HYPERSINDY: DEEP GENERATIVE MODELING OF NONLINEAR STOCHASTIC GOVERNING EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

The discovery of governing differential equations from data is an open frontier in machine learning. The *sparse identification of nonlinear dynamics* (SINDy) (Brunton et al., 2016) framework enables data-driven discovery of interpretable models in the form of sparse, deterministic governing laws. Recent works have sought to adapt this approach to the stochastic setting, though these adaptations are severely hampered by the curse of dimensionality. On the other hand, Bayesian-inspired deep learning methods have achieved widespread success in high-dimensional probabilistic modeling via computationally efficient approximate inference techniques, suggesting the use of these techniques for efficient stochastic equation discovery. Here, we introduce *HyperSINDy*, a framework for modeling stochastic dynamics via a deep generative model of sparse governing equations whose parametric form is discovered from data. HyperSINDy employs a variational encoder to approximate the distribution of observed states and derivatives. A hypernetwork (Ha et al., 2016) transforms samples from this distribution into the coefficients of a differential equation whose sparse form is learned simultaneously using a trainable binary mask (Louizos et al., 2018). Once trained, HyperSINDy generates stochastic dynamics via a differential equation whose coefficients are driven by a Gaussian white noise. In experiments, HyperSINDy accurately recovers ground truth stochastic governing equations, with learned stochasticity scaling to match that of the data. Finally, HyperSINDy provides uncertainty quantification that scales to high-dimensional systems. Taken together, HyperSINDy offers a promising framework for model discovery and uncertainty quantification in real-world systems, integrating sparse equation discovery methods with advances in statistical machine learning and deep generative modeling.

1 INTRODUCTION

Across numerous disciplines, large amounts of measurement data have been collected from dynamical phenomena lacking comprehensive mathematical descriptions. It is desirable to model these data in terms of governing equations involving the state variables, which typically enables insight into the physical interactions in the system. To this end, recent years have seen considerable progress in the ability to distill such governing equations from data alone (e.g., (Schmidt & Lipson, 2009; Brunton et al., 2016)). Nonetheless, this remains an outstanding challenge for systems exhibiting apparently stochastic nonlinear behavior, particularly when lacking even partial knowledge of the governing equations. Such systems thus motivate probabilistic approaches that not only reproduce the observed stochastic behavior (e.g., via generic stochastic differential equations (SDEs) (Friedrich et al., 2011) or neural networks (Girin et al., 2021; Lim & Zohren, 2021)), but do so via discovered analytical representations that are parsimonious and physically informative (Boninsegna et al., 2018).

We are particularly interested in model-free methods that seek to discover both the parameters and functional form of governing equations describing the data. To this end, the sparse identification of nonlinear dynamics (SINDy) framework (Brunton et al., 2016) has emerged as a powerful data-driven approach that identifies both the coefficients and terms of differential equations, given a pre-defined library of candidate functions. The effectiveness of SINDy for sparse model discovery derives from the tendency of physical systems to possess a relatively limited set of active terms. Extensions of the SINDy framework have sought to increase its robustness to noise, offer uncertainty quantification

(UQ), and make it suitable for modeling stochastic dynamics (Boninsegna et al., 2018; Niven et al., 2020; Messenger & Bortz, 2021; Hirsh et al., 2021; Callaham et al., 2021; Fasel et al., 2022; Wang et al., 2022). However, these extensions have generally relied upon computationally expensive approaches to learn the appropriate probability distributions. As such, a unified and computationally tractable formulation of SINDy that meets these additional goals is presently lacking.

Variational inference (VI) methods represent a class of techniques for addressing the complex and often intractable integrals arising in exact Bayesian inference, instead approximating the true posterior via simple distribution(s). Recently, the combination of *amortized* VI (Ganguly et al., 2022) with the representational capacity of neural networks has emerged as a powerful, efficient approach to probabilistic modeling, with widespread application in the form of deep generative models (Kingma & Welling, 2014; Rezende & Mohamed, 2015). Despite the success of these approaches for dynamical modeling (e.g., (Girin et al., 2021)), applications thus far have utilized generic state space formulations or parameter inference on a known functional form of the dynamics. Thus, the potential for VI to facilitate probabilistic equation discovery remains largely unexplored.

1.1 Contributions

In this work, we propose HyperSINDy, a VI-based SINDy implementation that learns a parameterized distribution of ordinary differential equations (ODEs) sharing a common sparse form. Specifically, HyperSINDy employs a variational encoder to parameterize a latent distribution over observed states and derivatives, then uses a hypernetwork (Ha et al., 2016; Pawlowski et al., 2018) to translate samples from this distribution into the coefficients of a sparse ODE whose functional form is learned in a common optimization. In this way, HyperSINDy is able to model complex stochastic dynamics through an interpretable analytical expression – technically, a *random* ODE (Han & Kloeden, 2017) – whose coefficients are parameterized by a white noise process.

Specific contributions of the HyperSINDy framework include:

- **Efficient and Accurate Modeling of Stochastic Dynamics at Scale.** Through VI, we circumvent the curse of dimensionality that hampers other methods in identifying sparse stochastic equations. Specifically, HyperSINDy can accurately discover governing equations for stochastic systems having well beyond two spatial dimensions, which existing approaches have not exceeded (e.g., (Boninsegna et al., 2018; Callaham et al., 2021; Wang et al., 2022; Huang et al., 2022)).
- **Generative Modeling of Dynamics.** Once trained, HyperSINDy generates a random dynamical system whose vector field is parameterized by a Gaussian white noise. Hence, our approach efficiently arrives at a generative model for both the system dynamics and the exogenous disturbances (representing, e.g., unresolved scales). This permits simulations that reproduce the stochastic dynamical behavior of the observed process, while providing a natural method for quantifying uncertainty of the model parameters and propagating uncertainty in the probabilistic model forecast.
- **Interpretable Governing Equations Discovery.** In contrast to other deep generative approaches for modeling stochastic dynamics, HyperSINDy discovers the analytical form of a sparse governing equation without a priori knowledge. Sparsity promotes human readable models where each term corresponds to an interpretable physical mechanism. This notion of interpretability, based on sparsity, is appealing in the traditional perspective of engineering and physics.

In section 1.2, we discuss relevant literature. In section 2, we provide a background on the specific methods and mathematics that inspired our method. In section 3, we describe HyperSINDy. In section 4, we show results on various experiments. In section 5, we conclude with a discussion of our method, its limitations, and possible future directions.

1.2 Related Work

HyperSINDy bridges two parallel lines of work concerning data-driven modeling for stochastic dynamics: namely, probabilistic sparse equation discovery and deep generative modeling.

Most probabilistic implementations of SINDy have concerned UQ and noise robustness in the deterministic setting, rather than modeling stochastic dynamics per se. Of these approaches, ensembling methods (E-SINDy) (Fasel et al., 2022) have achieved state-of-the-art UQ and noise robustness for deterministic SINDy models, and were recently shown (Gao et al., 2023) to offer a computationally

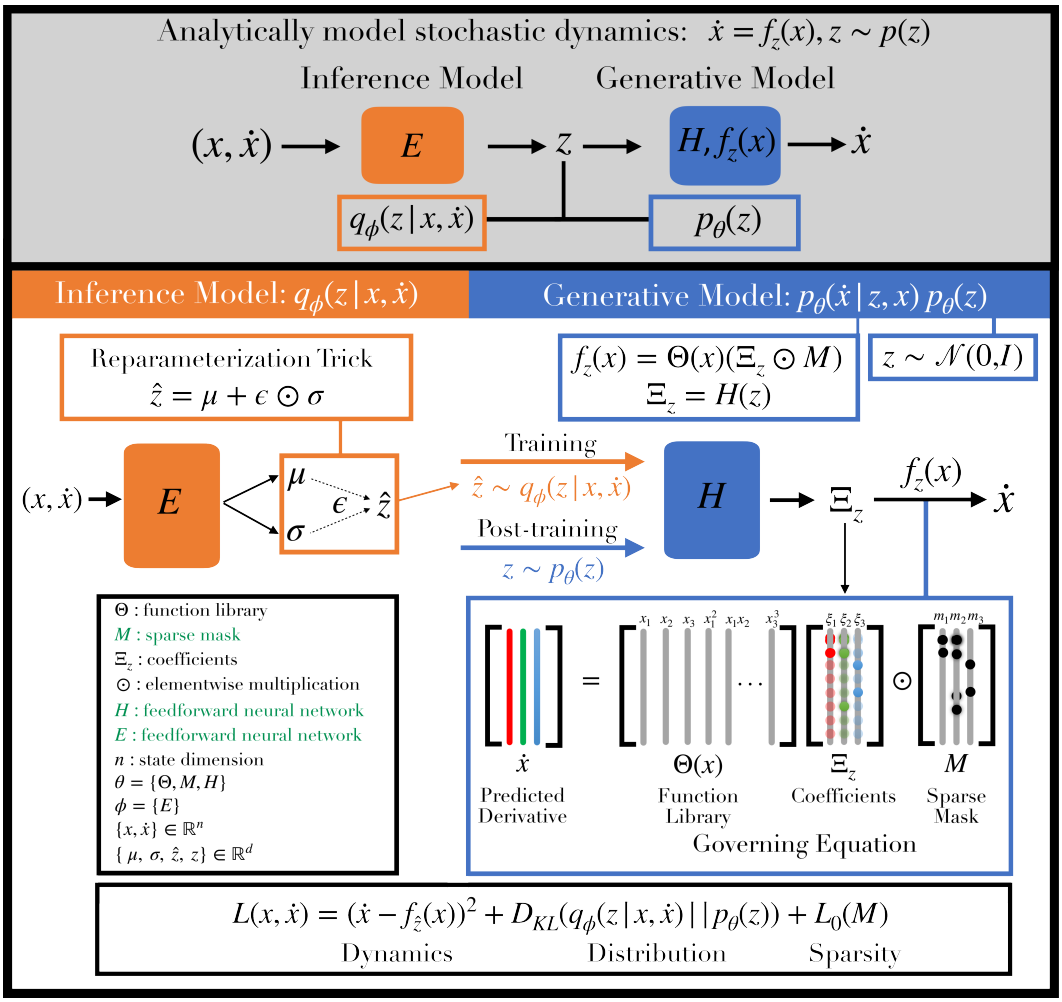


Figure 1: **HyperSINDy Framework**. HyperSINDy employs an inference model and generative model to discover an analytical representation of observed stochastic dynamics in the form of a random (nonlinear) ODE $f_{\mathbf{z}}(x)$. The inference model is an encoder neural network that maps $(\mathbf{x}, \dot{\mathbf{x}})$ to the parameters μ and σ of $q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})$. $\hat{\mathbf{z}}$ can be sampled using a simple reparameterization of μ and σ. The generative model predicts the derivative via a hypernetwork H, which transforms $\mathbf{z}$ into $\Xi_{\mathbf{z}}$, the coefficients of the ODE. $f_{\mathbf{z}}(\mathbf{x})$ comprises a function library Θ, the coefficients $\Xi_{\mathbf{z}}$, and sparse mask M. Once trained, stochastic dynamics can be generated by iteratively sampling $\mathbf{z}$ from its prior $\mathbf{z} \sim p_\theta(\mathbf{z})$, thus yielding new sample paths of the noise-paramterized vector field, $\Xi_{\mathbf{z}}$. In the legend, trainable parameters are shown in green. The loss function comprises terms related to 1) the derivative reconstructions, 2) the latent distribution $q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})$, and 3) sparsity of the discovered equation. See accompanying pseudocode 1 and 2 for details on batch-wise training.

efficient alternative to earlier Bayesian implementations of SINDy (Niven et al., 2020; Hirsh et al., 2021) leveraging costly sampling routines to compute posterior distributions. Nonetheless, a model of the process noise is crucial for accurate UQ in the stochastic dynamics setting. Multiple studies have generalized the SINDy framework for the identification of parametric SDEs (Boninsegna et al., 2018; Callaham et al., 2021), with three such studies recently performed in the Bayesian setting (Wang et al., 2022; Huang et al., 2022; Tripura & Chakraborty, 2023). However, as discussed in these works, existing methods for approximating the drift and diffusion terms of the SDE (e.g., constructing histograms for the Kramers-Moyal expansion) are severely hampered by the curse of dimensionality, with computational cost generally scaling exponentially with SDE state dimension. Thus, an efficient and scalable formulation of SINDy for stochastic dynamics remains lacking.

