Sparse model selection via integral terms

Hayden Schaeffer and Scott G. McCalla

Phys. Rev. E **96**, 023302 — Published 2 August 2017

DOI: 10.1103/PhysRevE.96.023302

Sparse Model Selection via Integral Terms

Hayden Schaeffer* and Scott G. McCalla
Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA, USA 15213 and
Department of Mathematical Sciences, Montana State University, Bozeman, MT, USA 59717

Model selection and parameter estimation are important for the effective integration of experimental data, scientific theory, and precise simulations. In this work, we develop a learning approach for the selection and identification of a dynamical system directly from noisy data. The learning is performed by extracting a small subset of important features from an overdetermined set of possible features using a non-convex sparse regression model. The sparse regression model is constructed to fit the noisy data to the trajectory of the dynamical system while using the smallest number of active terms. Computational experiments detail the model's stability, robustness to noise, and recovery accuracy. Examples include nonlinear equations, population dynamics, chaotic systems, and fast-slow systems.

Scientific progress is based on advancements in theory, experiment, and simulation. Experiments produce quantitive data that provide sufficient evidence to formulate theoretical equations that model the true behavior of the observed system. A famous example includes the electromagnetic equations of Coulomb, Gauss, Ampère, Faraday, and Maxwell which were formulated from several nineteenth century physical experiments and unified under one large system of differential equations. Computational methods utilize theoretical models to simulate results, thereby providing scientists with a tool to validate theory with experimental data and study the behavior of the system under many physical conditions. In computational fluid dynamics, theory and simulation are used together to help scientists understand the Navier-Stokes equations, whose solutions are often difficult to construct analytically.

With the increasing size of experimental datasets and advances in computational methods for analytics, there is a growing demand for data-driven exploration of scientific laws. In particular, machine learning and fitting algorithms can be used to explore large datasets with the goal of extracting important features, recognizing significant patterns, and parameterizing the observed behaviors. In this work, a computational method is developed for exploration, analytics, and model identification from noisy time-series data controlled by a dynamical system. The differential equation defines the governing model for some physical process. By learning the underlying dynamical system from noisy measurements, the method is able to extract information on the system, garner insight into the collective behavior of the data and, in some cases, provide a smooth approximation of the dataset.

In recent years, several methods have been proposed for fitting dynamical systems to data. In [1, 2], symbolic regression is used to extract physical laws (conservation laws, Hamiltonians, Lagrangians, equations of motion, *etc*) from a combination of synthetic and experimental data. The computational approach estimates the partial derivatives of each state variable with respect to each other and compares them to candidate functions. The candidate is chosen by measuring the accuracy of the fit as well as the efficiency of the candidate model. In another direction, several sparsity-promoting optimization methods have been proposed for fitting dynamical systems to a set of candidate functions. In [3], a sequential thresholded least-squares algorithm is used to fit a set of candidate polynomials to computed velocity data. Sparse optimization for learning partial differential equations from spatio-temporal data is detailed in [4, 5]. In [6], a sparse optimization problem is proposed for joint model selection and outlier detection, which allows for learning in the presence of time-intervals with large corruption. In this work, we propose a sparse optimization method for identifying the underlying dynamical system from a given data set, where the entire dataset is corrupted by noise. It is worth noting that sparse optimization and modeling has seen some recent applications to differential equation, scientific computing, and data-driven modeling, see for example [7–12].

A key tool in fitting the dataset to the most important features (candidate functions) is the use of sparsity, typically via the ℓ^0 or ℓ^1 penalty and the related thresholding operator, first proposed in [13]. The least absolute shrinkage and selection operator (LASSO) was introduced in [14], which regularizes the least-squares problem by penalizing the number of terms used in the fitting. Also, algorithms using soft and hard thresholding (the proximal operators of ℓ^1 and ℓ^0, respectively) are often used for denoising, reconstruction, and learning. In [15, 16], the ℓ^1 regularized problem (used as a relaxation to the non-convex ℓ^0) was shown to recover sparse signals in underdetermined system under certain assumptions. This has lead to many application in compressive sensing and image processing.

MODEL

Consider the evolution $x(t) \in \mathbb{R}^n$ governed by

$$\dot{x} = F(x) \text{ with } x(0) = x_0. \tag{1}$$

Our goal is to extract the model $F(x)$ from noisy measurements of $x(t)$. The function $F : \mathbb{R}^n \to \mathbb{R}^n$ should be Lipschitz continuous and each component should reside in the span of a set of (redundant) trial functions $\{f_j(x)\}_{j=1}^N$, *i.e.*:

$$F_i(x) = C_{i,j}\, f_j(x) \text{ with } C \in \mathbb{R}^{n\times N}.$$

Given data $\tilde{x}(t)$ over some interval $t \in [0, T]$, we would like to recover the function F, and thus the governing equation. We assume that the set of trial functions is large; however, the set of active terms in the underlying dynamics is sparse. As $F(x)$ is unknown, we could estimate it by the time derivative:

$$\dot{x}_i = C_{i,j}\, f_j(x). \tag{2}$$

In the ideal case, x solves Eq. [1] and C can be estimated through Eq. [2]. In particular, the support of the coefficient matrix C defines the model selected from the set of trial functions and the learned coefficients estimate the model parameters. In practice, measurement noise η corrupts the dataset $\tilde{x}(t) = x(t) + \eta(t)$. We assume that the measurement error is mean zero Gaussian noise $\eta(t) \sim \mathcal{N}(0, \sigma^2)$ with variance $\sigma^2 > 0$. Taking derivatives of noisy data is known to be unstable, making it difficult to use Eq. [2]. Instead, consider the integral formulation:

