Derivative-based SINDy (DSINDy): Addressing the challenge of discovering governing equations from noisy data

Jacqueline Wentz\textsuperscript{a}, Alireza Doostan\textsuperscript{a}

\textsuperscript{a}Smead Aerospace Engineering Sciences, University of Colorado, 3775 Discovery Dr, Boulder, CO, 80303, United States

Abstract

Recent advances in the field of data-driven dynamics allow for the discovery of ODE systems using state measurements. One approach, known as Sparse Identification of Nonlinear Dynamics (SINDy), assumes the dynamics are sparse within a predetermined basis in the states and finds the expansion coefficients through linear regression with sparsity constraints. This approach requires an accurate estimation of the state time derivatives, which is not necessarily possible in the high-noise regime without additional constraints. We present an approach called Derivative-based SINDy (DSINDy) that combines two novel methods to improve ODE recovery at high noise levels. First, we denoise the state variables by applying a projection operator that leverages the assumed basis for the system dynamics. Second, we use a second order cone program (SOCP) to find the derivative and governing equations simultaneously. We derive theoretical results for the projection-based denoising step, which allow us to estimate the values of hyperparameters used in the SOCP formulation. This underlying theory helps limit the number of required user-specified parameters. We present results demonstrating that our approach leads to improved system recovery for the Van der Pol oscillator, the Duffing oscillator, the Rössler attractor, and the Lorenz 96 model.

Keywords: sparse regression, nonlinear dynamics, SINDy, denoising, data-driven modeling

2020 MSC: 62J07, 65D10, 34A55, 37M10, 90C25, 15A04

1. Introduction

Physical systems are often represented using differential equations to describe how the systems evolve over time. From simple systems, such as the pendulum, to complex systems, such as the interaction of hundreds of proteins within a cell, differential equations provide a way to predict future states and study stability properties within a system. However, often the governing equations of systems are either partially or fully unknown. In these cases, methods for data-driven equation discovery can be applied to learn the governing equations directly from state measurements \cite{1}. These data-driven methods have shown success, for example, in learning chemical laws \cite{2}, biological networks \cite{3}, ecological systems \cite{4}, and fluid dynamics \cite{5}. However, some fundamental challenges in equation discovery remain, including accurate state time derivative estimation in the presence of noise, robustness with respect to noise, and the need for user-specified hyperparameters.

We focus specifically on the data-driven discovery of ordinary differential equation (ODE) systems. The best method for discovering a given ODE system depends on characteristics of the problem, such as the number of states, prior information, and the goal of the discovery, e.g., interpretability, forecasting, or stability analysis. For instance, although symbolic regression is highly expressive and leads to interpretable governing equations \cite{6} \cite{7} \cite{2}, it is a computationally infeasible approach for high dimensional systems. Similarly, neural-networks allow for high expressivity and require limited information on the form of the nonlinearities.
but the resulting solution is often uninterpretable and requires many user-specific hyperparameters, e.g., in defining the neural network architecture. Recently developed sparsity-promoting methods, e.g., Sparse Identification of Nonlinear Dynamics (SINDy), assume the governing equations are sparse in some known basis of the state variables [9, 10, 11]. This leads to parsimonious and interpretable solutions but requires potentially unavailable knowledge on the form of the basis. In contrast to neural networks and symbolic regression, sparsity-promoting methods allow for the rapid discovery of an ODE system by solving a sparsity regularized least-squares optimization problem.

Here, we will focus on sparsity-promoting techniques as we are interested in interpretable equations. One of the main challenges with these methods is the accurate approximation of the state time derivatives in the presence of measurement noise. A variety of methods have been proposed to circumvent this issue such as a priori smoothing [12, 13], using an integral formulation [14], and modeling the state variables with a neural network to find derivatives through automatic differentiation [15]. Several groups have explored simultaneous denoising and recovery of the system dynamics [16, 17, 18, 19]. Although promising, the resulting optimization problem is non-convex and often involves the tuning of hyperparameters. One of the most successful approaches uses the weak form of the dynamics [20, 21, 22, 23, 24] and, known as Weak SINDy or WSINDy, applies integration by parts and places the derivative on test functions to avoid the derivative calculation from noisy data. One potential downside to WSINDy is there are several hyperparameters that may need tuning in order to obtain reasonable results [23].

Our approach for discovery governing equations, which we call Derivative-based SINDy (DSINDy), seeks to improve upon the aforementioned methods in several ways. Instead of a priori smoothing without any physical information [12], we apply a denoising step that leverages the assumed basis for the system dynamics. Additionally, in contrast to simultaneous denoising and recovery methods [16, 17, 18, 19], our approach for system recovery only involves solving a convex program. Finally, by using theoretical results from the denoising step, we recover the system dynamics without requiring the tuning of hyperparameters. Although a comprehensive comparison of DSINDy with other equation discovery methods in the high noise regime is beyond the scope of this work, we do provide a comparison of DSINDy with both WSINDy and an $\ell_1$-minimization version of SINDy [25], which we refer to as $\ell_1$-SINDy. Note that in $\ell_1$-SINDy, Tikhonov regularization is used to find smooth time derivatives from the noisy measurements.

The DSINDy algorithm can be broken down into two steps. First we use a novel algorithm, called Projection-based State Denoising (PSDN), which leverages, through a projection operation, the assumed basis for the system dynamics. We next formulate an iteratively reweighted, second order cone program (IRW-SOCP) that allows us to directly find the state time derivatives, as opposed to the states themselves, while enforcing sparsity of the coefficients, hence the name “Derivative”-based SINDy. A theoretical analysis of PSDN allows us to estimate the values of hyperparameters needed for IRW-SOCP. When used in concert, PSDN and IRW-SOCP lead to the best performance in terms of coefficient estimation and system recovery. Thus, although these two methods can be applied independently, we present them together as a comprehensive approach for learning the governing equations of an ODE system.

Our paper is outlined as follows. In Section 2 we state the equation discovery problem and give relevant notation. In Section 3 we describe DSINDy in detail, and, in Section 4 we present theoretical convergence results. Finally, in Sections 5 and 6 we introduce four example ODE systems and compare the recovery performance of DSINDy with WSINDy and $\ell_1$-SINDy.

2. Problem statement and notation

We consider a system with $m$ state variables that change over time according to

$$\dot{u}_k^*(t) = F_k(u_1^*(t), u_2^*(t), ..., u_m^*(t)) \quad \text{for} \quad k = 1, 2, ..., m,$$

where we use dot notation to refer to the time derivative. Here, $F_k$ for $k = 1, 2, ..., m$ are unknown functions of the state variables. Note we use an asterisk ($*$) to refer to true values and derivatives of the state variables, in contrast to noisy estimates.
One approach for discovering the ODE system is to assume that $F_k$ can be written as a linear combination of known basis functions \([11]\). We make this assumption here, but note that for many systems this information may not be available. We define the known set of basis functions as \( \{ \theta_j \}_{j=1}^{p} \), where each function acts on the state space and returns a scalar, i.e., $\theta_j : \mathbb{R}^m \to \mathbb{R}$ for $j = 1, 2, ..., p$. By assumption, for each state variable $u_k$ there is a vector $c_k = [c_{k,1}, ..., c_{k,p}]^T \in \mathbb{R}^p$, such that

$$
\dot{u}_k(t) = \sum_{j=1}^{p} c_{k,j} \theta_j(u_1^*(t), u_2^*(t), ..., u_m^*(t)). \quad (2)
$$

As in SINDy, the goal of this work is to recover $c_k$, for $k = 1, 2, ..., m$, using a set of noisy state measurements.

We model noise as an additive term on the true state variable trajectory. That is, for a system with $N$ measurements obtained at times $0 = t_1, t_2, ..., t_N = t_{\text{end}}$, the $k$-th state measurement vector $u_k = [u_{k,1}, u_{k,2}, ..., u_{k,N}]^T$ is

$$
u_k = u_k^* + \epsilon_k \in \mathbb{R}^N, \quad \text{for } k = 1, 2, ..., m, \quad (3)$$

where $u_k^* = [u_{k,1}^*(t_1), u_{k,2}^*(t_2), ..., u_{k,N}^*(t_N)]^T$ and $\epsilon_k = [\epsilon_{k,1}, \epsilon_{k,2}, ..., \epsilon_{k,N}]^T$. We assume the noise is an i.i.d. Gaussian random variable with zero mean and variance $\sigma^2$, i.e., $\epsilon_{k,i} \sim \mathcal{N}(0, \sigma^2)$ for $k = 1, 2, ..., m$ and $i = 1, 2, ..., N$. In this work, we only consider systems with uncorrelated measurement noise.

We define the libraries $\Theta \in \mathbb{R}^{N \times p}$ and $\Theta^* \in \mathbb{R}^{N \times p}$ to contain evaluations of the basis functions at the noisy true state variable values, respectively. Each row corresponds to a measurement time and each column corresponds to a basis function such that, for $i = 1, 2, ..., N$ and $j = 1, 2, ..., p$,

$$
\Theta_{ij} = \theta_j(u_{1,i}, u_{2,i}, ..., u_{m,i}) \quad \text{and} \quad \Theta^*_{ij} = \theta_j(u_{1,i}^*, u_{2,i}^*, ..., u_{m,i}^*), \quad (4)
$$

where $u_{k,i}^* = u_k^*(t_i)$. Let $\dot{\mathbf{u}}_k^* \in \mathbb{R}^N$ be the vector of true time derivatives of state variable $u_k$ at times $t_1, ..., t_N$. It follows that

$$\Theta^* c_k = \dot{\mathbf{u}}_k^*. \quad (5)$$

In practice, the true values and time derivatives of the state variables are unknown. Therefore, we use an approximated version of Equation (5), i.e.,

$$\Theta c_k \approx \dot{\mathbf{u}}_k, \quad (6)$$

to find the coefficients. Here, $\dot{\mathbf{u}}_k$ is an approximation of the true time derivative vector, $\dot{\mathbf{u}}_k^*$. Noise in the state measurements impacts both sides of Equation (6), which may cause inaccurate coefficient recovery. As discussed in Section \([12]\), many groups have started with Equation (6) to recover the governing equations of ODE systems, e.g., \([12, 13, 14, 17, 22, 23, 25]\). Note that similar to these works we only consider the case where the system is overdetermined, i.e., $N \geq p$.

In Section \([3]\) we present our novel DSINDy approach for finding the coefficient vector $c_k$. In Section \([5]\) we compare DSINDy to two other approaches, i.e., WSINDy and $l_1$-SINDy, which are described in detail in Appendix \([A]\).

### 2.1. Multivariate monomial basis

We define the basis $\{ \theta_j \}_{j=1}^{p}$ to be the set of multivariate monomials up to total degree $d$. The $j$-th basis function is given as,

$$
\theta_j(u_1, u_2, ..., u_m) = \prod_{k=1}^{m} (u_k)^{\alpha_k^{(j)}}. \quad (7)
$$

We use $\alpha^{(j)} = [\alpha_1^{(j)}, \alpha_2^{(j)}, ..., \alpha_m^{(j)}] \in \mathbb{N}_0^m$ to denote the multi-index vector corresponding to the $j$-th element of the monomial basis such that $\sum_{k=1}^{m} \alpha_k^{(j)} \leq d$. As an example, for a system where $m = 2$ and $d = 2$, there are $p = 6$ basis terms where

$$
\begin{align*}
\alpha^{(1)} &= [0, 0], & \alpha^{(2)} &= [1, 0], & \alpha^{(3)} &= [0, 1] \\
\alpha^{(4)} &= [2, 0], & \alpha^{(5)} &= [1, 1], & \alpha^{(6)} &= [0, 2].
\end{align*} \quad (8)
$$

Although we use the set of multivariate monomials as the basis, the methods presented in this paper could be applied to systems with other types of basis elements.
2.2. Additional notation

Throughout the paper we use an asterisk (*) and tilde (~) to refer to true state values and denoised measurements, respectively. When referencing the original noisy data no decorations are used. Analogous decorations are used to refer to functions or matrices evaluated with a given state version. When the same method or logic can be applied to each state in the system, we often drop the subscript $k$ and, e.g., refer to an arbitrary state vector as $u \in \mathbb{R}^N$.

In DSINDy we use a discrete integral operator $T \in \mathbb{R}^{N \times N}$, which performs Trapezoidal quadrature, i.e.,

$$
T := \frac{\Delta t}{2} \begin{bmatrix}
0 & 0 & 0 & \ldots & 0 & 0 \\
1 & 1 & 0 & \ldots & 0 & 0 \\
1 & 2 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
1 & 2 & 2 & \ldots & 1 & 0 \\
1 & 2 & 2 & \ldots & 2 & 1
\end{bmatrix},
$$

(9)

where $\Delta t = t_{\text{end}}/(N-1)$. If $u^*, \dot{u}^* \in \mathbb{R}^N$ are the vectors of the true values and true time derivatives of a state variable, then

$$
u_0^* + Tu^* = u^* + e_q,$$

(10)

where $e_q \in \mathbb{R}^N$ is error introduced by performing trapezoidal quadrature (note that $e_{q,1} = 0$). We additionally use a finite difference matrix $D \in \mathbb{R}^{(3N-3) \times N}$ to penalizes non-smooth solutions, i.e.,

$$D := \begin{bmatrix}I \\ D_1 \\ D_2 \end{bmatrix},$$

(11)

where $D_1 \in \mathbb{R}^{(N-1) \times N}$ and $D_2 \in \mathbb{R}^{(N-2) \times N}$ are the first and second order finite difference operators, respectively. That is

$$D_1 := \frac{1}{\Delta t} \begin{bmatrix}-1 & 1 \\ -1 & 1 \\ \vdots & \ddots & \ddots \end{bmatrix}, \quad D_2 := \frac{1}{\Delta t^2} \begin{bmatrix}1 & -2 & 1 \\ 1 & -2 & 1 \\ \vdots & \ddots & \ddots \end{bmatrix}.$$  

(12)

Our denoising strategy involves projecting a vector $x \in \mathbb{R}^N$ onto the column space of a matrix $A \in \mathbb{R}^{N \times M}$. The result is given as $P_A x$ where

$$P_A := AA^\dagger.$$

(13)

Here and in other sections, we use a dagger ($\dagger$) to represent the pseudoinverse, i.e., $A^\dagger := (A^T A)^{-1} A^T$.

