Discovery of Nonlinear Dynamical Systems using a Runge-Kutta Inspired Dictionary-based Sparse Regression Approach

Pawan Goyal* Peter Benner*

*Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany.
Email: goyalp@mpi-magdeburg.mpg.de, ORCID: 0000-0003-3072-7780
†Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany.
Email: benner@mpi-magdeburg.mpg.de, ORCID: 0000-0003-3362-4103

Abstract: Discovering dynamical models to describe underlying dynamical behavior is essential to draw decisive conclusions and engineering studies, e.g., optimizing a process. Experimental data availability notwithstanding has increased significantly, but interpretable and explainable models in science and engineering yet remain incomprehensible. In this work, we blend machine learning and dictionary-based learning with numerical analysis tools to discover governing differential equations from noisy and sparsely-sampled measurement data. We utilize the fact that given a dictionary containing huge candidate nonlinear functions, dynamical models can often be described by a few appropriately chosen candidates. As a result, we obtain interpretable and parsimonious models which are prone to generalize better beyond the sampling regime. Additionally, we integrate a numerical integration framework with dictionary learning that yields differential equations without requiring or approximating derivative information at any stage. Hence, it is utterly effective in corrupted and sparsely-sampled data. We discuss its extension to governing equations, containing rational nonlinearities that typically appear in biological networks. Moreover, we generalized the method to governing equations that are subject to parameter variations and externally controlled inputs. We demonstrate the efficiency of the method to discover a number of diverse differential equations using noisy measurements, including a model describing neural dynamics, chaotic Lorenz model, Michaelis-Menten Kinetics, and a parameterized Hopf normal form.

Keywords: Artificial intelligence, machine learning, dictionary learning, nonlinear dynamical systems, differential equations

Novelty statement: This work combines machine learning (dictionary-based) with a numerical integration scheme, namely a Runge-Kutta scheme to discover governing equations using corrupted and sparsely-sampled data. The method does not require the computation of derivative information to discover governing equations. Hence, it holds a key advantage when data are corrupted and sparsely sampled.

1 Introduction

Data-driven discovery of dynamic models has recently picked up much attention as there are revolutionary breakthroughs in data science and machine learning [22,29]. With the increasing ease of data availability and advances in machine learning, we can delve into analyzing data and identifying patterns to uncover dynamic models that faithfully describe the underlying dynamical behavior. Though inferring dynamic models have

been intensively studied in the literature, drawing conclusions and interpretations from them still remains strenuous. Moreover, extrapolation and generalization of models are limited beyond the training regime.

The sphere of identifying models using data is often referred to as system identification. For linear systems, there is an extensive collection of approaches [26, 42]. However, despite several decades of research on learning nonlinear systems [23, 25, 39], it is still far away from being as mature as linear systems. Inferring nonlinear systems often require a prior model hypothesis by practitioners. A compelling breakthrough towards discovering nonlinear governing equations appeared in [3, 37], where an approach based on genetic programming or symbolic regression is developed to identify nonlinear models using measurement data. It provides interpretable analytic models that accomplish a long-standing desire to the engineering community. A parsimonious model is determined by examining the Pareto font that discloses a tread-off between the identified model's complexity and accuracy. In a similar spirit, there have been efforts to develop sparsity promoting approaches to discover nonlinear dynamical systems [5, 6, 31, 32, 43]. It is often observed that the dynamics of physical processes can be given by collecting a few nonlinear feature candidates from a high-dimensional nonlinear function space, referred to as a feature dictionary. These sparsity-promoting methods are prone to discover models that are interpretable and parsimonious. Significant progress in solving sparse regression [16, 20, 40] and compressed sensing [7, 8, 14, 41] support developments of these approaches. Although all these methods have gained much popularity, the success of these methods largely depends on the feature candidates included in the dictionary and the ability to accurately approximating the derivative information using measurement data. A derivative approximation using sparsely sampled and noisy measurements impose a tough challenge though there are approaches to deal with noise, see, e.g., [10] We also highlight additional directions explored in the literature to discover nonlinear governing equations, which include discovery of models using time-series data [11], automated inference of dynamics [3, 12, 38], and equation-free modeling [24, 32, 46].

In this work, we re-conceptualize the problem of discovering nonlinear differential equations by blending sparse identification with a classical numerical integration tool. We here focus on a widely known integration scheme, namely *Runge-Kutta* 4th-order [1]. In contrast to previously studied sparse identification approaches, e.g., [3, 5, 43], our approach would not require direct access or approximation of temporal gradient information. Therefore, we do not commit errors due to a gradient approximation. The approach becomes an attractive choice when the collected measurement data are sparsely sampled and corrupted with noise. We mention that numerical integration-inspired (e.g., Runge-Kutta) neural network architecture designs have also studied in the literature and have observed their supreme performances in deep learning, see, e.g., [18, 19], and from the perspective of dynamical modeling, see, e.g., [33–35]. These methods yield black-box models, thus interpretable and generalization of these models are ambiguous.

What is more, we discuss an essential class of dynamic models that typically explains the dynamics of biological networks. It is also witnessed that regulatory and metabolic networks are sparse in nature, i.e., not all components influence each other. Furthermore, such dynamic models are often given by rational nonlinear functions. Consequently, the classical dictionary-based sparse identification ideology is not applicable as building all possible rational feature candidates is infeasible. To deal with this, the authors in [28] have recast the problem as finding the sparsest vector in a given null-space. However, computing a null space using corrupted measurement data is a non-trivial task though there is some work in the direction [17]. In this work, we instead characterize identifying rational functions as a ratio of two functions, where each function is identified using dictionary learning. Hence, we inherently retain the primary principle of sparse identification in the course of discovering models. In addition to these, we discuss the case where a dictionary contains parameterized candidates, e.g., $e^{\alpha x}$, where x is dependent variables, and α is an unknown parameter. We extend our discussion to parametric and controlled dynamic processes.

The organization of the paper is as follows. In Section 2, we briefly recap the Runge-Kutta 4th-order scheme that is typically used to integrate differential equations. After that, we propose a methodology to discover differential equations by synthesizing the integration scheme with sparse identification. Furthermore, since the method involves solving nonlinear and non-convex optimization problems that promote sparse solutions, Section 3 discusses algorithms inspired by a sparse-regression approach in [5, 40]. In Section 4, we examine a number of extensions to other classes of models, e.g., when governing equations are given by a ratio of two functions and involve model parameters and external control inputs. In the subsequent section, we illustrate the efficiency of the proposed methods by discovering a broad variety of benchmark examples, namely the chaotic Lorenz model, Fitz-Hugh Nagumo models, Michaelis-Menten Kinetics, and

parameterized Hopf normal norm. We extensively study the performance of the proposed approach even under noisy measurements and compare it to the approach proposed in [5]. We conclude the paper with a summary and high-priority research directions.

2 Discovering Nonlinear Governing Equations using a Runge-Kutta Inspired Sparse Identification

In this section, we are determined to discover nonlinear governing equations using sparsely sampled measurement data. These may be corrupted using experimental and/or sensor noise. We establish approaches by combining a numerical integration method and dictionary-based learning of the gradient field. As a result, we develop methodologies that allow us to discover nonlinear differential equations without the explicit need for derivative information, unlike the approach proposed in [5, 12, 43]. In this work, we utilize the widely employed approach to integrate differential equations, namely *Runge-Kutta* 4th*-order* (RK4) scheme, which is briefly outlined in the following.

2.1 Runge-Kutta 4th order scheme

The RK4 scheme is a widely-used method to solve an initial value problem. Let us consider an initial value problem as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.1}$$

where $\mathbf{x}(t) := \left[\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_n(t)\right]$ with $\mathbf{x}_j(t)$ being the jth element of the vector $\mathbf{x}(t)$. Assume that we aim at predicting $\mathbf{x}(t_{k+1})$ for a given $\mathbf{x}(t_k)$, where $k \in \{0, 1, \ldots, \mathcal{N}\}$. Then, using the RK4 scheme, $\mathbf{x}(t_{k+1})$ can be given as a weighted sum of four increments, which are the product of the time-step and gradient field information $\mathbf{f}(\cdot)$ at the specific locations. Precisely, it is given as

$$\mathbf{x}(t_{k+1}) \approx \mathbf{x}(t_k) + \frac{1}{6} h_k \left(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3 + \mathbf{k}_4\right), \quad h_k = t_{k+1} - t_k, \tag{2.2}$$

where

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}(t_k)), \quad \mathbf{k}_2 = \mathbf{f}\left(\mathbf{x}\left(t_k + h_k \frac{\mathbf{k}_1}{2}\right)\right), \quad \mathbf{k}_3 = \mathbf{f}\left(\mathbf{x}\left(t_k + h_k \frac{\mathbf{k}_2}{2}\right)\right), \quad \mathbf{k}_4 = \mathbf{f}\left(\mathbf{x}\left(t_k + h_k \mathbf{k}_3\right)\right).$$

The RK4 scheme as a network is illustrated in Figure 2.1(a). The local integration error due to the RK4 scheme is of $\mathcal{O}(h_k^5)$; hence, the approach is very accurate for smaller time-steps. Furthermore, if we integrate the equation (2.1) from the time t_0 to t_f, we can take $\mathcal{N}$ steps with time-steps $h_k, k \in \{1, \ldots, \mathcal{N}\}$ so that $t_f = t_0 + \sum_{i=0}^{\mathcal{N}} h_k$. In the rest of the paper, we use a short-hand notation for the step in (2.2) by $\mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_k), h_k\right)$, i.e.,

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k + h_k) \approx \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_k), h_k\right). \tag{2.3}$$

Lastly, we stress a point that the RK4 scheme readily handles integration backward in time, meaning that h_k in (2.2) can also be negative. Hence, we can predict both $y(t_{k+1})$ and $y(t_{k-1})$ using $y(t_k)$ very accurately using RK4 scheme.

2.2 Discovering nonlinear dynamical systems

Next, we develop a RK4-inspired sparse identification approach to discover governing equations. Precisely, we aim at disclosing the most parsimonious representation of the gradient field $\mathbf{f}(\mathbf{x}(t))$ in (2.1) using only a time-history of $\mathbf{x}(t)$. Assume that the data is sampled at the time instances $\{t_0, \ldots, t_{\mathcal{N}}\}$ and let us define time-steps $h_k := t_{k+1} - t_k$. Furthermore, for simplicity of notation, we assume that the data follows RK4 exactly, but the method is not limited to it. Consequently, we form two data matrices:

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}(t_2) \\ \mathbf{x}(t_3) \\ \vdots \\ \mathbf{x}(t_{\mathcal{N}}) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1(t_2) & \mathbf{x}_2(t_2) & \cdots & \mathbf{x}_n(t_2) \\ \mathbf{x}_1(t_3) & \mathbf{x}_2(t_3) & \cdots & \mathbf{x}_n(t_3) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1(t_{\mathcal{N}}) & \mathbf{x}_2(t_{\mathcal{N}}) & \cdots & \mathbf{x}_n(t_{\mathcal{N}}) \end{bmatrix} \quad \text{and} \quad \mathbf{X}_{\mathcal{F}}(\mathbf{f}) := \begin{bmatrix} \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_0), h_1\right) \\ \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_1), h_2\right) \\ \vdots \\ \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_{\mathcal{N}-1}), h_{\mathcal{N}}\right) \end{bmatrix}. \tag{2.4}$$

The next important ingredients to sparse identification is the construction of a huge symbolic dictionary $\mathbf{\Phi}$, containing potential nonlinear features. So, the function $\mathbf{f}(\cdot)$ can be given by a linear combination of few terms from the dictionary. For example, one can consider a dictionary containing, polynomial, exponential, and trigonometric functions, which, for any given vector $\mathbf{v} := \left[\mathbf{v}_1, \ldots, \mathbf{v}_n\right]$ can be given as:

$$\mathbf{\Phi}(\mathbf{v}) = \left[1, \mathbf{v}, \mathbf{v}^{\mathcal{P}_2}, \mathbf{v}^{\mathcal{P}_3}, \ldots, \mathbf{e}^{-\mathbf{v}}, \mathbf{e}^{-2\mathbf{v}}, \ldots, \mathbf{sin}(\mathbf{v}), \mathbf{cos}(\mathbf{v}), \ldots\right] \tag{2.5}$$

in which $\mathbf{v}^{\mathcal{P}_i}, i \in \{2, 3\}$ denote high-order polynomials, e.g., $\mathbf{v}^{\mathcal{P}_2}$ contains all possible degree-2 polynomials of elements of $\mathbf{v}$ as:

$$\mathbf{v}^{\mathcal{P}_2} = \left[\mathbf{v}_1^2, \mathbf{v}_1\mathbf{v}_2, \ldots, \mathbf{v}_2^2, \mathbf{v}_2\mathbf{v}_3, \ldots, \mathbf{v}_n^2\right] \tag{2.6}$$

Each element in the dictionary $\mathbf{\Phi}$ is a potential candidate to describe the function $\mathbf{f}$. Moreover, depending on applications, one may take the help of experts and include empirical knowledge to construct a meaningful feature dictionary.