A separate line of work has leveraged advances in probabilistic deep learning for modeling stochastic dynamics, with deep generative models achieving state-of-the-art performance across a range of modeling tasks ((Yoon et al., 2019; Girin et al., 2021)). Although these models do not typically involve explicit dynamical representations, advances in physics-informed machine learning (Karniadakis et al., 2021) have motivated numerous developments at this intersection (e.g., (Lopez & Atzberger, 2021; Takeishi & Kalousis, 2021; Yang et al., 2020; Zhang et al., 2019). As for (stochastic) equation discovery, several recent works have successfully employed VAEs to learn the coefficients of a generic (or pre-specified) SDE within a (potentially lower-dimensional) latent space(Hasan et al., 2022; García et al., 2022; Nguyen et al., 2021; Zhong & Meidani, 2023). We propose to similarly leverage a VAE-like architecture to perform inference on a latent stochastic process; however, we seek to additionally discover a structural representation of the governing laws, which can yield considerable physical insight into the system (Boninsegna et al., 2018; Nayek et al., 2021; Wang et al., 2022). In sum, we seek to bridge the above fields via a unified deep learning architecture (trainable end-to-end with backpropagation) that enables discovery of the functional form of a governing stochastic process, along with posterior distributions over the discovered system coefficients (e.g., for UQ).

2 BACKGROUND

We briefly overview the SINDy and VAE frameworks, as well as an implementation of an L_0 loss, before describing their integration within the HyperSINDy architecture.

Sparse Identification of Nonlinear Dynamics The SINDy (Brunton et al., 2016) framework leverages sparse regression to enable discovery of a parsimonious system of differential equations from time-ordered snapshots. Thus, consider a system with state $\mathbf{x}(t) \in \mathbb{R}^d$ governed by the ODE:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) \tag{1}$$

Given m observations of the system in time $\mathbf{X} = [\mathbf{x}(t_1), \mathbf{x}(t_2), ..., \mathbf{x}(t_m)]^T$ and the estimated time derivatives $\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \dot{\mathbf{x}}(t_2), ..., \dot{\mathbf{x}}(t_m)]^T$, we construct a library of candidate functions $\Theta(\mathbf{X}) = [\theta_1(\mathbf{X}), \theta_2(\mathbf{X}), ..., \theta_l(\mathbf{X})]$. We then solve the regression problem, $\dot{\mathbf{X}} = \Theta(\mathbf{X})\boldsymbol{\Xi}$, to identify the optimal functions and coefficients in Θ and $\boldsymbol{\Xi}$, respectively. A sparsity-promoting regularization function R is typically added to this model discovery problem, yielding the final optimization, $\hat{\boldsymbol{\Xi}} = \arg\min_{\boldsymbol{\Xi}}(\dot{\mathbf{X}} - \Theta(\mathbf{X})\boldsymbol{\Xi})^2 + R(\boldsymbol{\Xi})$. Although we focus on this basic implementation, we note that there have been numerous extensions of the original SINDy framework (for a recent overview, see (Kaptanoglu et al., 2022)), many of which can be easily incorporated into the present framework.

Variational Autoencoder The VAE framework (Kingma & Welling, 2014) elegantly integrates variational inference (VI) with deep learning architectures, providing an efficient and powerful approach toward probabilistic modeling. VAEs assume that a set of observations $\mathbf{x}$ derives from a corresponding set of latent states $\mathbf{z}$. VAEs construct an approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and maximize the evidence lower bound (ELBO) of the log likelihood of the data $p_\theta(\mathbf{x})$:

$$\log p_\theta(\mathbf{x}) \geq ELBO(\mathbf{x}, \mathbf{z}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \tag{2}$$

where ϕ and θ and are the parameters of the inference (encoder) and generative (decoder) models, respectively. The "reparameterization trick" enables sampling from $q_\phi(\mathbf{z}|\mathbf{x})$ using $\mathbf{z} = \mu(\mathbf{z}) + \sigma(\mathbf{z}) \odot \epsilon$ while still training the network end-to-end with backpropagation. After training, new observations are easily generated by sampling from the prior $p_\theta(\mathbf{z})$, typically a unit Gaussian with diagonal covariance.

L_0 Regularization The L_0 norm is ideal for sparse regression problems as it penalizes all nonzero weights equally, regardless of magnitude. As L_0 regularization poses an intractable optimization problem, the L_1 regularization (lasso) – which penalizes the actual values of the learned weights – is a more common technique to achieve sparsity in practice. Nonetheless, incorporation of an L_0-norm penalty (Zheng et al., 2019) into SINDy was recently found to have considerable advantages (Champion et al., 2020), motivating us to adopt a backpropagation-compatible L_0 regularization. Accordingly, we implement one such method recently proposed by Louizos et al. (2018), which penalizes a trainable mask using the hard-concrete distribution.

Specifically, let $M \in \mathbb{R}^d$ be the desired sparse mask. Let s be a binary concrete random variable (Maddison et al., 2017; Jang et al., 2017) distributed in $(0, 1)$ with probability density $q_\phi(s)$, cumulative density $Q_\phi(s)$, location $\log \alpha$, and temperature β. Let $\phi = (\log \alpha, \beta)$. Suppose we have $\gamma < 0$

and $\zeta > 1$. We define each element m in M as a hard concrete random variable computed entirely as a transformation of s. Thus, learning an optimal m necessitates learning $q_\phi(s)$, which simplifies to optimizing $\log \alpha$ (we fix β). Sampling from $q_\phi(s)$ and backpropagating into $\log \alpha$ motivates use of the reparameterization trick (as in the VAE above) with $\epsilon \sim \mathcal{U}(0, 1)$. Then, m is computed.

$$s = \text{Sigmoid}((\log \epsilon - \log(1 - \epsilon) + \log \alpha)/\beta) \qquad m = \min(1, \max(0, s(\zeta - \gamma) + \gamma)) \tag{3}$$

After training, we obtain m using our optimized $\log \alpha$ parameter:

$$m = \min(1, \max(0, \text{Sigmoid}(\log \alpha)(\zeta - \gamma) + \gamma)) \tag{4}$$

We train M using the following loss:

$$L_0(M) = \sum_{j=1}^{d} \text{Sigmoid}(\log \alpha_j - \beta \log \frac{\gamma}{\zeta}) \tag{5}$$

Refer to (Louizos et al., 2018) for the full derivation. In short, this provides a backpropagation-compatible approach to enforce sparsity via a trainable, element-wise mask.

3 HYPERSINDY

We combine advances in Bayesian deep learning with the SINDy framework to propose HyperSINDy, a hypernetwork (Ha et al., 2016; Pawlowski et al., 2018) approach to parsimoniously model stochastic nonlinear dynamics via a noise-parameterized vector field whose sparse, time-invariant functional form is discovered from data. In brief, HyperSINDy uses a variational encoder to learn a latent distribution over the states and derivatives of a system, whose posterior is regularized to match a Gaussian prior. Once trained, a white noise process generates a time-varying vector field by updating the coefficients of the discovered (random) ODE. Across a range of experiments, new noise realizations generate stochastic nonlinear dynamics that recapitulate the behavior of the original system, while also enabling UQ on the learned coefficients. Fig. 1 provides an overview of our approach and problem setting, which we detail below.

Problem Setting Stochastic equations are fundamental tools for mathematically modeling dynamics under uncertainty. In general, the precise physical source of uncertainty is unknown and/or of secondary importance (Friedrich et al., 2011; Duan, 2015; Särkkä & Solin, 2019); as such, several formulations exist. A common choice is the Langevin-type SDE with explicitly separated deterministic (drift) and stochastic (diffusion) terms. Alternatively, we may consider a deterministic ODE with stochastic parameters, i.e., a *random* ODE (RDE), which is another well-established framework (Arnold, 1998; Duan, 2015) with wide-ranging real-world applications (e.g., fluctuating resources in biological systems (Kloeden & Pötzsche, 2013)). Here, we adopt the RDE formulation in the widely studied setting of i.i.d. noise (Arnold, 1998; Caraballo & Han, 2016). We find this formulation practically advantageous for integration with deep generative modeling and VI, enabling a powerful and scalable approach to stochastic dynamics. Importantly, as any (finite-dimensional) SDE can be transformed into an equivalent RDE and vice versa (Han & Kloeden, 2017); these practical advantages can be exploited without compromising relevance to canonical SDE representations (as we will empirically demonstrate).

As above, let $\mathbf{x}_{0:T}$ be the observations from times 0 to T of the state of a system, $\mathbf{x}_t \in \mathbb{R}^n$. We assume these data are generated from some stochastic dynamics $\dot{\mathbf{x}} = f_{\mathbf{z}}(\mathbf{x}_t)$, where $\mathbf{z}$ is a latent random variable modeled as an i.i.d. noise process. We wish to identify a family of sparse vector field functions $f_{\mathbf{z}}$ constrained to a common functional form for all $\mathbf{z} \in \mathbb{R}^d$ (i.e., only the coefficients of f are time-varying, reflecting the system's dependence on fluctuating quantities).

With this framing, we seek to approximate both the functional form $f_{\mathbf{z}}$ and a posterior estimate of the latent noise trajectory $\mathbf{z} = [\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_T]^{\mathrm{T}}$ associated with each observed trajectory $\mathbf{x}_{\mathbf{0:T}}$. To do so, we employ a variational encoder to learn an inference model for the latent space $p(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})$ and a generative model $p(\dot{\mathbf{x}}|\mathbf{x}, \mathbf{z})$ subject to $\dot{\mathbf{x}} = f_{\mathbf{z}}(\mathbf{x})$, as detailed below. Ultimately, once trained, we may generate new trajectories of $\mathbf{x}$ simply by iteratively sampling $\mathbf{z}$ from its Gaussian prior (i.e., constructing new sample paths of the driving noise).

Generative Model Consider a factorization of the conditional generative model with parameters θ: $p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x}) = p_\theta(\dot{\mathbf{x}}|\mathbf{z}, \mathbf{x})p_\theta(\mathbf{z})$. We assume that $\mathbf{z}$ is independent of $\mathbf{x}$, so $p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z})$. $p_\theta(\dot{\mathbf{x}}|\mathbf{z}, \mathbf{x})$ describes how the state $\mathbf{x}$ and latent $\mathbf{z}$ are transformed into the derivative, while $p_\theta(\mathbf{z})$ is a prior over the latent distribution of states and their derivatives. We take $p_\theta(\mathbf{z})$ to be a standard Gaussian with diagonal covariance: $p_\theta(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$. We seek to parameterize $p_\theta(\dot{\mathbf{x}}|\mathbf{z}, \mathbf{x})$ according to a nonlinear function, $f_\mathbf{z}(\mathbf{x})$. Following the SINDy framework, which seeks interpretability in the form of sparse governing equations, we adapt 1 to obtain the following implementation of $f_\mathbf{z}(\mathbf{x})$:

$$f_\mathbf{z}(\mathbf{x}) = \Theta(\mathbf{x})(\Xi_\mathbf{z} \odot M). \tag{6}$$

where $\odot$ indicates an element-wise multiplication. $\Theta(\mathbf{x})$ is a matrix expansion of $\mathbf{x}$ using a predefined library of basis functions, which can include any rational functions, such as polynomial (e.g., $\mathbf{x}_1^2, \mathbf{x}_1\mathbf{x}_2$) or trigonometric (e.g., $\sin \mathbf{x_1}$) functions. $\Xi_\mathbf{z}$ is a matrix of coefficients that is output by a hypernetwork H that takes in $\mathbf{z}$ as input: $\Xi_\mathbf{z} = H(\mathbf{z})$. M is a matrix of values $M_{ij} \in [0, 1]$ that is trained with a close approximation to a differentiable L_0 norm. Specifically, the values of M are simulated using a hard concrete distribution. As such, M enforces sparsity in the terms of each equation through the element-wise multiplication $(\Xi_\mathbf{z} \odot M)$. See Background for more details.