Remark 1. To avoid numerical instabilities, we find $P_A$ using the singular value decomposition, i.e., $P_A := U U^T$, where $U$ contains the left singular vectors of $A$ that correspond to nonzero singular values.

For an arbitrary vector $x$, we use $x_i$ to denote the $i$th element. The expected value and variance of a random variable $X$ are denoted as $E[X]$ and $\mathcal{V}[X]$, respectively. We use $1_N \in \mathbb{R}^N$ to represent a vector of ones and unless otherwise noted $\| \cdot \| = \| \cdot \|_2$.

3. DSINDy Overview

The DSINDy algorithm involves first denoising the state measurements (see Section 3.1) and then using an SOCP formulation to find the coefficients and derivatives (see Section 3.2). Here, we drop the subscript $k$ and use, for example, $u$ to refer to an arbitrary state in the system.
3.1. Projection-based state denoising

To motivate the PSDN approach, we start with the equation $\Theta^* c = \dot{u}^*$; see Section 2. We discretely integrate both sides, using the operator $T$ given by Equation (9), to obtain:

$$T\Theta^* c = \dot{u}^* - u_0^* + \epsilon_q,$$

(14)

where $\epsilon_q$ is the error introduced by quadrature. Ignoring this error, Equation (14) can be rewritten as

$$\Phi^* b \approx \dot{u}^*,$$

(15)

where

$$b := \begin{bmatrix} u_0^* \\ c \end{bmatrix} \quad \text{and} \quad \Phi^* := \begin{bmatrix} 1_N & T\Theta^* \end{bmatrix}.$$

(16)

From Equation (15) it is clear that the true state vector $\dot{u}^*$ approximately lies in the column space of $\Phi^*$. With this in mind, our denoising strategy is to project the measurements, $u_i$, onto the column space of an approximation of $\Phi^*$.

If $\Phi^*$ were known, we could directly perform this projection to find

$$\hat{u}^* := P_{\Phi^*} u,$$

(17)

where $P_{\Phi^*}$ is the projection operator as defined in Equation (13). As $N \to \infty$ and $t_{\text{end}}$ is held constant, we can show that $\hat{u}^*$ approaches $\dot{u}^*$ (see Lemma 1). In practice, $\Phi^*$ must be approximated from the noisy measurements as follows,

$$\hat{\Phi} := \begin{bmatrix} 1_N & T\hat{\Theta} \end{bmatrix},$$

(18)

where $\hat{\Theta}$ is an unbiased estimator of $\Theta^*$ such that $E[\hat{\Theta}] = \Theta^*$ (see Remark 2). For details on how to construct $\hat{\Theta}$ for the monomial basis see Appendix B.4.1. The PSDN approach is then summarized as a projection of the data onto the column space of $\hat{\Phi}$,

$$\tilde{u} := P_{\hat{\Phi}} u \quad \text{(PSDN)}.$$

(19)

Even though we are introducing additional error due to the noise in $\hat{\Phi}$, the result given by Equation (19) has similar asymptotic properties as $\hat{u}^*$. Specifically, as $N \to \infty$ and $t_{\text{end}}$ is held constant, $\tilde{u}$ approaches $\dot{u}^*$ (see Theorem 1).

Remark 2. Using $\hat{\Theta}$ to find $\tilde{u}$ helps simplify the proofs in Section 4. The calculation of $\hat{\Theta}$ may not be possible for alternative basis functions, but, in practice, we did not observe a significant change in performance when the projection was performed onto the column space of an uncentered library.

For some systems, PSDN led to near optimal denoising, but this was not always the case (see Section 6.1). Therefore, we also consider an iterative approach, i.e., IterPSDN, where we gradually project the data onto the column space of a sequence of $\Phi^*$ estimates (see Algorithm 1). At each iteration, we first find an estimate of $\Phi^*$ based on the current state vector estimates and then perform a partial projection onto the column space of the $\Phi^*$ estimate as shown on Line 6 of Algorithm 1.

Algorithm 1 requires a few additional input parameters. First, we introduce $\alpha \in [0, 1]$. If $\alpha = 1$, a full projection is performed and, if $\alpha = 0$, the state vector estimates are not changed. For $\alpha < 1$, a weighted average of the projection result and current state vector estimate is used to generate the new state vector estimate. In practice, we set $\alpha = 0.1$ but note that for $\alpha < 0.1$ similar results were obtained. The ‘CheckDiverg’ flag is included as Algorithm 1 might lead to diverging estimates. When ‘CheckDiverg=True’, an estimate of the standard deviation of the noise, i.e., $\sigma_k$ for $k = 1, 2, \ldots, m$, must be specified.
Algorithm 1 \text{IterPSDN}(\{u_k\}_{k=1}^m; \alpha, \{\sigma_k\}_{k=1}^m, \text{CheckDiverg})

1: Set $u_k^{(0)} = u_k$ for $k=1,2,...,m$
2: for $i=0,1,2,...$ do
3: Let $\Theta^{(i)}$ be monomial library evaluated at $\{u_k^{(i)}\}_{k=1}^m$
4: Set $\Phi^{(i)} = \begin{bmatrix} 1_N & T\Theta^{(i)} \end{bmatrix}$
5: for $k=1,2,...,m$ do
6: Set $u_k^{(i+1)} = \alpha P_{\sigma_k} u_k^{(i)} + (\alpha - 1) u_k^{(i)}$ \{Perform partial projection\}
7: if CheckDiverg and $\frac{1}{\sqrt{N}} \|u_k^{(i+1)} - u_k^{(0)}\| > \sigma_k$ then
8: Set $u_k^{(i+1)} = u_k^{(i)}$ \{If diverging, revert state values\}
9: end if
10: end for
11: if $\max_k \left( \|u_k^{(i+1)} - u_k^{(i)}\| \right) < 10^{-8}$ then
12: Break
13: end if
14: end for
15: return $\{u_k^{(i)}\}_{k=1}^m$

3.2. Finding derivatives and coefficients simultaneously

To find the state time derivatives while enforcing sparsity of the coefficients, we write the coefficients as a function of the derivative. We start with the equation of the system dynamics, $\Theta^* c = \dot{u}^*$, and multiply both sides of this equation by the transpose of $\Theta$ (the monomial library evaluated at the smoothed states) to obtain the oblique projection equation, $\Theta^T \Theta^* c = \Theta^T \dot{u}^*$. Then, assuming $\Theta^T \Theta^*$ is invertible, we have that

$$c = G^{-1} \tilde{\Theta}^T \dot{u}^* \quad \text{where} \quad G := \tilde{\Theta}^T \Theta^*.$$ \hspace{1cm} (20)

As $G$ cannot be directly evaluated, we explored two methods for estimating this matrix. First, we derived an estimator $\tilde{G}$ such that $\tilde{G}/N$ is consistent under certain assumptions (see Appendix B.4.2). Second, we approximated $\Theta^*$ as $\tilde{\Theta}$ and $G$ as

$$\tilde{G} := \tilde{\Theta}^T \tilde{\Theta}. \quad \hspace{1cm} (21)$$

Since we did not observe a significant difference between these two approaches, we only show results for the second method in Section 6. However, for alternative systems or sample sizes the choice of the estimator of $G$ may play a more significant role.

Remark 3. The matrix $\tilde{\Theta}$ could be any approximated version of $\Theta^*$, e.g., found using state variable estimates from Gaussian process regression. However, in practice we obtain significantly better results when $\tilde{\Theta}$ is obtained using PSDN or IterPSDN.

To find the state time derivatives we solve a convex optimization problem, where the objective enforces sparsity of the coefficients and the constraints require the derivative be smooth and agree with the data. Similar to iteratively reweighted Lasso (IRW-Lasso), see Appendix A.1, we solve this problem multiple times, where, at each iteration, we weight the coefficients based on their previous values. Because this problem can be formulated as a SOCP, we refer to the algorithm as IRW-SOCP.

More specifically, for $i=1,2,...$, we find $\tilde{u}^{(i)}$ and the corresponding coefficient estimate $c^{(i)} = \tilde{G}^{-1} \tilde{\Theta}^T \tilde{u}^{(i)}$ by solving,

$$\begin{align*}
\min_{u_0, \tilde{u}} & \quad \|W^{(i-1)} G^{-1} \tilde{\Theta}^T \tilde{u}\|_1 \quad \text{(sparsity of coefficients)} \\
\text{subject to} & \quad \|D\tilde{u}\| \leq C \quad \text{(smooth derivative)} \\
& \quad \left[\begin{array}{cc} 1 & T \end{array}\right] \begin{bmatrix} u_0 \\ \tilde{u} \end{bmatrix} - P_{\Phi}\tilde{u} \leq \gamma_i \quad \text{(match smoothed data)}.
\end{align*} \hspace{1cm} (22)$$
For the first iteration, $W^{(0)} = I$ and for $i > 0$, $W^{(i)}$ is a diagonal matrix where

$$W_{jj}^{(i)} = \frac{1}{|c_{j}^{(i)}| + \varepsilon \max_j |c_{j}^{(i)}|}.$$  \hfill (23)

In practice, we set $\varepsilon = 10^{-4}$. The finite difference matrix $D$ and discrete integral operator $T$ are as presented in Section 2.2. In the second constraint, we include a projection of $\hat{u}$ onto the column space of $\Phi$ in order to guarantee feasibility (see Section 3.2.1).

There are two hyperparameters in Equation (22) that need to be determined at each iteration, $C$ and $\gamma_i$. In practice, we set $C$ to guarantee feasibility at small values of $\gamma_i$ (see Section 3.2.1). The remaining parameter, $\gamma_i$, is determined either using the Pareto corner criteria or by directly using a theoretical value (see Section 3.2.2). Both approaches for finding $\gamma_i$ rely on theoretical results given in Section 4. We include a subscript on $\gamma$ to emphasize that its value may change at each iteration.

### 3.2.1. Setting the smoothing hyperparameter $C$ to guarantee feasibility

We estimate $C$ in Equation (22) using the PSDN results. Specifically, we find an initial estimate of the derivative,

$$\hat{u} := \hat{\Theta} \left( (\hat{\Phi}^T \hat{\Phi})^{-1} \hat{\Phi}^T \right)_{2:} \hat{u},$$  \hfill (24)

where the ‘2 :’ subscript implies that we are removing the top row of the matrix. We then use this estimate to set the value of $C$ as,

$$C := \|D \hat{u}\|.$$  \hfill (25)

At this $C$, Equation (22) is feasible for all $\gamma > 0$ (see Appendix B.5).

**Remark 4.** In practice, the $C$ found with Equation (25) was often larger than the optimal value. However, generally the results were not very sensitive to this parameter.

### 3.2.2. Setting the data-matching hyperparameter $\gamma$

Consider the data-matching constraint in Equation (22) where $\hat{u}$ is replaced with the true time derivative vector, i.e., $\hat{u}^*$. The constraint is then approximately

$$\|u^* - P_{\Phi} \hat{u}\| \leq \gamma,$$  \hfill (26)

where here we are ignoring quadrature error. In Section 4, we show that at large $N$ the best possible expected value of this data-matching error is given as,

$$E[\|u^* - P_{\Phi} u\|] \approx \sigma \sqrt{p + 1} =: \gamma_{\text{exp}},$$  \hfill (27)

where $\sigma$ is the standard deviation of the measurement noise. Although this optimal value will not necessarily be obtained by PSDN, in Section 6.1 we show that IterPSDN does obtain this value for the systems considered. Therefore, we predict the optimal value of $\gamma_i$, for $i = 1, 2, ...,\ $ will be close to this estimate, i.e., close to $\gamma_{\text{exp}}$.

With this in mind, we consider two methods for finding the data-matching hyperparameter. First, we use the corner point of the Pareto curve, see Appendix A.2 for details, but restrict the search space to lie within an order of magnitude of $\gamma_{\text{exp}}$, i.e.,

$$\gamma_i \in [0.1 \cdot \gamma_{\text{exp}}, 10 \cdot \gamma_{\text{exp}}] \quad \text{for} \quad i = 1, 2, ...,\ $$  \hfill (28)

In practice, we find that this improves the performance of the corner finding algorithm and a clear corner point exists within this range. This is the approach used in the main results section, i.e. Section 6. We also explore the option of setting $\gamma_i = \gamma_{\text{exp}}$ for all $i$. In Appendix C.3, we compare the performance of these two methods.

**Remark 5.** By using the results from PSDN or IterPSDN in Equation (22), we predict a hyperparameter $\gamma_{\text{exp}}$ that is independent of $N$. If we had instead used the original measurements, the data-matching error $\|u^* - u\|$ would grow with the square root of $N$, leading to a less restrictive constraint.
4. Theoretical work

In this section we derive error bounds for PSDN as given by Equation (19). We show that, in the limit as the number of measurements goes to infinity and the training time is held constant, the projected state variables approach their true values with respect to the relative ℓ2 error. For notational simplicity we introduce the following definitions:

\[
\Psi^* := \frac{1}{\sqrt{N}}\Phi^*, \quad \Delta \Theta := \hat{\Theta} - \Theta^*, \quad \Delta \Phi := \hat{\Phi} - \Phi^*, \quad \Delta P := P_{\Phi} - P_{\Phi^*},
\]

where \(\Theta^*, \Phi^*,\) and \(\hat{\Phi}\) are as given by Equations (4), (16) and (18). The matrix \(\hat{\Theta}\) is an unbiased estimate of the noisy monomial library (see Appendix B.4.1), and the projection operator \(P\) is as defined in Equation (13). Throughout this section we drop the subscript on \(u_k\) as the results can be applied to any of the \(m\) states.

