A unified sparse optimization framework to learn parsimonious physics-informed models from data

Kathleen Champion[1*], Peng Zheng[1], Aleksandr Y. Aravkin[1], Steven L. Brunton[2,1], J. Nathan Kutz[1]

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

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

Abstract

Machine learning (ML) is redefining what is possible in data-intensive fields of science and engineering. However, applying ML to problems in the physical sciences comes with a unique set of challenges: scientists want physically interpretable models that can (i) generalize to predict previously unobserved behaviors, (ii) provide effective forecasting predictions (extrapolation), and (iii) be certifiable. Autonomous systems will necessarily interact with changing and uncertain environments, motivating the need for models that can accurately extrapolate based on physical principles (e.g. Newton's universal second law for classical mechanics, $F = ma$). Standard ML approaches have shown impressive performance for predicting dynamics in an interpolatory regime, but the resulting models often lack interpretability and fail to generalize.

In this paper, we introduce a unified sparse optimization framework that learns governing dynamical systems models from data, selecting relevant terms in the dynamics from a library of possible functions. The resulting models are parsimonious, have physical interpretations, and can generalize to new parameter regimes. Our framework allows the use of non-convex sparsity promoting regularization functions and can be adapted to address key challenges in scientific problems and data sets, including outliers, parametric dependencies, and physical constraints. We show that the approach discovers parsimonious dynamical models on several example systems, including a spiking neuron model. This flexible approach can be tailored to the unique challenges associated with a wide range of applications and data sets, providing a powerful ML-based framework for learning governing models for physical systems from data.

1 Introduction

With abundant data being generated across scientific fields, researchers are increasingly turning to machine learning (ML) methods to aid scientific inquiry. In addition to standard techniques in clustering and classification, ML is now being used to discover models that characterize and predict the behavior of physical systems. Unlike many applications in ML, *interpretation*, *generalization* and *extrapolation* are the primary objectives for engineering and science, hence we must identify *parsimonious* models that have the fewest terms required to describe the dynamics. This is in contrast to neural networks (NNs), which are defined by exceedingly large parametrizations which typically lack interpretability or generalizability. A breakthrough approach in model discovery used symbolic regression to learn the form of governing equations from data [4, 30]. Sparse identification of nonlinear dynamics (SINDy) [5] is a related approach that uses sparse regression to find the fewest terms in a library of candidate functions required to model the dynamics. 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 (e.g., conservation of mass, momentum, and energy) [16]. In this work, we develop

* Corresponding author (kpchamp@uw.edu).

a unified sparse optimization framework for dynamical system discovery that enables one to simultaneously discover models, trim corrupt training data, enforce known physics, and identify parametric dependency in the equations.

Although studied for decades in dynamical systems [9, 21], the universal approximation capabilities of NNs [10, 12] have generated a resurgence of approaches to model time-series data [2, 17, 19, 23, 25, 26, 32, 35, 36, 39]. NNs can also learn coordinate transformations that simplify the dynamical system representation [17, 19, 22, 32, 36, 39] (e.g. Koopman representations [14, 20]). However, NNs generally struggle with extrapolation, are difficult to interpret, and cannot readily enforce known or partially known physics.

In contrast, the SINDy algorithm [5] has been shown to produce interpretable and generalizable dynamical systems models from limited data. SINDy has been applied broadly to identify models for optical systems [31], fluid flows [16], chemical reaction dynamics [11], plasma convection [6], structural modeling [15], and for model predictive control [13]. It is also possible to extend SINDy to identify partial differential equations [28, 29], to trim corrupt data [34], and to incorporate partially known physics and constraints [16]. Because the approach is fundamentally based on a sparsity-regularized regression, there is an opportunity to unify these innovations via the sparse relaxed regularized regression (SR3) [40], resulting in a unified sparse model discovery framework.

1.1 Basic problem formulation

The sparse identification of nonlinear dynamics (SINDy) method [5] enables the discovery of nonlinear dynamical systems models from data. Assume we have data from a dynamical system

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

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state of the system at time t. We want to find the terms in $\mathbf{f}$ given the assumption that $\mathbf{f}$ has only a few active terms: it is sparse in the space of all possible functions of $\mathbf{x}(t)$. Given snapshot data $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m\end{bmatrix}^T$ and $\dot{\mathbf{X}} = \begin{bmatrix}\dot{\mathbf{x}}_1 & \dot{\mathbf{x}}_2 & \cdots & \dot{\mathbf{x}}_m\end{bmatrix}^T$, we build a library of candidate functions $\boldsymbol{\Theta}(\mathbf{X}) = [\boldsymbol{\theta}_1(\mathbf{X}) \cdots \boldsymbol{\theta}_p(\mathbf{X})]$. We then seek a solution of

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}. \tag{2}$$

where $\boldsymbol{\Xi} = (\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \cdots\ \boldsymbol{\xi}_n)$ are sparse coefficient (loading) vectors. A natural optimization is given by

$$\min_{\boldsymbol{\Xi}} \frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|^2 + \lambda R(\boldsymbol{\Xi}) \tag{3}$$

where $R(\cdot)$ is a regularizer that promotes sparsity. When R is convex, a range of well-known algorithms for (3) are available. The standard approach is to choose R to be the sparsity-promoting ℓ_1 norm, which is the convex relaxation of ℓ_0 norm. In this case, SINDy is solved via LASSO [33]. In practice, LASSO does not perform well at coefficient selection (see Section 2.1 for details). In the context of dynamics discovery we would like to use non-convex R, specifically the ℓ_0 norm.

The standard SINDy algorithm performs sequential thresholded least squares (STLSQ): given a parameter η that specifies the minimum magnitude for a coefficient in $\boldsymbol{\Xi}$, perform a least squares fit and then zero out all coefficients with magnitude below the threshold. This process of fitting and thresholding is performed until convergence. While this method works remarkably well, it is customized to the least squares formulation and does not readily accommodate extensions including incorporation of additional constraints, robust formulations, or nonlinear parameter estimation. A number of extensions to SINDy have been developed and required adaptations to the optimization algorithm [13, 16, 28, 29, 34].