We constrain $f_\mathbf{z}$ to a d-parameter family of ODEs sharing a common functional form. Specifically, H may be interpreted as an implicit distribution (Pawlowski et al., 2018) for $p_\theta(\mathbf{\Xi}) = \int p_\theta(\mathbf{\Xi}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}$. Although we cannot compute the density of $p_\theta(\mathbf{\Xi})$ exactly, we can sample derivative functions by feeding samples $\mathbf{z} \sim p_\theta(\mathbf{z})$ into the hypernetwork: $\Xi_\mathbf{z} = H(\mathbf{z})$.

Inference Model Our inference model is defined by the approximate posterior, $q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})$, with parameters ϕ. $q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})$ is implemented by a neural network E and the reparameterization trick, i.e., $\mu_q, \sigma_q = E(\mathbf{x}, \dot{\mathbf{x}})$; $\hat{\mathbf{z}} = \mu_q + \epsilon \odot \sigma_q$.

Training We train the model end-to-end with backpropagation to minimize the following loss:

$$loss = (\dot{\mathbf{x}} - f_{\hat{\mathbf{z}}}(\mathbf{x}))^2 + \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})||p_\theta(\mathbf{z})) + \lambda L_0(M) \tag{7}$$

where β and λ are hyperparameters. The loss function optimizes the parameters ϕ and θ, where ϕ are the parameters of E (i.e., the variational parameters) and θ are the parameters of H and M (note that $p_\theta(\mathbf{z})$ has fixed parameters). Refer to the Appendix for a full derivation of this loss function, and to Background for details on the sparsity-related loss $L_0(M)$ (especially equation 5). To speed up training, every set number of epochs, we permanently set values of M equal to 0 if the magnitude of corresponding coefficients fall below a specific threshold value.

4 RESULTS

We evaluate the performance of HyperSINDy on four stochastic dynamical systems. Across a range of (dynamical) noise levels, we seek to assess the accuracy of models identified by HyperSINDy and the degree to which uncertainty estimates faithfully reflect the level of simulated noise. Refer to the Appendix for full details on data generation, training, and simulations.

4.1 STOCHASTIC EQUATION DISCOVERY

First, we show results for 3D Stochastic Lorenz and 3D Stochastic Rössler datasets, simulated by:

$$\dot{x} = \omega(y - x) \qquad \dot{y} = x(\rho - z) - y \qquad \dot{z} = xy - \beta z \qquad \text{Lorenz} \tag{8}$$

$$\dot{x} = -y - z \qquad \dot{y} = x + ay \qquad \dot{z} = b + z(x - c) \qquad \text{Rössler} \tag{9}$$

where parameters (ω, ρ, β) and (a, b, c) are each modeled as random processes, simulated by iteratively sampling (at each timestep) from normal distributions with scale σ and means $(10, 28, \frac{8}{3})$ and $(0.2, 0.2, 5.7)$, respectively (i.e., each parameter is driven by an independent white noise). We train a HyperSINDy model on three trajectories from each system, with $\sigma = 1, 5, 10$.

We find that HyperSINDy correctly identifies most terms in each equation (Fig. 2). Notably, increasing noise has little impact on the mean coefficients learned by HyperSINDy; instead, the estimated standard deviations of these coefficients proportionately scale with the dynamical noise. Furthermore, HyperSINDy only increases the standard deviation on the terms modeled to have

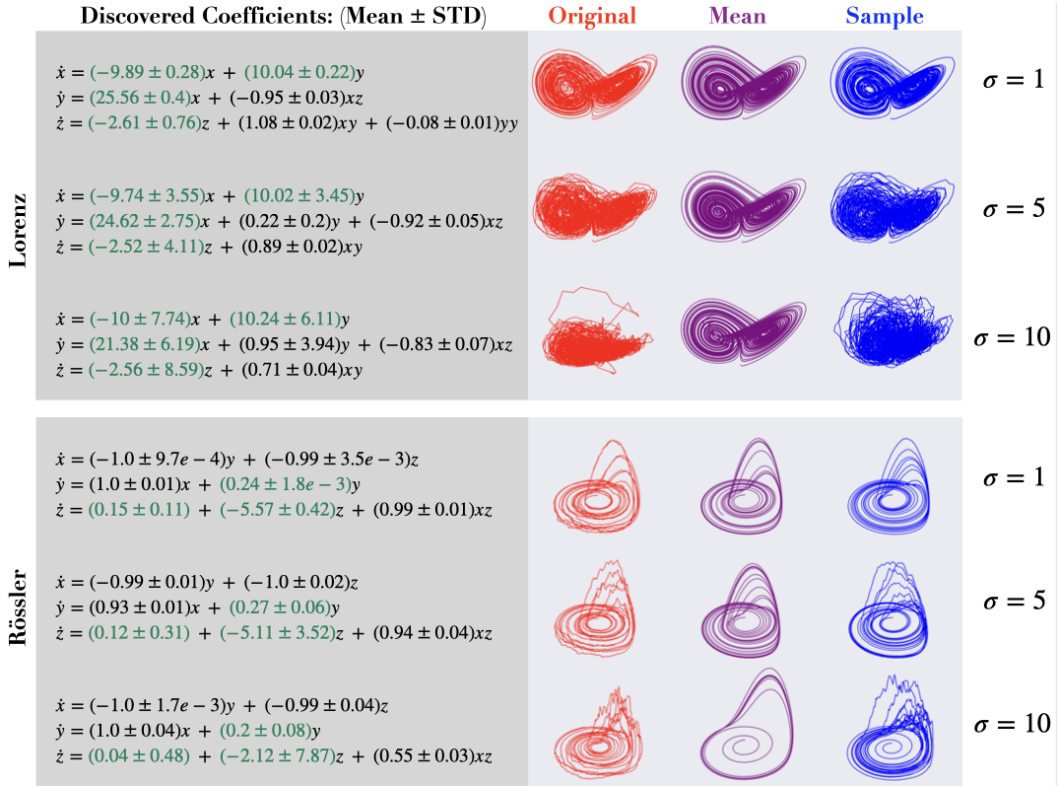


Figure 2: **3D Stochastic Lorenz and Rössler**. HyperSINDy models trained on trajectories simulated with three levels of noise (σ) for each of the two systems. For each σ, the mean and standard deviation of the discovered governing equation coefficients are shown (refer to 8 and 9 for ground truth equations). Coefficients in green are those iteratively sampled from Gaussian distributions in the original system. Red trajectories are sample test trajectories simulated with the given σ. Purple trajectories are generated from HyperSINDy using the mean of the discovered governing equations. Blue trajectories are generated by iteratively sampling from HyperSINDy's learned generative model. See Fig. S1 for additional samples; E-SINDy samples are shown in Fig. S2.

additional noise, while maintaining tight bounds on other terms (e.g. xy in $\dot{y}$ for Lorenz). Moreover, HyperSINDy is able to simulate the original (stochastic) dynamical behavior even as the noise level increases (blue trajectories). On the other hand, because HyperSINDy also successfully identifies the deterministic functional form despite process noise, it is able to produce smooth trajectories (purple) by forecasting with the mean of the discovered equation ensemble.

We sought to benchmark HyperSINDy performance against a leading model for probabilistic model discover, ensemble SINDy (E-SINDy) (Fasel et al., 2022). We trained a total of 30 HyperSINDy and 30 E-SINDy models using Lorenz and Rössler trajectories (see appendix for all simulation details). We evaluated the RMSE of the mean and standard deviation of the discovered equation coefficients, as compared to ground truth. HyperSINDy outperforms E-SINDy on both mean and standard deviation for each experiment (1). Refer to appendix Table S4 for complementary precision and recall results. This pattern of results was minimally sensitive to latent space dimension (Fig. S5). See Table S6 for further quantitative results validating HyperSINDy's strength at equation discovery on a 2D system.

4.2 Recovering drift-diffusion dynamics

The preceding analyses validate HyperSINDy's capacity for stochastic equation discovery. As HyperSINDy adopts an RDE-based modeling strategy (i.e., a noise-parameterized ODE, as opposed to an SDE with separable drift and diffusion), ground truth equations were explicitly modeled as RDEs rather than SDEs to enable straightforward comparison. As RDEs are conjugate to SDEs

Table 1: Total coefficient RMSE (↓) relative to ground truth equations

Param		Lorenz		Rossler	
		HyperSINDy	E-SINDy	HyperSINDy	E-SINDy
1	MEAN	**0.082** $\pm$ 0.004	0.18 $\pm$ 0.029	**0.029** $\pm$ 0.035	0.077 $\pm$ 0.04
	STD	**0.598** $\pm$ 0.045	1.296 $\pm$ 0.083	**0.828** $\pm$ 0.059	0.849 $\pm$ 0.012
5	MEAN	**0.117** $\pm$ 0.022	0.268 $\pm$ 0.064	**0.086** $\pm$ 0.047	0.296 $\pm$ 0.199
	STD	**0.4** $\pm$ 0.055	0.971 $\pm$ 0.024	**0.807** $\pm$ 0.012	0.875 $\pm$ 0.023
10	MEAN	**0.203** $\pm$ 0.047	0.349 $\pm$ 0.103	**0.228** $\pm$ 0.138	0.699 $\pm$ 0.551
	STD	**0.279** $\pm$ 0.085	0.913 $\pm$ 0.016	**0.812** $\pm$ 0.014	0.875 $\pm$ 0.028

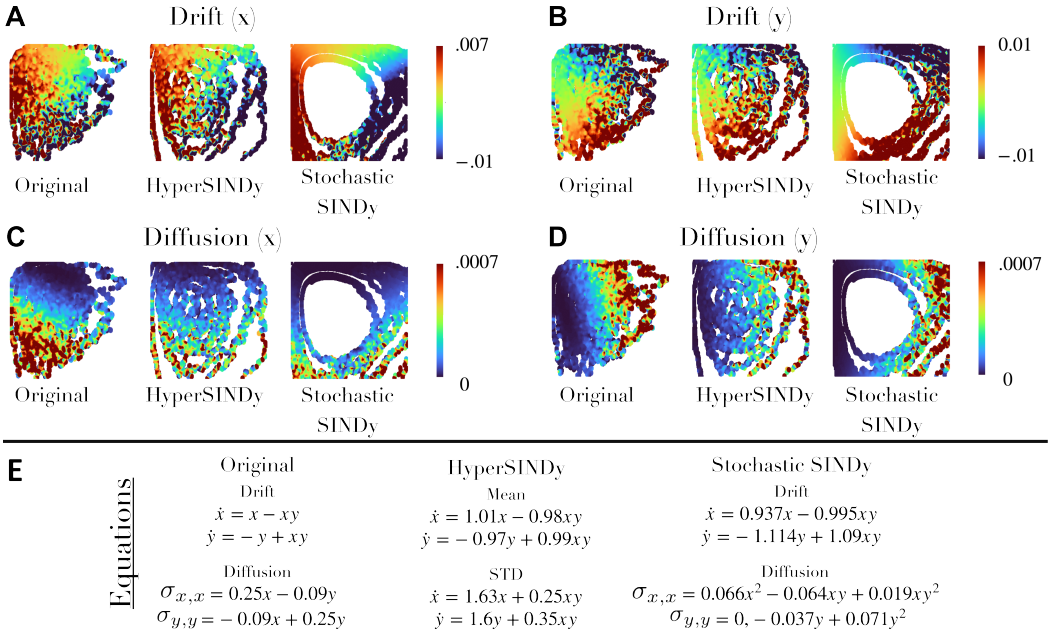


Figure 3: **Recovering drift and diffusion behavior in the stochastic Lotka-Volterra model.**. K-M coefficients computed on sample trajectories from each of the three models. From left to right: the ground truth SDE, the HyperSINDy-discovered system, and the Stochastic SINDy-discovered system.