4.1. An optimal error estimate for PSDN

We obtain an optimal error estimate by calculating the expected squared error of projecting the noisy data onto the column space of \(\Phi^*\). This result informs upon the best possible performance of PSDN, see Equation (19). Additionally, these results can be used to find the value of the data-matching hyperparameter \(\gamma\) in IRW-SOCP, i.e., Equation (22). We first derive an upper bound on the expected squared error of \(\hat{u}^*\) as given by Equation (17).

**Lemma 1.** Suppose the function \(u^*(t)\) is three times continuously differentiable for all \(t \in [0, t_{\text{end}}]\), then

\[
E\left[\|P_{\Phi}u - u^*\|^2\right] \leq \sigma^2(p + 1) + \frac{C_1^2}{(N - 1)^3},
\]

where

\[
C_1 = \frac{t_{\text{end}}^3}{12} \max_{t \in [0, t_{\text{end}}]} \left| \frac{d^3 u^*}{dt^3} \right|.
\]

**Proof.** First let

\[
b := \begin{bmatrix} u_0^* \\ c \end{bmatrix}, \quad e_q := \Phi^* b - u^*, \quad P_{\Phi^*} := I - P_{\Phi^*},
\]

where \(e_q \in \mathbb{R}^N\) is the trapezoidal quadrature error, see Equation (10). Then the error of projecting \(u^*\) onto the column space of \(\Phi^*\) is given as,

\[
P_{\Phi^*}u^* - u^* = P_{\Phi^*}(\Phi^* b - e_q) - u^* = \Phi^* b - P_{\Phi^*} e_q - u^*
\]

\[
= (e_q + u^*) - P_{\Phi^*} e_q - u^* = P_{\Phi^*} e_q.
\]

In turn, the error of projecting \(u = u^* + \epsilon\) onto the column space of \(\Phi^*\) is

\[
P_{\Phi^*}u - u^* = P_{\Phi^*} u^* + P_{\Phi^*} \epsilon - u^* = P_{\Phi^*} \epsilon + P_{\Phi^*} e_q,
\]

where we use Equation (32) to obtain the second equality. Since \(P_{\Phi^*}\) is an orthogonal projection, the norm of the right hand side of Equation (33) is

\[
\|P_{\Phi^*} \epsilon + P_{\Phi^*} e_q\|^2 = \|P_{\Phi^*} \epsilon\|^2 + \|P_{\Phi^*} e_q\|^2.
\]

We combine Equations (33) and (34) and take the expectation to obtain,

\[
E\left[\|P_{\Phi^*}u - u^*\|^2\right] = E\left[\epsilon^T P_{\Phi^*} \epsilon\right] + \sum_{i,j} E[(\epsilon_i \epsilon_j) (P_{\Phi^*})_{ij}] + \|P_{\Phi^*} e_q\|^2
\]

\[
= \sum_i E[\epsilon_i^2] (P_{\Phi^*})_{ii} + \|P_{\Phi^*} e_q\|^2 = \sigma^2 \text{Tr}(P_{\Phi^*}) + \|P_{\Phi^*} e_q\|^2.
\]
The trace of a projection matrix is equal to the dimension of the target space, i.e., \( \text{Tr}(P_{\Phi}) = \text{Rank}(\Phi^*) \), which implies
\[
\mathbb{E}\left[ \| P_{\Phi^*} \mathbf{u} - \mathbf{u}^* \|^2 \right] = \sigma^2 \text{Rank}(\Phi^*) + \| P_{\Phi^*}^\perp \mathbf{e}_q \|^2 \leq \sigma^2 (p + 1) + \| \mathbf{e}_q \|^2.
\] (36)
The lemma is then proven by applying a bound (see Lemma B1 in Appendix B) on the norm of the quadrature error, i.e.,
\[
\| \mathbf{e}_q \| \leq C_1 (N - 1)^{-3/2}.
\] (37)
This bound follows almost immediately from the well known trapezoidal rule error bound.

Remark 6. Since \( \mathbb{E}[X]^2 < \mathbb{E}[X^2] \), Lemma 4 provides an upper bound on the expected relative error. Although, we have not provided a lower bound on this error, we still find that \( \sigma \sqrt{p + 1} \) provides a reasonable estimate (see Section 6.1).

4.2. An upper bound on the expected error of PSDN

Our main result provides a bound on the expected value of the PSDN error. Under certain assumptions, this result shows that as the sample size increases the relative error of PSDN goes to zero.

Theorem 1. Let \( \hat{\Phi} \) be given by Equation (18) where \( T \) is the discrete integration matrix that performs trapezoidal quadrature. Suppose \( \text{rank}(\hat{\Phi}) = \text{rank}(\Phi^*) \), \( \| (\Phi^*)^\dagger \| \| \Delta \Phi \| < 1/4 \), and \( \mathbf{u}^*(t) \) is three times continuously differentiable for \( t \in [0, \tau_{\text{end}}] \). Then the expected relative error of PSDN is bounded as follows
\[
\mathbb{E}\left[ \| P_{\hat{\Phi}^*} \mathbf{u} - \mathbf{u}^* \| \right] \leq \frac{\sigma (\sqrt{p + 1} + C_2)}{\| \mathbf{u}^* \|} + \frac{C_1}{(N - 1)^{3/2} \| \mathbf{u}^* \|} + \frac{C_2}{\sqrt{N}},
\] (40)
where \( C_1 \) as defined in Lemma 4 and
\[
C_2 = \tau_{\text{end}} \| (\Phi^*)^\dagger \| \sum_{j=1}^{p} \max_k \mathbb{V} |\Delta \Theta_{jk}|^{1/2}.
\] (41)

A few comments on Theorem 1 are warranted. In Appendix B we show \( \| (\Phi^*)^\dagger \| \) is uniformly bounded over \( N \), which implies the bound given by Equation (40) goes to zero as the measurement density increases. The first assumption of Theorem 1, i.e., \( \text{rank}(\hat{\Phi}) = \text{rank}(\Phi^*) \), is valid if the library \( \Theta^* \) has full column rank. This is true for the systems examined here, see Section 5 and we leave it as future work to examine the performance of PSDN when this is not the case. In Corollary 2 given in Section 4.2.2 we show that the second assumption, i.e., \( \| (\Phi^*)^\dagger \| \| \Delta \Phi \| < 1/4 \), is satisfied in expectation at large \( N \).

We provide the proof of Theorem 1 in Section 4.2.2. In the proof we use the triangle and Cauchy-Schwarz inequalities to show that,
\[
\mathbb{E}\left[ \| P_{\hat{\Phi}^*} \mathbf{u} - \mathbf{u}^* \| \right] \leq \mathbb{E}\left[ \| P_{\Phi^*} \mathbf{u} - \mathbf{u}^* \| \right] + \left( \mathbb{E}\left[ \| \Delta P \|^2 \right] \right)^{1/2} \left( \| \mathbf{u}^* \| + \sqrt{N} \sigma \right).
\] (42)
Since Lemma 4 provides a bound for the first term, it remains to show the expected value of \( \| \Delta P \|^2 \) goes to zero as \( N \to \infty \), which we do next in Section 4.2.1.
4.2.1. Bounding the expected projection error \( \| \Delta P \| \)

We obtain a bound for the expected projection error using the results of two lemmas. In Lemma 2, we derive a bound for \( \mathbb{E} [\| \Delta \Phi \|] \) that depends on the elementwise variance of \( \Delta \Theta = \hat{\Theta} - \Theta^* \). We then, in Lemma 3, derive a bound for \( \mathbb{E} [\| \Delta P \|] \) that depends on \( \mathbb{E} [\| \Delta \Phi \|] \).

**Lemma 2.** The expected error of \( \hat{\Phi} \) is bounded as follows:

\[
\mathbb{E} [\| \Delta \Phi \|^2] \leq \frac{t^2}{2} \sum_{j=1}^{p} \max_k (V[\Delta \Theta_{kj}]) .
\]

**Proof.** Using \( \Delta \Phi = \hat{\Phi} - \Phi^* = [0 \ T \Delta \Theta] \) where \( T \) is given by Equation (9), we have that

\[
\| \Delta \Phi \|^2_F = \| T \Delta \Theta \|^2_F = \sum_{j=1}^{p} \sum_{i=2}^{N} \left( \sum_{k=1}^{i} T_{ik} \Delta \Theta_{kj} \right)^2 .
\]

(44)

We use this result, i.e., Equation (44), to bound the expected value of \( \| \Delta \Phi \|^2_F \),

\[
\mathbb{E} [\| \Delta \Phi \|^2_F] \leq \sum_{j=1}^{p} \sum_{i=2}^{N} \mathbb{E} \left[ \left( \sum_{k=1}^{i} T_{ik} \Delta \Theta_{kj} \right)^2 \right] = \sum_{j=1}^{p} \sum_{i=2}^{N} V \left[ \sum_{k=1}^{i} T_{ik} \Delta \Theta_{kj} \right] \leq \Delta t^2 \sum_{j=1}^{p} \sum_{i=2}^{N} \left( i - \frac{3}{2} \right) \max_k (V[\Delta \Theta_{kj}]) .
\]

The inequality follows because the sum of the squared elements in the \( i \)th row of \( T \) is

\[
\sum_{k=1}^{i} T_{ik}^2 = \frac{\Delta t^2}{4} (4i - 6) = \Delta t^2 \left( i - \frac{3}{2} \right) .
\]

Continuing, we have that

\[
\mathbb{E} [\| \Delta \Phi \|^2_F] \leq \Delta t^2 \sum_{j=1}^{p} \sum_{i=1}^{N-1} \left( i - \frac{1}{2} \right) \max_k (V[\Delta \Theta_{kj}])
\]

\[
= \Delta t^2 \frac{(N-1)^2}{2} \sum_{j=1}^{p} \max_k (V[\Delta \Theta_{kj}]) = \frac{\Delta t^2}{2} \sum_{j=1}^{p} \max_k (V[\Delta \Theta_{kj}]) .
\]

The proof is completed by noting that the 2-norm of a matrix is always less than the Frobenius norm. □

The following lemma provides a bound on \( \| \Delta P \| \) that depends on \( \| \Delta \Phi \| \). In order to obtain this bound we use results from [26] as presented in Appendix B.2

**Lemma 3.** Suppose \( \text{rank}(\hat{\Phi}) = \text{rank}(\Phi^*) \) and \( \| (\Phi^*)^\dagger \| \| \Delta \Phi \| < 1/4 \), then

\[
\| \Delta P \|^2 \leq 2 \| (\Phi^*)^\dagger \|^2 \| \Delta \Phi \|^2 .
\]

(45)

**Proof.** Let \( E_{11}, E_{21}, \) and \( \beta \) be as defined in Theorem B1 (a modified version of Theorem 4.1 from [26]) and define the following scalars,

\[
X := \| E_{11} \| \| (\Phi^*)^\dagger \|, \quad Y := \| E_{21} \| \| (\Phi^*)^\dagger \|. \]

First note that,

\[
\beta \| E_{21} \| \| \Phi^* \| = \frac{\| E_{21} \| \| (\Phi^*)^\dagger \|}{1 - \| E_{11} \| \| (\Phi^*)^\dagger \|} = \frac{Y}{1 - X} .
\]
Using $\|E_{11}\| < \|\Delta \Phi\|$ and the lemma assumptions, we have that,

$$X = \|E_{11}\| (\Phi^*)^\dagger < \|\Delta \Phi\| (\Phi^*)^\dagger < 1/4.$$  

We next square the bound given by Equation (B.9) and rewrite it as,

$$\|\Delta P\|^2 \leq \frac{Y^2}{1 + Y^2/(1 - X)^2} = \frac{Y^2}{(1 - X)^2 + Y^2} \leq 2Y^2. \quad (46)$$

The second inequality follows because, for $X < 1/4$,

$$2Y^2((1 - X)^2 + Y^2) = 2Y^2(1 - 2X + X^2 + Y^2) \geq 2Y^2(1 - 2X) \geq Y^2.$$

The lemma statement follows from Equation (46) since $Y \leq \|(\Phi^*)^\dagger\|\|\Delta \Phi\|$. \hfill \qed

### 4.2.2. Proof of the main result

We first state a corollary that shows an assumption of Theorem 1 is satisfied at large $N$. This corollary follows immediately from Lemma 2 and Corollary B1 which shows the value of $\|(\Phi^*)^\dagger\|$ is bounded from above for all $N$.

**Corollary 2.** Suppose the assumptions of Corollary B1 in Appendix B hold, i.e., $u_k(t)$ for $k = 1, 2, \ldots, m$ is twice continuously differentiable for $t \in [0, t_{\text{end}}]$ and in the limit as $N \to \infty$ the rank of $(\Phi^*)^T \Phi^*$ does not change. Then there exists $N_0$ such that for $N > N_0$

$$\|(\Phi^*)^\dagger\| \leq \|\Delta \Phi\| < 1/4. \quad (47)$$

We next provide the proof of Theorem 1 using the results of Lemmas 1 to 3.

