} = \frac{1}{pd}\|\boldsymbol{\Xi}\|_1 . \tag{12d}$$

The total loss function is

$$\mathcal{L}_{\text{recon}} + \lambda_1\mathcal{L}_{d\mathbf{x}/dt} + \lambda_2\mathcal{L}_{d\mathbf{z}/dt} + \lambda_3\mathcal{L}_{\text{reg}}. \tag{13}$$

$\mathcal{L}_{\text{recon}}$ ensures that the autoencoder can accurately reconstruct the data from the intrinsic coordinates. $\mathcal{L}_{d\mathbf{x}/dt}$ and $\mathcal{L}_{d\mathbf{z}/dt}$ ensure that the discovered SINDy model captures the dynamics of the system by ensuring that the model can predict the derivatives from the data. $\mathcal{L}_{\text{reg}}$ promotes sparsity of the coefficients in the SINDy model.

S1.3 Computing derivatives

Computing the derivatives of the encoder variables requires propagating derivatives through the network. Our network makes use of an activation function $f(\cdot)$ that operates elementwise. Given an input $\mathbf{x}$, we define the pre-activation values at the jth encoder layer as

$$\mathbf{l}_j = f(\mathbf{l}_{j-1})\mathbf{W}_j + \mathbf{b}_j. \tag{14}$$

The first layer applies the weights and biases directly to the input so that

$$\mathbf{l}_0 = \mathbf{x}\mathbf{W}_0 + \mathbf{b}_0. \tag{15}$$

The activation function is not applied to the last layer, so for an encoder with L hidden layers the autoencoder variables are defined as

$$\mathbf{z} = f(\mathbf{l}_{L-1})\mathbf{W}_L + \mathbf{b}_L. \tag{16}$$

Assuming that derivatives $d\mathbf{x}/dt$ are available or can be computed, derivatives $d\mathbf{z}/dt$ can also be computed:

$$\frac{d\mathbf{z}}{dt} = \left(f'(\mathbf{l}_{L-1}) \circ \frac{d\mathbf{l}_{L-1}}{dt} \right) \mathbf{W}_L$$

with

$$\frac{d\mathbf{l}_j}{dt} = \left(f'(\mathbf{l}_{j-1}) \circ \frac{d\mathbf{l}_{j-1}}{dt} \right) \mathbf{W}_j$$
$$\frac{d\mathbf{l}_0}{dt} = \frac{d\mathbf{x}}{dt}\mathbf{W}_0.$$

For the nonlinear pendulum example, we use a second order SINDy model that requires the calculation of second derivatives. Second derivatives can be computed using the following:

$$\frac{d^2\mathbf{z}}{dt^2} = \left(f''(\mathbf{l}_{L-1}) \circ \frac{d\mathbf{l}_{L-1}}{dt} \circ \frac{d\mathbf{l}_{L-1}}{dt} + f'(\mathbf{l}_{L-1}) \circ \frac{d^2\mathbf{l}_{L-1}}{dt^2} \right) \mathbf{W}_L$$
$$\frac{d^2\mathbf{l}_j}{dt^2} = \left(f''(\mathbf{l}_{j-1}) \circ \frac{d\mathbf{l}_{j-1}}{dt} \circ \frac{d\mathbf{l}_{j-1}}{dt} + f'(\mathbf{l}_{j-1}) \circ \frac{d^2\mathbf{l}_{j-1}}{dt^2} \right) \mathbf{W}_j$$
$$\frac{d\mathbf{l}_0}{dt} = \frac{d^2\mathbf{x}}{dt^2}\mathbf{W}_0.$$

S1.4 Training procedure

We train multiple models for each of the example systems. Each instance of training has a different random initialization of the network weights. The weight matrices $\mathbf{W}_j$ are initialized using the Xavier initialization: the entries are chosen from a random uniform distribution over $[-\sqrt{6/\alpha}, \sqrt{6/\alpha}]$ where α is the dimension of the input plus the dimension of the output [84]. The bias vectors $\mathbf{b}_j$ are initialized to 0 and the SINDy model coefficients $\boldsymbol{\Xi}$ are initialized so that every entry is 1. We train each model using the Adam optimizer for a fixed number of epochs [81]. The learning rate and number of training epochs for each example are specified in Section S2.