$$x_i(t) = x_i(0) + C_{i,j}\, d_j(x, t)$$

where we denote the integral terms d_j by

$$d_j(x, t) := \int_0^t f_j(x(\tau))\, \mathrm{d}\tau. \tag{3}$$

The collection of terms $d_j(x, t)$ will be referred to as the features since they are data-dependent and thus observed behaviors. In the integral formulation, Eq. [2] becomes the following linear system (in terms of C):

$$x_i(t) - x_i(0) = C_{i,j}\, d_j(x, t). \tag{4}$$

By discretizing the equation over time-stamps t_k for $k = 1, .., K$, Eq. [4] admits a linear inverse problem for C. Define the feature matrix $\mathcal{D} \in \mathbb{R}^{K\times N}$ as the collection of (column-wise) feature vectors $d_1(x, t)$, $d_2(x, t)$, $...d_p(x, t)$, where the columns represents the temporal components and the rows represent the features, *i.e.* $(\mathcal{D}(x))_{i,j} = d_j(x, t_i)$. Let $\delta x(t) := [x(t) - x_0]^{\mathrm{T}}$ be the displacement from the initial data and define the vector $X \in \mathbb{R}^{Kn}$ as the column-wise collection of all displacements:

$$X(x) = [\delta x_1(t_1) \cdots \delta x_1(t_K) \cdots \delta x_n(t_1) \cdots \delta x_n(t_K)]^{\mathrm{T}}.$$

Similarly, let $c \in \mathbb{R}^{nN}$ be the vector of all coefficients collected column-wise, *i.e.* $c := \mathrm{vec}(C)$. Lastly define $D(x) := \mathrm{diag}[\mathcal{D}(x), ..., \mathcal{D}(x)] \in \mathbb{R}^{Kn\times nN}$ as the block diagonal matrix generated by $\mathcal{D}$. Then the discrete linear system is

$$X(x) = D(x)c. \tag{5}$$

The vector X and matrix D are generated from the data itself. Even in the ideal case, the feature matrix may be redundant: if $x(t) = 0$ for all time and $f_j(x) = x^{j-1}$ with $j \geq 1$, the matrix $\mathcal{D}$ has one non-zero column and uniqueness is not expected.

In practice Eq. [5] does not hold because of the noise and we are left with the more difficult inverse problem, *i.e.* $X(\tilde{x}) \approx D(\tilde{x})c$. Instead, we determine the coefficient vector c that satisfies the least-squares constraint

$$\|X(\tilde{x}) - D(\tilde{x})c\|_2 \leq \epsilon \tag{6}$$

for some parameter $\epsilon(\sigma^2)$. To solve Eq. [6], we regularize the problem using the ℓ^0 penalty:

$$\min_C ||c||_0 \quad \text{s.t.} \quad \|X - Dc\|_2 \leq \epsilon \tag{7}$$

where we drop the dependence on $\tilde{x}$ for simplicity of notation. The constraint in Eq. [7] uses integrated rather than differentiated data, thus we expect this process to be more stable to noise. The ℓ^0 penalty, defined as $||z||_0 = \sum_j |\mathrm{sign}(z_j)|$, counts the number of non-zero coefficients limiting the number of features that may be used to represent the dataset. This penalty enforces model sparsity. This minimization problem is non-convex and will likely be algorithm and data dependent. One key benefit is that the method does not require pre-processing of the data in order to compute the trial functions. In particular, the features (trial functions) are calculated directly from the noisy data without smoothing or denoising the data. Another is that once the coefficients are learned, an approximate solution $x^L(t)$ is generated, $x^L(t) = \tilde{x}(0) + DC$, for a cleaner, and possibly more accurate, representation of the data. Alternatively a cleaner approximation can be found by solving $\dot{x}^S = D(x^S)C$.

ALGORITHM

Henceforth, we assume $F(x)$ is well-represented as a combination of polynomials (or in some examples, trigonometric functions) and define $f_j(x) = x^{j-1}$, unless otherwise stated. The number of feature vectors used in the learning process must be large enough to include all the terms present in the underlying system. Each of the feature vectors $d_j(x, t)$ are approximated using piecewise constant quadrature:

$$d_j(x, t_k) = \int_0^{t_k} f_j(x(\tau))\, \mathrm{d}\tau \approx \Delta t \sum_{\ell=1}^{k} f_j(x(t_\ell)) \tag{8}$$

where $k = 1, ..., K$ and $d_j(x, t_0) = d_j(x, 0) = 0$. The piecewise constant quadrature produces a closer approximation to the noiseless signal without smoothing. For polynomial trial functions, Eq. [8] effectively calculates a scaled expectation for a sum of random variables of the form $x^n\eta^p$. As the noise is sampled independently of the

sample $x(t)$, we have $\mathrm{E}(x^n\eta^p) \approx \mathrm{E}(x^n)\mathrm{E}(\eta^p)$. The second term arises from our Gaussian distribution and is 0 when p is odd and $\mathrm{E}(\eta^p) = \sigma^p(p-1)!!$ when p is even. Effectively, many of the noise dependent cross terms are near zero when piecewise constant quadrature is used to approximate the feature vector.