(Han & Kloeden, 2017), this distinction is not fundamental. Nonetheless, this leaves unaddressed the question of how HyperSINDy would learn to represent dynamics explicitly simulated as SDEs.

To address this question, we simulate a 2D SDE to enable direct comparison against the leading method, stochastic SINDy (Boninsegna et al., 2018; Nabeel et al., 2022) (which cannot easily scale to higher dimensions). Specifically, we simulate a widely used model for population dynamics, the stochastic Lotka-Volterra system with state-dependent diffusion:

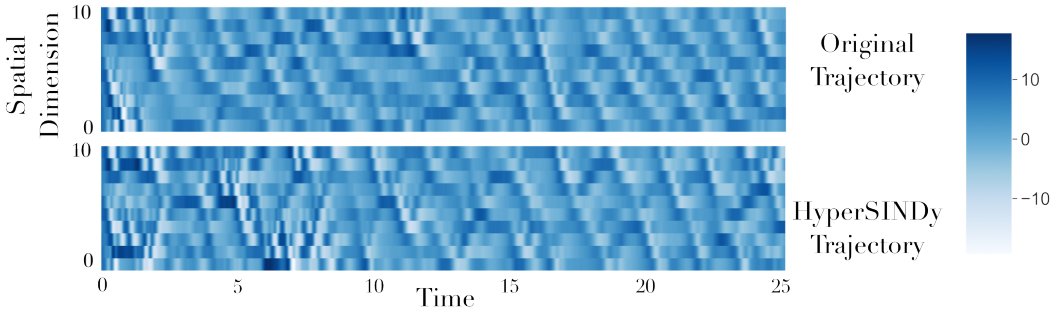
$$\begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \end{pmatrix} = \begin{pmatrix} x - xy \\ -y + xy \end{pmatrix} \mathrm{d}t + \begin{pmatrix} \sigma_{xx} & 0 \\ 0 & \sigma_{yy} \end{pmatrix} \begin{pmatrix} \mathrm{d}W_x \\ \mathrm{d}W_y \end{pmatrix} \tag{10}$$

where $\sigma_{xx}(t) = 0.25x - 0.09y$ and $\sigma_{yy}(t) = -0.09x + 0.25y$ give the state-dependent diffusion coefficients, and $W_x(t), W_y(t)$ are independent Wiener processes with i.i.d. increments (i.e., $\Delta W_t = W_{t+1} - W_t \sim \mathcal{N}(0, \Delta t)$). The system is simulated with Euler-Maruyama integration ($\Delta t = 0.01$).

Figure 3 illustrates the results of this analysis. Notably, HyperSINDy learns an expression whose terms correspond to those of the original drift function, thus enabling physical insight into the system. Nonetheless, the analytical expressions are not directly comparable, as the SDE represents stochasticity in terms of a separate diffusion term, while HyperSINDy represents stochasticity as coefficient noise. To enable direct comparison of the deterministic and stochastic aspects of the dynamics discovered by the two methods, we may numerically estimate drift and diffusion coefficients

from the simulated trajectories. Specifically, we may estimate the first two Kramers-Moyal (K-M) coefficients, which derive from a Taylor expansion of the master equation, and which fully describe the Markovian dynamics. Notably, HyperSINDy captures the appropriate deterministic (drift) and stochastic (diffusion) behavior of the system, recapitulating the state-dependence of these terms as seen in the original system – even performing favorably to stochastic SINDy in this setting.

4.3 High Dimensional Stochastic Discovery



Mean

$$\dot{x}_1 = 6.75 - 0.75x_1 + 0.99x_2x_{10} - 0.99x_9x_{10}$$
$$\dot{x}_2 = 8.46 - 0.74x_2 + 0.98x_1x_3 - 0.99x_1x_{10}$$
$$\dot{x}_3 = 7.59 - 0.71x_3 - 0.99x_1x_2 + 0.98x_2x_4$$
$$\dot{x}_4 = 7.89 - 0.77x_4 - 0.97x_2x_3 + 0.97x_3x_5$$
$$\dot{x}_5 = 6.7 - 0.81x_5 - 0.97x_3x_4 + 0.97x_4x_6$$
$$\dot{x}_6 = 7.86 - 0.76x_6 - 0.99x_4x_5 + 0.99x_5x_7$$
$$\dot{x}_7 = 7.31 - 0.75x_7 - 0.99x_5x_6 + 0.98x_6x_8$$
$$\dot{x}_8 = 7.56 - 0.76x_8 - 0.98x_6x_7 + 0.98x_7x_9$$
$$\dot{x}_9 = 7.45 - 0.74x_9 - 0.98x_7x_8 + 0.99x_8x_{10}$$
$$\dot{x}_{10} = 6.49 - 0.75x_{10} + 0.99x_1x_9 - 0.99x_8x_9$$

STD

$$\dot{x}_1 = 8.44 + 0.04x_1 + 0.02x_2x_{10} + 0.01x_9x_{10}$$
$$\dot{x}_2 = 7.98 + 0.05x_2 + 0.02x_1x_3 + 0.02x_1x_{10}$$
$$\dot{x}_3 = 8.1 + 0.05x_3 + 0.01x_1x_2 + 0.02x_2x_4$$
$$\dot{x}_4 = 7.72 + 0.04x_4 + 0.01x_2x_3 + 0.01x_3x_5$$
$$\dot{x}_5 = 7.65 + 0.04x_5 + 0.02x_3x_4 + 0.01x_4x_6$$
$$\dot{x}_6 = 8.47 + 0.05x_6 + 0.03x_4x_5 + 0.01x_5x_7$$
$$\dot{x}_7 = 8.09 + 0.06x_7 + 0.01x_5x_6 + 0.02x_6x_8$$
$$\dot{x}_8 = 7.89 + 0.03x_8 + 0.02x_6x_7 + 0.01x_7x_9$$
$$\dot{x}_9 = 7.97 + 0.04x_9 + 0.01x_7x_8 + 0.02x_8x_{10}$$
$$\dot{x}_{10} = 7.83 + 0.03x_{10} + 0.01x_1x_9 + 0.01x_8x_9$$

Figure 4: **10D Stochastic Lorenz-96**. A sample test trajectory with $\sigma = 10$ (top) and sample HyperSINDy trajectory (middle) after training on a dataset with $\sigma = 10$. The bottom boxes show the mean and standard deviation of coefficients in the discovered governing equations (cf. Eq. 11).

Lastly, we assess HyperSINDy's capacity for Bayesian inference/stochastic modeling for high dimensional stochastic systems, which are not amenable to existing analytical SDE discovery methods (e.g., (Boninsegna et al., 2018; Callaham et al., 2021)). Thus, we simulate a stochastic version of the Lorenz-96 system using:

$$\dot{x}_i = F_i + x_{i+1}x_{i-1} - x_{i-2}x_{i-1} - x_i \tag{11}$$

for $i = 1, ..., 10$ where $x_{-1} = x_9$, $x_0 = x_{10}$, and $x_{11} = x_1$. We iteratively sample each F_i from a normal distribution: $F_i \sim \mathcal{N}(8, 10)$. As shown in Fig. 4, HyperSINDy correctly identifies all terms in the system, while also correctly learning a high variance coefficient exclusively for the forcing terms, F_i. In addition, HyperSINDy produces sample trajectories that match the stochastic dynamical behavior of ground truth sample trajectories. Refer to appendix Table S5 for quantitative comparisons between HyperSINDy and E-SINDy on the 10D Lorenz-96 system.

5 Discussion

We have provided an overview of HyperSINDy, a neural network-based approach to sparse equation discovery for stochastic dynamics. HyperSINDy is unique in its ability to provide analytical representations and UQ in the setting of high-dimensional stochastic dynamics. The present work represents a proof of concept for this architecture. We envision numerous future directions for extending the algorithmic and theoretical aspects of HyperSINDy – e.g., evaluation in the context of other noise types and with respect to convergence in the continuous limit. Moreover, while we employ a fairly straightforward implementation of SINDy, numerous developments of the SINDy framework (Kaptanoglu et al., 2022) may be easily incorporated into the HyperSINDy architecture. Finally, the integration of SINDy into a neural network framework paves the way for future developments that incorporate advances in probabilistic machine learning with interpretable equation discovery.

REFERENCES

Ludwig Arnold. *Random Dynamical Systems*. Springer Monographs in Mathematics. Springer, Berlin, Heidelberg, 1998. ISBN 978-3-642-08355-6 978-3-662-12878-7. doi: 10.1007/978-3-662-12878-7. URL http://link.springer.com/10.1007/978-3-662-12878-7.

Stefan Bauer, Nico S Gorbach, Djordje Miladinovic, and Joachim M Buhmann. Efficient and flexible inference for stochastic systems. *Advances in Neural Information Processing Systems*, 30, 2017.

Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. Sparse learning of stochastic dynamical equations. *The Journal of Chemical Physics*, 148(24):241723, March 2018. ISSN 0021-9606. doi: 10.1063/1.5018409. URL https://doi.org/10.1063/1.5018409.

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. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1517384113. URL https://pnas.org/doi/full/10.1073/pnas.1517384113.

J. L. Callaham, J.-C. Loiseau, G. Rigas, and S. L. Brunton. Nonlinear stochastic modelling with Langevin regression. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2250):20210092, June 2021. ISSN 1364-5021, 1471-2946. doi: 10.1098/rspa.2021.0092. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2021.0092.

Tomás Caraballo and Xiaoying Han. *Applied Nonautonomous and Random Dynamical Systems*. SpringerBriefs in Mathematics. Springer International Publishing, Cham, 2016. ISBN 978-3-319-49246-9 978-3-319-49247-6. doi: 10.1007/978-3-319-49247-6. URL http://link.springer.com/10.1007/978-3-319-49247-6.

Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved Conditional VRNNs for Video Prediction. pp. 7608–7617, 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/html/Castrejon_Improved_Conditional_VRNNs_for_Video_Prediction_ICCV_2019_paper.html.

Kathleen Champion, Peng Zheng, Aleksandr Y. Aravkin, Steven L. Brunton, and J. Nathan Kutz. A Unified Sparse Optimization Framework to Learn Parsimonious Physics-Informed Models From Data. *IEEE Access*, 8:169259–169271, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.3023625. URL https://ieeexplore.ieee.org/document/9194760/.

Jinqiao Duan. *An Introduction to Stochastic Dynamics*. Cambridge University Press, New York, NY, 1st edition edition, April 2015. ISBN 978-1-107-42820-1.

U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton. Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478(2260):20210904, April 2022. doi: 10.1098/rspa.2021.0904. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2021.0904. Publisher: Royal Society.

Rudolf Friedrich, Joachim Peinke, Muhammad Sahimi, and M. Reza Rahimi Tabar. Approaching complexity by stochastic methods: From biological systems to turbulence. *Physics Reports*, 506(5):87–162, September 2011. ISSN 03701573. doi: 10.1016/j.physrep.2011.05.003. URL https://linkinghub.elsevier.com/retrieve/pii/S0370157311001530.

Ankush Ganguly, Sanjana Jain, and Ukrit Watchareeruetai. Amortized Variational Inference: Towards the Mathematical Foundation and Review, September 2022. URL http://arxiv.org/abs/2209.10888. arXiv:2209.10888 [cs, math, stat].

L. Mars Gao, Urban Fasel, Steven L. Brunton, and J. Nathan Kutz. Convergence of uncertainty estimates in Ensemble and Bayesian sparse model discovery, January 2023. URL http://arxiv.org/abs/2301.12649. arXiv:2301.12649 [cs, math, stat].