To obtain parsimonious dynamical models, we use a sequential thresholding procedure that promotes sparsity on the coefficients in $\boldsymbol{\Xi}$, which represent the dynamics on the latent variables $\mathbf{z}$. Every 500 epochs, we set all coefficients in $\boldsymbol{\Xi}$ with a magnitude of less than 0.1 to 0, effectively removing these terms from the SINDy model. This is achieved by using a mask $\boldsymbol{\Upsilon}$, consisting of 1s and 0s, that determines which terms remain in the SINDy model. Thus the true SINDy terms in the loss function are given by

$$\lambda_1 \frac{1}{m} \sum_{i=1}^{m} \left\| \dot{\mathbf{x}}_i - (\nabla_{\mathbf{z}} \psi(\varphi(\mathbf{x}_i))) \left(\boldsymbol{\Theta}(\varphi(\mathbf{x}_i)^T)(\boldsymbol{\Upsilon} \circ \boldsymbol{\Xi}) \right) \right\|_2^2 + \lambda_2 \frac{1}{m} \sum_{i=1}^{m} \left\| \nabla_{\mathbf{x}} \varphi(\mathbf{x}_i) \dot{\mathbf{x}}_i - \boldsymbol{\Theta}(\varphi(\mathbf{x}_i)^T)(\boldsymbol{\Upsilon} \circ \boldsymbol{\Xi}) \right\|_2^2 \tag{17}$$

where $\boldsymbol{\Upsilon}$ is passed in separately and not updated by the optimization algorithm. Once a term has been thresholded out during training, it is permanently removed from the SINDy model. Therefore the number of active terms in the SINDy model can only be decreased as training continues. The L_1 regularization on $\boldsymbol{\Xi}$ encourages the model coefficients to decrease in magnitude, which combined with the sequential thresholding produces a parsimonious dynamical model.

While the L_1 regularization penalty on $\boldsymbol{\Xi}$ promotes sparsity in the resulting SINDy model, it also encourages nonzero terms to have smaller magnitudes. This results in a trade-off between

accurately reconstructing the dynamics of the system and reducing the magnitude of the SINDy coefficients, where the trade-off is determined by the relative magnitudes of the loss weight penalties λ_1, λ_2 and the regularization penalty λ_3. The specified training procedure therefore typically results in models with coefficients that are slightly smaller in magnitude than those which would best reproduce the dynamics. To account for this, we add an additional coefficient refinement period to the training procedure. To perform this refinement, we lock in the sparsity pattern in the dynamics by fixing the coefficient mask $\boldsymbol{\Upsilon}$ and continue training for 1000 epochs without the L_1 regularization on $\boldsymbol{\Xi}$. This ensures that the best coefficients are found for the resulting SINDy model and also allows the training procedure to refine the encoder and decoder parameters. This procedure is analagous to running a debiased regression following the use of LASSO to select model terms [85].

S1.5 Model selection

Random initialization of the NN weights is standard practice for deep learning approaches. This results in the discovery of different models for different instances of training, which necessitates comparison among multiple models. In this work, when considering the success of a resulting model, one must consider the parsimony of the SINDy model, how well the decoder reconstructs the input, and how well the SINDy model captures the dynamics.

To assess model performance, we calculate the fraction of unexplained variance of both the input data $\mathbf{x}$ and its derivative $\dot{\mathbf{x}}$. This error calculation takes into account both the decoder reconstruction and the fit of the dynamics. When considering parsimony, we consider the number of active terms in the resulting SINDy model. While parsimonious models are desirable for ease of analysis and interpretability, a model that is too parsimonious may be unable to fully capture the dynamics. In general, for the examples explored, we find that models with fewer active terms perform better on validation data (lower fraction of unexplained variance of $\dot{\mathbf{x}}$) whereas models with more active terms tend to over-fit the training data.

For each example system, we apply the training procedure to ten different initializations of the network and compare the resulting models. For the purpose of demonstration, for each example we show results for a chosen "best" model, which is taken to be the model with the lowest fraction of variance unexplained on validation data among models with the fewest active coefficients. While every instance of training does not result in the exact same SINDy sparsity pattern, the network tends to discover a few different closely related forms of the dynamics. We discuss the comparison among models for each particular example further in Section S2.