To solve Eq. [7], we apply the Douglas-Rachford algorithm [17, 18] along with an auxiliary variable. Introducing the auxiliary variable w with the constraint $w = Dc$ and $w \in B_\epsilon(X)$ yields a helpful splitting of the equations. Define the variable $z := (c, w) \in \mathbb{R}^{nN\times Kn}$, then Eq. [7] becomes

$$\min_z \; G_1(z) + G_2(z) \tag{9}$$

where the two functions are defined as:

$$G_1(z) := \|c\|_0 + \mathrm{Ind}_{B_\epsilon(X)}(w), \quad G_2(z) := \mathrm{Ind}_{\mathcal{K}}(z) \tag{10}$$

with $\mathcal{K} = \{z = (w, c)|\; w = Dc\}$ and Ind is the indicator function on the set (zero if satisfied, infinite otherwise). Although both G_1 and G_2 are non-differentiable and the ℓ^0 penalty is non-convex, we only need that the functions emit an explicit proximal operator. The proximal operator for a functional $H(z)$ is defined as

$$\mathrm{prox}_{\gamma G}(z) = \underset{z'}{\mathrm{argmin}}\, \gamma\, G(z') + \frac{1}{2}\|z - z'\|^2$$

where z' is an auxiliary variable in the optimization (not to be confused with the time-derivative). The proximal operator for G_1 acts on each variable (c, w) separately (since their optimization subproblems decouple):

$$\mathrm{prox}_{\gamma G_1}(z) = \Big(H_\gamma(c),\; \mathrm{Proj}_{B_\epsilon(X)}(w)\Big)$$

where H_γ is the hard thresholding function which acts component-wise on c

$$\left(H_\gamma(c)\right)_j = \begin{cases} c_j, & \text{if } |c_j| \geq \sqrt{\gamma} \\ 0, & \text{otherwise} \end{cases}$$

and the projection onto the ball is defined as

$$\mathrm{Proj}_{B_\epsilon(X)}(w) = X + \max\left(1, \frac{\epsilon}{\|w - X\|}\right)(w - X).$$

The proximal operator for G_2 is simply the projection onto the set $\mathcal{K}$

$$\mathrm{prox}_{\gamma G_2}(z) = \\ \big((I + D^TD)^{-1}(c + D^Tw), D(I + D^TD)^{-1}(c + D^Tw)\big)$$

The Douglas-Rachford iteration is defined by the following two-step process:

$$\begin{aligned} \tilde{z}^{k+1} &= \left(1 - \frac{\mu}{2}\right)\tilde{z}^k + \frac{\mu}{2}\,\mathrm{rprox}_{\gamma G_2}\left(\mathrm{rprox}_{\gamma G_1}\left(\tilde{z}^k\right)\right) \\ z^{k+1} &= \mathrm{prox}_{\gamma G_1}(\tilde{z}^{k+1}) \end{aligned} \tag{11}$$

where $\mathrm{rprox}_{\gamma G}(z) = 2\mathrm{prox}_{\gamma G}(z) - z$. Two important properties are worth noting. First, the order of the proximal operators are important, in particular, using the order in Eq. [11] ensures that the output will remain sparse. Second, the solution z^k depends on the parameter γ although it does not appear in the original minimization problem. For sparse model identification, γ represents the meaningful scales that appear in the solution c^*.

NUMERICAL EXAMPLES

The method is tested on a variety of ordinary differential equations. The underlying data $x(t)$ is either explicitly found or approximated with Runge-Kutta 45. The data is then corrupted by additive Gaussian noise giving $\tilde{x}(t) = x(t) + \eta(t)$ for the learning algorithm. The set of features is pre-determined for each example, and may include either polynomial or trigonometric functions. Each term d_j is numerically approximated using Eq. [8]. The noise level

$$\text{noise level} := \frac{||\eta(t)||_2}{||x(t)||_2} \times 100\% = \frac{\sigma}{||x(t)||_2} \times 100\%$$

is defined as the ratio between the ℓ^2 norms of the noise and underlying data. The parameter μ used in the Douglas–Rachford algorithm is set to $\mu = 1$, while the thresholding parameter γ is example dependent. The sample size is denoted as n. The maximum number of iterations varies; however, a maximum iteration value of 5000 is more than sufficient as convergence is reached prior to the maximum iteration.

The examples below serve to highlight several important results of the method, features about the convergence, and some potential issues that may arise. The trajectories need to take sufficiently large excursions with respect to the noise level in each of the equation components if the reconstruction is to be successful. Additionally if these excursions are faster (*i.e.* large variations), then the reconstruction is more accurate. The main difficulties are then in learning near equilibrium behavior, or in learning the governing equations along a slow manifold. To some degree, the most instructive and difficult cases are low dimensional stable systems near an equilibrium. It is expected that chaotic systems are the most robust cases as there is uniform re-sampling of the system's phase space at different rates. In all of the examples below, the noisy data is plotted with red markers and the approximations are plotted with black curves.