**Proof of Theorem 1.** We first bound the value of $\|P_\Phi u - u^*\|$ using the triangle inequality

$$\|P_\Phi u - u^*\| \leq \|P_\Phi^\ast u - u^*\| + \|\Delta P\|\|u^* + \epsilon\| \leq \|P_\Phi^\ast u - u^*\| + \|\Delta P\|\|u^*\| + \|\epsilon\|, \quad (48)$$

where $\Delta P = P_\Phi - P_\Phi^\ast$. Taking expectations of both sides of Equation (48) and using Jensen’s inequality, we have that

$$\mathbb{E}[\|P_\Phi u - u^*\|] \leq T_1 + T_2 + T_3, \quad (49)$$

where

$$T_1 := \sqrt{\mathbb{E}[\|P_\Phi^\ast u - u^*\|^2]}, \quad T_2 := \|u^*\|\sqrt{\mathbb{E}[\|\Delta P\|^2]}, \quad T_3 := \sqrt{\mathbb{E}[\|\Delta P\|^2\|\epsilon\|^2]}.$$

We first obtain a bound for $T_1$ using Lemma 1 and the triangle inequality,

$$T_1 = \sqrt{\mathbb{E}[\|P_\Phi^\ast u - u^*\|^2]} \leq \sigma\sqrt{p + 1} + \frac{\sqrt{C_1}}{(N - 1)^{3/2}}. \quad (50)$$

To bound $T_2$ and $T_3$ we first use Lemmas 2 and 3 to bound $\mathbb{E}[\|\Delta P\|^2]$,

$$\mathbb{E}[\|\Delta P\|^2] \leq 2\|(\Phi^*)^\dagger\|^2\|\Delta \Phi\|^2 \leq \frac{\|\langle \Phi^\ast \rangle\|^2}{N} \sum_{j=1}^{p} \max_k (V[\Delta \Theta_{kj}]). \quad (51)$$

Equation (51) can immediately be used to bound $T_2$. For $T_3$ we first use Cauchy Schwarz to show

$$T_3^2 = \mathbb{E}[\|\Delta P\|^2\|\epsilon\|^2] \leq \mathbb{E}[\|\Delta P\|^2] \mathbb{E}[\|\epsilon\|^2] = \sigma^2 N \mathbb{E}[\|\Delta P\|^2]. \quad (52)$$

Putting Equations (49) to (52) together, we obtain the final result. \hfill \qed
Table 1: Implementation details for ℓ₁-SINDy, WSINDy, and DSINDy. In parentheses, we specify how hyperparameters were found. For more details see Appendix A.

| Method     | Denoising | Derivative Estimation       | Coefficient Estimation       |
|------------|-----------|-----------------------------|------------------------------|
| ℓ₁-SINDy   | IterPSDN  | Tikhonov Reg. (Pareto)      | IRW-Lasso (Pareto)           |
| WSINDy     | Weighted Avg. | Not Applicable               | Modified STLS (see [24])    |
| DSINDy     | IterPSDN  | IRW-SOCP (Pareto or theory)  |                              |

5. Methods for obtaining and comparing numerical results

5.1. Implementation details for equation discovery algorithms

We compared the performance of DSINDy with multiple alternative methods, including Gaussian process (GP) regression [27], ℓ₁-SINDy [11, 25], and WSINDy [23]. To perform WSINDy we used the most recent MATLAB code available on GitHub [1]. In general, when running WSINDy we used the default parameters given in the MATLAB code. The one exception to this was for the Lorenz 96 model, where WSINDy did poorly at low noise levels. To fix this we set the parameter ‘alpha_loss’ = 0.4 (default: 0.8) and ‘overlap_frac’ = 0.1 (default: 1.0). The python code for all other computations, i.e., GP regression, ℓ₁-SINDy and DSINDy, is available on our group’s GitHub page [2].

Table 1 provides a summary and comparison of the major steps behind each of the equation discovery algorithms. For all three methods we include a priori denoising of the data. For ℓ₁-SINDy and DSINDy the denoising is done with IterPSDN, and for WSINDy we used the built-in smoothing algorithm in the WSINDy MATLAB code. The ℓ₁-SINDy algorithm obtains the state time derivatives using Tikhonov regularization and finds the basis coefficients using IRW-Lasso. In contrast, WSINDy does not require a derivative estimation and finds the coefficients using sequential-thresholding least squares (STLS). A complete mathematical description of these alternative methods is given in Appendix A.

Our results rely on several open-source python packages. To perform GP regression, we used the gaussian_process package from the scikit-learn python library [28] and constructed a GP with a radial basis function plus white noise kernel. To implement ℓ₁-SINDy we used the scikit-learn implementation of Lasso, and for DSINDy the optimization problem given by Equation (22) was solved using the python CVXOPT package [29] coupled with MOSEK Solver [30].

Remark 7. A review of both local and global smoothing methods, in the context of equation discovery, is provided by [12]. For the present work, we compare our denoising approach with GP regression, as a global denoising method. GP regression offers flexibility in fitting nonlinear functions and has available python implementations for finding the GP hyperparameters [28].

5.2. Example ODE systems

The equations and parameters for the ODE systems we considered are given in Table 2. For both the Duffing and Van der Pol oscillator, the two state variables, $u_1$ and $u_2$, represent the displacement and velocity, respectively. For the Duffing oscillator, we considered two parameter sets, which both result in a damped system with a stable equilibrium. The first parameter set (PS1) explores the ability of DSINDy to recover coefficients that vary significantly in magnitude, and the second parameter set (PS2) was used in the WSINDy work [23]. We next considered the Van der Pol oscillator where, by setting $\gamma > 0$, we are guaranteed the system enters a stable limit cycle. This example demonstrates the ability of DSINDy to capture this cyclical behavior. Finally, we considered two chaotic systems, the Rössler attractor and the Lorenz 96 model. Using the given parameter set, the Rössler attractor has a chaotic attractor and the third state $u_3$ often has values close to zero. We studied this system to explore how DSINDy performs in this more complex scenario. We additionally studied the Lorenz 96 system with $m = 6$ states to examine how DSINDy performs on larger dimensional chaotic systems.

1 https://github.com/MathBioCU/WSINDy_ODE (accessed 9/28/2022)
2 https://github.com/CU-UQ/DSINDy
5.3. Error metrics for comparing results

In the results section, we report on the error of denoising and system recovery. Typically, the error is reported as a relative $\ell_2$-error. For example, the relative error of a coefficient vector estimate $c$, given a true vector $c^*$, is given as

$$\mathcal{E}(c; c^*) := \frac{\|c - c^*\|}{\|c^*\|}. \quad (53)$$

For the Duffing oscillator, the Van der Pol oscillator, and the Rössler attractor, we report the reconstruction error. This error is calculated by simulating the system with the same initial conditions as the training data but for double the training time, i.e., $t_{\text{test}} = 2t_{\text{end}}$. For the Lorenz 96 model, we instead explore the ability of DSINDy to predict, new, short term dynamics. To do this, we use the same initial conditions as the training data and find the true system value at $t = t_{\text{end}} = 5$. We use this as the initial condition for testing, i.e. $u_{IC,\text{test}} = u^*(t_{\text{end}})$, and simulate the learned dynamics using the initial condition $u_{IC,\text{test}}$. We record the time over which the simulated system has a large predictive ability, i.e., the time for which the relative reconstruction error is less than 10% for all state variables. For all simulations, we record the state estimates using a time interval of $\Delta t = 0.01$ and compare the result to the true solution using Equation (53).

If the relative reconstruction error exceeds 1.0 for any state in the system, then this is classified as a failure, and the reconstruction error is set to 1.0. A system that becomes unbounded is also classified as a failure, and the reconstruction error is set to 1.0 for all state variables. This approach allows us to compare average reconstruction results for multiple methods and noise realizations.

In general, if figures showing the denoising and derivative estimation errors we provide the mean ± standard deviation (std) of 30 noise replications. However, in the figures showing the coefficient and reconstruction errors, we show the standard error of the mean (sem) instead of the standard deviation in order to make the plots more readable.

6. Results

We first compare numerical results for the projection-based denoising methods, i.e., PSDN and IterPSDN, with theoretical predictions (Section 6.1). We then examine the results of the complete DSINDy algorithm and compare the error of the denoising step with GP regression (Section 6.2.1) and the error of the IRW-SOCP derivative estimation with Tikhonov regularization (Section 6.2.2). Ultimately, DSINDy improves upon coefficient recovery and state variable estimation compared with $\ell_1$-SINDy and WSINDy (Section 6.2.3).
Relative $\tilde{u}_1$ error

$\tilde{u} = P\Phi^*u$

$E_{\text{noisy}}(u^*_1)$

$E_{\text{theory}}(u^*_1)$

$E(\tilde{u}_1;u^*_1)$

$\tilde{u} = P\hat{\Phi}u$

$\tilde{u} = \text{IterPSDN}(\alpha = 0.1)$

Figure 1: Comparing denoising error with theoretical predictions for the Duffing oscillator (PS1). Results show mean ± std for 50 sample replications at noise level $\sigma^2 = 0.1$ when the integrated library $\Phi^*$ is known (left), when using PSDN (middle), and when using IterPSDN, Algorithm 1 (right). Each row represents one of the system states. For system parameters see Table 2.

Relative $\tilde{u}_2$ error

$E_{\text{noisy}}(u^*_2)$

$E_{\text{theory}}(u^*_2)$

$E(\tilde{u}_2;u^*_2)$

Figure 2: Comparing denoising error with theoretical predictions for the Van der Pol oscillator. Results show mean ± std for 50 sample replications at noise level $\sigma^2 = 0.1$ when the integrated library $\Phi^*$ is known (left), when using PSDN (middle), and when using IterPSDN, Algorithm 1 (right). Each row represents one of the system states. For system parameters see Table 2.
6.1. Projection denoising results agree with theoretical predictions

Here, we compare denoising results with the theoretical bounds presented in Section 4. We compare the relative error of a denoised state vector, i.e., $\mathcal{E}(\tilde{u}; u^*)$, see Equation (53), with the optimal error estimate derived in Section 4, i.e.,

$$\mathcal{E}_{\text{theory}}(u^*) := \frac{\sigma \sqrt{p + 1}}{\|u^*\|}. \quad (54)$$

This value represents the minimal error we can expect to obtain in PSDN or IterPSDN. For comparison, we also show the expected error of the original noisy data, i.e.,

$$\mathcal{E}_{\text{noisy}}(u^*) := \frac{\sigma \sqrt{N}}{\|u^*\|}. \quad (55)$$

We compare $\mathcal{E}$, $\mathcal{E}_{\text{theory}}$, $\mathcal{E}_{\text{noisy}}$ at a range of sample sizes $N$ for the four ODE systems presented in Table 2 (Figures 1 to 3). Results for the Lorenz 96 model are given in Appendix C.1. At each $N$, we performed 50 sample replications to approximate the mean and standard deviation of the state prediction error following denoising. Results are shown for three variations of the projection-state denoising approach: denoising when $\Phi^*$ is known (first column of Figures 1 to 3), PSDN (second column), and IterPSDN (third column). When applying Algorithm 1, we set the ‘CheckDiverg’ flag equal to ‘True’ and used the true standard deviation.
of the measurement noise. For the Duffing and Van der Pol oscillators and the Lorenz 96 model, divergence did not occur. However, for the Rössler attractor, checking for divergence led to improved solutions.

**Remark 8.** The standard deviation of the noise $\sigma$ is often known based on the accuracy of the measurement system or can be estimated. For example, in Section 6.2.1, $\sigma$ is estimated using the result of GP regression.

For all four systems, we find that as expected (see Lemma 1 and Corollary 1) the projection error when $\Phi^*$ is known matches the theoretical value. For PSDN, some state variables have an increased error compared to this optimal value, i.e., see the second state in the Duffing oscillator system (middle column, bottom row of Figure 1), the second state of the Van der Pol oscillator (middle column, bottom row of Figure 2), and the third state of the Rössler attractor (middle column, bottom row of Figure 3). However, IterPSDN improves upon PSDN and obtains the optimal value for the aforementioned states. The one exception to this is the third state of the Rössler attractor where IterPSDN improves upon PSDN, but the error is still slightly larger than the optimal value. These results suggest that IterPSDN helps remove the error introduced in the estimation of $\Phi^*$.

**Remark 9.** Although IterPSDN improves upon the performance of PSDN for the systems considered, this error reduction is not guaranteed. We leave a more in depth analysis and comparison of PSDN and IterPSDN to future work.

### 6.2. DSINDy Results

We applied the complete DSINDy algorithm (see Table 1) to the example systems in Section 5 using the parameters given in Table 2. Unless otherwise noted, for all systems we consider a sample size of $N = 1000$ and noise levels ranging from $\sigma = 10^{-3}$ to $\sigma = 1$, where $\sigma$ represents the standard deviation of the Gaussian measurement noise. Since the magnitude of each state is different, the signal-to-noise level also varies for each state and system. The DSINDy results in this section were obtained by setting the hyperparameter $\gamma$ in Equation (22) using the corner point of the Pareto curve where the search space was limited as shown in Equation (28). In general when applying Algorithm 1 we set the ‘CheckDiverg’ flag equal to ‘True’ and set the standard deviations $\sigma_k$ for $k = 1, ..., m$ using the estimate obtained from GP regression. For the Duffing oscillator results at $N = 8000$, we ran IterPSDN with the ‘CheckDiverg=False’. Divergence was not an issue, and this avoided performing GP regression to find the standard deviation.

#### 6.2.1. The IterPSDN step of DSINDy improves state estimation

For all systems, IterPSDN (Algorithm 1) outperforms or is equivalent to GP regression (Figure 4). IterPSDN and GP regression performed equally well at denoising the state measurement in the Duffing oscillator (PS1) (top row of Figure 4) as well as the Duffing oscillator (PS2) and the Lorenz 96 model (results not shown). For the Van der Pol oscillator and Rössler attractor, IterPSDN improves the state estimation (middle and bottom rows of Figure 4). In addition to improving state estimates, IterPSDN is faster than GP regression (given the standard deviation of the noise can be estimated quickly), and using projection state denoising allows us to estimate the relevant hyperparameters in the IRW-SOCP formulation (see Section 3).

#### 6.2.2. The IRW-SOCP step of DSINDy improves state time derivative estimation

The IRW-SOCP step of DSINDy, see Equation (22), led to improved derivative estimation compared with Tikhonov regularization (see Figure 5). Results for the Lorenz 96 model are given in Appendix C.1. Although we observe improvement across multiple noise levels, this improvement becomes less significant as the standard deviation of the noise increases. Indeed, in some cases Tikhonov regularization slightly outperforms DSINDy at the highest noise level considered, e.g., see second state of Duffing oscillator (PS1) (Figure 5 top row). However, as described below, increasing the sample size may alleviate this issue.