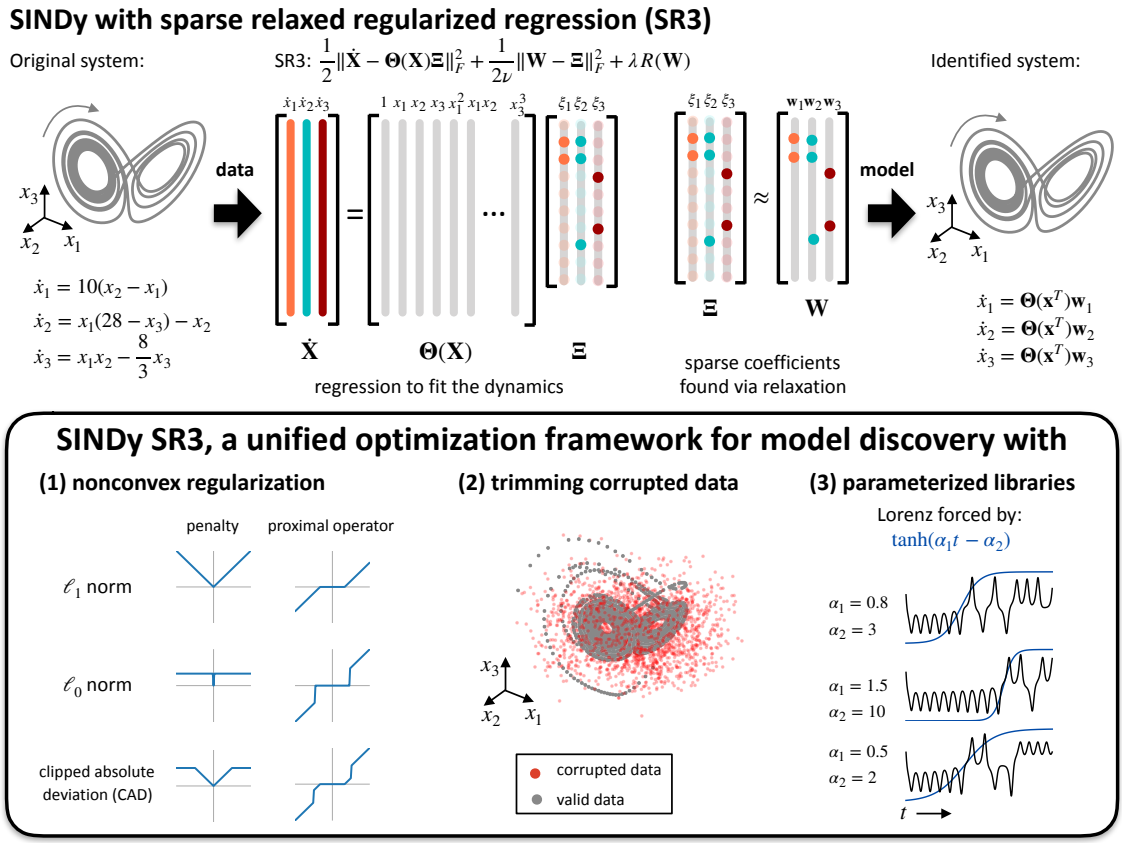

Figure 1: Overview of the SINDy method for identifying nonlinear dynamical systems. SINDy sets up the system identification problem as a sparse regression problem, selecting a set of active governing terms from a library. Sparse relaxed regularized regression (SR3) provides a flexible, unified framework that can be adapted to address a number of challenges that might occur with data from physical systems, including outlier identification, parameterized library functions, and forcing.

2 Formulation and approach

We extend (3) to include additional structure, robustness to outliers, and nonlinear parameter estimation using the sparse relaxed regularized regression (SR3) approach that uses relaxation and partial minimization [40]. SR3 for (3) introduces the auxiliary variable $\mathbf{W}$ and relaxes the optimization to

$$\min_{\mathbf{\Xi},\mathbf{W}} \frac{1}{2}\|\dot{\mathbf{X}} - \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi}\|^2 + \lambda R(\mathbf{W}) + \frac{1}{2\nu}\|\mathbf{\Xi} - \mathbf{W}\|^2. \tag{4}$$

We can solve (4) using the alternating update rule in Algorithm 1. This requires only least squares solves and prox operators [40]. The resulting solution approximates the original problem (3) as $\nu \downarrow 0$. When R is taken to be the ℓ_0 penalty, the prox operator is hard thresholding, and Algorithm 1 is similar, but not equivalent, to thresholded least squares, and performs similarly in practice. However, unlike thresholded least squares, the SR3 approach easily generalizes to new problems and features.

Algorithm 1 Basic SR3 Algorithm

Input ϵ, $\mathbf{W}^0$
Initialize $k = 0$, err$= 2\epsilon$.
while err$> \epsilon$ **do**
  $k \leftarrow k + 1$
  $\boldsymbol{\Xi}^k = \underset{\boldsymbol{\Xi}}{\operatorname{argmin}} \frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|^2 + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}^{k-1}\|^2$
  $\mathbf{W}^k = \operatorname{prox}_{\lambda\nu R}(\boldsymbol{\Xi}^k)$
  err $= \|\mathbf{W}^k - \mathbf{W}^{k-1}\|/\nu$
end while