Linear Scalar Equations– Consider the linear model

$$\dot{x} = -\alpha x, \quad x_0 = 1 \Rightarrow x(t) = \mathrm{e}^{-\alpha\, t} \tag{12}$$

where $\alpha > 0$ is the decay parameter and the trial functions are given by $f_i = x^{i-1}$ for $i \leq 25$. This simple model is used to investigate the relationship between (computational) recovery rates and the maximum noise level.

TABLE I. **Sensitivity to Linear Decay Rate:** The data is simulated with the parameters $t \in [0, 15]$, $\Delta t = 0.005$, $x_0 = 1$, and $\epsilon = 1.1\sqrt{n}\sigma$. The value of γ is adapted to the example. The first column contains the values of α, the second shows the total distance traveled by the trajectory, the third displays the largest noise level achieved while maintaining recoverability, the fourth shows the percentage of non-equilibrium to equilibrium data that is used in the learning, and the fifth records the L^2 relative error between the approximate solution using the learned equation and the original data.

α	Δx	Noise Ratio	Equilibrium Ratio	L^2 Relative Error
0.25	0.976	0.449	0	0.055
0.3	0.989	0.511	0.090	0.041
0.4	0.998	0.579	0.270	0.069
0.5	0.999	0.672	0.386	0.060
0.75	≈ 1	0.775	0.555	0.058
1	≈ 1	0.859	0.647	0.057
1.25	≈ 1	0.866	0.706	0.056

In particular, this example describes the dependence between the size and location of the samples needed for nearly-exact recovery. In both cases below $\epsilon = 1.1\sqrt{n}\sigma$ is fixed, but γ is varied until the learned solution is sufficiently accurate.

For the first experiment, we discretize the solution to Eq. [12] over $t \in [0, 15]$ with time-step $\Delta t = 0.005$. Fixing the decay rate α, the learning is performed with increasing noise levels in order to find the largest possible corruption that can occur while maintaining recoverability with respect to coefficient identification (see Table I). It is worth noting that the relative error for each α is comparable to the least squares fitting method when the form of the governing equation is known. As α increases, the data decays faster and the portion of data near equilibrium increases. Note that the total variation in x is nearly one in all cases. These results suggest that for comparable sized variations in the data faster transit times in the dependent variables improve the robustness, thereby allowing our learning model to correctly identify the governing equation with higher noise levels. Also, the relative error appears to be stable with respect to the noise and decay rate variations, indicating that the method is robust to the coefficient values.

To test the robustness of this approach to variations in the sample size n, we discretize the solution to Eq. [12] over $t \in [0, 10]$ while varying $\Delta t = 10/n$. Starting with 25 data points and increasing the size to 5000 data points, the method recovers the governing equation with higher accuracy and in the presence of larger noise variance (see Table II). Again, it is worth noting that the relative errors are comparable to the least squares fitting method applied to the exact model. The results from Table II confirm that increasing the sample size improves recovery, but relatively strong results can be found with a small number of samples.

TABLE II. **Sensitivity to Sample Size:** The data is simulated with $t \in [0, 10]$, $x_0 = 1$, $\Delta t = 10/n$, and $\epsilon = 1.1\sqrt{n}\sigma$. The parameter γ is adapted to the example.

Data Size, n	Noise Ratio	Relative Error
25	0.145	0.229
50	0.185	0.049
100	0.353	0.057
250	0.404	0.053
500	0.558	0.032
1000	0.692	0.036
5000	0.744	0.036

Generalized Logistic Equations– Consider the generalized logistic growth model, aka the Richards' differential equation,

$$\dot{x} = \delta(x - x^p); \tag{13}$$

this is a classical model for population dynamics, and the solutions correspond to a family of sigmoid curves. By varying δ and p, we will explore how the learning algorithm responds to different size excursions in the dependent variable, different governing equations (including governing equations not exactly represented by the trial functions), and what regions of the trajectory contribute most to the learned governing equation.

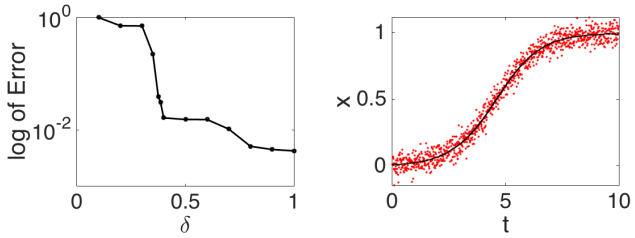

FIG. 1. **Logistic Growth.** The left plot displays the log of the relative ℓ^2 error on the learned coefficients versus the parameter δ. The data is simulated with $t \in [0, 10]$, $\Delta t = 0.01$, and $x_0 = 0.01$. The parameters are fixed at $\epsilon = 1.1\sqrt{n}\sigma$ and $\gamma^2 = 0.5$. The noise level is fixed at 2% with the same realizations between experiments. The jump at between $\delta = 0.3$ and $\delta = 0.4$ marks a transition, where the approximation identifies the correct terms (and continues to refine the values). On the right, the data is plotted with $t \in [0, 10]$, $\Delta t = 0.01$, $x_0 = 0.01$, $\epsilon = 1.1\sqrt{n}\sigma$, and $\gamma^2 = 0.5$. The noise level is 5% and the error between the learned solution and the original data in the L^2 norm is 9.4×10^{-3}.