Having paradise of an extensive dictionary, one has many choices to choose candidates from the dictionary. However, our goal is to choose as few candidates as possible, describing the nonlinear function $\mathbf{f}$ in (2.1). Hence, we set up a sparsity-promoting optimization problem to pick few candidate functions from the dictionary, e.g.,

$$\mathbf{f}_k(\mathbf{x}(t)) = \mathbf{\Phi}(\mathbf{x}(t))\boldsymbol{\theta}_k, \tag{2.7}$$

where $\mathbf{f}_k : \mathbb{R}^n \to \mathbb{R}$ is the kth element of $\mathbf{f}$, and $\boldsymbol{\theta}_k$ a sparse vector; hence, selecting appropriate candidates from the dictionary determines governing equations. As a result, we can write the function $\mathbf{f}(\cdot)$ in (2.1) as follows:

$$\mathbf{f}(\mathbf{x}) = \left[\mathbf{f}_1(\mathbf{x}),\ \mathbf{f}_2(\mathbf{x}),\ \ldots,\ \mathbf{f}_n(\mathbf{x})\right] = \left[\mathbf{\Phi}(\mathbf{x})\boldsymbol{\theta}_1, \mathbf{\Phi}(\mathbf{x})\boldsymbol{\theta}_2, \cdots, \mathbf{\Phi}(\mathbf{x})\boldsymbol{\theta}_n\right] = \mathbf{\Phi}(\mathbf{x})\mathbf{\Theta}, \tag{2.8}$$

where $\mathbf{\Theta} = \left[\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_n\right]$. This allows to articulate our optimization problem that aims at discovering governing equations – that is to find the sparsest $\mathbf{\Theta}$, satisfying

$$\mathbf{X} = \mathbf{X}_{\mathcal{F}}\left(\mathbf{f}\right), \quad \text{where} \quad \mathbf{f}(\mathbf{x}) = \mathbf{\Phi}(\mathbf{x})\mathbf{\Theta}. \tag{2.9}$$

Once we identify $\mathbf{\Theta}$ or $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\}$, the dynamic model can be given as

$$\left[\mathbf{x}_1(t),\ \mathbf{x}_2(t),\ \ldots,\ \mathbf{x}_n(t)\right] = \left[\mathbf{\Phi}(\mathbf{x})\boldsymbol{\theta}_1,\ \mathbf{\Phi}(\mathbf{x})\boldsymbol{\theta}_2,\ \ldots, \mathbf{\Phi}(\mathbf{x})\boldsymbol{\theta}_n\right].$$

We referred to the proposed approach as Runge-Kutta inspired sparse identification (RK4-SINDy). We depict all the essential steps for RK4-SINDy to discover governing equations in Figure 2.1 through the Fitz-Hugh Nagumo model (details of the model are provided later).

We take the opportunity to stress imperative advantages of RK4-SINDy. That is – to discover nonlinear differential equations, we do not require derivative information of $\mathbf{x}(t)$ at any step. We only hypothesize that the gradient field can be given by selecting appropriate features from a dictionary containing a vast number of possible nonlinear features. Consequently, we expect to discover good quality models when data are sparsely collected and/or are corrupted, and this is what we manifest in our results in Section 5. Interestingly, the approach readily handles irregular time-steps.

When the data are corrupted with noise or does not follow RK4 exactly, then we may need to regularize the above optimization problem. Since the l_1-regularization promotes sparsity in the solution, one can solve an l_1-regularized optimization problem:

$$\min_{\mathbf{\Theta}} \left\|\mathbf{X} - \mathbf{X}_{\mathcal{F}}\left(\mathbf{\Phi}(\cdot)^{\otimes}\mathbf{\Theta}\right)\right\| + \lambda \left\|\mathbf{\Theta}\right\|_{l_1}. \tag{2.10}$$

As discussed in Subsection 2.1, the RK4 scheme can accurately predict both $\mathbf{x}(t_{i+1})$ and $\mathbf{x}(t_{i-1})$ using $\mathbf{x}(t_i)$. Therefore, the following also holds:

$$\mathbf{X}^{\mathsf{b}} = \mathbf{X}^{\mathsf{b}}_{\mathcal{F}}\left(\mathbf{f}\right),$$

where

$$\mathbf{X}^{\mathsf{b}} := \begin{bmatrix} \mathbf{x}(t_0) \\ \mathbf{x}(t_1) \\ \vdots \\ \mathbf{x}(t_{\mathcal{N}-1}) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1(t_0) & \mathbf{x}_2(t_0) & \cdots & \mathbf{x}_n(t_0) \\ \mathbf{x}_1(t_1) & \mathbf{x}_2(t_1) & \cdots & \mathbf{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1(t_{\mathcal{N}-1}) & \mathbf{x}_2(t_{\mathcal{N}-1}) & \cdots & \mathbf{x}_n(t_{\mathcal{N}-1}) \end{bmatrix} \quad \text{and} \quad \mathbf{X}^{\mathsf{b}}_{\mathcal{F}}(\mathbf{f}) := \begin{bmatrix} \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_1), -h_1\right) \\ \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_2), -h_2\right) \\ \vdots \\ \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{f}, \mathbf{x}(t_{\mathcal{N}}), -h_{\mathcal{N}}\right) \end{bmatrix}.$$

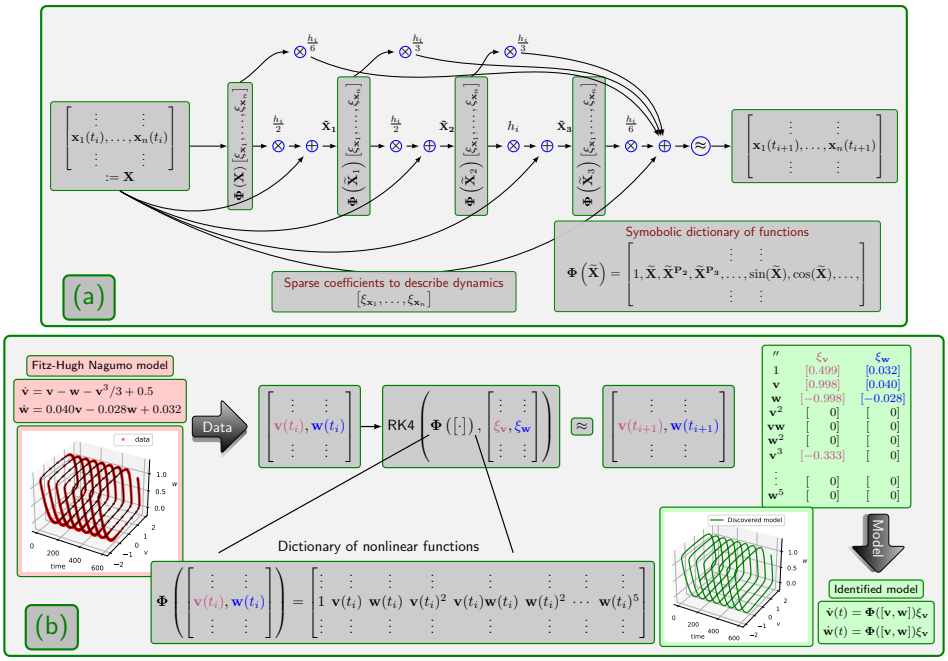


Figure 2.1: In (a), we show the RK4 scheme to predict variables at the next time-step as a network. It resembles a residual-type network with skip connections. In (b), we present a systematic illustration of RK4-SINDy approach to discover governing equations using the Fitz-Hugh Nagumo model. In the first step, we collect a time history of variables $\mathbf{v}(t)$ and $\mathbf{w}(t)$. Next, we build a symbolic feature dictionary $\mathbf{\Phi}$, containing potential features. It is followed by solving a nonlinear sparse regression problem to pick the right features from the dictionary (encoded in sparse vectors $\xi_{\mathbf{v}}$ and $\xi_{\mathbf{w}}$). Here, we presume that variables at the next time steps are given by following the RK4 scheme. The non-zero entries in vectors $\xi_{\mathbf{v}}$ and $\xi_{\mathbf{w}}$ determine the right-hand side of the differential equations. As shown, we pick the right features from the dictionary upon solving the optimization problem, and corresponding coefficients are 0.1% accurate.

Therefore, we can have a more involved optimization by including both forward and backward predictions in time. This helps particularly in noisy measurement data. In the next subsection, we discuss an efficient procedure to solve the optimization problem (2.9).

3 Algorithms to Solve Nonlinear Sparse Regression Problems

Several methodologies exist to solve linear optimization problems that yield a sparse solution, see ,e.g., LASSO [16, 40]. However, the optimization problem (2.9) is nonlinear and likely non-convex. There are some developments in solving sparsity-constrained nonlinear optimization problems; see, e.g., [2, 45]. Though these methods enjoy many nice theoretical properties, they typically require a priory the maximum number of non-zero elements in the solutions, which is often unknown to us. Also, they are computationally demanding.

Here, we propose two simple gradient-based sequential thresholding schemes, similar to the one discussed in [5] for linear problems. In these schemes, we first solve the nonlinear optimization problem (2.9) using a (stochastic-) gradient descent method to obtain $\mathbf{\Theta}_1$, followed by applying a thresholding to $\mathbf{\Theta}_1$.

3.1 Fix cutoff thresholding

In the first approach, we define a cutoff value λ and set all the coefficients smaller than λ to zero. We then update the remaining non-zero coefficients by solving the optimization problem (2.9) again, followed by employing the thresholding. We repeat the procedure until all the non-zero coefficients are equal to or larger than λ. This procedure is efficient as the current value of non-zero coefficients can be used as an initial guess for the next iteration, and the optimal $\mathbf{\Theta}$ can be found with a little computational effort. Note the

Algorithm 1 Fix Cutoff Thresholding Procedure

Input: Measurement data $\{\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_{\mathcal{N}})\}$ and the cutoff parameter λ.

Solve the optimization problem (2.9) to get $\boldsymbol{\Theta}$.
$\texttt{small\_idx} = (|\boldsymbol{\Theta}| < \lambda)$ ▷ Determine indices at which coefficients are less λ
$\texttt{Err} = \|\boldsymbol{\Theta}\,(\texttt{small\_idx})\,\|$
while $\texttt{Err} \neq 0$ **do**
  Update $\boldsymbol{\Theta}$ by solving the optimization problem (2.9) with the constraint $\boldsymbol{\Theta}\,(\texttt{small\_idx}) = 0$
  $\texttt{small\_idx} = (|\boldsymbol{\Theta}| < \lambda)$ ▷ Determine indices at which coefficients are less λ
  $\texttt{Err} = \|\boldsymbol{\Theta}\,(\texttt{small\_idx})\,\|$

Output: The sparse $\boldsymbol{\Theta}$ that picks right features from the dictionary.

Algorithm 2 Iterative Cutoff Thresholding Procedure

Input: Measurement data $\{\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_{\mathcal{N}})\}$.