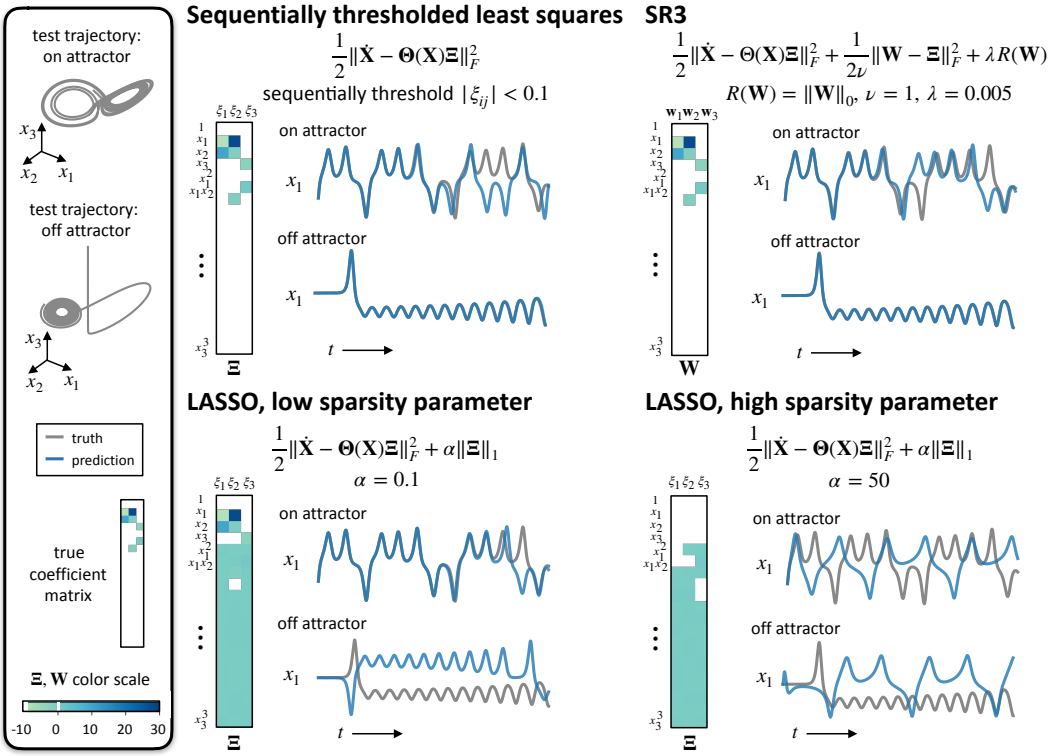

Figure 2: Comparison of optimization methods for identifying the active coefficients in a SINDy model. The standard approach has been STLSQ, which is able to identify a sparse model that fits the data well. However, this approach lacks flexibility and is not easily adapted to incorporate other optimization challenges. LASSO is a standard approach for performing a sparse regression, but does not do well at performing coefficient selection: many of the terms in the coefficient matrix are small but nonzero. Increasing the regularization strength leads to a model that is still not sparse and has a poor fit of the data. SR3 relaxes the regression problem in a way that enables the use of nonconvex regularization functions such as the ℓ_0 norm or hard thresholding. This results in a truly sparse model, and provides a flexible framework that can easily incorporate additional optimizations such as trimming outliers and fitting parameterized library functions.

2.1 Performance of SR3 for SINDy

SR3 for SINDy provides an optimization framework that both (1) enables the identification of truly sparse models and (2) can be adapted to include additional features. We first compare SR3 to both STLSQ and the LASSO algorithm. While STLSQ works well for identifying sparse models

that capture the behavior of a system, it is a standalone method without a true optimization cost function, meaning the algorithm must be reformulated to work with other adaptations to the SINDy problem [18]. LASSO provides a standard optimization approach but does not successfully identify sparse models. Even with clean data, LASSO models for SINDy typically have many coefficients that are small in magnitude but nonzero. Obtaining a sparse set of coefficients is key for interpretability. SR3 works with nonconvex regularization functions such as the ℓ_0 norm, enabling the identification of truly sparse models.

In Fig. 2 we compare these algorithms using data from the canonical chaotic Lorenz system:

$$\dot{x}_1 = 10(x_2 - x_1), \quad \dot{x}_2 = x_1(28 - x_3) - x_2, \quad \dot{x}_3 = x_1 x_2 - (8/3)x_3. \tag{5}$$

We simulate the system from 20 initial conditions and fit a SINDy model with polynomials up to order 3 using the following optimization approaches: STLSQ with threshold 0.1, SR3 with ℓ_0 regularization, LASSO with a regularization weight of 0.1, and LASSO with a regularization weight of 50. For each model we analyze (1) the sparsity pattern of the coefficient matrix and (2) simulations of the resulting model on test trajectories. As shown in Figure 2, STLSQ and SR3 yield the same correct sparsity pattern. In simulation, both track a Lorenz test trajectory for several trips around the attractor before eventually falling off. The eventual deviation is expected due to the chaotic nature of the Lorenz system, as a slight difference in coefficient values or initial conditions can lead to vastly different trajectories (although the trajectories continue to fill in the Lorenz attractor). These models also track the behavior well for a trajectory that starts off the attractor. The LASSO models both have many terms that are small in magnitude but still nonzero. As the regularization penalty is increased, rather than removing the unimportant terms in the dynamics the method removes many of the true coefficients in the Lorenz model. The LASSO model with heavy regularization has a very poor fit for the dynamics, as seen via simulation. While the LASSO model with less regularization provides a good fit for the dynamics on the attractor, it does not generalize off the attractor.

3 Simultaneous Sparse Inference and Data Trimming

Many real world data sets contain corrupted data and/or outliers, which is problematic for model identification methods. For SINDy, outliers can be especially problematic, as derivative computations are corrupted. Many data modeling methods have been adapted to deal with corrupted data, resulting in "robust" versions of the methods (such as robust PCA). The SR3 algorithm for SINDy can be adapted to incorporate trimming of outliers, providing a robust optimization algorithm for SINDy. Starting with least trimmed squares [27], extended formulations that simultaneously fit models and trim outliers are widely used in statistical learning. Trimming has proven particularly useful in the high-dimensional setting when used with the LASSO approach and its extensions [37, 38].

The high-dimensional trimming extension applied to (3) takes the form

$$\min_{\boldsymbol{\Xi},\mathbf{v}} \sum_{i=1}^{m} \frac{1}{2} v_i \|(\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi})_i\|^2 + \lambda R(\boldsymbol{\Xi}) \quad \text{s.t.} \quad 0 \le v_i \le 1, \quad \mathbf{1}^T\mathbf{v} = h, \tag{6}$$

where h is an estimate of the number of 'inliers' out of the potential m rows of the system. The set $\Delta_h := \{\mathbf{v} : 0 \le v_i \le 1, \quad \mathbf{1}^T\mathbf{v} = h\}$ is known as the capped simplex. Current algorithms for (6), such as those of [38], rely on LASSO formulations and thus have significant limitations (see

Algorithm 2 SR3-Trimming Algorithm

Input ϵ, β, $\mathbf{W}^0$, $\mathbf{v}^0$
Initialize $k = 0$, err$= 2\epsilon$.
while err$> \epsilon$ **do**
$k \leftarrow k + 1$
$\boldsymbol{\Xi}^k = \underset{\boldsymbol{\Xi}}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{2} v_i \|(\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi})_i\|^2 + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}^{k-1}\|^2$
$\mathbf{W}^k = \operatorname{prox}_{\lambda\nu R}(\boldsymbol{\Xi}^k)$
$\mathbf{v}^k = \operatorname{proj}_{\Delta_h}\left(\mathbf{v}^{k-1} - \beta \mathbf{g}_{\mathbf{v}}\right), \quad (\mathbf{g}_{\mathbf{v}})_i = \|(\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}^k)_i\|^2$
err $= \|\mathbf{W}^k - \mathbf{W}^{k-1}\|/\nu + \|\mathbf{v}^k - \mathbf{v}^{k-1}\|/\beta$
end while