First, consider the standard logistic equation with $p = 2$ and $\delta = 1$. Figure 1 illustrates the recovery of this equation from noisy data with two noise levels. The

learned equations are given by

$$\text{Noise} = 2\% : \ \dot{x} = \sum_{j=0,\ldots,24} 1.003\,\delta_{j1}\,x^j - 1.004\,\delta_{j2}\,x^j$$

$$\text{Noise} = 5\% : \ \dot{x} = \sum_{j=0,\ldots,24} 1.033\,\delta_{j1}\,x^j - 1.036\,\delta_{j2}\,x^j$$

where the Kronecker delta notation is used to emphasize the terms used in the learning process with a learned coefficient of zero.

Fixing $p = 2$ and $t \in [0, T]$, we vary $\delta > 0$ to show that the method will undergo a transition between low accuracy and nearly-exact recovery once the data is sufficiently far from the equilibrium state. The graph in Fig. 1 shows the log of the relative error on the learned coefficients versus the scale δ. A sharp decrease in the error occurs between $\delta = 0.3$ and $\delta = 0.4$ and marks the transition from learning only the linear term (for $\delta < 0.3$) to learning both the linear and quadratic terms (for $\delta \geq 0.4$). This transition occurs as the solution to Eq. [13] introduces sharper gradients and takes a larger excursion in x.

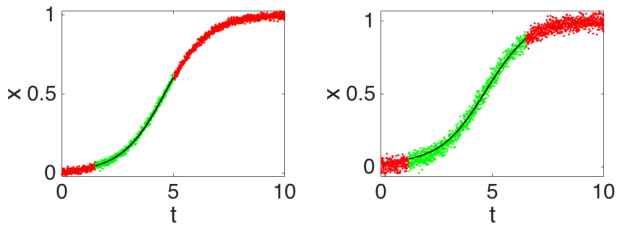

FIG. 2. **Logistic Growth.** The data is simulated over $t \in [0, 10]$, with time-step $\Delta t = 0.01$, and initial data $x_0 = 0.01$. The learning parameters are set to $\epsilon = 1.1\sqrt{n}\sigma$ and $\gamma^2 = 0.5$. The noisy data is plotted with red markers (darker gray), the interval of strongest influence is marked in green (lighter gray), and the learned solution is in black. On the left (2% noise level), the interval of strongest influence is given by $[1.5, 5]$ and on the right, (noise level 5%) the interval of strongest influence is given by $[1.25, 6.5]$.

Next, we look for the interval of strongest influence to the recovery. Considering two noise levels, we perform a coarse search for the smallest interval that leads to an accurate result. The results in Fig. 2 suggest that the region of high variation (away from the equilibrium) controls the accuracy of the solution to the learning model.

As the logistic equation is typically used to model saturating population dynamics, we apply this method to U.S. census data (population in millions) collected from 1790 to 1990, with time-step $\Delta t = 10$. The data is fit without assuming the model form– using a total of 25 terms in the dictionary (more terms than data samples). Even though the noise variance in the raw data is not known *a priori*, we choose the parameters so that the learned solution has minimal error while using the minimal amount of terms in the governing equation. With

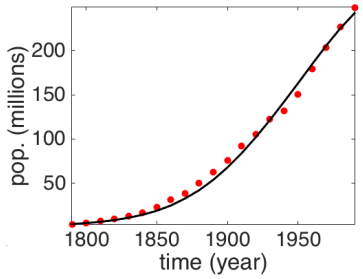

FIG. 3. **U.S. Census Data.** Using the US census data (population in millions) collected from 1790 to 1990, with time-step $\Delta t = 10$, we fit the data without assuming the model form. The learning parameters are set to $\epsilon = 1$, and $\gamma^2 = 1.9 \times 10^{-3}$. The learned governing equation corresponds to a parameterized logistic equation and agrees closely with data.

TABLE III. **Generalized Logistic Equations:** The data is simulated from $t \in [0, 10]$, with time-step $\Delta t = 0.01$, and initial data $x_0 = 0.01$. The noise is fixed at 2%. The learning parameters are set to $\epsilon = 1.1\sqrt{n}\sigma$ and $\gamma^2 = 0.5$. The method identifies the correct coefficients and estimates the parameters within the scale of the noise. The error between the approximated solution and the original data in the L^2 norm is 5.3×10^{-3}, 5.8×10^{-3}, 7.2×10^{-3}, and 8.0×10^{-3}, for $p = 2, 3, 4$, and 5.1 respectively.

Terms	$p = 2$	$p = 3$	$p = 4$	$p = 5.1$
1	0	0	0	0
x	1.003	1.002	1.001	1.006
x^2	−1.002	0	0	0
x^3	0	−0.999	0	0
x^4	0	0	−0.997	0
x^5	0	0	0	−0.999
|	|	|	|	|
x^{24}	0	0	0	0

this choice of parameters, the learned governing equation corresponds to a parameterized logistic equation. To check the efficiency of the learned equations, γ is varied and fitting-error is checked. Shrinking values of γ dramatically increase the number of terms in the recovered model (due to overfitting). On the other hand, increasing γ leads to high recovery error with less terms. As seen in Fig. 3, the learned solution (in black) agrees closely with raw data (in red).