Constantino A. García, Paulo Félix, Jesús M. Presedo, and Abraham Otero. Stochastic embeddings of dynamical phenomena through variational autoencoders. *Journal of Computational Physics*, 454:110970, April 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2022.110970. URL https://www.sciencedirect.com/science/article/pii/S0021999122000328.

Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical Variational Autoencoders: A Comprehensive Review. *Foundations and Trends® in Machine Learning*, 15(1-2):1–175, December 2021. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000089. URL https://www.nowpublishers.com/article/Details/MAL-089. Publisher: Now Publishers, Inc.

David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks, December 2016. URL http://arxiv.org/abs/1609.09106. arXiv:1609.09106 [cs].

Xiaoying Han and Peter E. Kloeden. *Random Ordinary Differential Equations and Their Numerical Solution*, volume 85 of *Probability Theory and Stochastic Modelling*. Springer Singapore, Singapore, 2017. ISBN 978-981-10-6264-3 978-981-10-6265-0. doi: 10.1007/978-981-10-6265-0. URL http://link.springer.com/10.1007/978-981-10-6265-0.

Ali Hasan, João M. Pereira, Sina Farsiu, and Vahid Tarokh. Identifying Latent Stochastic Differential Equations. *IEEE Transactions on Signal Processing*, 70:89–104, 2022. ISSN 1941-0476. doi: 10.1109/TSP.2021.3131723. Conference Name: IEEE Transactions on Signal Processing.

Seth M. Hirsh, David A. Barajas-Solano, and J. Nathan Kutz. Sparsifying Priors for Bayesian Uncertainty Quantification in Model Discovery, July 2021. URL http://arxiv.org/abs/2107.02107. arXiv:2107.02107 [math].

Yunfei Huang, Youssef Mabrouk, Gerhard Gompper, and Benedikt Sabass. Sparse inference and active learning of stochastic differential equations from data. *Scientific Reports*, 12(1):21691, December 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-25638-9. URL https://www.nature.com/articles/s41598-022-25638-9. Number: 1 Publisher: Nature Publishing Group.

Peter Imkeller and Björn Schmalfuss. The conjugacy of stochastic and random differential equations and the existence of global attractors. *Journal of Dynamics and Differential Equations*, 13:215–249, 2001.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax, August 2017. URL http://arxiv.org/abs/1611.01144. arXiv:1611.01144 [cs, stat].

Alan A. Kaptanoglu, Brian M. de Silva, Urban Fasel, Kadierdan Kaheman, Andy J. Goldschmidt, Jared Callaham, Charles B. Delahunt, Zachary G. Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, January 2022. ISSN 2475-9066. doi: 10.21105/joss.03994. URL https://joss.theoj.org/papers/10.21105/joss.03994.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, June 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00314-5. URL https://www.nature.com/articles/s42254-021-00314-5. Number: 6 Publisher: Nature Publishing Group.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv:1312.6114 [cs, stat].

Diederik P. Kingma and Max Welling. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, November 2019. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000056. URL https://www.nowpublishers.com/article/Details/MAL-056. Publisher: Now Publishers, Inc.

Peter E. Kloeden and Christian Pötzsche (eds.). *Nonautonomous Dynamical Systems in the Life Sciences*, volume 2102 of *Lecture Notes in Mathematics*. Springer International Publishing, Cham, 2013. ISBN 978-3-319-03079-1 978-3-319-03080-7. doi: 10.1007/978-3-319-03080-7. URL https://link.springer.com/10.1007/978-3-319-03080-7.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, February 2021. doi: 10.1098/rsta.2020.0209. URL https://royalsocietypublishing.org/doi/full/10.1098/rsta.2020.0209. Publisher: Royal Society.

Ryan Lopez and Paul J. Atzberger. Variational Autoencoders for Learning Nonlinear Dynamics of Physical Systems, March 2021. URL http://arxiv.org/abs/2012.03448. arXiv:2012.03448 [cs, eess, math].

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs, math].

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning Sparse Neural Networks through L_0 Regularization, June 2018. URL http://arxiv.org/abs/1712.01312. arXiv:1712.01312 [cs, stat].

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables, March 2017. URL http://arxiv.org/abs/1611.00712. arXiv:1611.00712 [cs, stat].

Daniel A. Messenger and David M. Bortz. Weak SINDy: Galerkin-Based Data-Driven Model Selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, January 2021. ISSN 1540-3459, 1540-3467. doi: 10.1137/20M1343166. URL http://arxiv.org/abs/2005.04339. arXiv:2005.04339 [cs, math].

Arshed Nabeel, Ashwin Karichannavar, Shuaib Palathingal, Jitesh Jhawar, Danny Raj M, and Vishwesha Guttal. PyDaddy: A Python package for discovering stochastic dynamical equations from timeseries data, November 2022. URL http://arxiv.org/abs/2205.02645. arXiv:2205.02645 [cs, math, q-bio].

R. Nayek, R. Fuentes, K. Worden, and E. J. Cross. On spike-and-slab priors for Bayesian equation discovery of nonlinear dynamical systems via sparse linear regression. *Mechanical Systems and Signal Processing*, 161:107986, December 2021. ISSN 0888-3270. doi: 10.1016/j.ymssp.2021.107986. URL https://www.sciencedirect.com/science/article/pii/S0888327021003812.

Duong Nguyen, Said Ouala, Lucas Drumetz, and Ronan Fablet. Variational Deep Learning for the Identification and Reconstruction of Chaotic and Stochastic Dynamical Systems from Noisy and Partial Observations, February 2021. URL http://arxiv.org/abs/2009.02296. arXiv:2009.02296 [cs, stat].

Robert K. Niven, Ali Mohammad-Djafari, Laurent Cordier, Markus Abel, and Markus Quade. Bayesian Identification of Dynamical Systems. *Proceedings*, 33(1):33, 2020. ISSN 2504-3900. doi: 10.3390/proceedings2019033033. URL https://www.mdpi.com/2504-3900/33/1/33. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

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 *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Nick Pawlowski, Andrew Brock, Matthew C. H. Lee, Martin Rajchl, and Ben Glocker. Implicit Weight Uncertainty in Neural Networks, May 2018. URL http://arxiv.org/abs/1711.01297. arXiv:1711.01297 [cs, stat].

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond, April 2019. URL http://arxiv.org/abs/1904.09237. arXiv:1904.09237 [cs, math, stat].

Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1530–1538. PMLR, June 2015. URL https://proceedings.mlr.press/v37/rezende15.html. ISSN: 1938-7228.

Michael Schmidt and Hod Lipson. Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923):81–85, April 2009. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1165893. URL https://www.science.org/doi/10.1126/science.1165893.

Arno Solin, Ella Tamir, and Prakhar Verma. Scalable inference in sdes by direct matching of the fokker–planck–kolmogorov equation. *Advances in Neural Information Processing Systems*, 34: 417–429, 2021.

Luning Sun, Daniel Huang, Hao Sun, and Jian-Xun Wang. Bayesian spline learning for equation discovery of nonlinear dynamics with quantified uncertainty. *Advances in Neural Information Processing Systems*, 35:6927–6940, 2022.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 1 edition, April 2019. ISBN 978-1-108-18673-5 978-1-316-51008-7 978-1-316-64946-6. doi: 10.1017/9781108186735. URL https://www.cambridge.org/core/product/identifier/9781108186735/type/book.

Naoya Takeishi and Alexandros Kalousis. Physics-Integrated Variational Autoencoders for Robust and Interpretable Generative Modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 14809–14821. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7ca57a9f85a19a6e4b9a248c1daca185-Abstract.html.

Tapas Tripura and Souvik Chakraborty. A sparse Bayesian framework for discovering interpretable nonlinear stochastic dynamical systems with Gaussian white noise. *Mechanical Systems and Signal Processing*, 187:109939, March 2023. ISSN 0888-3270. doi: 10.1016/j.ymssp.2022.109939. URL https://www.sciencedirect.com/science/article/pii/S088832702201007X.

Yasen Wang, Huazhen Fang, Junyang Jin, Guijun Ma, Xin He, Xing Dai, Zuogong Yue, Cheng Cheng, Hai-Tao Zhang, Donglin Pu, Dongrui Wu, Ye Yuan, Jorge Gonçalves, Jürgen Kurths, and Han Ding. Data-Driven Discovery of Stochastic Differential Equations. *Engineering*, 17: 244–252, October 2022. ISSN 2095-8099. doi: 10.1016/j.eng.2022.02.007. URL https://www.sciencedirect.com/science/article/pii/S209580992200145X.

Yaniv Yacoby, Weiwei Pan, and Finale Doshi-Velez. Failure Modes of Variational Autoencoders and Their Effects on Downstream Tasks, March 2022. URL http://arxiv.org/abs/2007.07124. arXiv:2007.07124 [cs, stat].

Liu Yang, Dongkun Zhang, and George Em Karniadakis. Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 42 (1):A292–A317, January 2020. ISSN 1064-8275. doi: 10.1137/18M1225409. URL https://epubs.siam.org/doi/10.1137/18M1225409. Publisher: Society for Industrial and Applied Mathematics.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/c9efe5f26cd17ba6216bbe2a7d26d490-Abstract.html.

Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, November 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.07.048. URL https://www.sciencedirect.com/science/article/pii/S0021999119305340.

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. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2886528. Conference Name: IEEE Access.

Weiheng Zhong and Hadi Meidani. PI-VAE: Physics-Informed Variational Auto-Encoder for stochastic differential equations. *Computer Methods in Applied Mechanics and Engineering*, 403:115664, January 2023. ISSN 0045-7825. doi: 10.1016/j.cma.2022.115664. URL https://www.sciencedirect.com/science/article/pii/S0045782522006193.

APPENDIX A DERIVATION OF LOSS FUNCTION

We assume independence of $\mathbf{z}$ with respect to $\mathbf{x}$, such that $p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z})$. Then, as described in the Methods section, our generative model factorizes as follows (Bayes' rule):

$$p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x}) = p_\theta(\dot{\mathbf{x}}|\mathbf{z}, \mathbf{x})p_\theta(\mathbf{z}) \tag{12}$$

Given the chain rule for conditional probability, we also have:

$$\begin{aligned} p_\theta(\dot{\mathbf{x}}|\mathbf{z}, \mathbf{x}) &= \frac{p_\theta(\dot{\mathbf{x}}, \mathbf{z}, \mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} && \text{Conditional Probability} \\ &= \frac{p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})p_\theta(\dot{\mathbf{x}}|\mathbf{x})p_\theta(\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{x})} \\ &= \frac{p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})p_\theta(\dot{\mathbf{x}}|\mathbf{x})}{p_\theta(\mathbf{z})} && p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}) \end{aligned}$$

By substituting into the first factorization, we obtain the second factorization:

$$p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})p_\theta(\dot{\mathbf{x}}|\mathbf{x}) \tag{13}$$

We seek to learn a model that captures the dynamics $\dot{\mathbf{x}}$, given the state $\mathbf{x}$. Specifically, we seek to maximize the log-likelihood $\log p_\theta(\dot{\mathbf{x}}|\mathbf{x})$ by performing inference over $\mathbf{z}$. We follow a similar derivation as in (Kingma & Welling, 2019):

$$\begin{aligned} \log p_\theta(\dot{\mathbf{x}}|\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log p_\theta(\dot{\mathbf{x}}|\mathbf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log \frac{p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})}\right] && \text{see Eq. 13} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log \frac{p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x})q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})}{p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log \frac{p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})}\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})}{p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})}\right] \\ &= ELBO + D_{KL}(q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})||p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})) \end{aligned}$$

Since $D_{KL}(q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})||p_\theta(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})) \geq 0$, we maximize the $ELBO$, which lower bounds $\log p_\theta(\dot{\mathbf{x}}|\mathbf{x})$.

$$\begin{aligned} ELBO &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log \frac{p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log p_\theta(\dot{\mathbf{x}}, \mathbf{z}|\mathbf{x}) - \log q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log p_\theta(\dot{\mathbf{x}}|\mathbf{z}, \mathbf{x}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})\right] && \text{see Eq. 12} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\dot{\mathbf{x}},\mathbf{x})}\left[\log p_\theta(\dot{\mathbf{x}}|\mathbf{z}, \mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\dot{\mathbf{x}}, \mathbf{x})||p_\theta(\mathbf{z}))\right] \end{aligned}$$

Equivalently, we can minimize the $-ELBO$, given by the following loss:

$$loss = (\dot{\mathbf{x}} - f_{\hat{\mathbf{z}}}(\mathbf{x}))^2 + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})||p_\theta(\mathbf{z}))$$

Given our goal to learn a sparse set of governing equations, we need to train M, which is multiplied elementwise with Ξ_z. To do so, we add $L_0(M)$ (main text Eq. 5) to the loss, yielding:

$$\begin{aligned} loss &= (\dot{\mathbf{x}} - f_{\hat{\mathbf{z}}}(\mathbf{x}))^2 + \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})||p_\theta(\mathbf{z})) + \lambda L_0(M) \\ &= (\dot{\mathbf{x}} - f_{\hat{\mathbf{z}}}(\mathbf{x}))^2 + \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})||p_\theta(\mathbf{z})) + \lambda \sum_{j=1}^{k} \text{Sigmoid}(\log \alpha_j - \beta_{L_0} \log \frac{\gamma}{\zeta}) \end{aligned}$$

where β, λ, β_{L_0}, γ, and ζ are hyperparameters, k is the dimension of a vectorized M, and $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})$ using the reparameterization trick. $log\alpha_j$ are location parameters for the distribution that M is transformed from, as described in the Background section of the main text.