S2 Example Systems

S2.1 Chaotic Lorenz system

To create a high-dimensional data set with dynamics defined by the Lorenz system, we choose six spatial modes $\mathbf{u}_1, \ldots, \mathbf{u}_6 \in \mathbb{R}^{128}$ and take

$$\mathbf{x}(t) = \mathbf{u}_1 z_1(t) + \mathbf{u}_2 z_2(t) + \mathbf{u}_3 z_3(t) + \mathbf{u}_4 z_1(t)^3 + \mathbf{u}_5 z_2(t)^3 + \mathbf{u}_6 z_3(t)^3.$$

where the dynamics of $\mathbf{z}$ are specified by the Lorenz equations

$$\dot{z}_1 = \sigma(z_2 - z_1) \tag{18a}$$

$$\dot{z}_2 = z_1(\rho - z_3) - z_2 \tag{18b}$$

$$\dot{z}_3 = z_1 z_2 - \beta z_3 \tag{18c}$$

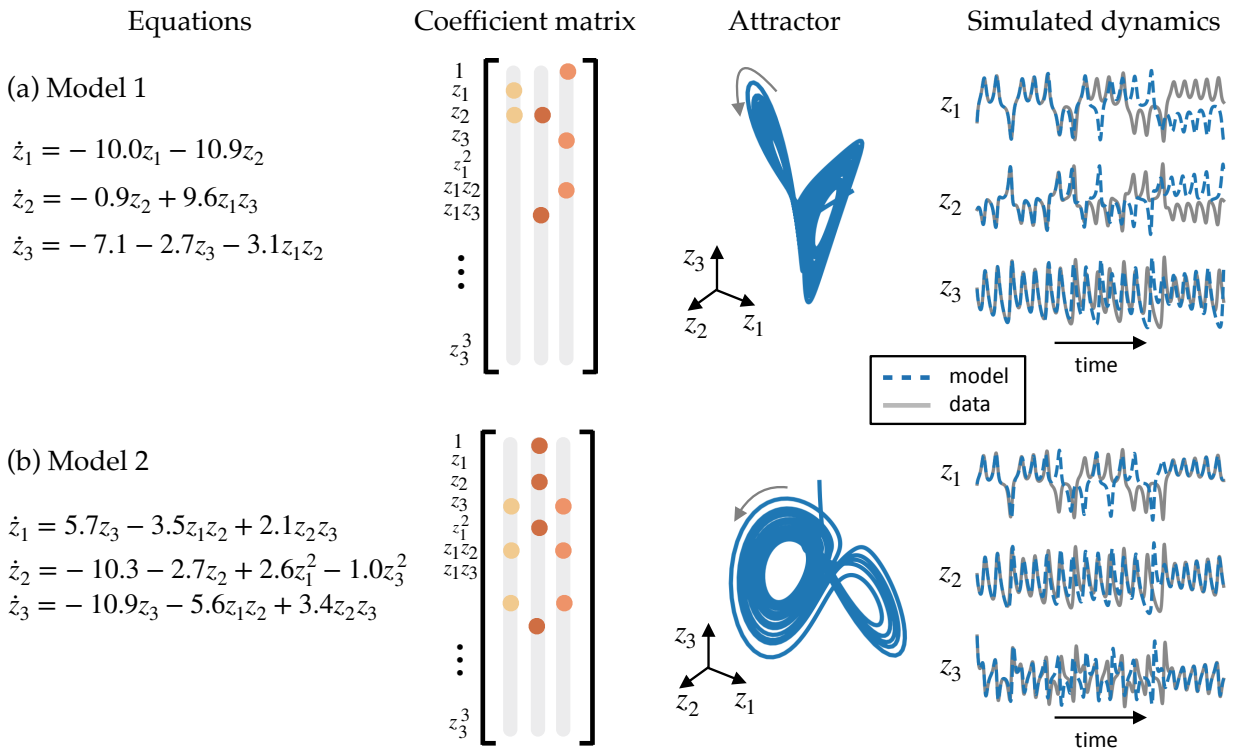

Figure S1: Comparison of two discovered models for the Lorenz example system. For both models we show the equations, SINDy coefficients $\boldsymbol{\Xi}$, attractors, and simulated dynamics for two models discovered by the SINDy autoencoder. (a) A model with 7 active terms. This model can be rewritten in the same form as the original Lorenz system using the variable transformation described in Section S2.1. Simulation of the model produces an attractor with a two lobe structure and is able to reproduce the true trajectories of the dynamics for some time before eventually diverging due to the chaotic nature of the system. (b) A model with 10 active terms. The model has more terms than the true Lorenz system, but has a slightly lower fraction of unexplained variance of $\mathbf{x}, \dot{\mathbf{x}}$ than the model in (a). Simulation shows that the dynamics also lie on an attractor with two lobes. The model can accurately predict the true dynamics over a similar duration as (a).

with standard parameter values of $\sigma = 10, \rho = 28, \beta = 8/3$. We choose our spatial modes $\mathbf{u}_1, \ldots, \mathbf{u}_6$ to be the first six Legendre polynomials defined at 128 grid points on a 1D spatial domain $[-1, 1]$. To generate our data set, we simulate the system from 2048 initial conditions for the training set, 20 for the validation set, and 100 for the test set. For each initial condition we integrate the system forward in time from $t = 0$ to $t = 5$ with a spacing of $\Delta t = 0.02$ to obtain 250 samples. Initial conditions are chosen randomly from a uniform distribution over $z_1 \in [-36, 36]$, $z_2 \in [-48, 48]$, $z_3 \in [-16, 66]$. This results in a training set with 512,000 total samples.