We examined whether increasing the sample size improved the prediction of the state time derivative. Specifically, we ran DSINDy using $N = 250, 500, 1000, 2000, 4000, 8000$ measurements from the Duffing oscillator (PS2) using the same parameters given in Table 2. In DSINDy, the error of the derivative estimation consistently decreases with the square root of $N$ (Figure 6). However, for Tikhonov regularization this

16
relationship is not always evident, e.g., see middle and right column of Figure 6. Additionally, we find that as the sample size increases, DSINDy starts to outperform Tikhonov regularization at $\sigma = 1$.0 (see right column of Figure 6).

6.2.3. DSINDy improves coefficient recovery and system reconstruction

DSINDy reduces the error of coefficient recovery and/or system reconstruction compared with $\ell_1$-SINDy and WSINDy (Figures 7 to 11). Note that results for Duffing Oscillator (PS1) are given in Appendix C.2. In terms of coefficient recovery, DSINDy generally outperforms $\ell_1$-SINDy and WSINDy, e.g., see top row of Figures 7 and 8 and top two rows of Figure 9. WSINDy only obtains better coefficients, compared with DSINDy, for the Rössler attractor system (Figure 8 top row). However, this does not translate to a better system recovery (Figure 8, bottom row). This is likely because WSINDy fails to identify one of the small nonzero coefficients. Indeed, at $\sigma = 0.1$, in 14 out of 30 replications, WSINDy fails to identify the coefficient that corresponds with $\beta$ in the Rössler attractor system (see Table 2).

In terms of system reconstruction, DSINDy consistently outperforms both $\ell_1$-SINDy and WSINDy, e.g., see Figures 8, 11 and 12. The one exception to this is the Van der Pol oscillator where the reconstruction error of DSINDy is equivalent to WSINDy (Figure 7, bottom row). DSINDy led to accurate predictions of short term dynamics in the chaotic Lorenz 96 model, even when the measurements were highly corrupted by noise (see Figure 9, bottom panel). Figure 10 shows an example of this predictive capability, alongside state measurements obtained at $\sigma = 1$. In contrast to DSINDy, neither $\ell_1$-SINDy nor WSINDy were able to predict short term dynamics of the Lorenz 96 model at the highest noise level considered.

The benefits of DSINDy are also apparent for the Duffing oscillator (PS2) system, where we consider the
Figure 5: DSINDy improves derivative estimation compared with Tikhonov regularization. Showing the mean ± std of 30 realizations with $N = 1000$ at each noise level. For system parameters see Table 2.
Figure 6: Increasing the number of training samples improves DSINDy derivative estimation for the Duffing oscillator system (PS2). Each column corresponds to a different noise level. Showing the mean ± std of 30 noise realizations at each sample size. For system parameters see Table 2.

Figure 7: Coefficient and reconstruction error for Van der Pol system. Showing the mean ± sem of 30 realizations with $N = 1000$ at each noise level. The failure rate represents the fraction of simulations out of 30 that failed. For system parameters see Table 2.
Figure 8: Coefficient and reconstruction error for the Rössler attractor. Showing the mean ± sem of 30 realizations with $N = 1000$ at each noise level. The failure rate represents the fraction of simulations out of 30 that failed. For system parameters see Table 2.
Figure 9: Coefficient errors for the 6 states of the Lorenz 96 system. Showing the mean ± sem of 30 realizations with $N = 2000$ at each noise level. The bottom plot shows the time at which the relative reconstruction error of any state exceeds 10% when the learned dynamics are initialized at $t = t_{end} = 5$. Note that a prediction time less than $10^{-2}$ implies zero predictive ability as this is time step of the simulation. For system parameters see Table 2. An example simulation result with the prediction time labeled is given in Figure 10.
Figure 10: DSINDy improves predictions of the Lorenz 96 model compared with $\ell_1$-SINDy and WSINDy. Left column: Example set of measurements obtained for $t \in [0, 5]$ ($\sigma = 1$, $N = 2000$). Right column: Predicted system dynamics using example measurements starting at $t_{IC, test} = 5$. The vertical black dotted line represents the time at which the relative DSINDy prediction error for one of the states, i.e., the third state, exceeds 10%.
effect of sample size (Figure 11). At low noise levels, DSINDy and ℓ₁-SINDy performed similarly (Figure 11, first column), whereas WSINDy struggled at the low sample sizes, i.e., at \( N = 250, 500 \). At the two highest noise levels, \( \sigma = 0.1 \) and \( \sigma = 1.0 \), DSINDy improves upon both ℓ₁-SINDy and WSINDy in terms of both the coefficient and reconstruction errors (Figure 11, second and third columns). In particular, at the highest noise level considered, increasing the sample size improves the performance of DSINDy but does not improve the performance of the other two methods.

### 6.2.4. Method for selecting data-matching hyperparameter \( \gamma \)

We did not observe a significant affect of the data-matching hyperparameter selection method on the performance of DSINDy in any of the ODE systems (see Appendix C for complete results). Recall, that we either found \( \gamma \) in Equation (22) using the corner point of the Pareto curve (as done in the previous section) or using the estimate given in Equation (27). Although both methods rely on our theoretical results, the second approach is much faster as we do not have to explore the hyperparameter space. These results suggest that directly using the theoretical estimate may improve algorithmic speed without compromising performance.

### 7. Discussion

We developed DSINDy to improve ODE discovery when state measurements are highly corrupted by noise. The method leverages the assumed (overcomplete) basis for the dynamics to denoise the state variable measurements, and finds the state variable time derivatives while enforcing sparsity of the coefficients (Section 3). Our goal was to improve derivative estimation in the presence of noise, and, in turn improve overall system recovery. We found that DSINDy successfully accomplished this goal for the Duffing oscillator, the Van der Pol oscillator, the Rössler attractor, and the higher dimensional Lorenz 96 model (Section 6).

The performance of DSINDy improves as the sample size increases. Theoretical results for the a priori denoising strategy PSDN show that the denoised state variables asymptotically approach their true values. Although theoretical results are not available for iterative projections (IterPSDN, Algorithm 1), we numerically found the error of IterPSDN also decreased with the number of samples (see Figures 1 to 3).

Additionally, for the Duffing oscillator we see improvements in system recovery as the sample size increases (see Figures 6 and 11). In particular, at high noise levels, this improvement is exclusive to DSINDy.

There are multiple areas of future research that may improve the DSINDy approach. For example, currently we set the value of the smoothing hyperparameter in IRW-SOCP to guarantee feasibility of Equation (22). In practice this value is often larger than necessary, and therefore, future work may involve exploring alternative ways to learn this hyperparameter. This may also provide insight into whether the Pareto curve or theoretical results should be used to set the value of the data-matching hyperparameter, as current results are inconclusive (Section 6.2.4). For the smoothing step, theoretical guarantees do not exist for IterPSDN (Algorithm 1). Therefore, a more thorough analysis and future modifications to this algorithm are warranted.

On top of the SINDy framework additional approaches have been developed to help alleviate the challenge of noise. These include approaches that use Bayesian inference [31, 32, 33] and ensemble-based/bootstrapping techniques [34, 35] to obtain estimates of coefficient uncertainty. In [13] a toolkit was proposed that provides extensions to the SINDy framework to help deal with high noise measurements. For example, the toolkit involves modifying the basis after comparing the obtained dynamics to the original measurements. Similar to SINDy, DSINDy could be integrated into these larger frameworks.

One of the biggest challenges with equation discovery is finding a suitable basis for the dynamics. The PSDN approach may potentially be modified to help with basis selection by observing how the inclusion of different basis elements impacts the results. Currently, the theory for PSDN does not hold if the library does not have full column rank, i.e., when there are redundant terms in the basis. This can be a challenging scenario to detect at high noise levels [25]. A topic of future work is to examine the performance of PSDN in this scenario and determine if identification of redundant basis elements is possible in the high-noise regime.
Figure 11: Coefficient and reconstruction error for Duffing oscillator system (PS2). Each column represents a different noise level. Showing the mean ± sem of 30 realizations at each noise level. The failure rate represents the fraction of simulations out of 30 that failed. For system parameters see Table 2. An example simulation result for this system is given in Figure 12.
Figure 12: Example results for the Duffing oscillator (PS2) system. Left column: Example set of measurements obtain for $t \in [0, 10], \sigma = 0.1, N = 1000$. Right column: Predicted system dynamics using example measurements over double the training time.

Acknowledgments

The authors would like to thank Professor David Bortz and Daniel Messenger for helpful discussions regarding this work. This work was supported by the Department of Energy, National Nuclear Security Administration, Predictive Science Academic Alliance Program (PSAAP) [Award Number DE-NA0003962].

Appendix A. Description of equation discovery methods that were compared with DSINDy

Here we present existing methods that are used for comparison purposes in the main manuscript. These include sparsity-promoting regularization techniques (Appendix A.1) and the Pareto corner criteria for finding the hyperparameter in regularized least squares problems (Appendix A.2). Note that the Pareto corner criteria is also used within the DSINDy algorithm. We then provide a description of $\ell_1$-SINDy and WSINDy (Appendices A.3 and A.4). Both equation discovery methods involve the use of sparsity-promoting regularization techniques discussed in Appendix A.1 and $\ell_1$-SINDy uses the Pareto corner criteria discussed in Appendix A.2.

Appendix A.1. Sparsity-promoting regularization

Here we discuss two sparsity-promoting regularization techniques: sequential-thresholding least squares (STLS) \[11\] and iteratively reweighted (IRW) Lasso \[36, 37, 25\]. Given a measurement matrix $A \in \mathbb{R}^{N \times p}$ and vector $b \in \mathbb{R}^N$, the goal of both methods is to reconstruct a sparse signal (or coefficient vector) $c \in \mathbb{R}^p$ such that $Ac \approx b$. In the paper we only considered overdetermined systems, i.e., $N \geq p$.

STLS is a sparsity-promoting method that iteratively sets coefficients to zero \[11\]. Briefly, STLS involves finding the least squares solution, $c^{LS} = A^T b$. Then, all coefficients with magnitudes less than a threshold, $\lambda_{STLS}$, are set equal to zero and the corresponding columns of $A$ are removed. These steps are repeated iteratively until the coefficient vector converges. For the complete algorithm see \[11\].
IRW-Lasso iteratively solves the Lasso $\ell_1$-regularization problem \cite{38}, i.e.,
\[
\text{minimize } \|Ac - b\|^2 + \lambda_{\text{Lasso}}\|We\|_1.
\] (A.1)
In standard Lasso as well as in the first iteration of IRW-Lasso, $W$ is the identity matrix. In subsequent iterations of IRW-Lasso, $W$ is a diagonal matrix such that
\[
W_{ii} = \frac{1}{|c_{\text{prior}}| + \epsilon}, \quad i = 1, 2, \ldots, p,
\] (A.2)
where $c_{\text{prior}}$ is the coefficient vector obtained in the previous iteration and $\epsilon$ is a small positive number to avoid dividing by zero. For more details on IRW-Lasso and its implementation see \cite{36,25}. In DSINDy we use a similar iterative process as IRW-Lasso to find a sparse vector $c$. However, the formulation differs from Equation (A.1) (see Section 3.2 for details).

Remark A1. Many additional strategies exist for finding a sparse vector $c$. These include Orthogonal Matching Pursuit (an $\ell_0$ minimization approach, \cite{39}) and Basis Pursuit Denoising (a version of Equation (A.1) where the $\|Ac - b\|$ loss is incorporated as a constraint, \cite{40}). However, these approaches are not considered in the present work.

Appendix A.2. Pareto corner finding algorithm

Let $A \in \mathbb{R}^{N \times p}$ and $b \in \mathbb{R}^N$, represent a measurement matrix and vector, respectively. Suppose we want to find a coefficient vector $c \in \mathbb{R}^p$, such that $Ac \approx b$, by solving
\[
\text{minimize } \|Ac - b\|^2 + \lambda\|c\|_s,
\] (A.3)
where $s$ could be 1 or 2 depending on the type of regularization desired. Here, we describe a method that uses the Pareto curve to find the value of the hyperparameter $\lambda$. The Pareto curve displays the trade-off between minimizing the solution residual, $\|Ac - b\|$, and the regularization residual $\|c\|_s$. \cite{41,42,25}.

To generate the Pareto curve, we solve Equation (A.3) at multiple values of the hyperparameter, i.e. set $\lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_i \in [0, \infty)$, to obtain $c(\lambda_i)$ for $i = 1, 2, \ldots, n$. We then plot the resulting solution residual verse the regularization residual, i.e., points contained in the set
\[
\{ (\|c(\lambda)\|_s, \|Ac(\lambda) - b\|) \mid \lambda \in [0, \infty) \}.
\] (A.4)
When plotted on a log scale, the resulting curve often has an L-shape characteristic. The value of $\lambda$ at the corner of this L-curve, i.e., $\lambda_c$, is thought to represent a balance between the regularization and solution error \cite{41}. Therefore, we choose $c(\lambda_c)$ as the final coefficient vector.

Various methods have been proposed for finding $\lambda_c$ efficiently \cite{43,44,45}. One algorithm iteratively performs a golden section search to narrow in on a point of the Pareto curve with large curvature \cite{44}. This is the algorithm that is used in the present work.