Table S1: Matrix shapes for different experiments

System	n	C	$\Theta(x)$	Ξ_z	M	$\Theta(x)(\Xi_z \odot M)$
Lorenz	3	F	250×19	$250 \times 19 \times 3$	$250 \times 19 \times 3$	250×3
Rössler	3	T	250×20	$250 \times 20 \times 3$	$250 \times 20 \times 3$	250×3
Lotka-Volterra	2	T	250×10	$250 \times 10 \times 2$	$250 \times 10 \times 2$	250×2
Lorenz-96	10	T	250×286	$250 \times 286 \times 10$	$250 \times 286 \times 10$	250×10

Table S2: Dataset initial conditions

System	Train	Test
Lorenz	(0, 1, 1.05)	(-1, 2, 0.5)
Rössler	(0, 1, 1.05)	(-1, 2, 0.5)
Lotka-Volterra	(4, 2)	(2.1, 1.0)
Lorenz-96	(8.01, 8, 8, 8, 8, 8, 8, 8, 8, 8)	(7.8, 8.7, 8.5, 6.0, 9.9, 9.5, 7.5, 6.9, 6.9, 8.7)

APPENDIX B GENERATIVE AND INFERENCE MODELS

Matrix dimensions are variable. Consider $x \in \mathbb{R}^n$ and a library with l terms. Then, we have:

$$\Theta(x) \in \mathbb{R}^l \quad \Xi_z \in \mathbb{R}^{l \times n} \quad M \in \mathbb{R}^{l \times n} \quad (\Xi_z \odot M) \in \mathbb{R}^{l \times n} \quad \Theta(x)(\Xi_z \odot M) \in \mathbb{R}^n$$

However, we use minibatches during training. Consider a batch of x of size b, meaning $x \in \mathbb{R}^{b \times n}$. Then, we have:

$$\Theta(x) \in \mathbb{R}^{b \times l} \quad \Xi_z \in \mathbb{R}^{b \times l \times n} \quad M \in \mathbb{R}^{b \times l \times n} \quad (\Xi_z \odot M) \in \mathbb{R}^{b \times l \times n} \quad \Theta(x)(\Xi_z \odot M) \in \mathbb{R}^{b \times n}$$

After training, we do not sample M using the reparameterization trick, since α has been learned. So, M has shape $l \times n$ (note that $k = l \cdot n$). For all experiments, we included polynomials up to order 3 in the library and used a batch size of 250 during training. Refer to Table $S1$ for a breakdown of matrix shapes for each experiment during training (note that C refers to whether a constant is included in the library).

APPENDIX C DATA

Each trajectory in the main manuscript was generated for 10000 timesteps with $\Delta t = 0.01$. Refer to Table S2 for data generation initial conditions. Note that the test initial condition for Lorenz-96 is rounded (the exact values can be found in the accompanying code, as we used Gaussian noise to choose the initial condition). Derivatives are estimated using finite differences without smoothing.

APPENDIX D TRAINING

D.1 ALGORITHM

A "best practice" for the HyperSINDy training algorithm is choosing low initial β and λ values and evaluating the results before adjusting in future runs. Moreover, we also utilize beta warmup Castrejon et al. (2019), which is useful for avoiding posterior collapse in VAEs. Specifically, we increase the β value from 0.01 to the chosen low initial β value over 100 epochs. If the prior did not learn the function well enough, we increased ("spiked") the β value at a later epoch in training to β_{spike}. Note that, although we knew the ground truth coefficients in our simulations, one can determine whether the prior learned "well enough" by comparing the similarity between the coefficients generated from the prior to the coefficients generated from the approximate posterior. See (Yacoby et al., 2022) for more information on this tradeoff between the posterior and prior. If the learned model was not sparse enough, we increased the λ value at a later epoch in training to λ_{spike}.

Every 100 epochs, we permanently set values in M to be zero if the corresponding coefficients (using the mean over a batch of coefficients) falls below the threshold value T. This is done using an auxiliary

matrix of shape $l \times n$, where the values are all initially set to 1 and then set to 0 throughout training if the corresponding value in M should be 0 permanently. This mask is multiplied elementwise with M to enforce this permanent sparsity. During training, we sample one M for each example in a minibatch of data using the reparameterization trick.

D.2 HYPERPARAMETERS

Hyperparamater tuning mostly consists of adjusting the β and λ value for the Kl divergence and L_0 terms in the loss function, respectively. Hyperparameters that stay constant for all experiments are listed here: $learning\_rate = 0.005$, $num\_hidden = 5$, $stat\_size = 250$, $batch\_size = 250$. $stat\_size$ refers to the number of coefficient matrices that are sampled from the prior to calculate the coefficient means used for the permanent thresholding described in the Training section. We used a hidden dimension of 64 in all neural networks for all experiments except on Lorenz-96, for which we used a hidden dimension of 128. We warm up to a low initial β value of 10 for every experiment. We use an initial L_0 regularization weight of $\lambda = 0.01$. For M, we use $\beta_{L_0} = 2/3$, $\gamma = -0.1$, and $\zeta = 1.1$. Note that an exhaustive list of hyperparameters and training settings can be found in the accompanying code. All experiments were run on an NVIDIA GeForce RTX 2080 Ti GPU. We ran all experiments in PyTorch (Paszke et al., 2019) using the AdamW optimizer (Loshchilov & Hutter, 2019) with a weight decay value of 0.01 and amsgrad (Reddi et al., 2019) enabled. Refer to Table S3 for a list of hyperparameters that we tuned to obtain the results in the main text. Refer to the attached code for more details on the RMSE experiments, as well as information on settings used to generate the supplemental figures.

APPENDIX E RMSE, PRECISION, AND RECALL METRICS

We use the following RMSE, precision, and recall metrics (as computed in Sun et al. (2022)):

$$
\begin{aligned}
rmse &= \frac{||\mathbf{C}_{True} - \mathbf{C}_{Pred}||_2}{||\mathbf{C}_{True}||_2} \\
precision &= \frac{||\mathbf{C}_{True} \odot \mathbf{C}_{Pred}||_0}{||\mathbf{C}_{Pred}||_0} \\
recall &= \frac{||\mathbf{C}_{True} \odot \mathbf{C}_{Pred}||_0}{||\mathbf{C}_{True}||_0}
\end{aligned}
$$

where $\mathbf{C}_{True}$ is the true mean or standard deviation of a given coefficient, and $\mathbf{C}_{Pred}$ is the mean or standard deviation of the predicted coefficients. For terms that are not included in the ground truth or predicted equations, we consider their mean or standard deviation to be zero.

APPENDIX F RDE-SDE TRANSFORMATION

Any finite dimensional SDE can be transformed into an RDE (Imkeller & Schmalfuss, 2001; Han & Kloeden, 2017). For an SDE with additive noise, the standard procedure involves replacement of the white noise with a stationary Ornstein Uhlenbeck process. Thus, following Han & Kloeden (2017) (section 3.5), the SDE

$$
\mathrm{d}X_t = f(X_t)\mathrm{d}t + \mathrm{d}W_t \tag{14}
$$

becomes

$$
\dot{Z}_t = f(Z_t + O_t) + O_t, \tag{15}
$$

where $Z_t \coloneqq X_t - O_t$ and O_t is the stationary Ornstein-Uhlenbeck process, which is the solution to the SDE $\mathrm{d}O_t = -O_t\mathrm{d}t + \mathrm{d}W_t$ (with W_t denoting the Wiener process).

In the multiplicative case, again following Han & Kloeden (2017), the SDE:

Table S3: Hyperparameters

Lorenz							
σ	d	β_{spike}	$epoch_{\beta_{spike}}$	λ_{spike}	$epoch_{\lambda_{spike}}$	$epochs$	T
1	6	100	400	10	400	999	0.05
5	6	400	400	10	400	999	0.05
10	6	400	400	10	400	999	0.05
Rössler							
σ	d	β_{spike}	$epoch_{\beta_{spike}}$	λ_{spike}	$epoch_{\lambda_{spike}}$	$epochs$	T
1	6	100	200	0.1	200	499	0.01
5	6	100	200	0.1	300	600	0.01
10	6	100	200	1	300	600	0.01
Lorenz RMSE							
σ	d	β_{spike}	$epoch_{\beta_{spike}}$	λ_{spike}	$epoch_{\lambda_{spike}}$	$epochs$	T
1	6	100	400	10	400	999	0.05
5	6	400	400	10	400	999	0.05
10	6	400	400	10	400	999	0.05
Rössler RMSE							
σ	d	β_{spike}	$epoch_{\beta_{spike}}$	λ_{spike}	$epoch_{\lambda_{spike}}$	$epochs$	T
1	6	100	200	0.1	200	499	0.01
5	6	200	200	0.1	300	600	0.01
10	6	300	200	1	300	600	0.01
Lotka-Volterra							
σ	d	β_{spike}	$epoch_{\beta_{spike}}$	λ_{spike}	$epoch_{\lambda_{spike}}$	$epochs$	T
N/A	4	None	None	0.1	100	250	0.1
Lorenz-96							
σ	d	β_{spike}	$epoch_{\beta_{spike}}$	λ_{spike}	$epoch_{\lambda_{spike}}$	$epochs$	T
10	20	None	None	10	400	999	0.05

$$\mathrm{d}X_t = f(t, X_t)\mathrm{d}t + b(t)X_t\mathrm{d}Wt \tag{16}$$

may be combined with the random transformation

$$z(t) = T(t)X_t, T(t) := \exp\left(\frac{1}{2}\int_0^t b^2(s)\,\mathrm{d}s - \int_0^t b(s)\,\mathrm{d}W_s\right) \tag{17}$$

to obtain the RDE:

$$\frac{\mathrm{d}z}{\mathrm{d}t} = T(t)f\left(t, T^{-1}(t)z(t)\right). \tag{18}$$

In general, explicit RDE-SDE transformations may not always be straightforward to implement (see Imkeller & Schmalfuss (2001); Caraballo & Han (2016); Han & Kloeden (2017) for extended discussion of this topic). The present manuscript does not seek to establish equivalence or explicit mappings between particular RDE and SDE expressions, nor are claims contingent on the ability to carry out this transformation. Rather, for our purposes, we simply note that the conjugacy between

the two formulations implies that results obtained within one framework are directly pertinent to the other Han & Kloeden (2017).