We now vary $p = 2, 3, 4$, and 5.1 to see how the algorithm performs with respect to changes in the governing equations. Qualitatively, the solutions are similar in structure, with small differences in the slope of the transition towards the equilibrium $x = 1$. The addition of noise makes it difficult to differentiate the solutions in the "eye-ball" norm. In Table III, the resulting learned coefficients are presented for various p's. For $p = 2, 3$, and 4, the method identifies the correct coefficients and

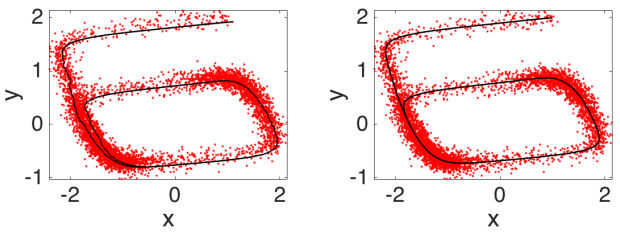

FIG. 4. **FitzHugh-Nagumo Model** The data uses $t \in [0, 50]$, $\Delta t = 0.01$, and $x_0 = (1, 2)^T$. The parameters are set to$\epsilon = 1.1\sqrt{n}\sigma$, $\gamma^2 = 0.05$, and noise level 9%. The left graph plots the approximate solution in black and the right graph the solution is re-simulated using the learned governing equation in black.

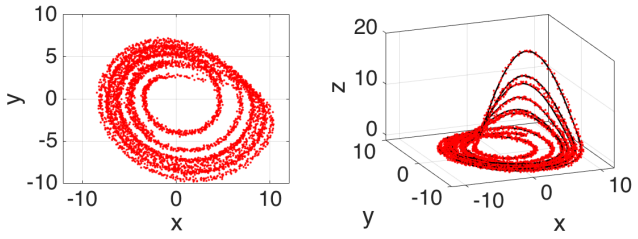

FIG. 5. **Fast-Slow Approximation** The data uses $t \in [0, 50]$, $\Delta t = 0.01$, $x_0 = (5, 2, 1, 1)^T$, and $\epsilon_1 = 5 \times 10^{-3}$. The learning parameters are set to $\epsilon = 0.1$, and $\gamma^2 = 0.1$. The left figure plots the noisy data projected onto the xy-plane with noise level 3.5%. The right figure plots the noisy data along with the data-driven approximation.

estimates the parameters within the scale of the noise. In the $p = 5.1$ case, the method, as expected, approximates the governing equation with $p = 5$. The error between the approximated solution and the original data in the L^2 norm is 5.3×10^{-3}, 5.8×10^{-3}, 7.2×10^{-3}, and 8.0×10^{-3}, for $p = 2, 3, 4$, and 5.1 respectively. Note that even though $x^{5.1}$ is not within the span of the feature space, the method is able to approximate the data and equation using $p = 5$. This suggest that the method could approximate systems where the feature space may be difficult to parameterize.

Singularly Perturbed Systems– The following two examples both address learning in systems with mixed time scales. First consider the FitzHugh–Nagumo equation

$$\begin{cases} \dot{x} = 0.1 + x - \frac{x^3}{3} - y \\ \dot{y} = 0.1(x - y) \end{cases}$$

for excitable systems such as a neuron.

In Fig. 4, the noisy data (noise level 9%) is plotted in red. The left graph plots the approximate solution in black and the right graph the solution is re-simulated using the learned governing equation (in black). Comparing the two plots, it can be seen that for large noise levels it may be necessary to use the re-simulated solution in order to maintain an accurate approximation. The learned solution is accurate to within the noise level (trial functions $f_{i,j} = x^{i-1}y^{j-1}$ for $i + j \leq 5$):

$$\text{Noise} = 2\% : \begin{cases} \dot{x} = 0.101 + 1.001\,x - .334\,x^3 - 1.001\,y \\ \dot{y} = 0.1\,(x - y) \end{cases}$$
$$\text{Noise} = 9\% : \begin{cases} \dot{x} = .105 + .991\,x - .327\,x^3 - .986\,y \\ \dot{y} = .1\,x - .101\,y \end{cases}.$$

Next consider the multiscale system which can be shown to reduce to the 3D Rössler System as the scale parameter $\epsilon_1 \to 0^+$

$$\begin{cases} \dot{x} &= -y - z \\ \dot{y} &= x + 0.2y \\ \dot{z} &= 0.2 + w - 5z \\ \epsilon_1 \dot{w} &= -w - xz \end{cases} \Rightarrow \begin{cases} \dot{x} = -y - z \\ \dot{y} = x + 0.2y \\ \dot{z} = 0.2 + z(x - 5). \end{cases} \quad (14)$$

For ϵ_1 small, the first three variable are slow and the fourth variable is fast. In this experiment, we simulate the solution using the 4D fast-slow system, corrupt the data with additive noise, and then only use the first three components in the learning algorithm. By restricting our attention to the first three components, we test the methods ability to identify the slow (dominant) behavior: see Fig. 5. In the 3.5% noise case, the corrupted trajectories are self-intersecting, which can cause difficulty for learning methods. The learned equations are

$$\text{Noise} = 1\% : \begin{cases} \dot{x} = -y - z \\ \dot{y} = x + 0.2y \\ \dot{z} = 0.2 + z(x - 5) \end{cases}$$
$$\text{Noise} = 3.5\% : \begin{cases} \dot{x} = -y - z \\ \dot{y} = x + 0.2y \\ \dot{z} = 0.2 + z(x - 5) \end{cases}.$$

with trial functions $f_{i,j,k} = x^{i-1}y^{j-1}z^{k-1}$ for $i+j+k \leq 4$. In both cases, the method is able to correctly identify the active features and approximate the coefficients.