Appendix A.3. $\ell_1$-SINDy

The $\ell_1$-SINDy algorithm is a variation of SINDy \cite{11} that uses $\ell_1$-minimization to find the coefficient vector. Specifically, the $\ell_1$-SINDy algorithm incorporates the modifications to SINDy presented in \cite{25}. In $\ell_1$-SINDy we solve Equation (6) directly by approximating the state time derivatives and assuming that the coefficient vector is sparse. The recovery of the ODE system then reduces to a regularized linear regression problem. The original SINDy paper \cite{11} used STLS to enforce sparsity of the coefficient vector. Later, \cite{25} found that IRW-Lasso led to improved coefficient prediction. Therefore, in $\ell_1$-SINDy we also use IRW-Lasso (see Appendix A.1). Additionally, we will follow \cite{25} and estimate the sparsity-hyperparameter $\lambda_{\text{lasso}}$ using the corner point of the Pareto curve (see Appendix A.2). We refer to this algorithm as $\ell_1$-SINDy to emphasize this departure from the original SINDy framework.
The SINDy algorithm, and hence $\ell_1$-SINDy, requires that the derivative be estimated from the noisy measurements, $u \in \mathbb{R}^N$. Again, we follow the approach of [24] and use Tikhonov regularization to find the derivative. We solve

$$\min_{\dot{u}} \| T \dot{u} - u \|^2 + \lambda T_{ik} \| D \dot{u} \|^2,$$

where $T$ and $D$ are as defined in Section 2.2. The first term of Equation (A.5) requires that the integrated time derivative predict the measurements and the second term promotes smoothness of the derivative. When implementing this method, we used the Pareto corner finding algorithm to find the hyperparameter $\lambda_{Tik}$ (see Appendix A.2) [44].

One modification we use here in $\ell_1$-SINDy that was not used in [25] is that we perform a priori denoising of the state measurements. We use the smoothed measurements from IterPSDN as input to $\ell_1$-SINDy. We do this to more accurately compare $\ell_1$-SINDy with the IRW-SOCP step of DSINDy.

**Appendix A.4. WSINDy**

WSINDy uses the weak formulation to learn the governing equations [23]. In WSINDy, we construct a set of test functions that are compactly supported within the time domain of the training data, i.e., $\phi(t) \in C_c([0,t_{end}])$ for $t = 1, ..., L$. The inner product, i.e., $\langle u, v \rangle = \int_{0}^{t_{end}} u(t)v(t)dt$, of the dynamics for state $k$ with each test function is given as, i.e.,

$$\langle \phi_t, \dot{u}_k \rangle = \langle \phi_t, F_k(u_1, u_2, ..., u_m) \rangle,$$

and modified using integration by parts to obtain,

$$-\langle \dot{\phi}_t, u_k \rangle = \langle \phi_t, F_k(u_1, u_2, ..., u_m) \rangle.$$

Note that since the test functions are compactly supported on $[0,t_{end}]$, the boundary terms go to zero. A discrete approximation to the integral in Equation (A.7), based on the measurement acquisition times, is given as

$$-\sum_{i=1}^{N} \phi_t(t_i)u_k(t_i) \approx \sum_{i=1}^{N} \phi_t(t_i)\Theta \cdot c_k,$$

where $\Theta$ represents the $i$th row of the basis function library $\Theta$, defined in Equation (1). Note that Equation (A.8) is not exact as the state variables are approximated from noisy measurements and error is introduced in discretization.

Finally, we consider Equation (A.8) for all test functions to obtain a linear system,

$$b_k \approx H c_k,$$

where $b_k = -\sum_{i=1}^{N} \phi_t(t_i)u_k(t_i)$ and the $\ell$th row of $H$ is $H_{\ell} = \sum_{i=1}^{N} \phi_t(t_i)\Theta_i$. One key benefit of WSINDy is that it avoids the state derivative calculation through the use of integration by parts.

To find the sparse coefficient vector $c_k \in \mathbb{R}^p$ for $k = 1, 2, ..., m$, the most recent WSINDy code available uses a modified version of STLS, i.e., MSTLS. This version differs from STLS in that an upper bound on the coefficient magnitude is incorporated. To find the optimal hyperparameter $\lambda$, MSTLS is performed at a range of $\lambda$ values, and the chosen $\lambda$ minimizes the following loss function,

$$L(\lambda) = \frac{\| H(c(\lambda) - c^{LS}) \|}{\| Hc^{LS} \|} + \frac{\| c(\lambda) \|_0}{p},$$

where $c^{LS}$ is the least squares solution. The first term of this loss function encourages the coefficient vector to match the least squares solution, while the second term encourages sparsity (for additional details see [24]). Note that MSTLS was not used in the original WSINDy paper for discovering ODE systems [23].

The current implementation of WSINDy on GitHub also performs prior smoothing of the measurements. The method smooths the data by finding a weighted average of each point with its nearest neighbors. The weighting and size of the smoothing window is calculated automatically.
Appendix B. Additional theoretical results and proofs

In this section we provide supplemental theoretical results. We first present an error bound on the \( \ell_2 \)-error of discrete integration which follows almost immediately from the well known trapezoidal rule error bound (Appendix B.1). We then present existing matrix perturbation results that provide a bound on the projection operator error (Appendix B.2), and we then provide a bound for the pseudoinverse of \( \Psi^* \) (Appendix B.3). Next, we explicitly present the unbiased monomial library \( \hat{\Theta} \) and derive an estimator of the matrix \( (\Psi^*)^T \Psi^* \) (Appendix B.4). Finally, we show that the IRW-SOCP formulation given by Equation (22) is feasible for our chosen value of the derivative-smoothing hyperparameter (Appendix B.5).

Appendix B.1. Trapezoidal quadrature error

The proof of Lemma 1 requires a bound on the error introduced by discrete integration in Equation (10). This error bound, specific to the trapezoidal rule, is given in the following lemma.

**Lemma B1.** Let \( T \in \mathbb{R}^{N \times N} \) be the discrete integral operator that performs trapezoidal quadrature as defined in Equation (9). Let \( 0 = t_1, t_2, ..., t_N = t_{\text{end}} \) represent a set of \( N \) equispaced time points in the interval \( [0, t_{\text{end}}] \) and let \( u(t) \) be defined by the following integral equation

\[
u(t) = u(0) + \int_0^t \dot{u}(t') dt', \tag{B.1}\]

where \( \dot{u} = du/dt \) and assume that the constant

\[
M = \max_{t \in [0,t_{\text{end}}]} \left| \frac{d^3u}{dt^3} \right| \tag{B.2}
\]

exists and is finite.

Suppose we estimate \( u = [u(t_1), ..., u(t_{\text{end}})] \) as

\[
u = u(0) + T\dot{u} \tag{B.3}
\]

where \( u(0) \) and \( \dot{u} = [\dot{u}(t_1), ..., \dot{u}(t_N)] \) are known. Define the quadrature error vector \( e_q := \nu - u \). Then

\[
\|e_q\| \leq \frac{t_{\text{end}}^3 M}{12} \left( N - 1 \right)^{-3/2}. \tag{B.4}
\]

**Proof.** Using the error bound for the trapezoidal rule, we have that

\[
|e_{q,i}| \leq \frac{(i - 1) (\Delta t)^3 M}{12(i - 1)^2} = \frac{(i - 1) (\Delta t)^3 M}{12} = \frac{(i - 1) t_{\text{end}}^3 M}{12(N - 1)^3}. \tag{B.5}
\]

Using Equation (B.5), we next derive a bound on \( \|e_q\|^2 \) noting that \( e_{q,1} = 0, \)

\[
\|e_q\|^2 = \sum_{i=2}^{N} e_{q,i}^2 \leq \sum_{j=1}^{N-1} \left( \frac{j t_{\text{end}}^3 M}{12(N - 1)^3} \right)^2 \tag{B.6}
\]

\[
= \left( \frac{t_{\text{end}}^3 M}{12(N - 1)^3} \right)^2 \sum_{j=1}^{N-1} j^2 \tag{B.6}
\]

\[
= \left( \frac{t_{\text{end}}^3 M}{12(N - 1)^3} \right)^2 \frac{(N - 1)N(2N - 1)}{6} \tag{B.6}
\]

\[
= \left( \frac{t_{\text{end}}^3 M}{12} \right)^2 \frac{N(2N - 1)}{6(N - 1)^5} \tag{B.6}
\]

\[
\leq \left( \frac{t_{\text{end}}^3 M}{12} \right)^2 \frac{1}{(N - 1)^3}. \tag{B.6}
\]
The last inequality follows because for $N \geq 2$,
\[
\frac{N}{N-1} \leq 2 \quad \text{and} \quad \frac{2N-1}{N-1} \leq 3.
\]
Taking the square root of Equation (B.6) completes the proof. \hfill \Box

Appendix B.2. Existing results on projection operator error

In this section we present results from a study that examined how matrix perturbations impact projection operations [26]. These existing results are used in the proof of Lemma 3 and are applicable to any two matrices $\Phi^*$ and $\hat{\Phi}$, where $\hat{\Phi}$ is a perturbed version of $\Phi^*$. Note that we use these matrix names to place the results of [26] in the context of the equation discovery problem presented in Section 2. We will use the following definitions,
\[
\Delta \Phi := \hat{\Phi} - \Phi^*, \quad \Delta P := P_{\hat{\Phi}} - P_{\Phi^*}. \tag{B.8}
\]
where the projection operators $(P_{\hat{\Phi}}$ and $P_{\Phi^*})$ are calculated using Equation (13).

**Theorem B1** (modified version of Theorem 4.1 in [26]). Let the SVD of $\Phi^*$ be given as
\[
\Phi^* = [\hat{U}_{\Phi^*} \quad \hat{V}_{\Phi^*}] \begin{bmatrix} \Sigma_{\Phi^*} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{U}_{\Phi^*}^T \\ \hat{V}_{\Phi^*}^T \end{bmatrix}.
\]

Define the error matrices
\[
E_{11} := \hat{U}_{\Phi^*}^T \Delta \Phi \hat{V}_{\Phi^*}, \quad E_{21} := \hat{U}_{\Phi^*}^T \Delta \Phi \hat{V}_{\Phi^*},
\]
and scalar value
\[
\beta := \frac{\|\Phi^*\|\|\Phi^*_\dag\|}{1 - \|E_{11}\|\|\Phi^*_\dag\|}.
\]
If $\text{Rank}(\hat{\Phi}) = \text{Rank}(\Phi^*)$ and $\|\Phi^*_\dag\|\|E_{11}\| < 1$, then
\[
\|\Delta P\| = \|P_{\hat{\Phi}} - P_{\Phi^*}\| \leq \frac{\beta\|E_{21}\|\|\Phi^*\|}{(1 + (\beta\|E_{21}\|\|\Phi^*_\dag\|)^2)^{1/2}}. \tag{B.9}
\]

The statement of Theorem B1 differs from the statement of Theorem 4.1 in [26] in two ways. First, Theorem 4.1 in [26] requires that $\hat{\Phi}$ be an acute perturbation of $\Phi^*$. Here, we instead require that $\text{Rank}(\hat{\Phi}) = \text{Rank}(\Phi^*)$ and $\|\Phi^*_\dag\|\|E_{11}\| < 1$. Together, these requirements imply that $\hat{\Phi}$ is an acute perturbation of $\Phi^*$ (see Theorem 2.2 and Theorem 2.5 in [26]). Second, we replace the variables $\kappa$ and $\gamma$, referenced in [26] with $\beta = \kappa/\gamma$. For additional details and definitions see [26].

Appendix B.3. Bounding the pseudoinverse of $\Psi^*$

In this section we show that the pseudoinverse of $\Psi^*$ is bounded for all $N$ (see Corollary B1). This result relies on the following lemma, which shows $(\Psi^*)^T \Psi$ converges pointwise to a finite matrix.

**Lemma B2.** Suppose $u_k^*(t)$ for $k = 1, 2, ..., m$ is twice continuously differentiable for $t \in [0, t_{end}]$. Then, in the limit as $N \to \infty$, $(\Psi^*)^T \Psi^*$ converges pointwise to $B \in \mathbb{R}^{p \times p}$, defined elementwise as
\[
B_{ij} = \frac{1}{t_{end}} \int_0^{t_{end}} \xi_i(t)\xi_j(t)dt \quad \text{where} \quad \xi_j(t) = \begin{cases} 1 & \text{for } j = 1 \\ \int_0^t \theta_j^{t_{\eta-1}}(t')dt' & \text{for } j > 1. \end{cases}
\]
Here, $\theta_j^t(t) = \theta_j(u_1^*(t), u_2^*(t), ..., u_m^*(t))$ represents the $j$th basis function.
Proof. First consider $T\Theta^*$ and note that for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, p$,

$$(T\Theta^*)_{ij} = \sum_{k=1}^{i} T_{ik} \Theta_{kj} = \int_{0}^{t_i} \theta_j^*(t) dt + e_{i,j+1} = \xi_{j+1}(t_i) + e_{i,j+1}, \quad (B.10)$$

where $e_{i,j+1}$ is error due to the trapezoidal quadrature step, i.e.,

$$|e_{i,j+1}| \leq \frac{(i-1)t_{\text{end}}}{12(N-1)^3} \max_{t \in [0,t_{\text{end}}]} \left| \frac{d^2 \theta_j^*}{dt^2} \right| = i\mathcal{O}(N^{-3}). \quad (B.11)$$

We additionally define $e_{i,1} = 0$ for all $i = 1, 2, \ldots, N$.

Define $B^{(N)} := (\Psi^*)^T \Psi^*$, i.e.,

$$B^{(N)} = \frac{1}{N} \left[ \frac{N}{(T\Theta^*)^T \mathbf{1}} (T\Theta^*)^T \Theta^* \right]. \quad (B.12)$$

Therefore, $B^{(N)}$ is given elementwise as,

$$B^{(N)}_{ij} = \frac{1}{N} \sum_{i=2}^{N} \left( \xi_i(t_i) + e_{i,\ell} \right) \left( \xi_j(t_i) + e_{i,j} \right) = \frac{1}{N} \sum_{i=2}^{N} \xi_i(t_i) \xi_j(t_i) + \mathcal{O}(N^{-2})$$

$$= \frac{1}{t_{\text{end}}} \int_{0}^{t_{\text{end}}} \xi_i(t) \xi_j(t) dt + \delta_{ij} + \mathcal{O}(N^{-2}) = B_{ij} + \mathcal{O}(N^{-1}).$$

The summation following the second equality gives a right Riemann sum, implying, $\delta_{ij} \leq \mathcal{O}(N^{-1})$. This result gives us the lemma statement.