Construct $\mathbf{X}$ using measurement data as in (2.4).
Solve the optimization problem (2.9) to get $\boldsymbol{\Theta}$.
$\mathcal{E} := \|\mathbf{X} - \mathbf{X}_{\mathcal{F}}\,(\boldsymbol{\Phi}(\cdot)\boldsymbol{\Theta})\,\|$, where $\mathbf{X}_{\mathcal{F}}$ is defined in (2.4).
while $\mathcal{E} \leq \texttt{tol}$ **do**
  Determine the smallest non-zero coefficient of $\texttt{abs}(\boldsymbol{\Theta})$, denoted by $\lambda_{\texttt{small}}$.
  $\texttt{small\_idx} = (|\boldsymbol{\Theta}| < \lambda_{\texttt{small}})$ ▷ Determine indices at which coefficients are less λ
  Update $\boldsymbol{\Theta}$ by solving the optimization problem (2.9) with the constraint $\boldsymbol{\Theta}\,(\texttt{small\_idx})$.
  $\mathcal{E} := \|\mathbf{X} - \mathbf{X}_{\mathcal{F}}\,(\boldsymbol{\Phi}(\cdot)\boldsymbol{\Theta})\,\|$.

Output: The sparse $\boldsymbol{\Theta}$ that picks right features from the dictionary.

cutoff parameter λ is important to obtain a suited sparse solution, but it can be found using the concept of cross-validation. We sketch the discussed procedure in Algorithm 1.

3.2 Iterative cutoff thresholding

In the fix cutoff thresholding approach, we need to pre-define the cutoff value for thresholding. A suitable value of it needs to be found by an iterative procedure. In our empirical observations, applying fix thresholding at each iteration does not yield the most sparse solution in many instances. To circumvent this, we propose an iterative way of thresholding – that is as follows. In the first step, we solve the optimization problem (2.9) for $\boldsymbol{\Theta}$. Then, we determine the smallest non-zero coefficients of $|\boldsymbol{\Theta}|$ followed by setting all the coefficients smaller than this to zero. Like the previous approach, we update the remaining non-zero coefficients by solving the optimization problem (2.9). We repeat the step of finding the smallest non-zero coefficient of the updated $|\boldsymbol{\Theta}|$ and setting it to zero. We iterate the procedure until the loss of data fidelity is less than a given tolerance. Visually, it can be anticipated using the curve between the data-fitting and number of non-zero elements in Θ, which typically exhibit an *elbow*-type curve. We shall see in our result section (Section 5). We sketch the step of the procedure in Algorithm 2.

We note that the successive iterations converge faster to the optimal value after the first thresholding as we choose the coefficients after applying thresholding as the initial guess. Moreover, in our experiments, we observe that this thresholding approach yields better results, particularly when data are corrupted with noise. However, it may be computationally more expensive than the fixed cutoff thresholding approach as it may need more iterations to converge. Therefore, an efficient approach combining fixed and iterative thresholding approaches is a worthy future research direction.

4 A Number of Possible Extensions

In this section, we discuss several extensions to the methodology proposed in Section 2, generalizing to a large class of problems. First, we discuss the discovery of governing differential equations given by a ratio of two functions. Next, we investigate the case in which a symbolic dictionary is parameterized. This is of particular interest when governing equations are expected to have candidate features, e.g., $\mathbf{e}^{\alpha\mathbf{x}(t)}$, where α is unknown. We further extend our discussion to parameterized and externally controlled governing equations.

4.1 Governing equations as a ratio of two functions

There are many instances, where the governing equations are given as a ratio of two nonlinear functions. Such equations frequently appear in the modeling of biological networks. For simplicity, we here examine a scalar problem; however, the extension to multi-dimensional cases readily follows. Consider governing equations of the form:

$$\dot{\mathbf{x}}(t) = \frac{\mathbf{g}(\mathbf{x})}{1+\mathbf{h}(\mathbf{x})}, \tag{4.1}$$

where $\mathbf{g}(\mathbf{x}) : \mathbb{R} \to \mathbb{R}$ and $\mathbf{h}(\mathbf{x}) : \mathbb{R} \to \mathbb{R}$ are continuous nonlinear functions. Here again, the observation is that the functions $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ can be given as linear combinations of a few terms from corresponding dictionaries. Hence, we can cast the problem of identifying the model (4.1) as a dictionary-based discovery of governing equations. Let us consider two symbolic dictionaries:

$$\mathbf{\Phi}^{(\mathbf{g})}(\mathbf{x}) = \left[1, \mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \ldots, \sin(\mathbf{x}), \cos(\mathbf{x}), \sin(\mathbf{x}^2), \cos(\mathbf{x}^2), \sin(\mathbf{x}^3), \sin(\mathbf{x}^3), \ldots\right], \tag{4.2}$$

$$\mathbf{\Phi}^{(\mathbf{h})}(\mathbf{x}) = \left[\mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \ldots, \sin(\mathbf{x}), \cos(\mathbf{x}), \sin(\mathbf{x}^2), \cos(\mathbf{x}^2), \sin(\mathbf{x}^3), \sin(\mathbf{x}^3), \ldots\right]. \tag{4.3}$$

Consequently, the functions $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ can be given by

$$\mathbf{g}(\mathbf{x}) = \mathbf{\Phi}^{(\mathbf{g})}(\mathbf{x})\boldsymbol{\theta}_{\mathbf{g}}, \tag{4.4}$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{\Phi}^{(\mathbf{h})}(\mathbf{x})\boldsymbol{\theta}_{\mathbf{h}}, \tag{4.5}$$

where $\theta_{\mathbf{g}}$ and $\theta_{\mathbf{h}}$ are sparse vectors. Then, we can readily apply the framework discussed in the previous section by assuming $\mathbf{f}(\mathbf{x}) := \dfrac{\mathbf{g}(\mathbf{x})}{1+\mathbf{h}(\mathbf{x})}$ in (2.1). We can determine sparse coefficients $\boldsymbol{\theta}_{\mathbf{g}}$ and $\boldsymbol{\theta}_{\mathbf{h}}$ by employing the thresholding concepts presented in Algorithms 1 and 2. These are possible because the algorithms are gradient-based and we only need to compute gradients with respect to $\boldsymbol{\theta}_{\mathbf{g}}$ and $\boldsymbol{\theta}_{\mathbf{h}}$.

Furthermore, we notice that it is worthwhile to consider governing equations of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{k}(\mathbf{x}) + \frac{\mathbf{g}(\mathbf{x})}{1+\mathbf{h}(\mathbf{x})}. \tag{4.6}$$

Indeed, the model (4.6) can be rewritten in the form considered in (4.1). But it is rather efficient to consider the form (4.6). We illustrate it with the following example:

$$\dot{\mathbf{x}}(t) = -\mathbf{x}(t) - \frac{\mathbf{x}(t)}{1+\mathbf{x}(t)}, \tag{4.7}$$

which fits to the form considered in (4.6). In this case, all nonlinear functions $\mathbf{k}(\cdot), \mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ are of degree-1 polynomials. On the other hand, if the model (4.7) is written in the form (4.1), then we have

$$\dot{\mathbf{x}}(t) = \frac{-1-\mathbf{x}(t)-\mathbf{x}(t)^2}{1+\mathbf{x}(t)}. \tag{4.8}$$

Thus, the nonlinear functions $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ in (4.1) are of degrees 2 and 1, respectively. This gives a hint that if we aim at learning governing equations using sparse identification, it might be efficient to consider the form (4.6) from the complexity of the dictionary. It becomes even more adequate in multi-dimensional differential equations. To discover dynamic model of the form (4.6), we extend the idea of learning nonlinear functions using dictionaries. Let us construct three dictionaries as follows:

$$\mathbf{\Phi}^{(\mathbf{k})}(\mathbf{x}) = \left[1, \mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \ldots, \sin(\mathbf{x}), \cos(\mathbf{x}), \sin(\mathbf{x}^2), \cos(\mathbf{x}^2), \sin(\mathbf{x}^3), \sin(\mathbf{x}^3), \ldots\right], \tag{4.9}$$

$$\mathbf{\Phi}^{(\mathbf{g})}(\mathbf{x}) = \left[1, \mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \cdots, \sin(\mathbf{x}), \cos(\mathbf{x}), \sin(\mathbf{x}^2), \cos(\mathbf{x}^2), \sin(\mathbf{x}^3), \sin(\mathbf{x}^3), \cdots\right], \tag{4.10}$$

$$\mathbf{\Phi}^{(\mathbf{h})}(\mathbf{x}) = \left[\mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \ldots, \sin(\mathbf{x}), \cos(\mathbf{x}), \sin(\mathbf{x}^2), \cos(\mathbf{x}^2), \sin(\mathbf{x}^3), \sin(\mathbf{x}^3), \cdots\right]. \tag{4.11}$$

Then, we believe that the nonlinear functions in (4.6) can be given as a sparse linear combination of the dictionaries, i.e.,

$$\mathbf{k}(\mathbf{x}) = \mathbf{\Phi}^{(\mathbf{k})}(\mathbf{x})\boldsymbol{\theta}_{\mathbf{k}}, \qquad \mathbf{g}(\mathbf{x}) = \mathbf{\Phi}^{(\mathbf{g})}(\mathbf{x})\boldsymbol{\theta}_{\mathbf{g}}, \qquad \mathbf{h}(\mathbf{x}) = \mathbf{\Phi}^{(\mathbf{h})}(\mathbf{x})\boldsymbol{\theta}_{\mathbf{h}}. \tag{4.12}$$

To determine the sparse coefficients $\{\boldsymbol{\theta}_{\mathbf{k}}, \boldsymbol{\theta}_{\mathbf{g}}, \boldsymbol{\theta}_{\mathbf{h}}\}$, we can employ the RK4-SINDy framework, and Algorithms 1 and 2. We will illustrate this approach to discover an enzyme kinetics in Section 5.4 that is given as a rational function.

4.2 Discovering of parametric and externally controlled equations

The RK4-SINDy immediately embraces the discovering of governing equations that are parametric and externally controlled. Let us begin with an externally controlled dynamic models of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{4.13}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$ are state and controlled input vectors. The goal here is to discover $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ using the state trajectory $\mathbf{x}(t)$ generated using a controlled input $u(t)$. We aim at discovering governing equations using dictionary-based identification. Like discussed in Section 2, we construct a symbolic dictionary $\mathbf{\Phi}$ of possible candidate features using $\mathbf{x}$ and $\mathbf{u}$, i.e.,

$$\mathbf{\Phi}(\mathbf{x}, \mathbf{u}) = \left[1, \mathbf{x}^\top, \mathbf{u}^\top, \left(\mathbf{x}_{\mathbf{u}}^{\mathcal{P}_2}\right)^\top, \left(\mathbf{x}_{\mathbf{u}}^{\mathcal{P}_3}\right)^\top\right], \tag{4.14}$$

where $\left(\mathbf{x}_{\mathbf{u}}^{\mathcal{P}_i}\right)^\top$ consists polynomial terms of degree-i, i.e., $\left(\mathbf{x}_{\mathbf{u}}^{\mathcal{P}_2}\right)^\top$ contains degree-2 polynomial terms including cross terms:

$$\left(\mathbf{x}_{\mathbf{u}}^{\mathcal{P}_2}\right)^\top = \left[\mathbf{x}_1^2, \ldots, \mathbf{x}_n^2, \mathbf{u}_1^2, \ldots, \mathbf{u}_m^2, \mathbf{x}_1\mathbf{u}_1, \ldots, \mathbf{x}_n\mathbf{u}_1, \mathbf{x}_1\mathbf{u}_2, \ldots, \mathbf{x}_n\mathbf{u}_m\right], \tag{4.15}$$

where $\mathbf{u}_i$ is the i-th element of $\mathbf{u}$. Using measurements of $\mathbf{x}$ and $\mathbf{u}$, we can cast the problem exactly as done in Section 2 by assuming that $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ can be determined by selecting appropriate functions from the dictionary $\mathbf{\Phi}(\mathbf{x}, \mathbf{u})$. Similarly, system parameters can also be incorporated to discover parametric differential equations of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mu), \tag{4.16}$$

where $\mu \in \mathbb{R}^p$ is the system parameters. It can be considered as a special case of (4.13) since a constant input can be thought of as a parameter in the course of discovering governing equations. We illustrate RK4-SINDy for discovering parametrized Hopf normal form using measurement data (see Subsection 5.5).