Pendulum– Consider the nonlinear pendulum problem and its linearization

$$\begin{cases} \dot{x} = y \\ \dot{y} = -1.5\,\sin x. \end{cases} \approx \begin{cases} \dot{x} = y \\ \dot{y} = -1.5\,x \end{cases}$$

in the small-angle regime. In Fig. 6, we learn the solution using a polynomial and trigonometric feature space in the two different regimes. In the nonlinear regime on the left, we simultaneously use both the polynomial ($f_{i,j} = x^{i-1}y^{j-1}$ for $i + j \leq 5$) and trigonometric ($f_i^s(\cdot) = \sin(i\cdot)$ and $f_i^c(\cdot) = \cos(i\cdot)$ for $i \in \{1, \ldots, 4\}$) trial functions. For

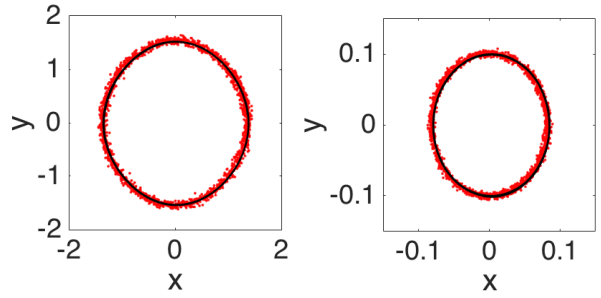

FIG. 6. **Pendulum Problem** The data uses $t \in [0, 18]$, $\Delta t = 0.1$, $x_0 = (1, 1)^T$ (left) and $x_0 = (0.01, 0.1)^T$ (right), $\epsilon = 1.1\sqrt{n}\sigma$, and $\gamma^2 = 0.2$. The noise level is fixed at 2.5%. On the left, the data-driven approximation (black) uses a polynomial and trigonometric feature space. On the right, the data-driven approximation (black) uses a polynomial feature space. The plot on the right is in the small-angle regime and the correct linearized system is identified.

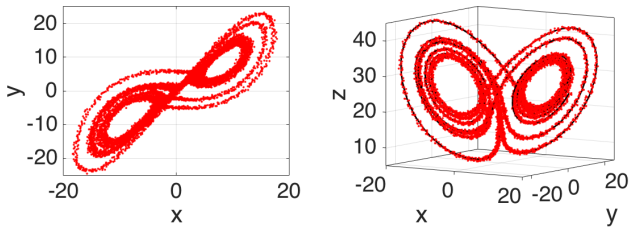

FIG. 7. **Lorenz System.** The data uses $t \in [0, 10]$, $\Delta t = 0.001$, and $x_0 = (-5, 1, 20)^T$. The parameter is $\epsilon = 1.1\sqrt{n}\sigma$. The left plot is the noisy data projected onto the xy-plane with noise level 3%. The right plot is the noisy data along with the data-driven approximation.

the small-angle regime only the polynomial functions $f_{i,j}$ are used. The learned equations are

$$\begin{cases} \dot{x} = 1.009\, y \\ \dot{y} = -1.500\, \sin x. \end{cases} \approx \begin{cases} \dot{x} = .997\, y \\ \dot{y} = -1.483\, x. \end{cases}$$

The right plot is in the small-angle regime, and as expected the correct linearization is learned (*i.e.* $\sin x \approx x$). The results in this example suggest that the method is able to discriminant between possible approximations, even when redundant features are used.

Lorenz– Finally consider the Lorenz system with the following parameters:

$$\begin{cases} \dot{x} = 10(y - x) \\ \dot{y} = x(28 - z) - y \\ \dot{z} = xy - \frac{8}{3} z \end{cases}$$

which is a model for atmospheric behavior and is known to exhibit a strange attractor. As seen in Fig. 7, the noisy trajectories are self-intersecting, which can cause difficulty for learning methods. The learned governing equations are:

$$\text{Noise} = 1.5\%: \begin{cases} \dot{x} = 10.000\, y - 10.005\, x \\ \dot{y} = x(27.799 - .995\, z) - .922\, y \\ \dot{z} = .999\, xy - 2.666\, z \end{cases}$$

$$\text{Noise} = 3\%: \begin{cases} \dot{x} = 9.872\, y - 9.899\, x \\ \dot{y} = x(27.963 - 1.005\, z) - .847\, y \\ \dot{z} = 1.003\, xy - 2.671\, z \end{cases}$$

with trial functions $f_{i,j,k} = x^{i-1} y^{j-1} z^{k-1}$ for $i+j+k \leq 4$. In both cases, the method is able to correctly identify the active features and approximate the coefficients. This shows the method's robustness to corrupted trajectories without the need for explicit denoising or pre-processing.