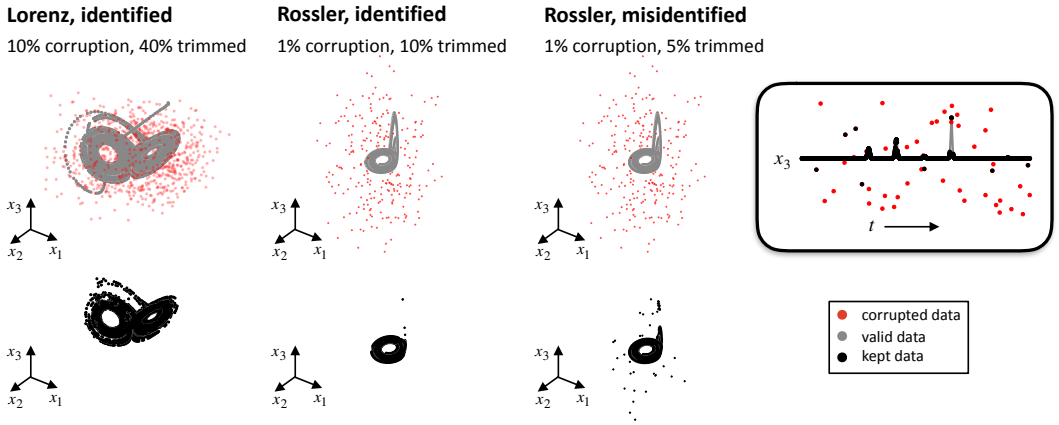

Figure 3: Demonstration of the trimming problem for the Lorenz and Rossler systems. For each system, we corrupt some subset of the data (corrupted values shown in red, valid data values shown in gray). We then apply SINDy SR3 with trimming. The black data points show the data that is left after trimming. For the Lorenz system, only data that is on the attractor remains and the system is correctly identified. For the Rossler system, the trimming algorithm also trims points from the portion of the attractor in the x_3 plane. The system is still correctly identified, but more data must be trimmed.

previous section). Here, we use the SR3 strategy (4) to extend to the trimmed SINDy problem (6):

$$\min_{\boldsymbol{\Xi},\mathbf{W},\mathbf{v}\in\Delta_h} \sum_{i=1}^{m} \frac{1}{2} v_i \|(\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi})_i\|^2 + \lambda R(\mathbf{W}) + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}\|^2. \tag{7}$$

We then use the alternating Algorithm 2 to solve the problem. The step size β is completely up to the user, as discussed in the convergence theory (see Supplementary Materials). The trimming algorithm requires specifying how many samples should be trimmed, which can be chosen by estimating the level of corruption in the data. Estimating derivatives using central differences, for instance, makes derivative estimates on either side of the original corrupted sample corrupt as well, meaning that three times as many samples as were originally corrupted will be bad. Thus trimming will need to be more than the initial estimate of how many samples were corrupted. Trimming ultimately can help identify and remove points with bad derivative estimates, leading to a better SINDy model fit.

3.1 Example: Lorenz

We demonstrate the use of SINDy SR3 for trimming outliers on data from the Lorenz system (5). We randomly select a subset of samples to corrupt, adding a high level of noise to these samples to create outliers. We apply the SINDy SR3 algorithm with trimming to simultaneously remove the corrupted samples and fit a SINDy model. Figure 3 shows the results of trimming on a dataset with 10% of the samples corrupted. The valid data points are shown in gray and the corrupt data points are highlighted in red. As derivatives are calculated directly from the data using central differences, this results in closer to 30% corruption (as derivative estimates on either side of each corrupt sample will also be corrupted). We find that the algorithm converges on the correct solution more often when a higher level of trimming is specified: in other words, it is better to remove some clean data along with all of the outliers than to risk leaving some outliers in the data set. Accordingly, we set our algorithm to trim 40% of the data. Despite the large fraction of corrupted samples, the method is consistently able to identify the Lorenz model (or a model with only 1-2 extra coefficients) from the remaining data in repeated simulations.

3.2 Example: Rossler

The Rossler system

$$\dot{x}_1 = -x_2 - x_3, \quad \dot{x}_2 = x_1 + 0.1x_2, \quad \dot{x}_3 = 0.1 + x_3(x_1 - 14). \tag{8}$$

exhibits chaotic behavior characterized by regular orbits around an attractor in the x_1, x_2 plane combined with occasional excursions into the x_3 plane. The Rossler attractor is plotted in Fig. 3 with 1% of samples corrupted (highlighted in red). While the excursions into the x_3 dimension occur consistently as a part of the Rossler dynamics, the fact that the majority of the attractor lies in the x_1, x_2 plane means that these excursions can be seen as outliers in the dynamics. The algorithm trims these events along with the corrupted samples, and therefore a higher percentage of the data must be trimmed to ensure outliers are not missed. Figure 3 shows the results of trimming when the outliers are all removed and the system is correctly identified (center panel, 10% trimmed) and when there is not enough trimming and the system is misidentified (right panel, 5% trimmed). We see that in the under-trimmed case, a significant portion of the attractor in the x_3 plane is removed whereas many of the corrupted samples are missed. In the case where the system is properly identified, the x_3 portion of the attractor is mostly removed but the system is still correctly identified.

4 Parameterized library functions

In standard examples of SINDy, the library is chosen to contain polynomials, which make a natural basis for many models in the physical sciences. However, many systems of interest may include more complicated terms in the dynamics, such as exponentials or trigonometric functions, that include parameters that contribute nonlinearly to the fitting problem. In addition to parameterized basis functions, systems may be subject to parameterized external forcing: for example, periodic forcing where the exact frequency of the forcing is unknown. SINDy with unknown parameters is given by

$$\min_{\boldsymbol{\Xi},\boldsymbol{\alpha}} \frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X},\boldsymbol{\alpha})\boldsymbol{\Xi}\|^2 + \lambda R(\boldsymbol{\Xi}). \tag{9}$$

This is a regularized nonlinear least squares problem. The SR3 approach makes it possible to devise an efficient algorithm for this problem as well. The relaxed formulation is given by

Algorithm 3 SR3-Parameter Estimation

Input ϵ, $\mathbf{W}^0$, $\boldsymbol{\alpha}^0$
Initialize $k = 0$, err$= 2\epsilon$.
while err$> \epsilon$ **do**
$k \leftarrow k + 1$
$\boldsymbol{\Xi}^k = \underset{\boldsymbol{\Xi}}{\operatorname{argmin}} \frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X}, \boldsymbol{\alpha}^{k-1})\boldsymbol{\Xi}\|^2 + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}^{k-1}\|^2$
$\mathbf{W}^k = \operatorname{prox}_{\lambda\nu R}(\boldsymbol{\Xi}^k)$
$\mathbf{g}_{\boldsymbol{\alpha}}^k = \nabla_{\boldsymbol{\alpha}} \left(\frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X}, \boldsymbol{\alpha})\boldsymbol{\Xi}^k\|^2\right)$, $\quad \mathbf{H}_{\boldsymbol{\alpha}}^k = \nabla_{\boldsymbol{\alpha}}^2 \left(\frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X}, \boldsymbol{\alpha})\boldsymbol{\Xi}^k\|^2\right)$,
$\boldsymbol{\alpha}^k = \boldsymbol{\alpha}^{k-1} - (\mathbf{H}_{\boldsymbol{\alpha}}^k)^{-1}\mathbf{g}_{\boldsymbol{\alpha}}^k$
err $= \|\mathbf{W}^k - \mathbf{W}^{k-1}\|/\nu + \|\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}\|$
end while