Corollary B.1. Suppose the assumptions of Lemma B2 hold where $(\Psi^*)^T \Psi^*$ converges to $B$ without changing rank, i.e., there exists $N_0$ such that for $N > N_0$ the rank of $(\Psi^*)^T \Psi^*$ and $B$ are equal. Then, the value of $\|\Psi^*\|_\Psi$ is bounded for all $N$.

Proof. The singular values and pseudoinverse matrix norm are related as follows:

$$\|\Psi^*\|_\Psi = \frac{1}{\sigma_{\text{min}}(\Psi^*)}, \quad (B.13)$$

where $\sigma_{\text{min}}(\Psi^*)$ is the smallest nonzero singular value of $\Psi^*$. Since $(\Psi^*)^T \Psi^*$ converges to $B$ without changing rank, $\sigma_{\text{min}}(\Psi^*)$ must approach a constant nonzero value. Therefore, there exists a $C$ such that $\|\Psi^*\|_\Psi < C$ for all $N$. □

Appendix B.4. Deriving estimators of $\Theta^*$ and $G$

In this section we set $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{u}^* \in \mathbb{R}^m$ to be vectors that contain the $m$ noisy and true state variables at an arbitrary time $\tau$, e.g., $\mathbf{u} = [u_1(\tau), u_2(\tau), \ldots, u_m(\tau)]$. This is in contrast to other sections of the paper where $\mathbf{u}$ and $\mathbf{u}^*$ represents the $N$ measurements and true values of an arbitrary state variable. We additionally define $\mathbf{e} \in \mathbb{R}^m$ to be the noise of the $m$ state variables at the arbitrary time $\tau$, i.e., $\mathbf{u} = \mathbf{u}^* + \mathbf{e}$.

To present the estimators of $\Theta^*$ and $G$, we define a partial ordering of the basis multi-indices. Specifically, $\alpha^{(k)} \leq \alpha^{(j)}$ if $\alpha^{(k)}_i \leq \alpha^{(j)}_i$ for all $i \in \{1, \ldots, m\}$. If additionally, $\alpha^{(k)} \neq \alpha^{(j)}$ then a strict inequality holds, i.e., $\alpha^{(k)} < \alpha^{(j)}$. We will use the following shorthand notation when considering the difference between two multi-indices,

$$\alpha_{j \setminus k} := \alpha^{(j)} - \alpha^{(k)}. \quad (B.14)$$

To keep the results general, we present the estimators of $\Theta^*$ and $G$ as functions that depend on the moments of the noise distribution and use the following shorthand notation,

$$\mathcal{M}_j = \mathbb{E}[\mathbf{e}^{\alpha^{(j)}}, \quad \mathcal{M}_{j \setminus k} = \mathbb{E}[\mathbf{e}^{\alpha^{(j)}} - \alpha^{(k)}]. \quad (B.15)$$
In the specific scenario where the measurement noise is a zero-mean random Gaussian variable with variance $\sigma^2$, we have that for a general multi-index $\alpha$,

$$
\mathbb{E}[\epsilon^{\alpha}] = \prod_{k=2, \alpha_k \text{ even}}^{m} \sigma^{\alpha_k} \alpha_k!!,
$$

(B.16)

where, for odd $n$, $n!! = \prod_{k=1}^{\frac{n-1}{2}} (2k - 1)$.

For notational simplicity we also use the following definition,

$$
C_{jk} := \binom{\alpha(j)}{\alpha(k)}.
$$

(B.17)

Appendix B.4.1. Unbiased estimator of the monomial library $\Theta^*$

Recall that the library $\Theta$ contains monomial basis functions evaluated at noisy state variable measurements, i.e., $\Theta_{ij} = \theta_j(u_1(t_i), u_2(t_i), ..., u_m(t_i))$. If $j$ is such that $|\alpha(j)| > 1$, then the corresponding monomial basis function $\theta_j(u_1, u_2, ..., u_m)$ is a biased estimator of $\theta_j^*(u_1^*, u_2^*, ..., u_m^*)$. To prove that the PSDN approximation approaches the true state variable values as we increase the density of measurements, i.e., Theorem 1, it is more convenient to work with the unbiased library, which we denote as $\hat{\Theta}$.

In this section, we present a claim that explicitly gives $\hat{\Theta}$ and shows that it is an unbiased estimator. Although, we use monomial basis functions, a similar approach can be applied for any set of polynomial basis functions, e.g., Legendre polynomials, to obtain an unbiased library.

Claim B1. Let $\alpha(j)$ for $j = 1, 2, ..., p$ represent the set of multi-indices used to define the monomial library of maximum total degree $d$. For $j = 1, 2, 3, ..., p$, iteratively define the following basis functions

$$
\hat{\theta}_j(u) = u^{\alpha(j)} - \sum_{\alpha(k) \subseteq \alpha(j)} C_{jk} \hat{\theta}_k(u)M_{j\setminus k}.
$$

(B.18)

Then $\hat{\theta}_j(u)$ is an unbiased estimate of $\theta_j(u^*)$ for $j = 1, 2, ..., p$.

Proof. First we find the expected value of the $j$th monomial basis function evaluated using the noisy measurements $u$, given as $\theta_j(u) = u^{\alpha(j)}$.

$$
\mathbb{E}\left[u^{\alpha(j)}\right] = \mathbb{E}\left[(u^* + \epsilon)^{\alpha(j)}\right] = \sum_{\alpha(k) \subseteq \alpha(j)} C_{jk}(u^*)^{\alpha(k)} M_{j\setminus k} = \theta_j(u^*) + \sum_{\alpha(k) \subseteq \alpha(j)} C_{jk} \theta_k(u^*)M_{j\set\setminus k},
$$

(B.19)

where here we use the multi-binomial theorem.

We can then show by induction that $\hat{\theta}_j(u)$ is an unbiased estimator of $\theta_j(u^*)$. First for $j = 1$, we have that $\alpha(1) = [0, 0, ..., 0]^T$ and $\hat{\theta}_1(u)$ is unbiased as

$$
\hat{\theta}_1(u) = 1 = \theta_1(u^*).
$$

(B.20)

Next, pick $j > 1$ and suppose that $\hat{\theta}_k(u)$ is unbiased for all $k < j$. Note that this immediately implies $\hat{\theta}_k(u)$ is unbiased for all $k$ such that $\alpha(k) < \alpha(j)$. By combining Equations (B.18) and (B.19), we have that

$$
\mathbb{E}\left[\hat{\theta}_j(u)\right] = \mathbb{E}\left[u^{\alpha(j)}\right] - \sum_{\alpha(k) \subseteq \alpha(j)} C_{jk} \mathbb{E}\left[\hat{\theta}_k(u)\right] M_{j\set\setminus k}
$$

$$
= \theta_j(u^*) - \sum_{\alpha(k) \subseteq \alpha(j)} C_{jk} \left(\mathbb{E}\left[\hat{\theta}_k(u)\right] \theta_k(u^*)\right) M_{j\set\setminus k} = \theta_j(u^*).
$$

(B.21)

Therefore, $\hat{\theta}_j(u)$ is unbiased.

When we refer to the centered monomial library $\hat{\Theta}$, we are referring to the library that contains the $p$ basis functions $\hat{\theta}_1, \hat{\theta}_2, ..., \hat{\theta}_p$ evaluated at the noisy measurements.

31
Appendix B.4.2. A consistent estimator of $G$

In this section, we present a consistent estimator for $H := G/N = \hat{G}^T \Theta^*/N$. We will assume $\hat{\Theta}$ is independent of the noise in the measurement library $\Theta$. In practice this is not the case, as $\hat{\Theta}$ is approximated from $\Theta$. However, this work provides a useful starting point for considering estimators for this class of Gramian matrices. Note that a similar approach can be used to find a consistent estimator of $(\Theta^*)^T \Theta^*/N$ that does not make this assumption of independence. We will use the following definitions, for $i = 1, 2, ..., N$,

$$u_i^* := \begin{bmatrix} u_1^*(t_i) \\ \vdots \\ u_m^*(t_i) \end{bmatrix}, \quad \tilde{u}_i := \begin{bmatrix} \tilde{u}_1(t_i) \\ \vdots \\ \tilde{u}_m(t_i) \end{bmatrix}, \quad \epsilon_i := \begin{bmatrix} \epsilon_1(t_i) \\ \vdots \\ \epsilon_m(t_i) \end{bmatrix}, \quad (B.22)$$

and $u_i := u_i^* + \epsilon_i$.

The consistent estimator of $G/N$, denoted as $\hat{H}$, is given in the following claim.

**Claim B2.** Let $\alpha^{(j)}$ for $j = 1, 2, ..., p$ represent the set of multi-indices used to define the monomial library of maximum total degree $d$ and suppose that $u_1^*(t), u_2^*(t), ..., u_m^*(t)$ are bounded for all $t \in [0, t_{end}]$. Iteratively define $H$ elementwise such that for $j, k = 1, 2, ..., p$,

$$\hat{H}_{jk} = \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_i^{\alpha^{(j)}} u_i^{\alpha^{(k)}} - \sum_{\ell \in \mathbb{Z}} C_{k\ell} \hat{H}_{j\ell} \mathcal{M}_{k\ell}. \quad (B.23)$$

Then $\hat{H}$ is a consistent estimator of $G/N$.

To prove this claim we first present and prove a helpful lemma.

**Lemma B3.** Suppose the assumptions of Claim B2 hold and pick $\ell, k \in \{1, ..., p\}$ such that $\alpha^{(\ell)} < \alpha^{(k)}$. Then for all $i' \in \{1, 2, ..., N\}$ and $j \in \{1, 2, ..., p\}$,

$$E[\hat{H}_{j\ell} \epsilon_{i'}^{\alpha^{(k)}]} = H_{j\ell} \mathcal{M}_{k\ell} + O\left(\frac{1}{N}\right). \quad (B.24)$$

**Proof.** We prove the statement using induction on $\ell$. First note that for $\ell = 1$ and $k \in \{2, 3, ..., p\}$ we have that

$$E \left[ \hat{H}_{j\ell} \epsilon_{i'}^{\alpha^{(k)}} \right] = \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_i^{\alpha^{(j)}} \mathcal{M}_{k\ell} = H_{j\ell} \mathcal{M}_{k\ell}. \quad (B.25)$$

Therefore, at $\ell = 1$, Equation (B.24) is satisfied.

Then, for $\ell > 1$ we assume Equation (B.24) holds for $\ell' < \ell$ and note that

$$\frac{1}{N} \sum_{i=1}^{N} \tilde{u}_i^{\alpha^{(j)}} u_i^{\alpha^{(k)}} = \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_i^{\alpha^{(j)}} \sum_{\ell' \in \mathbb{Z}} C_{\ell'\ell}(u_i^{(\ell')})^{\alpha^{(\ell')}} \epsilon_i^{\alpha^{(\ell')}} = \sum_{\ell' \in \mathbb{Z}} \frac{C_{\ell'\ell}}{N} \sum_{i=1}^{N} \tilde{u}_i^{\alpha^{(j)}} (u_i^{(\ell')})^{\alpha^{(\ell')}} \epsilon_i^{\alpha^{(\ell')}}. \quad (B.26)$$

32
Using this, we then have that

\[
\mathbb{E}\left[ \hat{H}_{j\ell} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] = \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} \leq \alpha^{(\ell)}} \frac{C_{\ell'}}{N} \sum_{i=1}^{N} \tilde{u}_{i}^{(\ell')} (u_{i}^{*})^{\alpha^{(\ell')}} \mathbb{E}\left[ \epsilon_{i}^{\alpha_{\ell'}} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] - \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} < \alpha^{(\ell)}} C_{\ell'} \mathbb{E}\left[ \hat{H}_{j\ell'} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] \mathcal{M}_{\ell' \ell'}
\]

\[
= \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} \leq \alpha^{(\ell)}} \frac{C_{\ell'}}{N} \sum_{i=1}^{N} \tilde{u}_{i}^{(\ell')} (u_{i}^{*})^{\alpha^{(\ell')}} \mathbb{E}\left[ \epsilon_{i}^{\alpha_{\ell'}} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] - \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} < \alpha^{(\ell)}} C_{\ell'} \mathbb{E}\left[ \hat{H}_{j\ell'} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] \mathcal{M}_{\ell' \ell'}
\]

\[
+ \frac{1}{N} \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} < \alpha^{(\ell)}} C_{\ell'} \mathbb{E}\left[ \hat{H}_{j\ell'} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] \left( \mathcal{M}_{k' \ell'} - \mathcal{M}_{\ell' \ell'} \right)
\]

\[
= H_{j\ell} \mathcal{M}_{k' \ell'} + \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} < \alpha^{(\ell)}} C_{\ell'} \left( H_{j\ell'} \mathcal{M}_{k' \ell'} - \mathbb{E}\left[ \hat{H}_{j\ell'} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] \right) \mathcal{M}_{\ell' \ell'}
\]

\[
+ \frac{1}{N} \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} < \alpha^{(\ell)}} C_{\ell'} \mathbb{E}\left[ \hat{H}_{j\ell'} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] \left( \mathcal{M}_{k' \ell'} - \mathcal{M}_{\ell' \ell'} \right).
\]

Here, for the second equality we are using the fact that the measurement noise is independent across the N samples. For the third equality, we are pulling out the term from the first summation that corresponds to G_{j\ell} \mathcal{M}_{k' \ell'}.