Following the training procedure described in Section S1.4, we learn ten models using the single set of training data (variability among the models comes from the initialization of the network weights). The hyperparameters used for training are shown in Table S1. For each model we run the training procedure for 10^4 epochs, followed by a refinement period of 10^3 epochs. Of the ten models, two have 7 active terms, two have 10 active terms, one has 11 active terms, and five have 15 or more active terms. While all models have less than 1% unexplained variance for both $\mathbf{x}$ and $\dot{\mathbf{x}}$, the three models with 20 or more active terms have the worst performance predicting $\dot{\mathbf{x}}$. The

Table S1: Hyperparameter values for the Lorenz example

Parameter	**Value**
n	128
d	3
training samples	5.12×10^5
batch size	8000
activation function	sigmoid
encoder layer widths	$64, 32$
decoder layer widths	$32, 64$
learning rate	10^{-3}
SINDy model order	1
SINDy library polynomial order	3
SINDy library includes sine	no
SINDy loss weight $\dot{\mathbf{x}}$, λ_1	10^{-4}
SINDy loss weight $\dot{\mathbf{z}}$, λ_2	0
SINDy regularization loss weight, λ_3	10^{-5}

two models with 10 active terms have the lowest overall error, followed by models with 7, 15, and 18 active terms. While the models with 10 active terms have a lower overall error than the models with 7 terms, both have a very low error and thus we choose to highlight the model with the fewest active terms. A model with 10 active terms is shown in Figure S1 for comparison.

For analysis, we highlight the model with the lowest error among the models with the fewest active terms. The discovered model has equations

$$\dot{z}_1 = -10.0z_1 - 10.9z_2 \tag{19a}$$
$$\dot{z}_2 = -0.9z_2 + 9.6z_1z_3 \tag{19b}$$
$$\dot{z}_3 = -7.1 - 2.7z_3 - 3.1z_1z_2. \tag{19c}$$

While the structure of this model appears to be different from that of the original Lorenz system, we can define an affine transformation that gives it the same structure. The variable transformation $z_1 = \alpha_1\tilde{z}_1$, $z_2 = \alpha_2\tilde{z}_2$, $z_3 = \alpha_3\tilde{z}_3 + \beta_3$ gives the following transformed system of equations:

$$\dot{\tilde{z}}_1 = \frac{1}{\alpha_1}\left(-10.0\alpha_1\tilde{z}_1 - 10.9\alpha_2\tilde{z}_2\right) \tag{20a}$$
$$= -10.0\tilde{z}_1 - 10.9\frac{\alpha_2}{\alpha_1}\tilde{z}_2 \tag{20b}$$
$$\dot{\tilde{z}}_2 = \frac{1}{\alpha_2}\left(-0.9\alpha_2\tilde{z}_2 + 9.6\alpha_1\tilde{z}_1(\alpha_3\tilde{z}_3 + \beta_3)\right) \tag{20c}$$
$$= 9.6\frac{\alpha_1}{\alpha_2}\beta_3\tilde{z}_1 - 0.9\tilde{z}_2 + 9.6\frac{\alpha_1\alpha_3}{\alpha_2}\tilde{z}_1\tilde{z}_3 \tag{20d}$$
$$\dot{\tilde{z}}_3 = \frac{1}{\alpha_3}\left(-7.1 - 2.7(\alpha_3\tilde{z}_3 + \beta_3) - 3.1\alpha_1\alpha_2\tilde{z}_1\tilde{z}_2\right) \tag{20e}$$
$$= \frac{1}{\alpha_3}(-7.1 - 2.7\beta_3) - 2.7\tilde{z}_3 - 3.1\frac{\alpha_1\alpha_2}{\alpha_3}\tilde{z}_1\tilde{z}_2. \tag{20f}$$

By choosing $\alpha_1 = 1$, $\alpha_2 = -0.917$, $\alpha_3 = 0.524$, $\beta_3 = -2.665$, the system becomes

$$\dot{\tilde{z}}_1 = -10.0\tilde{z}_1 + 10.0\tilde{z}_2 \tag{21a}$$

$$\dot{\tilde{z}}_2 = 27.7\tilde{z}_1 - 0.9\tilde{z}_2 - 5.5\tilde{z}_1\tilde{z}_3 \tag{21b}$$

$$\dot{\tilde{z}}_3 = -2.7\tilde{z}_3 + 5.5\tilde{z}_1\tilde{z}_2. \tag{21c}$$

This has the same form as the original Lorenz equations with parameters that are close in value, apart from an arbitrary scaling that affects the magnitude of the coefficients of $\tilde{z}_1\tilde{z}_3$ in (21b) and $\tilde{z}_1\tilde{z}_2$ in (21c). The attractor dynamics for this system are very similar to the original Lorenz attractor and are shown in Figure 3c.

The learning procedure discovers a dynamical model by fitting coefficients that predict the continuous-time derivatives of the variables in a dynamical system. Thus it is possible for the training procedure to discover a model with unstable dynamics or which is unable to predict the true dynamics through simulation. We assess the validity of the discovered models by simulating the dynamics of the discovered low-dimensional dynamical system. Simulation of the system shows that the system is stable with trajectories existing on an attractor very similar to the original Lorenz attractor. Additionally, the discovered system is able to predict the dynamics of the original system. The fourth panel in Figure S1a shows the trajectories found by stepping the discovered model forward in time as compared with the values of $\mathbf{z}$ obtained by mapping samples of the high-dimensional data through the encoder. Although this is done on a new initial condition, the trajectories match very closely up to $t = 5$, which is the duration of trajectories contained in the training set. After that the trajectories diverge, but the predicted trajectories remain on an attractor. The Lorenz dynamics are chaotic, and thus slight differences in coefficients or initial conditions cause trajectories to diverge quickly.

For comparison, in Figure S1b we show a second model discovered by the training procedure. This model has 10 active terms, as compared with 7 in the true Lorenz system. While the model contains additional terms not present in the original system, the dynamics lie on an attractor with a similar two lobe structure. Additionally, the system is able to predict the dynamics through simulation. This model has a lower error on test data than the original 7 term model, with a fraction of unexplained variance of 2×10^{-6} for $\mathbf{x}$, 6×10^{-5} for $\dot{\mathbf{x}}$, and 3×10^{-4} for $\dot{\mathbf{z}}$.