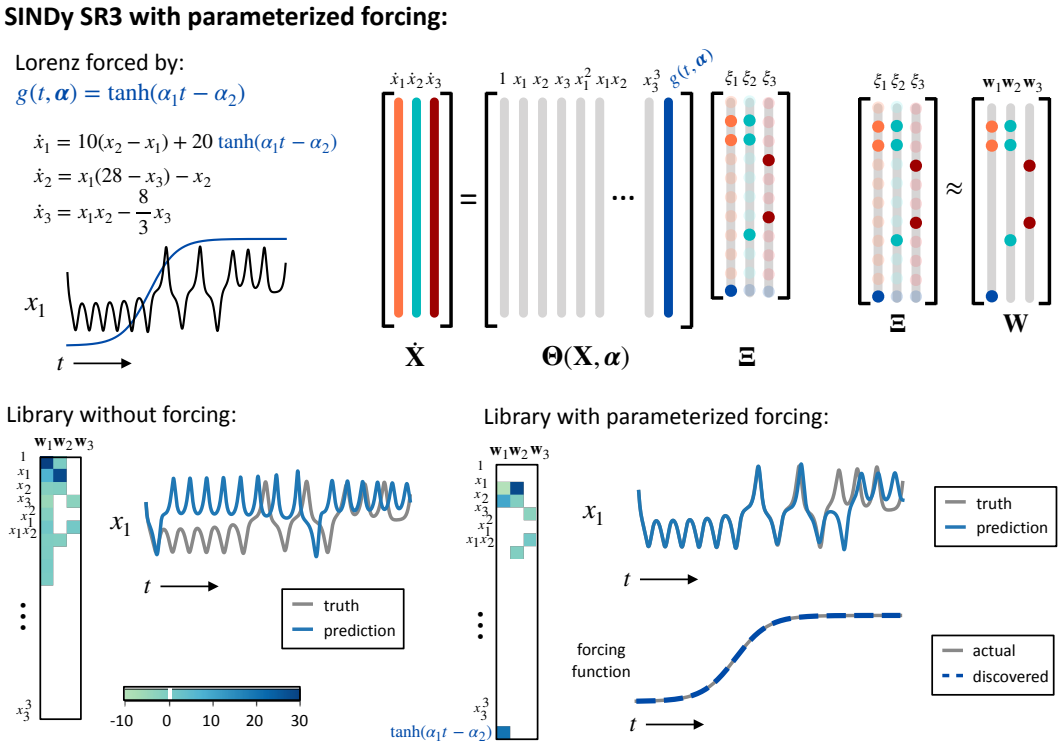

Figure 4: Depiction of SINDy SR3 with parameterized library terms, using the example of the Lorenz system forced by a hyperbolic tangent. The library includes a parameterized forcing term and a joint optimization is performed to find the parameter $\boldsymbol{\alpha}$ along with the SINDy model. Without the forcing term, a sparse model is not identified and the resulting model does not reproduce the behavior in simulation. With parameterized forcing in the library, both the forcing parameters and the library can be correctly identified given a sufficiently close initialization of the parameters $\boldsymbol{\alpha}$.

$$\min_{\boldsymbol{\Xi}, \mathbf{W}, \boldsymbol{\alpha}} \frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X}, \boldsymbol{\alpha})\boldsymbol{\Xi}\|^2 + \lambda R(\mathbf{W}) + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}\|^2. \tag{10}$$

We solve (10) using Algorithm 3. The $\boldsymbol{\alpha}$ variable is updated using a true Newton step, where the gradient and Hessian are computed using algorithmic differentiation.

The joint optimization for the parameterized case yields a nonconvex problem with potentially many local minima depending on the initial choice of the parameter(s) $\boldsymbol{\alpha}$. This makes it essential

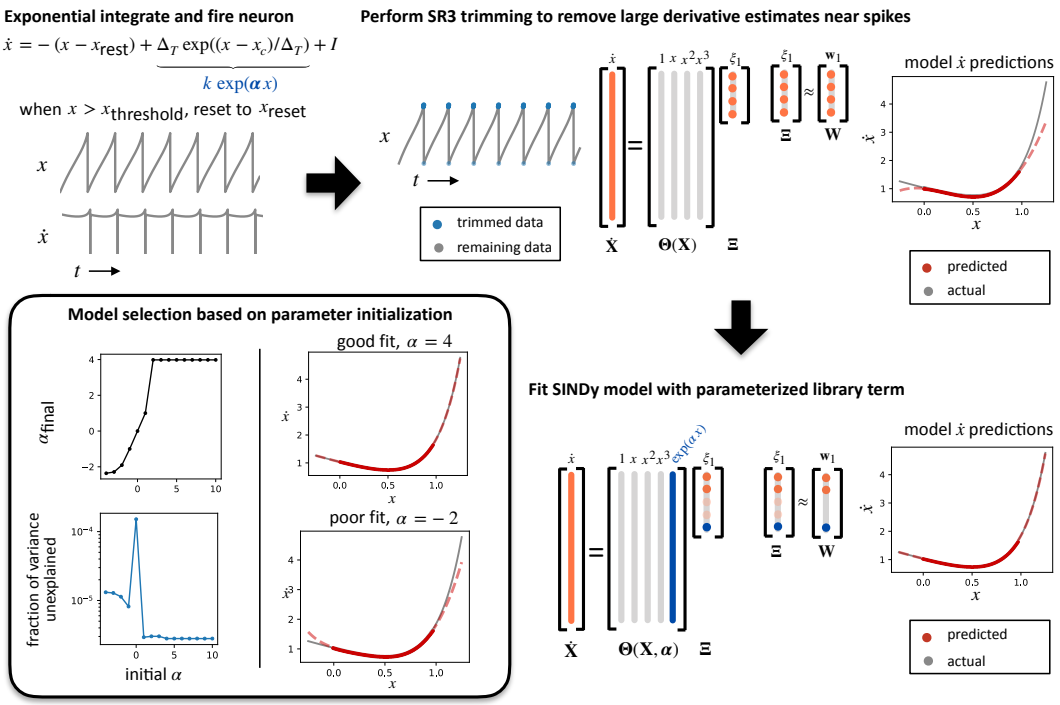

Figure 5: Workflow for identifying the exponential integrate and fire neuron, a spiking neuron model. First SINDy SR3 with trimming is performed, which removes the data points near the spikes but does not capture the exponential term. However, the SINDy algorithm with a parameterized exponential term is able to identify the correct model given proper initialization of the parameter. The inset shows how initialization affects the discovered parameter value: some initializations do not recover the correct parameter, and in this case the SINDy model error is higher (error shown on log scale). Model selection should therefore be used to identify the correct parameter value.

to assess the fit of the discovered model through model selection criteria. While the best choice of model may be clear (see Section 4.2 and Figure 5 for further details), this means parameterized SINDy works best for models with only a small number of parameters in the library, as scanning through different initializations scales combinatorially with added parameters.