4.3 Parameterized dictionary

The success of the sparse identification highly depends on the quality of a constructed feature dictionary. In other words, the dictionary should contain right features in which governing differential equations can be given as a linear combination of few terms from the dictionary. However, it becomes a challenging task when one aims at including, for instance, trigonometry or exponential functions (e.g., $\sin(ax), \mathbf{e}^{(bx)}$), where $\{a, b\}$ are unknown. In an extreme case, one might think of including $\sin(\cdot)$ and $\mathbf{e}(\cdot)$ for each possible value of a and b. This would lead to the dictionary of infinite dimension, hence becomes intractable. To illustrate it, we consider the governing equation as follows:

$$\dot{\mathbf{x}}(t) = -\mathbf{x}(t) + \exp(-1.75\mathbf{x}(t)). \tag{4.17}$$

Let us assume that we concern about discovering the model (4.17) using a time history of $\mathbf{x}(t)$ without any prior knowledge except that we expect exponential nonlinearities. It may be gathered with the help of

experts or from empirical knowledge. For instance, an electrical circuit modeling containing diode components typically involves exponential nonlinearities, but the corresponding coefficient is unknown.

We conventionally build a dictionary containing exponential functions using several possible coefficients as follows:

$$\mathbf{\Phi}(\mathbf{x}) = \left[1, \mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \ldots, \mathbf{e}^{\mathbf{x}}, \mathbf{e}^{-\mathbf{x}}, \mathbf{e}^{2\mathbf{x}}, \mathbf{e}^{-2\mathbf{x}} \ldots, \sin(\mathbf{x}), \cos(\mathbf{x}), \ldots\right]. \tag{4.18}$$

However, it is impossible to add all infinitely exponential terms with different coefficients in the dictionary. As a remedy, we discuss the idea of a parameterized dictionary that was also discussed in [9]:

$$\mathbf{\Phi}_{\Xi}(\mathbf{x}) = \left[1, \mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \ldots, \sin(\eta_1\mathbf{x}), \cos(\eta_2\mathbf{x}), \sin(\eta_3\mathbf{x}^2), \cos(\eta_4\mathbf{x}^2), \ldots, \mathbf{e}^{\eta_5\mathbf{x}}, \mathbf{e}^{\eta_6\mathbf{x}^2}, \ldots,\right] \tag{4.19}$$

where $\Xi = \{\xi_1, \xi_2, \ldots\}$. In this case, we do not need to include all frequencies for trigonometric functions and coefficients for exponential functions. However, it comes at the cost of finding suitable coefficients $\{\eta_i\}$'s, along with a vector, selecting right features from the dictionary. Since we solve optimization problems, e.g., (2.9) using a gradient descent, we can easily incorporate the parameters η_i's along with θ_i's as learning parameters and can readily employ Algorithms 1 and 2 with a little alteration.

5 Results

Here, we demonstrate the success of RK4-SINDy to discover governing equations using measurement data through a number of examples of different complexity[1]. In the first example, we consider simple illustrative examples, namely, linear and nonlinear damped oscillators. Using the linear damped oscillator, we perform a comprehensive study under various conditions, i.e., the robustness of the approach to sparsely sampled and highly corrupted data. We compare the performance of our approach to discover governing equations with [5]; we refer to it as Std-SINDy[2]. In the second example, we study the chaotic Lorenz example and show that RK4-SINDy determines the governing equations, exhibiting the chaotic behavior accurately. In the third example, we discover neural dynamics from measurement data using RK4-SINDy. As the fourth example, we illustrate the discovery of a model that describes the dynamics of enzyme activity and contains rational nonlinearities. In the last example, we showcase that RK4-SINDy also successfully discovers the parametric Hopf normal form from collected noisy measurement data for various parameters.

5.1 Two-dimensional Damped Oscillators

As simple illustrative examples, we consider two-dimensional damped harmonic oscillators. These can be given by linear and nonlinear models. We begin by considering the linear one.

5.1.1 Linear damped oscillator

Consider a 2D linear damped oscillator whose dynamics is given by:

$$\dot{\mathbf{x}}(t) = -0.1\mathbf{x}(t) + 2.0\mathbf{y}(t), \tag{5.1a}$$

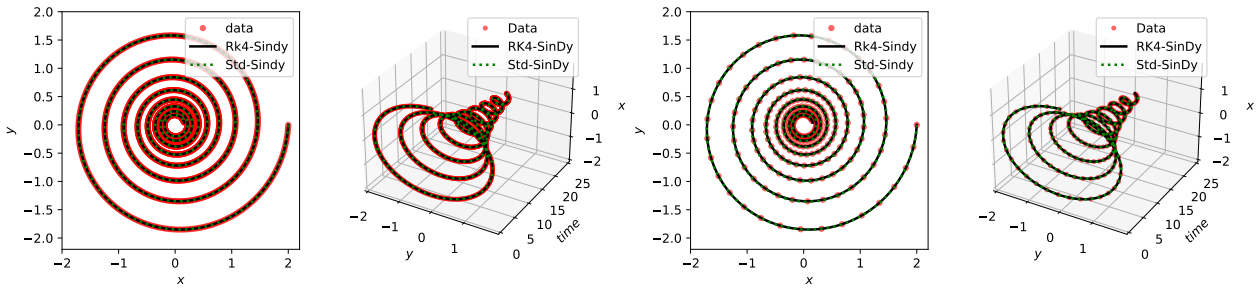
$$\dot{\mathbf{y}}(t) = -2.0\mathbf{x}(t) - 0.1\mathbf{y}(t). \tag{5.1b}$$

To infer governing equations from measurement data, we first assume to have clean data at a regular time-step dt. We then build a symbolic dictionary containing polynomial nonlinearities up to the degree of 5. Next, we learn governing equations using RK4-SINDy (Algorithm 1 with $\lambda = 5 \cdot 10^{-2}$) and observe the quality of inferred equations for different dt. We also present a comparison with Std-SINDy.

The results are shown in Figure 5.1 and Table 5.1. We notice that RK4-SINDy is impressively robust with the variation in time-step as compared to Std-SINDy, and discovers the governing equations accurately. We also emphasis that for large time-steps, Std-SINDy fails to capture dynamics; in fact, for a time-step $\mathsf{dt} = 5 \cdot 10^{-1}$, Std-SINDy even yields unstable models, see Figure 5.1d.

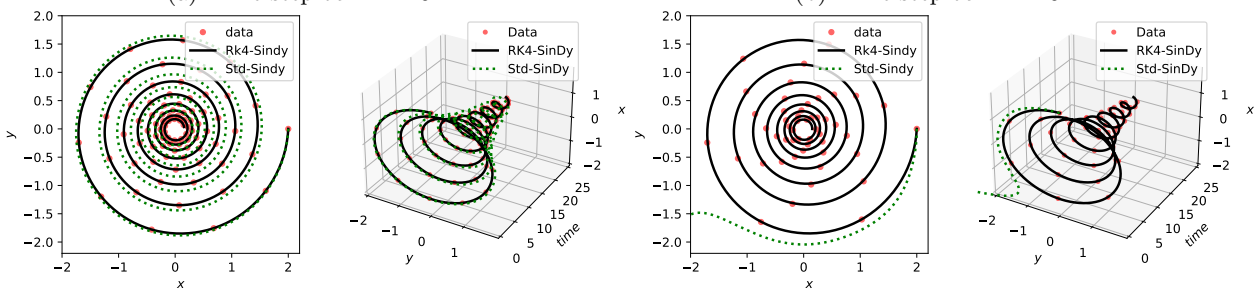
[1] Most of all examples are taken from [5]

[2] We use the Python implementation of the method, the so-called `PySINDy` [13].



(a) Time step $\mathsf{dt} = 1 \cdot 10^{-2}$.

(b) Time step $\mathsf{dt} = 1 \cdot 10^{-1}$.



(c) Time step $\mathsf{dt} = 3 \cdot 10^{-1}$.

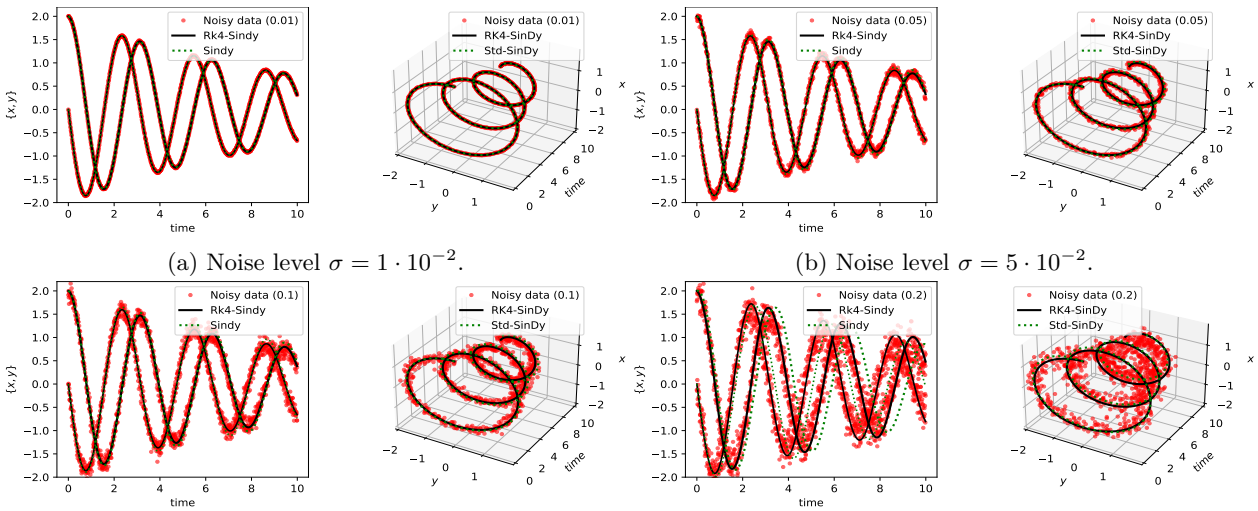
(d) Time step $\mathsf{dt} = 5 \cdot 10^{-1}$.

Figure 5.1: Linear 2D model: Identified models using data at various regular time-step.

Time step	RK4-SINDy	Std-SINDy
$1 \cdot 10^{-2}$	$\dot{\mathbf{x}}(t) = -0.100\mathbf{x}(t) + 2.000\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.001\mathbf{x}(t) - 0.100\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = -0.100\mathbf{x}(t) + 2.000\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.000\mathbf{x}(t) - 0.100\mathbf{y}(t)$
$1 \cdot 10^{-1}$	$\dot{\mathbf{x}}(t) = -0.100\mathbf{x}(t) + 2.001\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.001\mathbf{x}(t) - 0.100\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = -0.098\mathbf{x}(t) + 1.987\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -1.988\mathbf{x}(t) - 0.098\mathbf{y}(t)$
$3 \cdot 10^{-1}$	$\dot{\mathbf{x}}(t) = -0.101\mathbf{x}(t) + 2.002\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.002\mathbf{x}(t) - 0.101\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = -0.078\mathbf{x}(t) + 1.884\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -1.906\mathbf{x}(t) - 0.084\mathbf{y}(t)$
$5 \cdot 10^{-1}$	$\dot{\mathbf{x}}(t) = -0.103\mathbf{x}(t) + 2.011\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.011\mathbf{x}(t) - 0.103\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = 1.688\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -1.864\mathbf{x}(t) - 0.123\mathbf{x}(t)^2$ $- 0.146\mathbf{x}(t)^2\mathbf{y}(t) + 0.115\mathbf{x}(t)^4\mathbf{y}(t)$ $+ 0.133\mathbf{x}(t)^3\mathbf{y}(t)$

Table 5.1: Linear 2D model: The discovered governing equations using RK4-SINDy and Std-SINDy are reported for different regular time-steps at which data are collected.

Next, we study the performance of both methodologies under corrupted data. We corrupt the measurement data by adding zero-mean Gaussian white noise of different variances. We present the results in Figure 5.2 and Table 5.2 and notice that RK4-SINDy can discover better quality sparse parsimonious models as compared to Std-SINDy even under significantly corrupted data. It is predominately visible in Figure 5.2d.



(a) Noise level $\sigma = 1 \cdot 10^{-2}$.

(b) Noise level $\sigma = 5 \cdot 10^{-2}$.

(c) Noise level $\sigma = 1 \cdot 10^{-1}$.

(d) Noise level $\sigma = 2 \cdot 10^{-1}$.

Figure 5.2: Linear 2D model: The transient responses of discovered models using corrupted data are compared.