Note on Parameter Tuning– To detail a procedure for tuning the parameter, consider the logistic equation $\dot{x} = x - x^2$. The original data is simulated over $t \in [0, 15]$, with time-step $\Delta t = 0.01$, and initial data $x_0 = 0.01$. Noise is added directly to the simulated solution. Using our method with a optimally tuned γ and fixed $\epsilon = 2$ yields:

$$\dot{x} = 1.00x - 1.00x^2$$

the exact solution up to two significant digits. To obtain the optimal parameter, one may start with a relatively large value (compared to the coefficients), $\gamma = 2$. This results in all coefficients being zero. Next, reducing the parameter to $\gamma = 1.25$, yields a 1-sparse solution, $\dot{x} = 1.2657$. Next, reducing the parameter to the interval $0.05 < \gamma < 1$, the method selects 2-sparse solutions, in particular, x and x^2 are the only two terms selected, with a L^2 error between the approximation and the data of 0.05. If we decrease to $\gamma < 0.05$, the algorithm outputs 3-sparse or denser models; however, the denser models do not decrease the L^2 error between the approximation and the data. Thus, we see that when $\gamma < 0.05$, the method begins to overfit the data. The results are in Fig. 8, along side a comparison.

Automated procedures for parameter tuning can be constructed, for example, using various information criteria. In [19], the authors propose a method for learning dynamical models from an appropriate selection of trial functions. The trial functions that are used form a complete basis for smooth dynamics. Their method iteratively increases the number of allowed trial functions until the Bayesian information criterion exhibits a local maximum (which can prevent overfitting). In [20], the Akaike information criterion is used to compare several selected models that were learned using the sparse model selection approach. Methods for selecting parameters and comparing between learned model are important for applications.

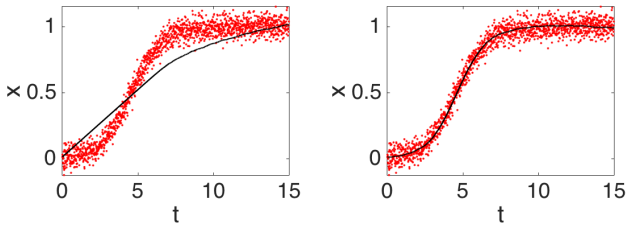

FIG. 8. **Comparison, Logistic Growth.** The data is simulated over $t \in [0, 15]$, with time-step $\Delta t = 0.01$, and initial data $x_0 = 0.01$. The black curves are the various solutions of the ODE when using the learned coefficients for each model. In both plots, two terms are chosen from the 15 term trial set. The first plot is shows the result of a sparse optimization method for fitting the trial functions directly to the ODE, inspired by the trial functions used in [3–6] but using a similar model to Eq. [7]. The terms 1 and x^{14} were chosen by the method. The second plot shows the solution using the method in this work, which selected the terms x and x^2.

CONCLUSION

A sparse regression approach for the identification of a dynamical system directly from noisy data is presented. Using the integral, or weak, formulation, noisy data is handled in a stable and robust way. For moderate noise levels, the algorithm produces an approximation to the noisy data without the need for completely re-solving the learned system of differential equations. Computational experiments show that this approach is stable with respect to data-size, robust to noise, and accurate when dynamic behavior is included in the dataset. The results presented here suggest the ability of sparse regression to reduce the dynamics of the data onto the important terms and to extract meaningful mathematical equations from the data.

* H.S. was supported by the AFOSR, FA9550-17-1-0125.

[1] J. Bongard and H. Lipson, Proceedings of the National Academy of Sciences **104**, 9943 (2007).
[2] M. Schmidt and H. Lipson, science **324**, 81 (2009).
[3] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Proceedings of the National Academy of Sciences **113**, 3932 (2016).
[4] H. Schaeffer, Proc. R. Soc. A **473**, 20160446 (2017).
[5] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Science Advances **3**, e1602614 (2017).
[6] G. Tran and R. Ward, arXiv preprint arXiv:1607.01067 (2016).
[7] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher, Proceedings of the National Academy of Sciences **110**, 6634 (2013).
[8] A. Mackey, H. Schaeffer, and S. Osher, Multiscale Modeling & Simulation **12**, 1800 (2014).
[9] T. Y. Hou, Q. Li, and H. Schaeffer, Journal of Computational Physics **288**, 150 (2015).
[10] G. Tran, H. Schaeffer, W. M. Feldman, and S. J. Osher, SIAM Journal on Applied Mathematics **75**, 1424 (2015).
[11] R. E. Caflisch, S. J. Osher, H. Schaeffer, and G. Tran, Communications in Mathematical Sciences **13**, 2155 (2015).
[12] V. Ozoliņš, R. Lai, R. Caflisch, and S. Osher, Proceedings of the National Academy of Sciences **111**, 1691 (2014).
[13] D. L. Donoho, IEEE transactions on information theory **41**, 613 (1995).
[14] R. Tibshirani, Journal of the Royal Statistical Society. Series B (Methodological) , 267 (1996).
[15] E. J. Candes, J. K. Romberg, and T. Tao, Communications on pure and applied mathematics **59**, 1207 (2006).
[16] E. Candes and J. Romberg, Inverse problems **23**, 969 (2007).
[17] P.-L. Lions and B. Mercier, SIAM Journal on Numerical Analysis **16**, 964 (1979).
[18] P. L. Combettes and J.-C. Pesquet, in *Fixed-point algorithms for inverse problems in science and engineering* (Springer, 2011) pp. 185–212.
[19] B. C. Daniels and I. Nemenman, Nature communications **6** (2015).
[20] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, arXiv preprint arXiv:1701.01773 (2017).