4.1 Lorenz with parameterized forcing

We consider (5) with x_1 forced by a parameterized hyperbolic tangent function $\tanh(\alpha_1 t - \alpha_2)$. The parameters α_1, α_2 determine the steepness and location of the sigmoidal curve in the forcing function. We simulate the system with forcing parameters $\alpha_1 = 0.8, \alpha_2 = 3$. Figure 4 shows the results of fitting the SINDy model with and without the parameterized forcing term in the library. In the case without forcing, the equation for x_1 is loaded up with several active terms in an attempt to properly fit the dynamics. The model is not able to reproduce the correct system behavior through simulation. In the case with forcing, we start with an initial guess of $\alpha_1 = 5, \alpha_2 = 10$ and perform the joint optimization to fit both the parameters and the coefficient matrix. The algorithm correctly identifies the forcing and finds the correct coefficient matrix. The resulting system matches the true dynamics for several trips around the attractor.

4.2 Exponential integrate and fire: trimming and parameterized library

We also consider the exponential integrate and fire (EIF) neuron model. The EIF model is a spiking neural model where the membrane potential x of a neuron is governed by

$$\dot{x} = -(x - x_{\text{rest}}) + \Delta_T \exp((x - x_c)/\Delta_T) + I \tag{11}$$

and a set of rules that determine when the neuron spikes. In modeling the system, when the value of the potential reaches a threshold potential $x > x_{\text{threshold}}$, the neuron is considered to have fired a spike and the potential is reset to $x = x_{\text{reset}}$. While the EIF model is a simplified model that does not capture the rich dynamics of real neurons, it serves as an ideal example for illustrating issues that may arise in scientific data. This model has sharp discontinuities at the spikes. While these discontinuities are artificial, true neural data also typically contains sharp peaks at the spikes, leading to inaccurate derivative estimates in these regions. It can therefore be useful to trim this data when derivative estimates are likely to be inaccurate. Additionally, the model has parameterized terms in the dynamics: there is an exponential term in the governing equation determined by a parameter that cannot be fit using a simple linear least squares regression.

We simulated the EIF model with a constant input current at a level that results in 7 spikes over the course of the simulation. The discontinuities at the spikes lead to bad derivative estimates around these points. We therefore run the trimming algorithm introduced in Sec. 3, which removes these points and fits a SINDy model. In this case, a sparse SINDy model is not identified as the regression uses all polynomial terms to try to approximate the exponential term present in the dynamics. Therefore, following the trimming we run the algorithm introduced in Sec. 4 to fit a SINDy model with a parameterized exponential term. The model predictions of the derivatives $\dot{x}$ are shown in Fig. 5. The parameterized model results in an optimization that may have multiple local minima, thus the initial guess for the parameters influences the optimization results. Figure 5 shows an analysis of how initialization of the parameter α affects the discovered model. For some initializations the optimization does not find the correct value for α. However, in these cases the model error is higher and looking at the resulting model predictions shows that the discovered model is incorrect.

5 Discussion

Machine learning for model discovery in physics, biology and engineering is of growing importance for characterizing complex systems for the purpose of control and technological applications. Critical for the design and implementation in new and emerging technologies is the ability to interpret and generalize the discovered models, thus requiring that parsimonious models be discovered which are minimally parametrized. Moreover, model discovery architectures must be able to incorporate the effects of constraints, provide robust models, and/or give accurate nonlinear parameter estimates. We here propose the SINDy-SR3 method which integrates a sparse regression framework for parsimonious model discovery with a unified optimization algorithm capable of incorporating many of the critical features necessary for real-life applications. We demonstrate its accuracy and efficiency on a number of example problems, showing that SINDy-SR3 is a viable framework for the engineering sciences.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1256082. The authors acknowledge support from the

Defense Advanced Research Projects Agency (DARPA PA-18-01-FP-125), the Army Research Office (W911NF-17-1-0422), and the Air Force Office of Scientific Research (FA9550-17-1-0329). Aleksandr Aravkin is supported by the Washington Research Foundation Data Science Professorship.

References

[1] Aleksandr Aravkin and Damek Davis. A smart stochastic algorithm for nonconvex optimization with applications to robust machine learning. *arXiv preprint arXiv:1610.01101*, 2016.

[2] 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.

[3] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[4] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proc. Nat. Acad. Sci.*, 104(24):9943–9948, 2007.

[5] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Nat. Acad. Sci.*, 113(15):3932–3937, April 2016.

[6] 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.

[7] Damek Davis. The asynchronous palm algorithm for nonsmooth nonconvex problems. *arXiv preprint arXiv:1604.00526*, 2016.

[8] Gene H Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

[9] R Gonzalez-Garcia, R Rico-Martinez, and IG Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Comp. & Chem. Eng.*, 22:S965–S968, 1998.

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

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

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

[13] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proc. R. Soc. A*, 474(2219), 2018.

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

[15] Zhilu Lai and Satish Nagarajaiah. Sparse structural system identification method for nonlinear dynamic systems with hysteresis/inelastic behavior. *Mech. Sys. & Sig. Proc.*, 117:813–842, 2019.

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

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

[18] 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.

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

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

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

[22] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9258–9268. Curran Associates, Inc., 2018.

[23] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. *Phys. Rev. Lett.*, 120(2):024102, 2018.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] 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.

[26] 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.

[27] Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

[28] 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.

[29] 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.

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

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

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

[34] Giang Tran and Rachel Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, 15(3):1108–1129, 2017.

[35] 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.

[36] 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.

[37] Eunho Yang and Aurélie C Lozano. Robust Gaussian graphical modeling with the trimmed graphical lasso. In *Advances in Neural Information Processing Systems*, pages 2602–2610, 2015.

[38] Eunho Yang, Aurélie C Lozano, and Aleksandr Aravkin. A general family of trimmed estimators for robust high-dimensional data analysis. *Electronic Journal of Statistics*, 12(2):3519–3553, 2018.

[39] 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.

[40] 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.