Noise level	RK4-SINDy	Std-SINDy
$1 \cdot 10^{-2}$	$\dot{\mathbf{x}}(t) = -0.099\mathbf{x}(t) + 1.999\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.000\mathbf{x}(t) - 0.101\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = -0.102\mathbf{x}(t) + 1.999\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.002\mathbf{x}(t) - 0.101\mathbf{y}(t)$
$5 \cdot 10^{-2}$	$\dot{\mathbf{x}}(t) = -0.095\mathbf{x}(t) + 1.999\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -1.995\mathbf{x}(t) - 0.105\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = -0.078\mathbf{x}(t) + 2.001\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -1.995\mathbf{x}(t) - 0.105\mathbf{y}(t)$
$1 \cdot 10^{-1}$	$\dot{\mathbf{x}}(t) = -0.091\mathbf{x}(t) + 1.985\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -1.997\mathbf{x}(t) - 0.103\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = -0.076\mathbf{x}(t) + 1.969\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.008\mathbf{x}(t) - 0.095\mathbf{y}(t)$
$2 \cdot 10^{-1}$	$\dot{\mathbf{x}}(t) = -0.177\mathbf{x}(t) + 2.053\mathbf{y}(t)$ $- 0.063\mathbf{x}^2\mathbf{y} + 0.059\mathbf{x}\mathbf{y}^2$ $\dot{\mathbf{y}}(t) = -1.960\mathbf{x}(t)$	$\dot{\mathbf{x}}(t) = -0.173\mathbf{x}(t) + 1.950\mathbf{y}(t) - 0.056\mathbf{y}(t)^2$ $+ 0.059\mathbf{x}(t)^3 - 0.079\mathbf{x}(t)^2\mathbf{y} + 0.095$ $\dot{\mathbf{y}}(t) = -2.005\mathbf{x}(t) - 0.095\mathbf{y}(t) + 0.069\mathbf{x}(t)\mathbf{y}(t)$ $+ 0.062\mathbf{x}(t)^3 + 0.060\mathbf{x}(t)\mathbf{y}(t)^2$

Table 5.2: Linear 2D model: The discovered governing equations, by employing RK4-SINDy and Std-SINDy, are reported. In this scenario, the measurement data are corrupted using zero-mean Gaussian white noise of different variances.

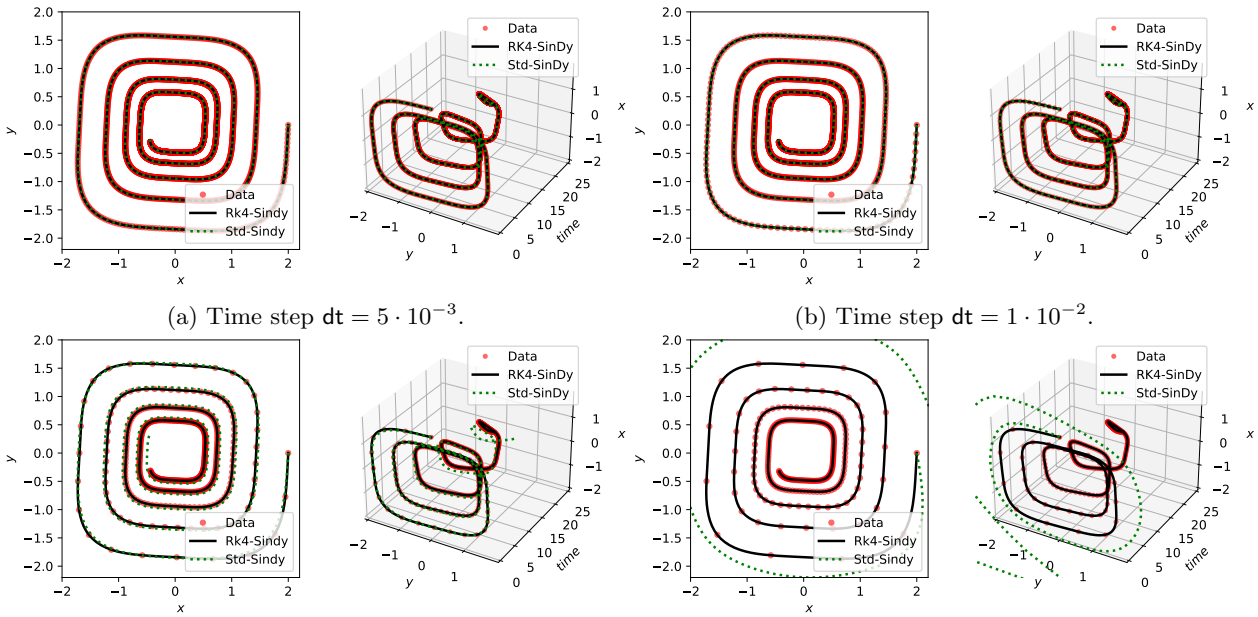
5.1.2 Cubic damped oscillator

Next, we consider a cubic damped oscillator, governed by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= -0.1\mathbf{x}(t)^3 + 2.0\mathbf{y}(t)^3, \\ \dot{\mathbf{x}}(t) &= -2.0\mathbf{x}(t)^3 - 0.1\mathbf{y}(t)^3. \end{aligned} \tag{5.2}$$

Like the linear case, we aim at discovering the governing equation using measurement data. We repeat the study done in the previous example using different regular time-steps. We report the quality of discovered models using RK4-SINDy and Std-SINDy in Figure 5.3 and Table 5.3. We observe that RK4-SINDy successfully discovers the governing equations quite accurately, whereas Std-SINDy struggles to identify the governing equations when measurements data are collected at a larger time-step. It simply fails to obtain a stable

model for a time-step $\mathsf{dt} = 0.1$. It showcases the robustness of RK4-SINDy to discover interpretable models even when data are collected sparsely.



(a) Time step $\mathsf{dt} = 5 \cdot 10^{-3}$.

(b) Time step $\mathsf{dt} = 1 \cdot 10^{-2}$.

(c) Time step $\mathsf{dt} = 5 \cdot 10^{-2}$.

(d) Time step $\mathsf{dt} = 1 \cdot 10^{-1}$.

Figure 5.3: Cubic 2D model: A comparison of the transient responses of discovered models using data at different regular time-steps.

Time step	RK4-SINDy	Std-SINDy
$5 \cdot 10^{-3}$	$\dot{\mathbf{x}}(t) = -0.099\mathbf{x}(t)^3 + 1.996\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.997\mathbf{x}(t)^3 - 0.100\mathbf{y}(t)^3$	$\dot{\mathbf{x}}(t) = -0.099\mathbf{x}(t)^3 + 1.995\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.996\mathbf{x}(t)^3 - 0.099\mathbf{y}(t)^3$
$1 \cdot 10^{-2}$	$\dot{\mathbf{x}}(t) = -0.099\mathbf{x}(t)^3 + 1.995\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.997\mathbf{x}(t)^3 - 0.100\mathbf{y}(t)^3$	$\dot{\mathbf{x}}(t) = -0.100\mathbf{x}(t)^3 + 1.994\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.996\mathbf{x}(t)^3 - 0.099\mathbf{y}(t)^3$
$5 \cdot 10^{-2}$	$\dot{\mathbf{x}}(t) = -0.100\mathbf{x}(t)^3 + 1.995\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.997\mathbf{x}(t)^3 - 0.100\mathbf{y}(t)^3$	$\dot{\mathbf{x}}(t) = -0.092\mathbf{x}(t)^3 + 2.002\mathbf{y}(t)^3$ $+ 0.076\mathbf{x}^4\mathbf{y} - 0.107\mathbf{x}^2\mathbf{y}^3$ $\dot{\mathbf{y}}(t) = -1.981\mathbf{x}(t)^3 - 0.092\mathbf{y}(t)^3$ $+ 0.078\mathbf{x}^3\mathbf{y}^2 - 0.068\mathbf{x}\mathbf{y}^4$
$1 \cdot 10^{-1}$	$\dot{\mathbf{x}}(t) = -0.103\mathbf{x}(t) + 2.000\mathbf{y}(t)$ $\dot{\mathbf{y}}(t) = -2.001\mathbf{x}(t) - 0.098\mathbf{y}(t)$	$\dot{\mathbf{x}}(t) = 0.090\mathbf{x}(t) - 0.097\mathbf{x}(t)^2 - 0.463\mathbf{x}(t)^3$ $+ \cdots + 0.381\mathbf{x}(t)^3\mathbf{y}(t)^2 - 0.258\mathbf{x}(t)\mathbf{y}(t)^4$ $\dot{\mathbf{y}}(t) = 0.100\mathbf{x}(t) + 0.104\mathbf{x}(t)^2 + 0.051\mathbf{x}(t)\mathbf{y}(t)$ $+ \cdots + 0.381\mathbf{x}(t)^3\mathbf{y}(t)^2 - 0.258\mathbf{x}(t)\mathbf{y}(t)^4$

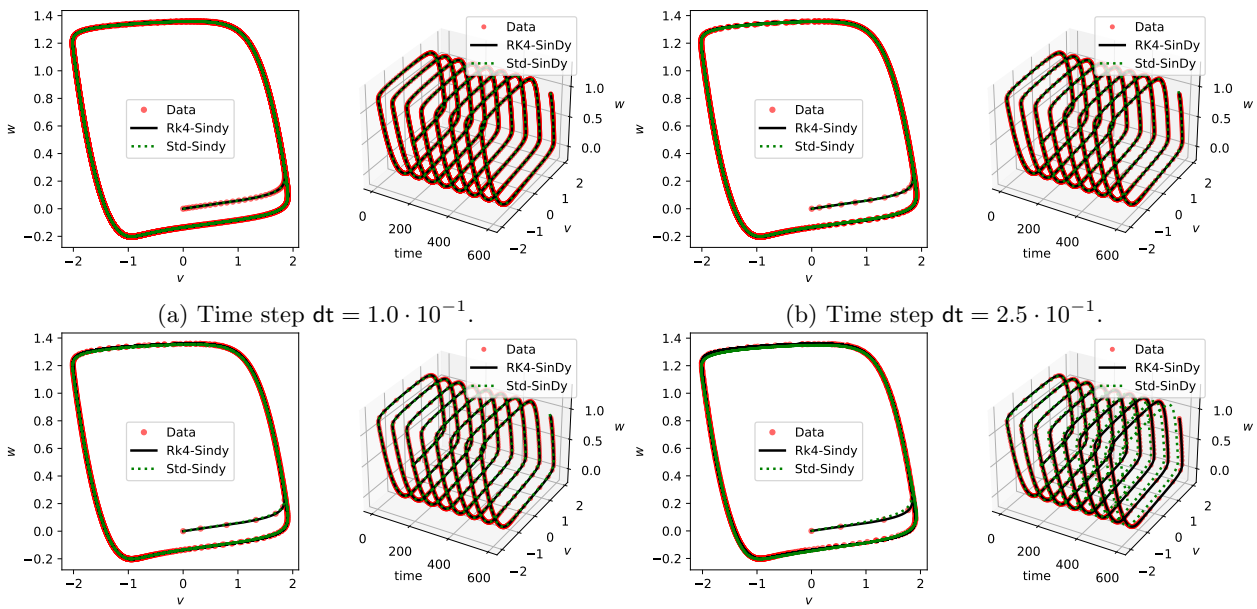
Table 5.3: Cubic 2D model: The table reports the discovered governing equations by employing RK4-SINDy and Std-SINDy.

5.2 Fitz-Hugh Nagumo model

Here, we explore discovery of the nonlinear Fitz-Hugh Nagumo (FHN) model that describes the activation and deactivation of neurons in a simplistic way [15]. The governing equations are:

$$\begin{aligned} \mathbf{v}(t) &= \mathbf{v}(t) - \mathbf{w}(t) - \frac{1}{3}\mathbf{v}(t)^3 + 0.5, \\ \mathbf{w}(t) &= 0.040\mathbf{v}(t) - 0.028\mathbf{w}(t) + 0.032. \end{aligned} \tag{5.3}$$

We collect the time-history data of $\mathbf{v}(t)$ and $\mathbf{w}(t)$ using the zero initial condition. We construct a dictionary containing polynomial terms up to the third degree. We employ RK4-SINDy (Algorithm 1 with $\lambda = 10^{-2}$) and Std-SINDy. We discover governing equations by using the data collected between the time interval $[0, 600]$s. We identify models under different conditions, namely, different time-steps at which data are collected. We report the results in Figure 5.4 and Table 5.4. It can be observed that RK4-SINDy faithfully discovers the underlying governing equations by picking the correct features from the dictionary and estimates the corresponding coefficients up to 1% accurately. On the other hand, Std-SINDy breaks down when data are taken at a large time-step.