S2.2 Reaction-diffusion

We generate data from a high-dimensional lambda-omega reaction-diffusion system governed by

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d_1(u_{xx} + u_{yy}) \tag{22a}$$

$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d_2(v_{xx} + v_{yy}) \tag{22b}$$

with $d_1, d_2 = 0.1$ and $\beta = 1$. The system is simulated from a single initial condition from $t = 0$ to $t = 10$ with a spacing of $\Delta t = 0.05$ for a total of 10,000 samples. The initial condition is defined as

$$u(y_1, y_2, 0) = \tanh\left(\sqrt{y_1^2 + y_2^2}\cos\left(\angle(y_1 + iy_2) - \sqrt{y_1^2 + y_2^2}\right)\right) \tag{23a}$$

$$v(y_1, y_2, 0) = \tanh\left(\sqrt{y_1^2 + y_2^2}\sin\left(\angle(y_1 + iy_2) - \sqrt{y_1^2 + y_2^2}\right)\right) \tag{23b}$$

over a spatial domain of $y_1 \in [-10, 10],\ y_2 \in [-10, 10]$ discretized on a grid with 100 points on each spatial axis. The solution of these equations results in a spiral wave formation. We apply our

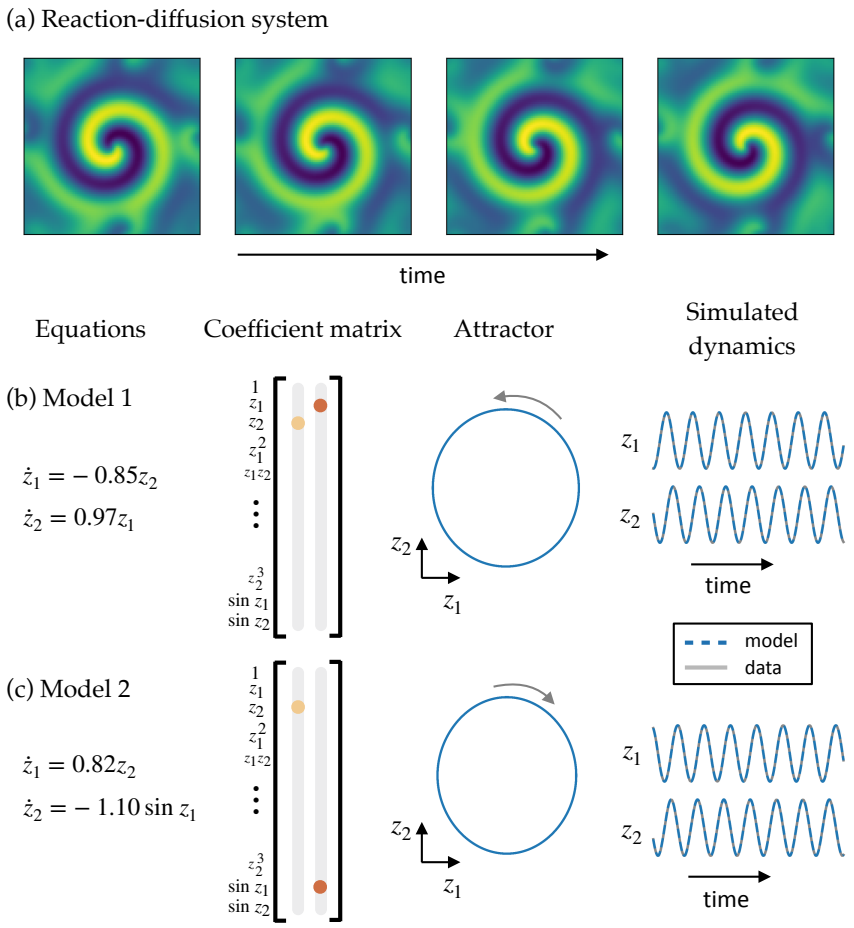

Figure S2: Resulting models for the reaction-diffusion system. (a) Snapshots of the high-dimensional system show a spiral wave formation. (b,c) Equations, SINDy coefficients $\boldsymbol{\Xi}$, attractors, and simulated dynamics for two models discovered by the SINDy autoencoder. The model in (b) is a linear oscillation, whereas the model in (c) is a nonlinear oscillation. Both models achieve similar error levels and can predict the dynamics in the test set via simulation of the low-dimensional dynamical system.

method to snapshots of $u(y_1, y_2, t)$ generated by the above equations, multiplied by a Gaussian $f(y_1, y_2) = \exp(-0.1(y_1^2 + y_2^2))$ centered at the origin to localize the spiral wave in the center of the domain. Our input data is thus defined as $\mathbf{x}(t) = f(:,:) \circ u(:,:,t) \in \mathbb{R}^{10^4}$. We also add Gaussian noise with a standard deviation of 10^{-6} to both $\mathbf{x}$ and $\dot{\mathbf{x}}$. Four time snapshots of the input data are shown in Figure S2a.