S1 Choice of parameters for SR3

The SR3 algorithm requires the specification of two parameters, ν and λ. The parameter ν controls how closely the relaxed coefficient matrix $\mathbf{W}$ matches $\boldsymbol{\Xi}$: small values of ν encourage $\mathbf{W}$ to be a close match for $\boldsymbol{\Xi}$, whereas larger values will allow $\mathbf{W}$ to be farther from $\boldsymbol{\Xi}$.

The parameter λ determines the strength of the regularization. If the regularization function is the ℓ_0 norm, the parameter λ can be chosen to correspond to the coefficient threshold used in the sequentially thresholded least squares algorithm (which determines the lowest magnitude value in the coefficient matrix). This is because the prox function for the ℓ_0 norm will threshold out coefficients below a value determined by ν and λ. In particular, if the desired coefficient threshold is η, we can take

$$\lambda = \frac{\eta^2}{2\nu}$$

and the prox update will threshold out values below η. In the examples shown here, we determine λ in this manner based on the desired values for ν, η. If the desired coefficient threshold is known (which is the case for the examples studied here, but may not be the case for unknown systems), this gives us a single parameter to adjust: ν. With λ defined in this manner, decreasing ν provides more weight to the regularization, whereas increasing ν provides more weight to the least squares model fit.

S2 Simulation details: performance of SR3 for SINDy

We illustrate a comparison of three algorithms for SINDy using data from the canonical example of the chaotic Lorenz system (5). To generate training data, we simulate the system from $t = 0$ to 10 with a time step of $\Delta t = 0.005$ for 20 initial conditions sampled from a random uniform distribution in a box around the attractor. This results in a data set with 40×10^4 samples. We add random Gaussian noise with a standard deviation of 10^{-2} and compute the derivatives of the data using the central difference method. The SINDy library matrix $\boldsymbol{\Theta}(\mathbf{X})$ is constructed using polynomial terms through order 3.

We find the SINDy model coefficient matrix using the following optimization approaches: sequentially thresholded least squares (STLSQ) with threshold 0.1, SR3 with ℓ_0 regularization, LASSO with a regularization weight of 0.1, and LASSO with a regularization weight of 50. The STLSQ algorithm is performed by doing 10 iterations of the following procedure: (1) perform a least squares fitting on remaining coefficients, (2) remove all coefficients with magnitude less than 0.1. The LASSO models are fit using the scikit-learn package [24]. LASSO models are fit without an intercept, and for both LASSO and SR3 we initialize the coefficient matrix using least squares. For the SR3 algorithm, we use parameters $\nu = 1$ and $\lambda = 0.005$ (which corresponds to a coefficient threshold of 0.1, see Section S1).

For each of the four resulting models we analyze (1) the sparsity pattern of the coefficient matrix and (2) the simulation of the resulting dynamical systems model. We compare the sparsity pattern of the coefficient matrix against the true sparsity pattern for the Lorenz system: SR3 and STLSQ identify the correct sparsity pattern, where as the LASSO models do not. For all models, we simulate the identified system on test trajectories using initial conditions not found in the training set. The initial conditions are $(-8, 7, 27)$ (on attractor) and $(0.01, 0.01, 80)$ (off attractor), and the systems are simulated for the same time duration used in the training set. These results are shown in Figure 2 in the main text.

S3 Simulation details: simultaneous sparse inference and data trimming

S3.1 Example: Lorenz

We demonstrate the use of the SR3-trimming algorithm 2 on data from the Lorenz system (5). We simulate the system over the same time as in Section S2 from 5 randomly sampled initial conditions. This results in a data set with 10^4 samples. We add Gaussian noise to the data with standard deviation 10^{-3}. We then randomly choose 10% of the samples to corrupt (1000 total samples). For each state variable of each corrupted sample, noise chosen from a random uniform distribution over $[-50, 50]$ is added. Derivatives are calculated from the data using central difference after the corruption is applied. We then apply the SR3-trimming algorithm, specifying that around 40% of the data points will be trimmed. We use SR3 parameters $\nu = 20$ and $\lambda = 0.00025$ (corresponding to a coefficient threshold of 0.1), and the step size is taken to be the default value $\beta = 1$. With repeated testing we find that the algorithm is consistently able to correctly remove the outliers from the data set and identify the Lorenz system.

S3.2 Example: Rossler

As an additional example, we test the trimming algorithm on data from the Rossler system (8). We generate sample data from 5 randomly sampled initial conditions around the portion of the attractor in the x_1, x_2 plane, simulating trajectories from $t = 0$ to 50 with a time step of $\Delta t = 0.01$. Our data set consists of 25000 samples. We add Gaussian noise with a standard deviation of 10^{-3} and add outliers to 1% of the data in the same manner as in Section S3.1, with the noise level chosen from a random uniform distribution over $[-100, 100]$. Derivatives are calculated from the corrupted data using central difference.

We apply the SR3-trimming algorithm with two different levels of trimming. In both cases we use SR3 parameters $\nu = 20$ and $\lambda = 6.25 \times 10^{-5}$ (corresponding to a coefficient threshold of 0.05) and default step size $\beta = 1$. On repeated trials we find that if we trim only 5% of the data, many of the outliers are typically missed and the system is not correctly identified (instead, the algorithm trims part of the attractor in the x_3 plane). However, if we trim 10% of the data the system is correctly identified in most cases (or only 1 or 2 coefficients are misidentified).

S4 Simulation details: parameterized library functions

S4.1 Lorenz with parameterized forcing

To demonstrate the use of SR3 for SINDy with parameter estimation, we look at an example of the Lorenz system (5) forced by a parameterized hyperbolic tangent function $\tanh(\alpha_1 t - \alpha_2)$. The full set of equations for the system is

$$\begin{aligned}
\dot{x}_1 &= 10(x_2 - x_1) + 20\tanh(\alpha_1 t - \alpha_2) \\
\dot{x}_2 &= x_1(28 - x_3) - x_2 \\
\dot{x}_3 &= x_1 x_2 - (8/3)x_3.
\end{aligned}$$

The parameters α_1, α_2 determine the steepness and location of the sigmoidal curve in the forcing function. We simulate the system as in Section S2 for a single initial condition $(8, -7, 27)$ with forcing parameters $\alpha_1 = 0.8, \alpha_2 = 3$. We add Gaussian noise of standard deviation 10^{-3} and compute the derivatives via central difference.