More generally, the chief motivation for stochastic equations (whether SDEs or RDEs) lies simply in the mathematical modeling of dynamics under uncertainty; for many (if not most) modeling applications, the precise physical nature of this uncertainty -– e.g., a diffusion process or fluctuating system parameters — is unknown and/or of secondary importance (Friedrich et al., 2011; Duan, 2015; Särkkä & Solin, 2019). From this perspective, SDEs and RDEs simply emerge as alternative modeling frameworks, each with unique practical (dis)advantages to be considered in context (e.g., see Bauer et al. (2017)). Nonetheless future work may examine the possibility of equipping our approach with an explicit SDE prior (e.g., (Solin et al., 2021)), thus strengthening theoretical connections to a broader stochastic dynamics literature.

APPENDIX G ALGORITHMS

Algorithm 1 Generation of Governing Equations Coefficients, Ξ_z, to predict $\dot{\mathbf{x}}$

if $\mathbf{z}$ not given **then**:
  $\mathbf{z} \sim \mathcal{N}(0, I)$ ▷ Generate batch of $\mathbf{z}$
$\Xi_{\mathbf{z}} = H(\mathbf{z})$ ▷ Generate 1 coefficient matrix for each $\mathbf{z}$ in batch
$\dot{\mathbf{x}} = f_{\mathbf{z}}(\mathbf{x}) = \Theta(\mathbf{x})(\Xi_{\mathbf{z}} \odot M)$ ▷ Unless training, uses Eq 4 to get M

Algorithm 2 Training Loop for Each Epoch

for each minibatch $\mathbf{x}, \dot{\mathbf{x}}$ **do**
  $\mu_q, \sigma_q = E(\mathbf{x}, \dot{\mathbf{x}})$ ▷ Encode each element of batch
  $\hat{\mathbf{z}} = \mu_q + \sigma_q \odot \epsilon$ ▷ Reparameterization Trick
  Obtain training M through transformations ▷ See Background, specifically Eq 3
  $\hat{\dot{\mathbf{x}}} = f_{\hat{\mathbf{z}}}(\mathbf{x}) =$ Algorithm 1($\hat{\mathbf{z}}$) ▷ Give $\hat{\mathbf{z}}$ to H
  $loss = (\dot{\mathbf{x}} - \hat{\dot{\mathbf{x}}})^2 + \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \dot{\mathbf{x}})||p_\theta(\mathbf{z})) + \lambda L_0(M)$
  Backprop $loss$ and update θ, ϕ
Sample batch of $\mathbf{z} \sim \mathcal{N}(0, 1)$
$\Xi_{\mathbf{z}} = H(\mathbf{z})$
$\Xi_{\mathbf{z}_{mean}}$ = mean over batch of $\Xi_{\mathbf{z}}$
if (epoch $\%$ threshold_interval) == 0 **then**: ▷ If we must threshold this epoch
  $M = 0$ permanently where $abs(\Xi_{\mathbf{z}_{mean}}) < threshold$ ▷ Permanently threshold M

APPENDIX H FURTHER SIMULATION DETAILS

To generate results for Figure 1 and Table S4, we ran separate experiments generating 10 trajectories (each with a different random seed, and each generated from a different initial condition) for each noise level of both systems. In total, we trained one HyperSINDy model and one E-SINDy model on each trajectory, yielding 30 HyperSINDy models and 30 E-SINDy models.

APPENDIX I FIGURES AND TABLES

We include here additional sample trajectories (all from the test initial condition) to highlight HyperSINDy's generative capabilities. Refer to Figure S1 for HyperSINDy trajectories generated for various noise levels on the Lorenz and Rössler system, and refer to Figure S2 for sample test trajectories. HyperSINDy captures the same dynamical behavior as the original system. Refer to Figure S3 for HyperSINDy trajectories generated for the Lorenz-96 system. Refer to Figure S5 for results on a Lotka-Voltera system simulated as an RDE with half-normal noise on the coefficients. We compare the distribution of discovered HyperSINDy coefficients and E-SINDy coefficients with the ground truth coefficients.

Furthermore, we include numerous tables of quantitative comparisons between HyperSINDy, E-SINDy, and Bayesian Spline Learning (BSL) (Sun et al., 2022). Refer to Table S4 for precision and recall comparisons between HyperSINDy and E-SINDy on the Lorenz and Rössler systems (the experimental setup in this comparison is analogous to that of 1). Refer to Table S5 for comparisons between HyperSINDy and E-SINDy on Lorenz-96 systems. Refer to Table S6 for comparisons between HyperSINDy, E-SINDy, and BSL on a Lotka-Voltera system simulated as a RDE (i.e. with random gaussian noise on each coefficient, which differs from Figure 3, which used an SDE formulation).

Note that, in our experiments, we simulate a total of three types of Lotka-Voltera systems. In Figure 3, we simulate the Lotka-Voltera system as an SDE, using equation 10. As noted in the main text, we use $dt = 0.01$. In Figure S5, we simulate the Lotka-Voltera system as an RDE:

$$\dot{x} = \alpha_1 x - \alpha_2 xy \qquad \dot{y} = -\beta_1 y + \beta_2 xy \tag{19}$$

At each timestep in the simulation, we draw samples of the coefficients from $HalfNormal$ (HN) distributions:

$$\{\alpha_1, \alpha_2, \beta_1, \beta_2\} \sim \{HN(1,5), HN(1,5), HN(1,5), HN(1,5)\}$$

Note that here, we use $dt = 0.005$. In Table S6, we again simulate the Lotka-Voltera system as an RDE using equation 19. However at each timestep in the simulation, we draw samples of the coefficients from Gaussian distributions:

$$\{\alpha_1, \alpha_2, \beta_1, \beta_2\} \sim \{\mathcal{N}(1,\sigma), \mathcal{N}(1,\sigma), \mathcal{N}(1,\sigma), \mathcal{N}(1,\sigma)\}$$

Here, we also use $dt = 0.005$.

Moreover, we attempted to fairly tune hyperparameters when training E-SINDy and BSL, using only the best model we could produce for comparisons. In the case of BSL, we tested out the following hyperparameters (refer to the publicly available BSL code): $lam = \{0.001, 0.0001, 0.000001\}, eta = \{0.05, 0.01, 0.025, 0.005, 0.0005\}, ADOlearningrate = \{0.05, 0.005\}$. Note that, for E-SINDy, HyperSINDy, and BSL, the strength of the sparsity parameter can significantly impact precision and recall, as it helps determines which terms get thresholded out.

Table S4: Total Term Precision and Recall relative to ground truth equations

		Lorenz		Rössler	
STD		HyperSINDy	E-SINDy	HyperSINDy	E-SINDy
1	Precision	**0.9857** $\pm$ 0.0452	0.6045 $\pm$ 0.0274	**0.9375** $\pm$ 0.0884	0.7534 $\pm$ 0.0850
	Recall	0.8714 $\pm$ 0.0452	**1.0000** $\pm$ 0.0000	0.9571 $\pm$ 0.0690	**1.0000** $\pm$ 0.0000
5	Precision	**0.9875** $\pm$ 0.0395	0.4588 $\pm$ 0.0468	**0.9250** $\pm$ 0.0645	0.5502 $\pm$ 0.0456
	Recall	0.9429 $\pm$ 0.0738	**1.0000** $\pm$ 0.0000	0.9857 $\pm$ 0.0452	**1.0000** $\pm$ 0.0000
10	Precision	**0.9875** $\pm$ 0.0395	0.6491 $\pm$ 0.0268	**0.9260** $\pm$ 0.0829	0.4846 $\pm$ 0.0699
	Recall	0.9857 $\pm$ 0.0452	**1.0000** $\pm$ 0.0000	0.9571 $\pm$ 0.0690	**1.0000** $\pm$ 0.0000

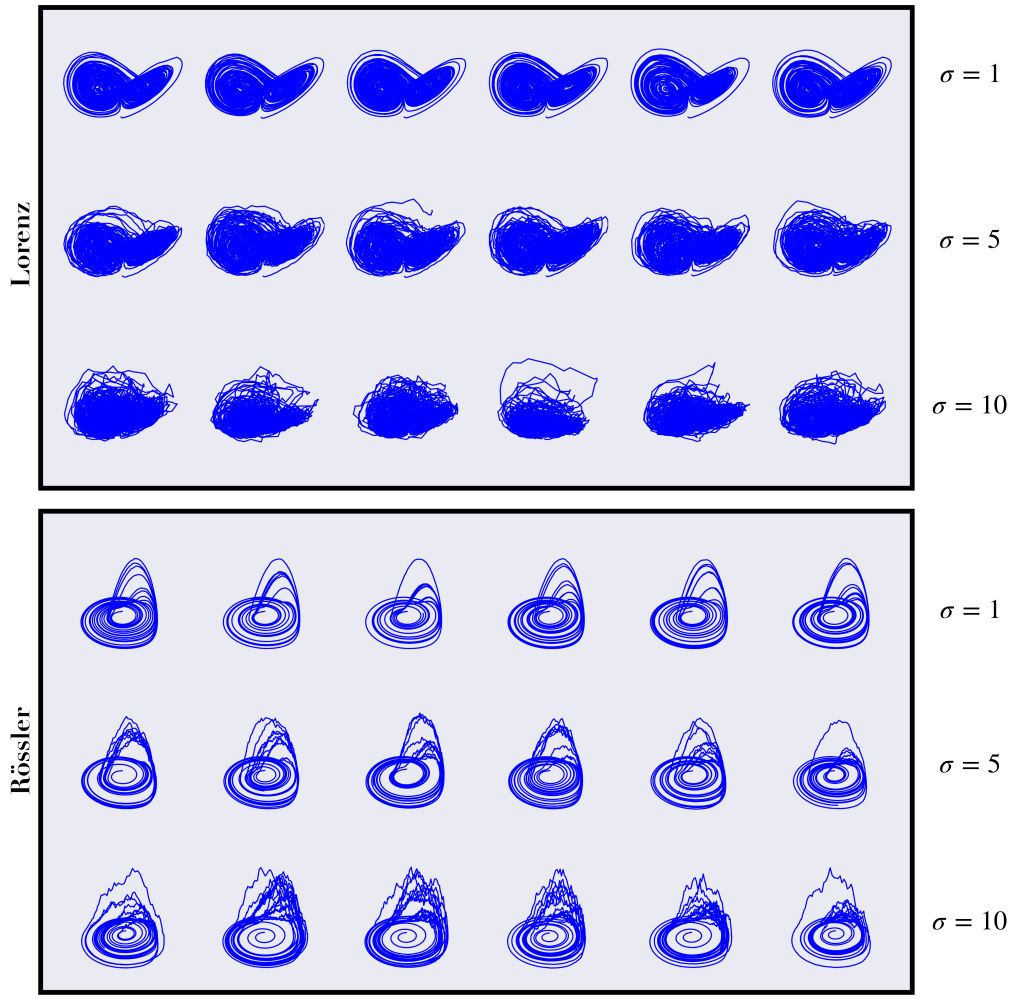


Figure S1: **Additional 3D Stochastic Lorenz and Rössler Samples**. HyperSINDy models trained on trajectories of varying noise (σ). Blue trajectories are generated by iteratively sampling from HyperSINDy's learned generative model.

Table S5: Lorenz-96

Experiment	Method	RMSE Mean	RMSE STD	Precision	Recall
$\sigma = 0$	HyperSINDy	0.05227	N/A	1.0	1.0
	E-SINDy	**0.006756**	N/A	1.0	1.0
$\sigma = 5$	HyperSINDy	**0.05370**	**0.7375**	**1.0**	1.0
	E-SINDy	0.1591	0.8240	0.4348	1.0
$\sigma = 10$	HyperSINDy	**0.1106**	**0.2117**	**1.0**	1.0
	E-SINDy	0.1729	0.8544	0.3077	1.0

We simulate the Lorenz-96 with varying levels of Gaussian noise (σ) on the forcing term. We report the RMSE between the mean and standard deviation of discovered coefficients, as compared to ground truth; we also report precision and recall of the terms. Note that for $\sigma = 0$, we cannot report the RMSE of the learned STD, as the ground truth standard deviation is 0.