(a) Time step $\mathsf{dt} = 1.0 \cdot 10^{-1}$.

(b) Time step $\mathsf{dt} = 2.5 \cdot 10^{-1}$.

(c) Time step $\mathsf{dt} = 5.0 \cdot 10^{-1}$.

(d) Time step $\mathsf{dt} = 7.5 \cdot 10^{-1}$.

Figure 5.4: FHN model: A comparison of the transient responses of the discovered differential equations using data collected at different regular time-steps.

5.3 Chaotic Lorenz system

As the next example, we consider the problem of discovering the nonlinear Lorenz model [27]. The dynamics of the chaotic system involves on an attractor and is governed by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= -10\mathbf{x}(t) + 10\mathbf{y}(t), \\ \dot{\mathbf{y}}(t) &= \mathbf{x}(28 - \mathbf{z}(t)) - \mathbf{y}(t), \\ \dot{\mathbf{z}}(t) &= \mathbf{x}(t)\mathbf{y}(t) - \tfrac{8}{3}\mathbf{z}(t). \end{aligned} \tag{5.4}$$

We collect the data by simulating the model from time $t = 0$ to $t = 20$ with a time-step of $\mathsf{dt} = 10^{-2}$. To discover the governing equations using the measurement data, we employ RK4-SINDy and Std-SINDy with the fixed cutoff parameter $\lambda = 0.5$. However, before employing the methodologies, we perform a normalization

dt	RK4-SINDy	Std-SINDy
$1.0 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.499 + 0.998\mathbf{v} - 0.998\mathbf{w} - 0.333\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.498 + 0.996\mathbf{v} - 0.996\mathbf{w} - 0.332\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$
$2.5 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.499 + 0.998\mathbf{v} - 0.998\mathbf{w} - 0.333\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.494 + 0.985\mathbf{v} - 0.989\mathbf{w} - 0.328\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$
$5.0 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.501 + 1.001\mathbf{v} - 1.001\mathbf{w} - 0.334\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.482 + 0.943\mathbf{v} - 0.959\mathbf{w}$ $- 0.034\mathbf{vw} - 0.311\mathbf{v}^3 + 0.024\mathbf{vw}^2$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$
$7.5 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.502 + 1.001\mathbf{v} - 1.003\mathbf{w} - 0.334\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.027\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.459 + 0.816\mathbf{v} - 0.982\mathbf{w}$ $- 0.013\mathbf{v}^2 + \cdots + 0.131\mathbf{vw}^2 - 0.021\mathbf{w}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$

Table 5.4: FHN model: Discovered models using data at various time-step using RK4-SINDy and Std-SINDy.

step. A reason behind is that the mean value of the variable $\mathbf{z}$ is large, and the standard deviations of all the three variables is much larger than 1. Consequently, a dictionary containing polynomial terms would be highly ill-conditioned. To circumvent this, we perform a normalization of data. Ideally, one performs normalization such that the mean and variance of the transformed data are 0 and 1. But for this particular example, we normalize such that the interactions between the transformed variables are similar to (5.4). Hence, we propose a transformation as

$$\tilde{\mathbf{x}}(t) := \tfrac{\mathbf{x}(t)}{8}, \quad \tilde{\mathbf{y}}(t) := \tfrac{\mathbf{y}(t)}{8}, \quad \tilde{\mathbf{z}}(t) := \tfrac{\mathbf{z}(t)-25}{8}. \tag{5.5}$$

Consequently, we obtain a model:

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}(t) &= -10\tilde{\mathbf{x}}(t) + 10\tilde{\mathbf{y}}(t), \\ \dot{\tilde{\mathbf{y}}}(t) &= \tilde{\mathbf{x}}(28 - 8\tilde{\mathbf{z}}(t)) - \tilde{\mathbf{y}}(t), \\ \dot{\tilde{\mathbf{z}}}(t) &= 8\tilde{\mathbf{x}}(t)\tilde{\mathbf{y}}(t) - \tfrac{8}{3}\tilde{\mathbf{z}}(t) - \tfrac{25}{3}. \end{aligned} \tag{5.6}$$

Notwithstanding, the models (5.4) and (5.6) look alike, and the basis features in which dynamics of both models lie are the same except a constant. However, the beauty of the model (5.6) or the transformed data is that the data becomes well-conditioned, hence the dictionary containing polynomial features. Next, we discover models by employing RK4-SINDy and Std-SINDy using the transformed data. For this, we construct a dictionary with polynomial nonlinearities up to degrees 3. We observe the result in Figure 5.5 and Table 5.5. We note that both methods identify correct features from the dictionary with coefficients that are close to the ground truth, but RK4-SINDy model coefficients are relatively closer to the ground-truth ones. It is also worthwhile to note that the coefficients of the obtained RK4-SINDy model are only 0.01% off to the ground-truth, but the dynamics still seem quite different, see Figure 5.5. A reason behind this is the highly chaotic behavior of the dynamics. As a result, a tiny deviation in the coefficients can significantly impact the transient behavior in an absolute sense; however, the dynamics on an attractor are perfectly captured.

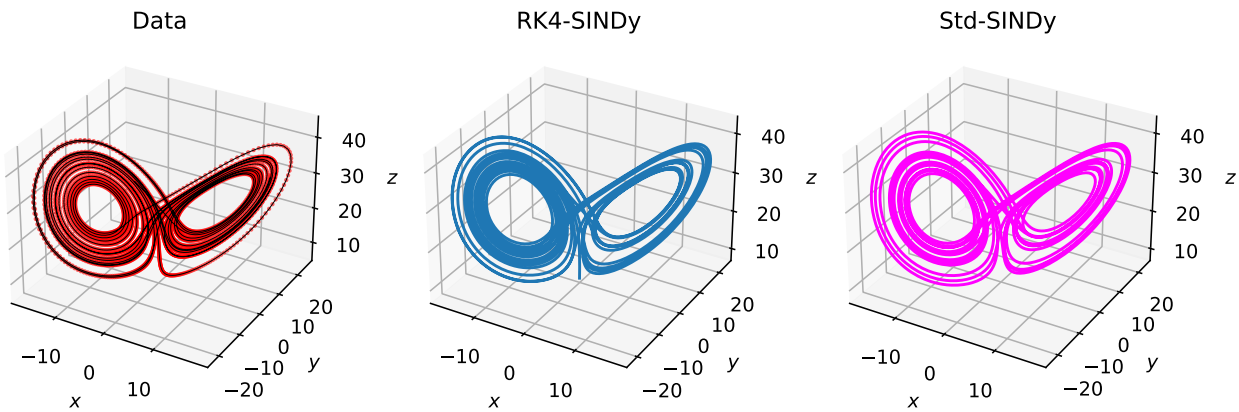


Figure 5.5: Chaotic Lorenz model: The left figures shows the collected data (in red) and a finely spaced trajectory of the ground truth is shown in black. The middle and right figures shows the trajectories obtained using the discovered models using RK4-SINDy and Std-SINDy, respectively.

RK4-SINDy	Std-SINDy
$\dot{\tilde{\mathbf{x}}}(t) = -10.004\tilde{\mathbf{x}}(t) + 10.004\tilde{\mathbf{y}}(t),$	$\dot{\tilde{\mathbf{x}}}(t) = -9.983\tilde{\mathbf{x}}(t) + 9.983\tilde{\mathbf{y}}(t),$
$\dot{\tilde{\mathbf{y}}}(t) = 2.966\tilde{\mathbf{x}} - 0.956\tilde{\mathbf{y}}(t) - 7.953\tilde{\mathbf{x}}(t)\tilde{\mathbf{z}}(t),$	$\dot{\tilde{\mathbf{y}}}(t) = 2.912\tilde{\mathbf{x}} - 0.922\tilde{\mathbf{y}}(t) - 7.911\tilde{\mathbf{x}}(t)\tilde{\mathbf{z}}(t),$
$\dot{\tilde{\mathbf{z}}}(t) = 7.944\tilde{\mathbf{x}}(t)\tilde{\mathbf{y}}(t) - 2.669\tilde{\mathbf{z}}(t) - 8.336$	$\dot{\tilde{\mathbf{z}}}(t) = 7.972\tilde{\mathbf{x}}(t)\tilde{\mathbf{y}}(t) - 2.662\tilde{\mathbf{z}}(t) - 8.313$

Table 5.5: Chaotic Lorenz model: Discovered governing equations using RK4-SINDy and Std-SINDy.

Next, we study the performance of the approaches under noisy measurements. For this, we add mean zero Gaussian noise of variance one. To employ RK4-SINDy, we first apply a Savitzky-Golay filter [36] to denoise the time-history data, see Figure 5.6. For Std-SINDy as well, we use the same filter to denoise the signal and approximate the derivative information. We plot the trajectories of the discovered models and ground-truth in Figure 5.7 and observe that dynamics on an attractor is still intact; however, we note that the discovered equations are very different from the ground truth, see Table 5.6. The learning can be improved by employing Algorithm 2, where we iteratively remove the smallest coefficient and determine the sparsest solutions by looking at the Pareto-front. However, it comes at a slightly higher computational cost. We discuss this approach more in detail in our following examples.

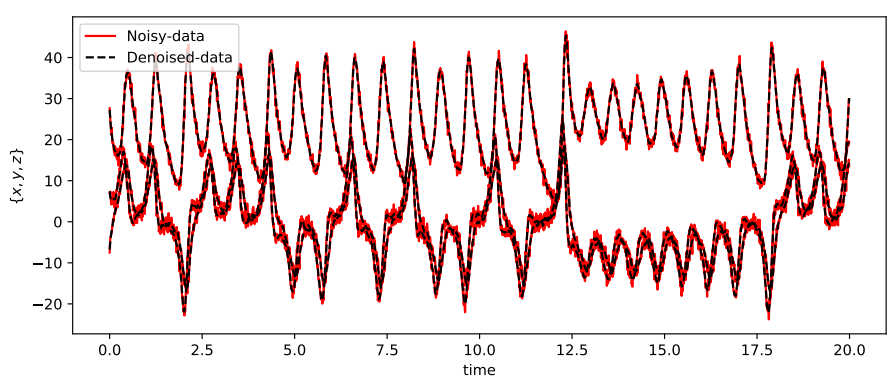


Figure 5.6: Chaotic Lorenz model: The figure shows the noisy measurements of $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ that are corrupted by adding zero-mean Gaussian noise of variance one. It also shows the denoised signal done using a Savitzky-Golay filter [36].

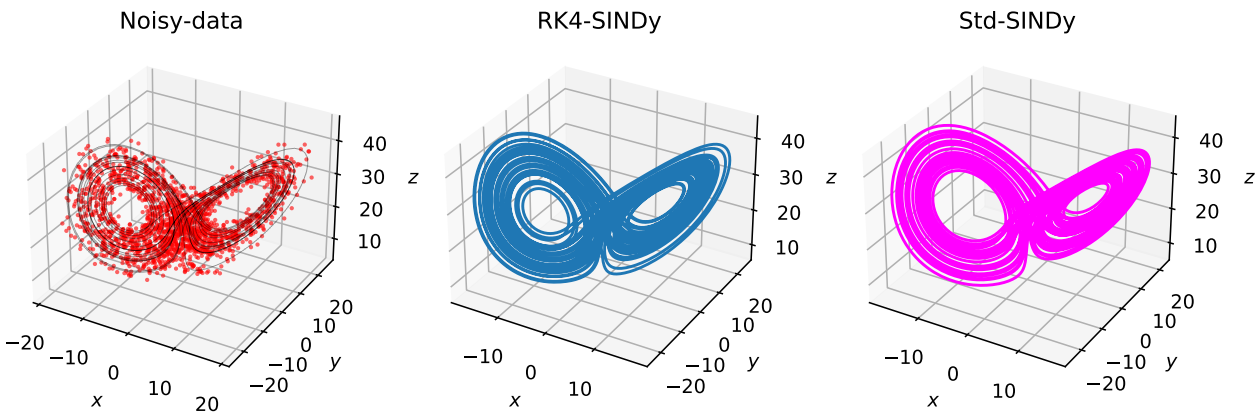


Figure 5.7: Chaotic Lorenz model: The left figure shows the collected noisy data (in red), and a continuous trajectory of the ground truth is shown in light black. We have added a Gaussian white noise of mean zero and variance one. The middle and right figures present the transient trajectories obtained using the discovered models using RK4-SINDy and Std-SINDy, respectively, and show that the dynamics of the discovered models are intact on an attractor.