As desired, since Equation (B.24) holds for \ell' < \ell,

\[
\mathbb{E}\left[ \hat{H}_{j\ell} \epsilon_{\ell'}^{\alpha_{k\ell}} \right] = H_{j\ell} \mathcal{M}_{k' \ell'} + \mathcal{O}\left( \frac{1}{N} \right).
\]

We use this result to prove Claim B.3.

**Proof of Claim B.3** To prove the result we need to show that \hat{H} is unbiased and that the variance goes to zero as we increase the sample size.

First we show that \hat{H} is unbiased. Taking the expectation of Equation (B.26), we immediately have that

\[
\mathbb{E}\left[ \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_{i}^{(\ell')} u_{i}^{(k)} \right] = H_{j\ell k} + \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} < \alpha^{(k)}} C_{\ell'} H_{j\ell} \mathcal{M}_{k' \ell}.
\]

Using the same inductive reasoning as that given in the proof to Claim B.1, we then have that \hat{H} is an unbiased estimator of \hat{H}.

Next, we show that the variance of each element of \hat{H} goes to zero in the limit as N \to \infty. First we rewrite \hat{H}_{j\ell k} using the definition of \hat{H} and calculation shown in Equation (B.29),

\[
\hat{H}_{j\ell k} = H_{j\ell k} + \sum_{\ell' \text{ s.t. } \alpha^{(\ell')} < \alpha^{(k)}} C_{\ell'} \left( \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_{i}^{(\ell')} (u_{i}^{*})^{\alpha^{(\ell')}} \epsilon_{i}^{\alpha_{k\ell} \ell'} - \hat{H}_{j\ell} \mathcal{M}_{k' \ell} \right).
\]

Therefore, we can show \sqrt{\hat{H}_{j\ell k}} goes to zero by showing that the variance of the second term on the right hand side of Equation (B.29) goes to zero. We again use a proof by induction.
For \( k = 1 \) we have that \( \hat{H}_{jk} = H_{jk} \) and, therefore \( \mathbb{V}[\hat{H}] = 0 \). For \( k > 0 \), we consider the following variance,

\[
\mathbb{V} \left[ \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_{i}^{(j)} (u_{i}^{*})^{(j)} e_{i}^{\alpha_{k}\ell} - \hat{H}_{j\ell} M_{k\ell} \right] = \frac{1}{N^2} \sum_{i,j'=1}^{N} (\tilde{u}_{i}^{*} \tilde{u}_{j'})^{(j)} (u_{i}^{*} u_{j'}^{*})^{(j')} \mathcal{M}_{k\ell}^{2} + \mathbb{E} \left[ \hat{H}_{j\ell}^{2} \right] \mathcal{M}_{k\ell}^{2} \\
- \frac{2}{N} \sum_{i=1}^{N} \tilde{u}_{i}^{(j)} (u_{i}^{*})^{(j)} \mathbb{E} \left[ e_{i}^{\alpha_{K\ell} \ell} \hat{H}_{j\ell} \right] \mathcal{M}_{k\ell}^{2} \\
+ \frac{1}{N^2} \sum_{i=1}^{N} (\tilde{u}_{i}^{*})^{2} (u_{i}^{*})^{2} \mathbb{E} [e_{i}^{2\alpha_{K\ell} \ell} - \mathcal{M}_{k\ell}^{2}] \\
= \mathcal{H}_{j\ell}^{2} \mathcal{M}_{k\ell}^{2} + \mathbb{E} \left[ \hat{H}_{j\ell}^{2} \right] \mathcal{M}_{k\ell}^{2} - 2 \mathcal{H}_{j\ell}^{2} \mathcal{M}_{k\ell}^{2} + \mathcal{O} \left( \frac{1}{N} \right).
\]

In this derivation we use the result from Lemma B.3 to replace \( \mathbb{E} \left[ e_{i}^{\alpha_{K\ell} \ell} \hat{H}_{j\ell} \right] \) with \( H_{j\ell} M_{k\ell} \). By induction, we have that for \( \ell < k \), \( \mathbb{V}[\hat{H}_{j\ell}] \) goes to zero and thus,

\[
\lim_{N \to \infty} \mathbb{E}[\hat{H}_{j\ell}^{2}] = \mathcal{H}_{j\ell}^{2}. \tag{B.30}
\]

Therefore, in the limit as \( N \to \infty \), \( \mathbb{V}[\hat{H}_{jk}] \) goes to zero.

\section*{Appendix B.5. Value of smoothing hyperparameter leads to feasible SOCP}

Using the value of \( C \) in Equation (25) guarantees feasibility of the SOCP problem, i.e., Equation (22), at small values of \( \gamma \). To see this, we will show that the constraints in Equation (22) are satisfied when \( \tilde{u} = \tilde{u}^{*} \) as defined by Equation (24). We immediately have that \( \|D \tilde{u}^{*}\| = C \), implying the first constraint in Equation (22) is satisfied. For the second constraint, we have that,

\[
\left\| \begin{bmatrix} 1 & T \end{bmatrix} \begin{bmatrix} \tilde{u}^{0} \\ \tilde{u} \end{bmatrix} - P_{\tilde{u}} \tilde{u} \right\| = \|u_{0} 1 + T \tilde{\Theta} \left( \tilde{\Phi}^{T} \tilde{\Phi} \right)^{-1} \tilde{\Phi}^{T} \tilde{u} - P_{\tilde{u}} \tilde{u} \| \\
= \|u_{0} 1 + \tilde{\Phi} \left( \tilde{\Phi}^{T} \tilde{\Phi} \right)^{-1} \tilde{\Phi}^{T} \tilde{u} - \left( \left( \tilde{\Phi}^{T} \tilde{\Phi} \right)^{-1} \tilde{\Phi}^{T} \right)_{1;:} \tilde{u} - P_{\tilde{u}} \tilde{u} \| \\
= 0,
\]

where the final equality holds by setting \( u_{0} = \left( \left( \tilde{\Phi}^{T} \tilde{\Phi} \right)^{-1} \tilde{\Phi}^{T} \right)_{1;:} \tilde{u} \). Thus, the problem is feasible for all \( \gamma \geq 0 \). Recall that the ‘2 :, ;’ subscript implies we are removing the top row of the matrix. Similarly, the ‘1, ;’ subscript implies we are considering only the top row of the matrix.

\section*{Appendix C. Additional results}

\subsection*{Appendix C.1. Additional results for the Lorenz 96 model}

Here, we provide additional results for the Lorenz 96 model. Figure C.13 shows results of the PSDN and IterPSDN compared to theoretical estimations (see Section 6.1 for additional details). Figure C.14 shows the error of the derivation estimation from the IRW-SOCP step of DSINDy.

\subsection*{Appendix C.2. Additional results for Duffing oscillator (PS1)}

Here, we provide results on coefficient error and state reconstruction error for the Duffing oscillator system when using Parameter Set 1 (see Figure C.15).

34
Figure C.13: Comparing denoising error with theoretical predictions for the Lorenz 96 model. Results show mean ± std for 50 sample replications at noise level $\sigma^2 = 1$ when the integrated library $\Phi^*$ is known (left), when using PSDN (middle), and when using IterPSDN, Algorithm 1 (right). Although we only show the results for the first state of the Lorenz 96 model, nearly identical results were observed for the other 5 states. For system parameters see Table 2.

Figure C.14: Derivative estimation errors for the 6 states of the Lorenz 96 system. Showing the mean ± sem of 30 realizations with $N = 2000$ at each noise level. For system parameters see Table 2.
Appendix C.3. Examining hyperparameter $\gamma$ selection method in DSINDy

Here, we present results showing that the two approaches for selecting the data-matching hyperparameter $\gamma$ in Equation (22) lead to roughly equivalent results (Figures C.16 and C.17). Note that for conciseness in these figures we show the relative error of the coefficients and system state reconstructions averaged across the states of a given system. For details on these systems see Section 5.

References

[1] A. Ghadami, B. I. Epureanu, Data-driven prediction in dynamical systems: recent developments, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 380 (2229) (aug 2022).
[2] P. Neumann, L. Cao, D. Russo, V. S. Vassiladis, A. A. Lapkin, A new formulation for symbolic regression to identify physico-chemical laws from experimental data, Chemical Engineering Journal 387 (2020) 123412.
[3] N. M. Mangan, S. L. Brunton, J. L. Proctor, J. N. Kutz, Inferring biological networks by sparse identification of nonlinear dynamics, IEEE Transactions on Molecular, Biological, and Multi-Scale Communications 2 (1) (2016) 52–63.
[4] D. Dam, M. Brøns, J. Juul Rasmussen, V. Naulin, J. S. Hesthaven, Sparse identification of a predator-prey system from simulation data of a convection model, Physics of Plasmas 24 (2) (2017) 022310.
[5] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[6] J. Bongard, H. Lipson, Automated reverse engineering of nonlinear dynamical systems, Proceedings of the National Academy of Sciences 104 (24) (2007) 9945–9948.
[7] M. Quade, M. Abel, K. Shafi, R. K. Niven, B. R. Noack, Prediction of dynamical systems by symbolic regression, Physical Review E 94 (1) (2016) 12214.
[8] M. Raisi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, Journal of Machine Learning Research 19 (2018) 1–24.
[9] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (5923) (2009) 81–85.
[10] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, C. Grebogi, Predicting catastrophes in nonlinear dynamical systems by compressive sensing, Physical Review Letters 106 (15) (2011) 154101.
[11] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences 113 (15) (2016) 3932–3937.
[12] A. Cortiella, K. C. Park, A. Doostan, A priori denoising strategies for sparse identification of nonlinear dynamical systems: A comparative study, Journal of Computing and Information Science in Engineering 23 (1) (2022) 1–34.
[13] C. B. Delahunt, J. N. Kutz, A Toolkit for Data-Driven Discovery of Governing Equations in High-Noise Regimes, IEEE Access 10 (2022) 31210–31234.
Figure C.16: Coefficient and reconstruction error for example systems. Showing the mean ± sem of 30 realizations with $N = 1000$ at each noise level ($N = 2000$ for Lorenz 96 model). The failure rate represents the fraction of simulations out of 30 that failed. For system parameters see Table 2.
Figure C.17: Coefficient and reconstruction error for Duffing oscillator system (PS2). Each column represents a different noise level. Showing the mean ± sem of 30 realizations at each noise level. The failure rate represents the fraction of simulations out of 30 that failed. For system parameters see Table 2.

[14] H. Schaeffer, S. G. McCalla, Sparse model selection via integral terms, Physical Review E 96 (2) (2017) 023302, 7.
[15] Z. Chen, Y. Liu, H. Sun, Physics-informed learning of governing equations from scarce data, Nature Communications 12 (1) (2021) 6136.
[16] G. Tran, R. Ward, Exact recovery of chaotic systems from highly corrupted data, Multiscale Model. Simul. 15 (3) (2017) 1108–1129.
[17] F. Sun, Y. Liu, H. Sun, Physics-informed spline learning for nonlinear dynamics discovery, Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (2021) 2054–2061.
[18] J. M. Hokanson, G. Iaccarino, A. Doostan, Simultaneous Identification and Denoising of Dynamical Systems, SISC (2023, in press).
[19] K. Kaheman, S. L. Brunton, J. N. Kutz, Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data, Machine Learning: Science and Technology 3 (1) (2022) 015031.
[20] Z. Wang, X. Huan, K. Garikipati, Variational system identification of the partial differential equations governing the physics of pattern-formation: Inference under varying fidelity and noise, Computer Methods in Applied Mechanics and Engineering 356 (2019) 44–74.
[21] D. R. Gurevich, P. A. K. Reinbold, R. O. Grigoriev, Robust and optimal sparse regression for nonlinear PDE models, Chaos: An Interdisciplinary Journal of Nonlinear Science 29 (10) (2019) 103113.
[22] P. A. Reinbold, D. R. Gurevich, R. O. Grigoriev, Using noisy or incomplete data to discover models of spatiotemporal dynamics, Physical Review E 101 (1) (2020).
[23] D. A. Messenger, D. M. Bortz, Weak SINDy: Galerkin-Based Data-Driven Model Selection, Multiscale Modeling & Simulation 19 (3) (2021) 1474–1497.
[24] D. A. Messenger, D. M. Bortz, Weak SINDy for partial differential equations, Journal of Computational Physics 443 (2021) 110525.
A. Cortiella, K. C. Park, A. Doostan, Sparse identification of nonlinear dynamical systems via reweighted $\ell_1$-regularized least squares, Computer Methods in Applied Mechanics and Engineering 376 (2021) 113620.

G. W. Stewart, On the Perturbation of Pseudo-Inverses, Projections and Linear Least Squares Problems, SIAM Review 19 (4) (1977) 634–662.

E. Schulz, M. Speekenbrink, A. Krause, A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions, Journal of Mathematical Psychology 85 (2018) 1–16.

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

M. Anderson, J. Dahl, L. Vandenberghe, CVXOPT: Python software for convex optimization, version 1.1 (2015).

Mosek ApS, MOSEK Optimizer API (August) (2019).

R. K. Niven, A. Mohammad-Djafari, L. Cordier, M. Abel, M. Quade, Bayesian Identification of Dynamical Systems, Proceedings 33 (1) (2020) 33.

G. W. Stewart, On the Perturbation of Pseudo-Inverses, Projections and Linear Least Squares Problems, SIAM Review 19 (4) (1977) 634–662.

E. Schulz, M. Speekenbrink, A. Krause, A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions, Journal of Mathematical Psychology 85 (2018) 1–16.

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

M. Anderson, J. Dahl, L. Vandenberghe, CVXOPT: Python software for convex optimization, version 1.1 (2015).

Mosek ApS, MOSEK Optimizer API (August) (2019).

R. K. Niven, A. Mohammad-Djafari, L. Cordier, M. Abel, M. Quade, Bayesian Identification of Dynamical Systems, Proceedings 33 (1) (2020) 33.