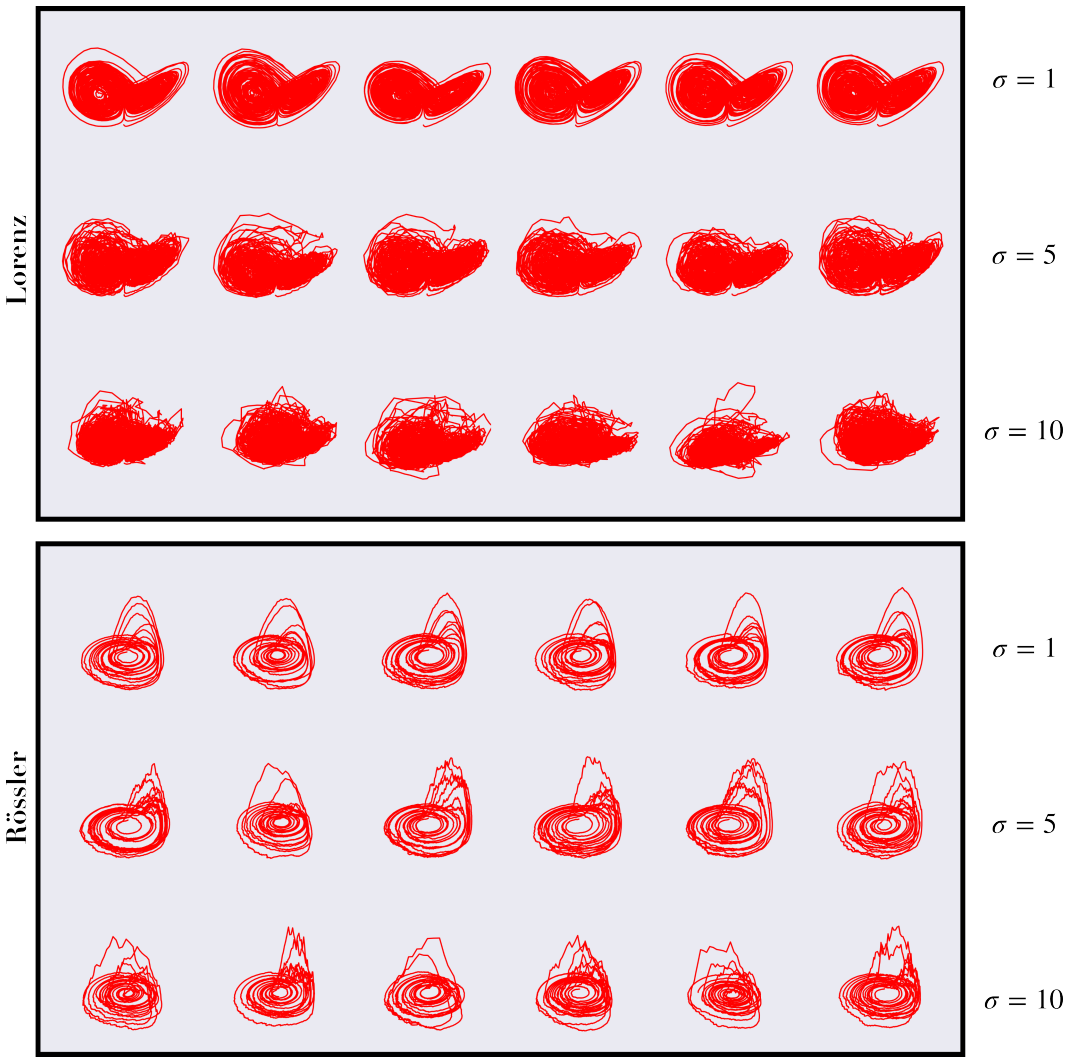


Figure S2: **3D Stochastic Lorenz and Rössler Ground Truth Samples**. Samples generated using the ground truth equations for varying noise levels (σ).

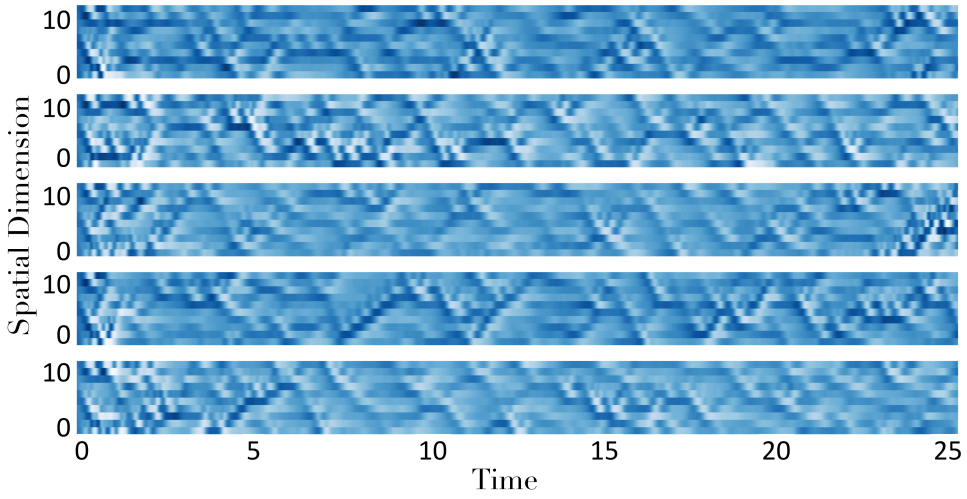


Figure S3: **Lorenz-96 Samples** ($\sigma = 10$). Each row contains a different sample trajectory. Trajectories are generated by iteratively sampling from HyperSINDy's learned generative model.

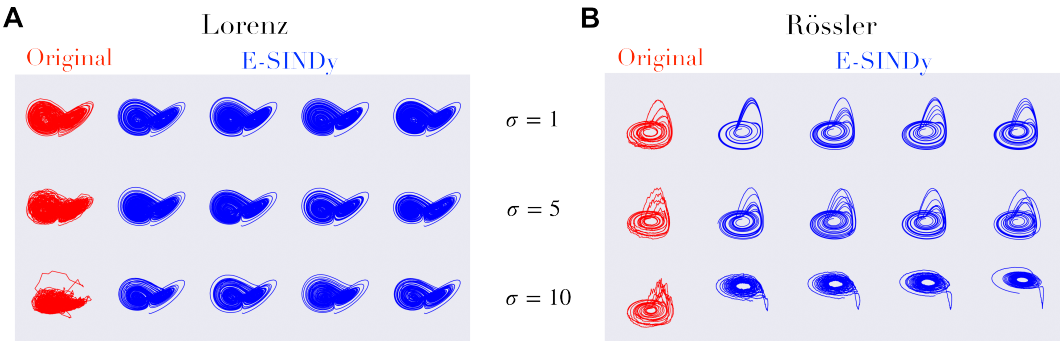


Figure S4: **E-SINDY 3D Stochastic Lorenz and Rössler Samples**. Samples generated using discovered E-SINDy equations for varying noise levels (σ).

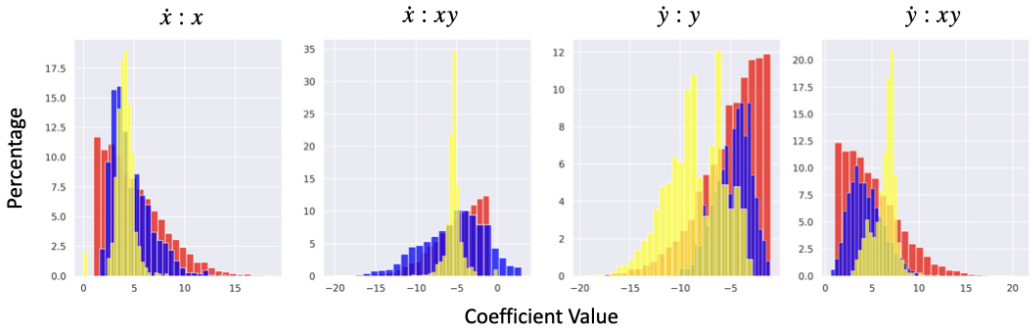


Figure S5: **Lotka-Volterra System with Half-Normal Noise** ($\sigma = 5$). Comparison of ground truth (red), HyperSINDy (blue) and E-SINDy (yellow) coefficient distributions for each of the dynamical terms. We simulate this system using an RDE formulation with half-normal noise (with the given σ) on each coefficient.

Table S6: Lotka-Volterra system with Gaussian noise on every coefficient.

Experiment	Method	RMSE Mean	Precision	Recall
$\sigma = 0$	HyperSINDy	0.0028	1.0	1.0
	E-SINDy	**0.0025**	1.0	1.0
	Bayesian Spline	0.1870	1.0	1.0
$\sigma = 1$	HyperSINDy	**0.0415**	**1.0**	1.0
	E-SINDy	0.1763	0.5714	1.0
	Bayesian Spline	0.2880	0.6667	1.0
$\sigma = 2$	HyperSINDy	**0.0902**	**1.0**	1.0
	E-SINDy	0.9480	0.2857	1.0
	Bayesian Spline	0.4665	0.4444	1.0
$\sigma = 3$	HyperSINDy	**0.1694**	**0.8000**	1.0
	E-SINDy	1.4953	0.3077	1.0
	Bayesian Spline	0.7577	0.4	1.0

We simulate the Lotka-Volterra system as an RDE, i.e. with Gaussian noise (with the given σ) on each coefficient. For each σ, we train a *HyperSINDy* model with the given z dimension, then evaluate the RMSE of the mean and standard deviation of the discovered coefficients, as compared to the ground truth mean and standard deviation.

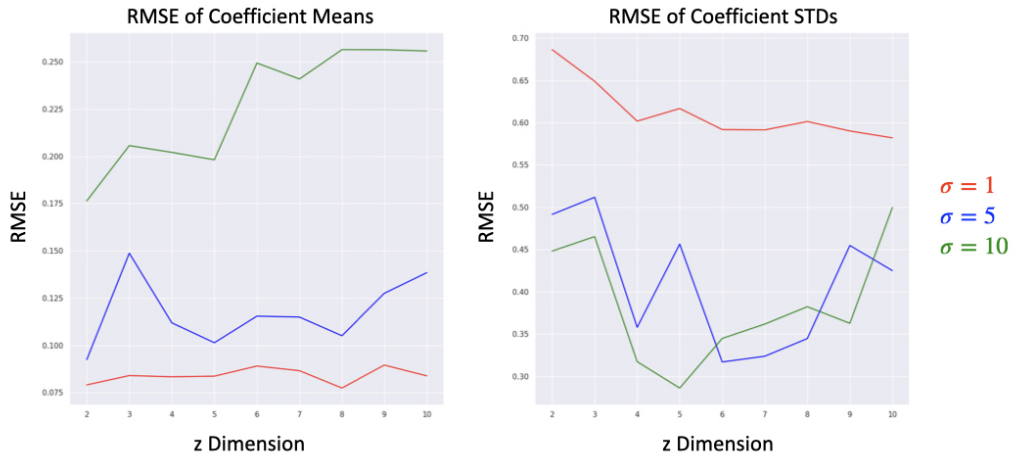


Figure S6: **3D Stochastic Lorenz with varying z dimension**. We simulate the 3D Lorenz system, as in 2, then train HyperSINDy models with different z dimension on each trajectory. Here, we plot the RMSE of the discovered mean and standard deviation of coefficients, as compared to the ground truth. Note that, for each σ, even though the RMSE can vary significantly for different z dimensions, it is still always lower than E-SINDy (see Table 1).