RK4-SINDy	Std-SINDy
$\dot{\tilde{\mathbf{x}}}(t) = -9.016\tilde{\mathbf{x}} + 8.221\tilde{\mathbf{y}}$ $- 1.675\tilde{\mathbf{x}}\tilde{\mathbf{z}} + 0.895\tilde{\mathbf{x}}^3$ $+ 0.603\tilde{\mathbf{x}}^2\tilde{\mathbf{y}} + -0.579\tilde{\mathbf{x}}\tilde{\mathbf{y}}^2$	$\dot{\tilde{\mathbf{x}}}(t) = -8.842\tilde{\mathbf{x}} + 8.373\tilde{\mathbf{y}}$ $- 3.107\tilde{\mathbf{x}}\tilde{\mathbf{z}} + 1.065\tilde{\mathbf{y}}\tilde{\mathbf{z}}$ $+ 2.165\tilde{\mathbf{x}}^3 + -1.215\tilde{\mathbf{x}}^2\tilde{\mathbf{y}}$
$\dot{\tilde{\mathbf{y}}}(t) = 1.025\tilde{\mathbf{y}} - 6.133\tilde{\mathbf{x}}\tilde{\mathbf{z}} - 1.033\tilde{\mathbf{y}}\tilde{\mathbf{z}}$	$\dot{\tilde{\mathbf{y}}}(t) = 1.811\tilde{\mathbf{x}} + -7.580\tilde{\mathbf{x}}\tilde{\mathbf{z}}$
$\dot{\tilde{\mathbf{z}}}(t) = -8.3451 - 2.708\tilde{\mathbf{z}} + 7.971\tilde{\mathbf{x}}\tilde{\mathbf{y}}$	$\dot{\tilde{\mathbf{z}}}(t) = -8.3441 + -2.710\tilde{\mathbf{z}} + 7.950\tilde{\mathbf{x}}\tilde{\mathbf{y}}$

Table 5.6: Lorenz model: Discovered governing equations using RK4-SINDy and Std-SINDy from noisy measurements.

5.4 Michaelis-Menten kinetics

To illustrate RK4-SINDy to discover governing equations that are given by a ratio of two nonlinear functions, we consider arguably the most well-known model for an enzyme kinetics, namely Michaelis-Menten model [21, 30]. The model explains the dynamics of binding and unbinding of enzyme with an substrate $\mathbf{s}$. In a simplistic way, the dynamics are governed by [4]:

$$\dot{\mathbf{s}}(t) = 0.6 - \frac{1.5\mathbf{s}(t)}{0.3 + \mathbf{s}(t)}. \tag{5.7}$$

As a first step, we generate data using four initial conditions $\{0.5, 1.0, 1.5, 2.0\}$. We collect data at a time-step $\mathsf{dt} = 5 \cdot 10^{-2}$, see Figure 5.8(a). Typically, governing equations, explaining biological process have rational functions. Therefore, we aim at discovering the enzyme kinetics model by assuming a rational form as shown in (4.1), i.e., the gradient field of $\mathbf{s}(t)$ takes the form $\frac{\mathbf{g}(\mathbf{s}(t))}{1+\mathbf{h}(\mathbf{s}(t))}$.

Next, in order to identify $\mathbf{g}(\mathbf{s})$ and $\mathbf{h}(\mathbf{s})$, we construct the polynomial dictionaries, containing terms up to degrees 4. After that, we employ RK4-SINDy to identify the precise features from the dictionaries to characterize $\mathbf{g}$ and $\mathbf{h}$. Moreover, we apply the iterative thresholding approach discussed in Algorithm 2, in contrast to previously considered examples where a fixed thresholding is applied. Note that the success of RK4-SINDy approach not only depends on a dictionary containing candidate features but the quality of data. We have marked that the dictionary data matrix's conditioning improves when data are normalized to mean-zero and variance-one. It is crucial for polynomial basis in the dictionary. For this example, we

normalize the data before employing RK4-SINDy. It means that the transformation is done as follows:

$$\tilde{\mathbf{s}}(t) = \frac{\tilde{\mathbf{s}} - \mu_{\mathbf{s}}}{\sigma_{\mathbf{s}}}, \tag{5.8}$$

where $\mu_{\mathbf{s}}$ and $\sigma_{\mathbf{s}}$ are the mean and standard deviation of the collected data. Next, using the normalized data, we learn the governing equation, describing the dynamics of $\tilde{\mathbf{s}}(t)$. Since we consider dictionaries for $\mathbf{g}$ and $\mathbf{h}$, containing polynomials of degree 4, there are total 9 coefficients. To identify the correct model while employing Algorithm 2, we keep track of the loss (data-fidelity) and the number of non-zero coefficients, which is shown in Figure 5.8(c). This allows us to built a Pareto front for the optimization problem and choose the most parsimonious model that describes the dynamics present in collected data. One of the most attractive features of learning parsimonious models is to avoid over-fitting and generalizing better in regions in which data are not collected. It is precisely what we observed as well. As shown in Figure 5.8(e), the learned model predicts dynamics very accurately in the region far away from the training one.

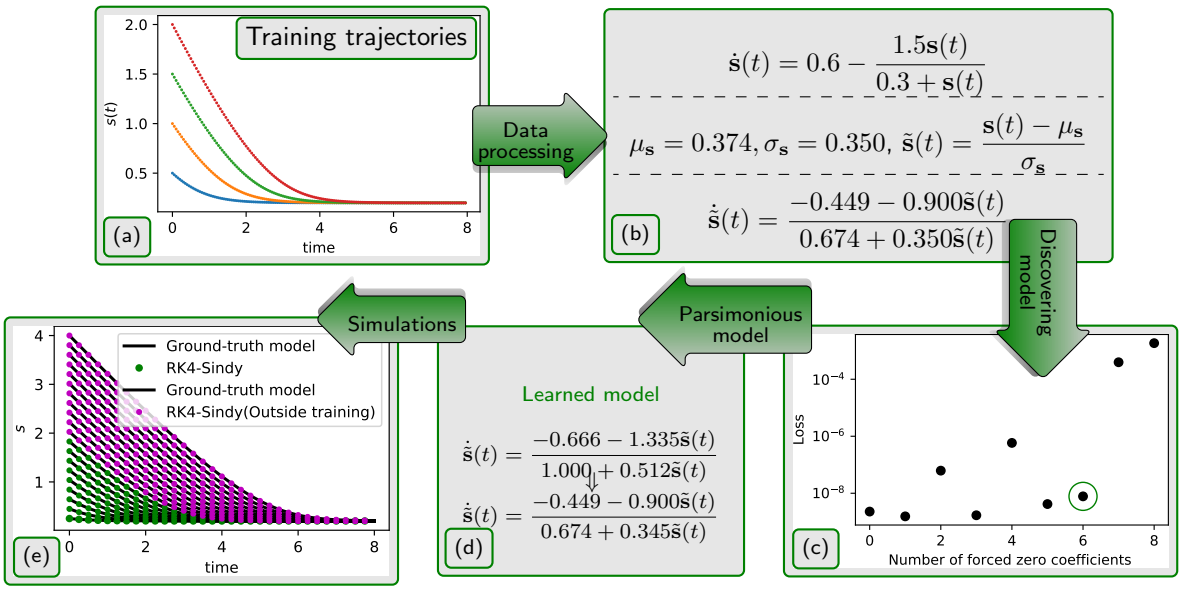


Figure 5.8: Michaelis-Menten kinetics: In the first step, we have collected data for 4 initial conditions at a time-stepping $\mathsf{dt} = 5 \cdot 10^{-2}$. In the second step, we performed data-processing to normalize the data using the mean and standard deviation. In the third step, we employed RK4-SINDy (Algorithm 2) to discover the most parsimonious model. For this, we observe the Pareto front and pick the model that has the best fit to the data yet having the maximum number of zero coefficients. We then compare the discovered model with the ground truth and find that the proposed methodology could find precise candidates from the polynomial dictionary. The corresponding coefficients have less than 1% errors.

Next, we study the performance of the method under noisy measurements. For this, we corrupt the collected data using zero-mean Gaussian noise of variance $\sigma = 2 \cdot 10^{-2}$. Then, we process the data by first employing a noise-reduction filter, namely Savitzky-Golay, followed by normalizing the data. In the third step, we focus on learning the most parsimonious model by picking appropriate candidates from the polynomial dictionary. Remarkably, the method allows us to find a model with correct features from the dictionary and coefficient accuracy up to 5%. Furthermore, the model faithfully generalizes to regimes outside the training, even using noisy measurements.

5.5 Hopf normal form

In our last example, we study discovering parameterized differential equations from noisy measurements. Many real-world dynamical processes have system parameters, and depending on them, the system may

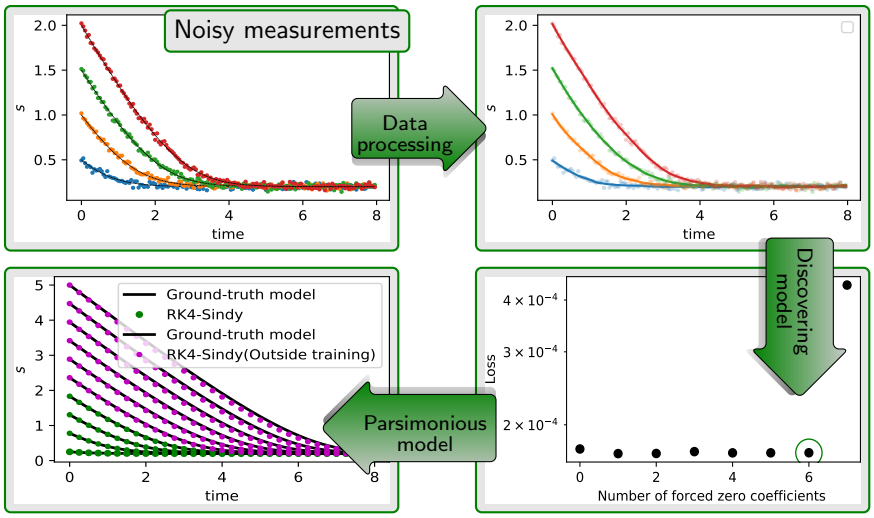


Figure 5.9: Michaelis-Menten kinetics: The figure demonstrates the necessary steps to uncover the most parsimonious model using noisy measurement data. It also testifies to the impressive capability of discovering the most parsimonious model – that is, they even generalize beyond the training regime.

exhibit very distinctive dynamical behaviors. To illustrate the efficiency of RK4-SINDy to discover parametric equations, we consider the Hopf system

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mu\mathbf{x}(t) - \omega\mathbf{y}(t) - \mathbf{A}\mathbf{x}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right),\\ \dot{\mathbf{y}}(t) &= \omega\mathbf{x}(t) + \mu\mathbf{y}(t) - \mathbf{A}\mathbf{y}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right)\end{aligned} \tag{5.9}$$

that exhibits bifurcation with respect to the parameter μ. For this example, we collect measurements for eight different parameter values μ at a time-step 0.2 by fixing $\omega = 1$ and $\mathbf{A} = 1$. Then, we corrupt the measurement data by adding a Gaussian sensor noise that is shown in Figure 5.10 (left top). Next, we aim at constructing a symbolic polynomial dictionary Φ by including the parameter μ as the dependent variables. While building a polynomial dictionary, it is important to choose the degree of the polynomial as well. Moreover, it is known that the polynomial basis becomes numerically unstable as the degree increases. Hence, solving optimization problem that discovers governing equations becomes challenging. With mean of this example, we discuss an assessment test to choose the appropriate degree of the polynomial in the dictionary. Essentially, we inspect data fidelity with respect to the degree of the polynomial in the dictionary. When the dictionary contains all essential polynomial features, then a sharp drop in the error is expected. We observe in Figure 5.10 (right-top) a sharp drop in the error at the degree 3, and the error remains almost the same even when higher polynomial features are added. It indicates that polynomial degree 3 is sufficient to describe the dynamics. Thereafter, using the dictionary containing degree 3 polynomial features, we seek to identify the minimum number of features from the dictionary that explains the underlying dynamics. We achieve this by employing RK4-SINDy, and compare the performance with Std-SINDy. We note down the discovered governing equations in Table 5.7, where we notice an impressive performance of RK4-SINDy to discover the exact form of the underlying parametric equations, and the coefficients are up to 1% accurate. On the other hand, Std-SINDy is not able to identify the correct form of the model. Furthermore, we compare the discovered model simulations using RK4-SINDy with ground truth beyond the training regime of the parameter μ in Figure 5.10 (bottom). It exposes the strength of the parsimonious and interpretable discovered models.