We apply Algorithm 2 to perform a joint discovery of both the coefficients $\mathbf{W}$ and forcing parameters $\boldsymbol{\alpha}$. We use parameters $\nu = 0.1$ and $\lambda = 0.05$ (corresponding to coefficient threshold 0.1). $\mathbf{W}$ is initialized using least squares, and as an initial guess for $\boldsymbol{\alpha}$ we use $\boldsymbol{\alpha}^0 = (5, 10)$. The algorithm discovers the correct parameters $\boldsymbol{\alpha}$ as well as the correct sparsity pattern in the coefficient matrix. We simulate the system and see that the discovered system tracks the behavior for several trips around the attractor. Results are shown in Figure 4.

For comparison, we apply the SR3 algorithm for SINDy with no forcing term in the library, using the same SR3 parameters as in the forcing case. The resulting model has many active terms in the equation for $\dot{x}_1$, as it attempts to capture the forcing behavior with polynomials of x_1, x_2, x_3. This model does not perform well in simulation, even from the same initial condition used in the training set. Figure 4 shows the coefficient matrix and model simulation for the discovered system.

S4.2 Exponential integrate and fire neuron: trimming and parameterized library

To demonstrate a work flow with both trimming and parameterized library functions, we perform systems identification on simulation of an exponential integrate and fire (EIF) neuron model

$$\dot{x} = -(x - x_{\text{rest}}) + \Delta_T \exp((x - x_c)/\Delta_T) + I$$

with parameters $x_{\text{rest}} = 0, x_c = 0.5, \Delta_T = 0.25$. The input current I is set to a constant value of 1. In the EIF model, the differential equation above is combined with a mechanism for spiking: when the potential x reaches a threshold $x_{\text{threshold}}$, its value at that time point is reset to a reset potential x_{reset} and the neuron is said to have fired a spike at that time point. We use $x_{\text{threshold}} = 1$ and $x_{\text{reset}} = 0$. We simulate the EIF model from $t = 0$ to 8 with a time step of $\Delta t = 10^{-3}$, using a forward Euler time stepping method and the spiking mechanism described above. At the given parameter values, there are a total of 7 spikes over the course of the simulation. This example is particularly sensitive to noise, and thus we demonstrate the results without added noise. Derivatives are computed using the central difference method.

Due to the discontinuities at the spikes, the derivative estimates for this data have sharp peaks near the spikes. We first apply Algorithm 2, which removes data points near the spikes. We apply the algorithm with parameters $\nu = 10$, $\lambda = 5 \times 10^{-6}$ (corresponding to a coefficient threshold of 0.01), telling the algorithm to trim 2% of the data. The result is that several data points near the spikes are trimmed. The resulting SINDy model is not sparse, as the coefficient library does not have an exponential term and the algorithm instead tries to approximate the exponential using polynomial terms.

To capture the true model for the neuron, we next apply Algorithm 3 to the trimmed data. Rather than including a forcing term in the library as in Section S4.1, we include a parameterized function of x in the form of an exponential: $g(\alpha, x) = \exp(\alpha x)$. The parameterized library is $\boldsymbol{\Theta}(\mathbf{x}, \alpha) = [1, \mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \exp(\alpha \mathbf{x})]$. We apply Algorithm 3 with the same parameters $\nu = 10$, $\lambda = 5 \times 10^{-6}$. Because the parameterized model results in an optimization with potentially many local minima, the initial guess α^0 significantly impacts the discovered parameter value α and coefficients $\mathbf{W}$. In Figure 5, we show the discovered parameter α and the prediction error of $\dot{x}$ for several initial values α^0. The correct value in this example is $\hat{\alpha} = 4$, and we use initial values ranging from $\alpha^0 = -4$ to 10. The prediction error shown is the fraction of variance of $\dot{x}$ unexplained by the resulting model (defined by the discovered parameters $\alpha, \mathbf{W}$), with error plotted on a log scale. Initializations close enough to the true value $\hat{\alpha}$ discover the right value and have a low error compared to models where the incorrect value is discovered. At these values,

the correct sparsity pattern in $\mathbf{W}$ is also discovered. This motivates the use of model selection to select among models with different initializations.

It should be possible to combine both the trimming and parameter search into a single optimization problem within the SR3 framework, but we leave this to future work.

S5 Convergence results

Here we state convergence results for Algorithm 1 and Algorithm 2. These algorithms fall under the framework of two classical methods, proximal gradient descent and the proximal alternating linearized minimization algorithm (PALM) [3]. While we demonstrate the use of Algorithm 3 on two example problems, this algorithm is much harder algorithm to analyze due to the complication from the Newton's step. We leave obtaining theoretical guarantees of Algorithm 3 as future work.

S5.1 Convergence of Algorithm 1

Using the variable projection framework [8], we partially optimize out $\boldsymbol{\Xi}$ and then treat Algorithm 1 as the classical proximal gradient method on $\mathbf{W}$. The convergence result for Algorithm 1 is provided in [40, Theorem 2] and is restated here:

Theorem 1 *Define the value function as,*

$$p(\mathbf{W}) = \min_{\boldsymbol{\Xi}} \frac{1}{2}\|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|^2 + \lambda R(\mathbf{W}) + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}\|^2$$

When p is bounded below, we know that the iterators from Algorithm 1 satisfy,

$$\frac{1}{N}\sum_{k=1}^{N} \|g^k\|^2 \leq \frac{1}{\nu N}(p(\mathbf{W}^0) - p^*),$$

where $g^k \in \partial p(\mathbf{W}^k)$ and $p^* = \min_{\mathbf{W}} p(\mathbf{W})$.

We obtain a sub-linear convergence rate for all prox-bounded regularizers R.

S5.2 Convergence of Algorithm 2

Following the same idea provided by the variable projection framework, the iterations from Algorithm 2 are equivalent with an alternating proximal gradient step between $\mathbf{W}$ and $\mathbf{v}$. This is the PALM algorithm, which is thoroughly analyzed in the context of trimming in [1] and [7]. We restate the convergence result here:

Theorem 2 *Consider the value function,*

$$p(\mathbf{W}, \mathbf{v}) = \min_{\boldsymbol{\Xi}} \sum_{i=1}^{m} \frac{1}{2} v_i \|(\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi})_i\|^2 + \lambda R(\mathbf{W}) + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}\|^2$$

And we know that the iterators (W^k, v^k) converge to the stationary point of p, with the rate,

$$\min_{k=0,\ldots,N} \operatorname{dist}(0, \partial p(\mathbf{W}^k, \mathbf{v}^k)) = o\left(\frac{1}{k+1}\right).$$

Algorithm 2 also requires the specification of a step size β for the proximal gradient step for $\mathbf{v}$. Because the objective is linear with respect to $\mathbf{v}$, the step size will not influence the convergence result in the above theorem. However, because the objective is non-convex, β will have an impact on where the solution lands. In this work we use a default step size of $\beta = 1$ for all examples.