We divide the total number of samples into training, validation, and test sets: the last 1000 samples are taken as the test set, 1000 samples are chosen randomly from the first 9000 samples as a validation set, and the remaining 8000 samples are taken as the training set. We train ten models using the procedure outlined in Section S1.4 for 3×10^3 epochs followed by a refinement period of 10^3 epochs. Hyperparameters used for training are shown in Table S2. Nine of the ten resulting dynamical systems models have two active terms and one has three active terms. The dynamical equations, SINDy coefficient matrix, attractors, and simulated dynamics for two example models are shown in Figure S2b,c. The models with two active terms all have one of the two forms shown

Table S2: Hyperparameter values for the reaction-diffusion example

Parameter	**Value**
n	10^4
d	2
training samples	8000
batch size	1024
activation function	sigmoid
encoder layer widths	256
decoder layer widths	256
learning rate	10^{-3}
SINDy model order	1
SINDy library polynomial order	3
SINDy library includes sine	yes
SINDy loss weight $\dot{\mathbf{x}}$, λ_1	0.5
SINDy loss weight $\dot{\mathbf{z}}$, λ_2	0.01
SINDy regularization loss weight, λ_3	0.1

Table S3: Hyperparameter values for the nonlinear pendulum example

Parameter	**Value**
n	2601
d	1
training samples	5×10^4
batch size	1024
activation function	sigmoid
encoder layer widths	$128, 64, 32$
decoder layer widths	$32, 64, 128$
learning rate	10^{-4}
SINDy model order	2
SINDy library polynomial order	3
SINDy library includes sine	yes
SINDy loss weight $\dot{x}$, λ_1	5×10^{-4}
SINDy loss weight $\dot{z}$, λ_2	5×10^{-5}
SINDy regularization loss weight, λ_3	10^{-5}

in the figure: three models have a linear oscillation and six models have a nonlinear oscillation. Both model forms have similar levels of error on the test set and are able to predict the dynamics in the test set from simulation, as shown in the fourth panel of Figure S2b,c.