6 Discussion

This work has manifested a compelling approach (RK4-SINDy) to discover nonlinear differential equations without imposing any prior structure on models. For this, we have blended sparsity-promoting identification with a numerical integration scheme, namely, Runge-Kutta 4th-order scheme. The beauty of the proposed methodology is that we do not require derivative information at any stage, notwithstanding we still discover differential equations. Hence, the proposed algorithm differs from previously suggested sparsity-promoting identification methods in the literature in this aspect. Consequently, we expect RK4-SINDy to perform better

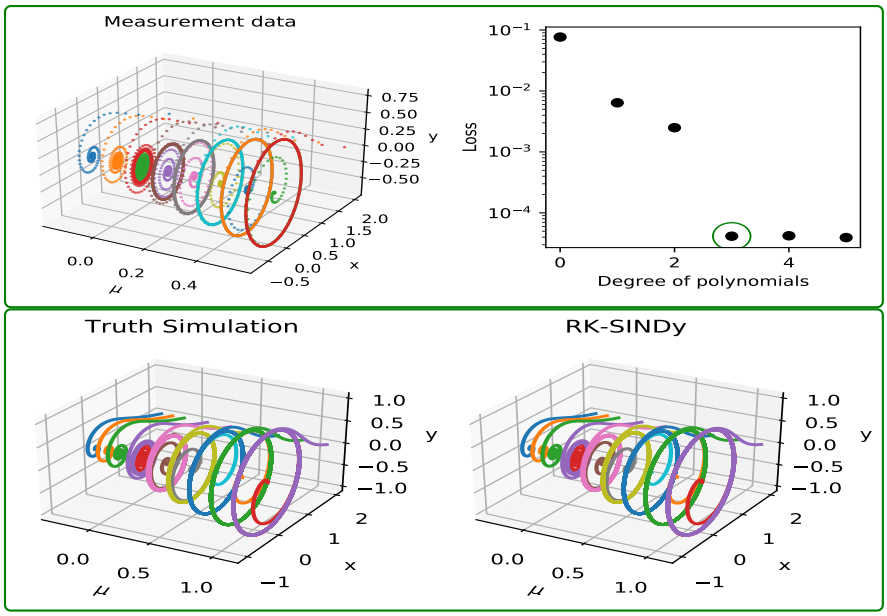


Figure 5.10: Hopf normal form: The top-left figure shows the noisy measurements that are obtained using various parameter μ. To identify correct degree polynomial basis in the dictionary, we do a assessment test, indicating degree-3 polynomial are sufficient to describe dynamics (top-right). The bottom figures shows a comparison of simulations of the ground truth model and identified models for the parameter μ, illustrating the capability of generalizing beyond the training parameter regime.

Method	Discovered model
RK4-SINDy	$\dot{\mathbf{x}}(t) = 1.001\mu\mathbf{x}(t) - 1.001\mathbf{y}(t) - 0.996\mathbf{x}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right)$ $\dot{\mathbf{y}}(t) = 1.001\mathbf{x}(t) + 1.010\mu\mathbf{y}(t) - 1.006\mathbf{x}(t)^2\mathbf{y}(t) - 1.004\mathbf{y}(t)^3$
Std-SINDy	$\dot{\mathbf{x}}(t) = -0.961\mathbf{y}(t) + 0.719\mu\mathbf{x}(t) + 0.822\mu\mathbf{y}(t) - 0.735\mathbf{x}(t)^3 - 1.044\mathbf{x}(t)^2\mathbf{y} - 0.686\mathbf{x}(t)\mathbf{y}(t)^2 - 0.846\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = 0.986\mathbf{x}(t) + 0.899\mu\mathbf{y}(t) - 0.882\mathbf{x}(t)^2\mathbf{y}(t) - 0.904\mathbf{y}(t)^3.$

Table 5.7: Hopf normal form: Here, we report discovered governing equations using noisy measurement data, representing dynamics of Hopf bifurcation and notice that RK4-SINDy recovers the Hopf normal form very accurately; on the other hand, Std-SINDy breaks down.

under sparsely sampled and corrupted data. We have demonstrated the efficiency of the approach on a variety of examples, namely linear and nonlinear damped oscillators, a model describing neural dynamics, chaotic behavior, and parametric differential equations. We have accurately discovered the Fitz-Hugh Nagumo model that describes the activation and de-activation of neurons. We have also illustrated the identification of the Lorenz model and have shown that dynamics of identified models are intact on an attractor as it is more important for chaotic dynamics. The example of Michaelis-Menten Kinetics highlights that the proposed algorithm can discover models that are given by a ratio of two functions. The example also shows the power of determining parsimonious models – that is, their generalization beyond the region in which data are selected. Furthermore, we have demonstrated the remarkable robustness of the proposed RK4-SINDy algorithm to sparsely-sampled data and to corrupted measurement data. In the case of large noise, a noise-reduction filter such as Savitzky-Golay helps to improve the quality of discovered governing equations. We have also reported a comparison with the sparse identification approach [6] and have observed the out-performance of RK4-SINDy over the latter approach.

This work opens many exciting doors for further research from both theory and practical perspectives. Since the approach aims at selecting the correct features from a dictionary containing a high-dimensional nonlinear feature basis, the construction of these feature bases in a dictionary plays a significant role in determining the success of the approach. There is no straightforward answer to this obstacle; however, there is some expectation that meaningful features may be constructed with the help of experts and empirical knowledge, or at least they may be realized in raw forms by them. Furthermore, we have solved the optimization problem (2.9) using a gradient-based method. We have observed that if feature functions in the dictionary are similar for given data, then the convergence is slow, and sometimes it even fails and is stuck in a local

minimum. In other words, the incoherency between the feature functions is low. Hence, there is a need for the normalization step. In Subsections 5.3 and 5.4, we have employed a normalization step to improve incoherency. However, it is worth investigating better-suited strategies to normalize either data or feature spaces as a pre-processing step so that sparsity in the feature space remains intact. In addition to these, a thorough study on the performance of various noise-reduction methods would provide deep insights into their appropriateness to RK4-SINDy, despite we noticed a good performance of the Sabitzky-Golay filter to reduced noise in our results.

Methods discovering interpretable models that generalize well beyond the training regime are limited, and the proposed method RK4-SINDy is among these. Additionally, approaches discovering governing equations that also obey physical laws are even rarer. A very recent paper [44] has stressed that discovering/learning models can be made even more efficient by incorporating the laws of nature in the course of discovering equations. A solid example comes from the discovering biological networks that often follow the mass-conversation law. Therefore, integrating physical laws in discovering models and sparse identification will hopefully shape the future of discovering explainable and generalizable differential equations.

References

[1] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. SIAM, 1998.

[2] Amir Beck and Yonina C Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM J. Optim.*, 23(3):1480–1509, 2013.

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

[4] George Edward Briggs. A further note on the kinetics of enzyme action. *Biochem. J.*, 19(6):1037, 1925.

[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. U.S.A.*, 113(15):3932–3937, 2016.

[6] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[7] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52(2):489–509, 2006.

[8] Emmanuel J Candès, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1207–1223, 2006.

[9] Kathleen Champion, Peng Zheng, Aleksandr Y Aravkin, Steven L Brunton, and J Nathan Kutz. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access*, 8:169259–169271, 2020.

[10] Rick Chartrand. Numerical differentiation of noisy, nonsmooth data. *Intern. Scholarly Res. Notices*, 2011, 2011.

[11] James P Crutchfield and Bruce S McNamara. Equations of motion from a data series. *Complex Sys.*, 1(417-452):121, 1987.

[12] Bryan C Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using S-systems and alternating regression. *PLoS One*, 10(3):e0119821, 2015.

[13] Brian de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *J. Open Source Software*, 5(49):2104, 2020.

[14] David L Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306, 2006.

[15] Richard FitzHugh. Mathematical models of threshold phenomena in the nerve membrane. *The Bulletin Math. Biophys.*, 17(4):257–278, 1955.

[16] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer, 2001.

[17] Matan Gavish and David L Donoho. Optimal shrinkage of singular values. *IEEE Trans. Inform. Theory*, 63(4):2137–2152, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 770–778, 2016.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 4700–4708, 2017.

[20] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[21] Kenneth A Johnson and Roger S Goody. The original Michaelis constant: translation of the 1913 Michaelis–Menten paper. *Biochemistry*, 50(39):8264–8269, 2011.

[22] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[23] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*, volume 7. Cambridge University Press, 2004.

[24] Ioannis G Kevrekidis, C William Gear, James M Hyman, Panagiotis G Kevrekidis, Olof Runborg, Constantinos Theodoropoulos, et al. Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. *Comm. Math. Sci.*, 1(4):715–762, 2003.

[25] S Narendra Kumpati and Parthasarathy Kannan. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, 1(1):4–27, 1990.

[26] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, NJ, 1999.

[27] Edward N Lorenz. Deterministic nonperiodic flow. *J. Atmospheric Sci.*, 20(2):130–141, 1963.

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

[29] Vivien Marx. The big challenges of big data. *Nature*, 498(7453):255–260, 2013.

[30] Leonor Michaelis and Maud L Menten. Die kinetik der invertinwirkung. *Biochem. z*, 49(333-369):352, 1913.

[31] Vidvuds Ozoliņš, Rongjie Lai, Russel Caflisch, and Stanley Osher. Compressed modes for variational problems in mathematics and physics. *Proc. Nat. Acad. Sci. U.S.A.*, 110(46):18368–18373, 2013.

[32] Joshua L Proctor, Steven L Brunton, Bingni W Brunton, and JN Kutz. Exploiting sparsity and equation-free architectures in complex systems. *Europ. Phy. J. Spec. Top.*, 223(13):2665–2684, 2014.

[33] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[34] 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.

[35] Samuel H Rudy, J Nathan Kutz, and Steven L Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *J. Comput. Phys.*, 396:483–506, 2019.

[36] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chem.*, 36(8):1627–1639, 1964.

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

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

[39] Johan AK Suykens, Joos PL Vandewalle, and Bart L de Moor. *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*. Springer, 1996.

[40] Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Stat. Soc.: Series B (Methodological)*, 58(1):267–288, 1996.

[41] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inform. Theory*, 53(12):4655–4666, 2007.

[42] Peter Van Overschee and Bart de Moor. *Subspace Identification of Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers, 1996.

[43] Wen-Xu Wang, Rui Yang, Ying-Cheng Lai, Vassilios Kovanis, and Celso Grebogi. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Phy. Rev. Letters*, 106(15):154101, 2011.

[44] Karen E Willcox, Omar Ghattas, and Patrick Heimbach. The imperative of physics-based modeling and inverse theory in computational science. *Nature Comput. Sci.*, 1(3):166–168, 2021.

[45] Zhuoran Yang, Zhaoran Wang, Han Liu, Yonina Eldar, and Tong Zhang. Sparse nonlinear regression: Parameter estimation under nonconvexity. In *Intern. Conf. on Machine Learning*, pages 2472–2481. PMLR, 2016.

[46] Hao Ye, Richard J Beamish, Sarah M Glaser, Sue CH Grant, Chih-hao Hsieh, Laura J Richards, Jon T Schnute, and George Sugihara. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proc. Nat. Acad. Sci. U.S.A.*, 112(13):E1569–E1576, 2015.