S2.3 Nonlinear pendulum

The nonlinear pendulum equation is given by

$$\ddot{z} = -\sin z. \tag{24}$$

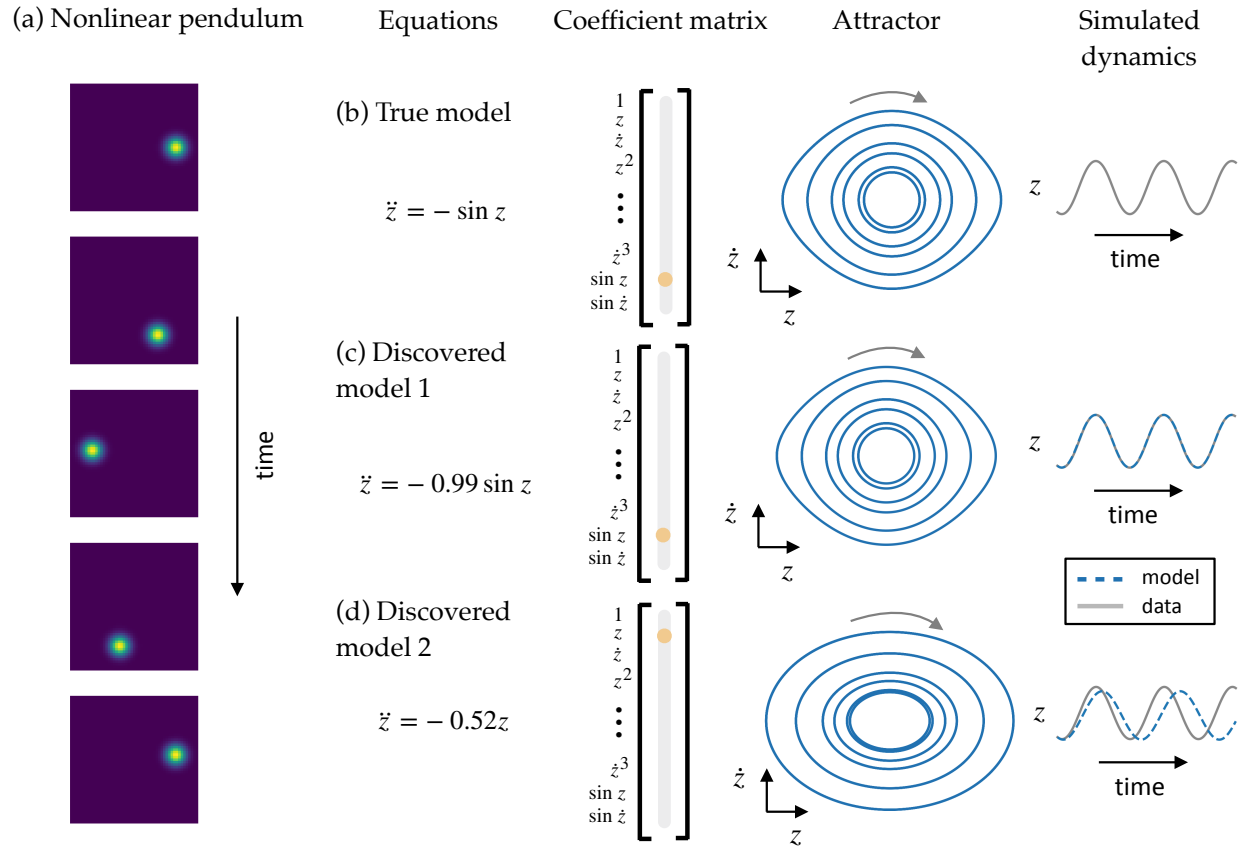

Figure S3: Resulting models for the nonlinear pendulum. (a) Snapshots of the high-dimensional system are images representing the position of the pendulum in time. (b,c,d) Comparison of two discovered models with the true pendulum dynamics. Equations, SINDy coefficients $\boldsymbol{\Xi}$, attractors, and simulated dynamics for the true pendulum equation are shown in (b). The model in (c) correctly discovered the true form of the pendulum dynamics. Both the image of the attractor and simulations match the true dynamics. (d) In one instance of training, the SINDy autoencoder discovered a linear oscillation for the dynamics. This model achieves a worse error than the model in (c).

We generate synthetic video of the pendulum in two spatial dimensions by creating high-dimensional snapshots given by

$$x(y_1, y_2, t) = \exp\left(-20\left((y_1 - \cos(z(t) - \pi/2))^2 + (y_2 - \sin(z(t) - \pi/2))^2\right)\right) \tag{25}$$

at a discretization of $y_1, y_2 \in [-1.5, 1.5]$. We use 51 grid points in each dimension resulting in snapshots $\mathbf{x}(t) \in \mathbb{R}^{2601}$. To generate a training set, we simulate (24) from 100 randomly chosen initial conditions with $z(0) \in [-\pi, \pi]$ and $\dot{z}(0) \in [-2.1, 2.1]$. The initial conditions are selected from a uniform distribution in the specified range but are restricted to conditions for which the pendulum does not have enough energy to do a full loop. This condition is determined by checking that $|\dot{z}(0)^2/2 - \cos z(0)| \leq 0.99$.

Following the training procedure outlined in Section S1.4, we train ten models for 5×10^3 epochs followed by a refinement period of 10^3 epochs. Hyperparameters used for this example are shown in Table S3. Five of the ten resulting models correctly recover the nonlinear pendulum equation. These five models have the best performance of the ten models. The attractor and

simulated dynamics for the best of these five models are shown in Figure S3. One model, also shown in Figure S3, recovers a linear oscillator. This model is able to achieve a reasonably low prediction error for $\ddot{\mathbf{x}}, \ddot{\mathbf{z}}$ but the simulated dynamics, while still oscillatory, appear qualitatively different from the true pendulum dynamics. The four remaining models all have two active terms in the dynamics and have a worse performance than the models with one active term.