

Symbolic Regression on Sparse and Noisy Data with Gaussian Processes

Junette Hsin¹, Shubhankar Agarwal², Adam Thorpe¹, Luis Sentis¹, and David Fridovich-Keil¹

Abstract—In this paper, we address the challenge of deriving dynamical models from sparse and noisy data. High-quality data is crucial for symbolic regression algorithms; limited and noisy data can present modeling challenges. To overcome this, we combine Gaussian process regression with a sparse identification of nonlinear dynamics (SINDy) method to denoise the data and identify nonlinear dynamical equations. Our simple approach offers improved robustness with sparse, noisy data compared to SINDy alone. We demonstrate its effectiveness on a Lotka-Volterra model, a unicycle dynamic model in simulation, and hardware data from an NVIDIA JetRacer system. We show superior performance over baselines including 20.78% improvement over SINDy and 61.92% improvement over SSR in predicting future trajectories from discovered dynamics.

I. INTRODUCTION

Accurate dynamic models are crucial for effective robot design and operation. In many cases, it is desirable to obtain analytic expressions over black box models, as analytic models extrapolate well beyond the training dataset and are more suitable for system analysis. One approach that has received significant attention is the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm [1]. It employs symbolic regression—a least-squares-based method—to learn the system dynamics purely from data using a predefined set of candidate functions. SINDy is simple in its approach but suffers from limitations in practice. In particular, the accuracy of the learned solution relies heavily on the selection of proper candidate function terms, and measurement noise in the data can significantly degrade the performance of SINDy for even simple systems [2]. Additionally, SINDy requires derivative data, which is often obtained through finite differencing or other approximation methods which can add additional error [3]. The sparsity of the data also impacts SINDy’s performance; while it can identify models with limited data [4], low frequency data may reduce model accuracy. *In this work, we devise a method that combines Gaussian process regression in conjunction with SINDy to learn system dynamics using sparse and noisy data.*

SINDy’s key advantage lies in discovering understandable models that balance accuracy and simplicity. While other data-driven methods have had success [5], limited data often limit their effectiveness, and the models they discover lack insight into the system’s structure [6]. In contrast, SINDy has been widely applied across various disciplines to understand the underlying structure of physical phenomena [7]–[9]. In the field of robotics, it has been used to learn the

dynamics of actuated systems for control purposes such as modeling jet engines for feedback linearization and sliding mode control [10]. SINDy’s appeal lies in its simple and highly adaptable sparse linear regression approach, requiring less data compared to methods like neural networks [6].

However, noise in the data remains a problem. A growing body of research based on SINDy seeks to mitigate the impact of noise on identifying the correct system dynamics. Reactive SINDy [11] uses vector-valued functions with SINDy to uncover underlying biological cell structures from noisy data, but can fail to converge to the correct reaction network with increasing levels of noise. PiDL-SINDy [12] utilizes a physics-informed neural network with sparse regression to learn the system dynamics, and DSINDy [3] and a modified SINDy using automatic differentiation (AD) [13] simultaneously de-noise the data and identify the governing equations. However, PiDL-SINDy [12] and AD-SINDy [13] run into computational bottlenecks and challenges with the structure of their optimization problems. Derivative-based approaches show promise, but DSINDy makes assumptions on the structure of its function library that may not be true in practice. ESINDy [6] proposes a statistical framework to compute the probabilities of candidate functions from an ensemble of models identified from noisy data, but its approach relies on Sequentially Thresholded Least Squares (STLS) regression. STLS can be effective for small levels of noise, but it deteriorates with larger noise levels [2].

Advancements in non-parametric based approaches also tackle the problem of learning governing equations from noisy data. Neural networks have been used to parameterize the state derivatives through a blackbox differential equation solver [14], and Gaussian processes have been used to infer parameters of linear equations from scarce and noisy observations [15] and generate vector fields of nonlinear differential equations [16]. Gaussian process regression is particularly effective as an interpolation tool and at reducing the noise in measurement data [5].

The novelty of our method lies in how we approximate the state derivatives by constructing the Gaussian process kernel using smoothed state measurements. This approach addresses issues caused by measurement noise and low temporal resolution for model learning. We learn the relationship between the state and time derivatives using Gaussian processes, and then determine analytic expressions for the dynamics with SINDy. We benchmark our method on both simulated and experimental hardware data, comparing it with SINDy, neural network models, and other baselines. The results show that our approach is notably more robust in identifying models from sparse and noisy data.

¹Departments of Aerospace Engineering & Engineering Mechanics and
²Electrical and Computer Engineering, University of Texas at Austin,
{jhsin, somi.agarwal, adam.thorpe, lsentis,
dfk}@utexas.edu

II. PROBLEM FORMULATION

Consider a system characterized by unknown dynamics

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad (1)$$

where $t \in \mathbb{R}$, $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the state of the system at time t , and $\mathbf{u}(t) \in \mathbb{R}^m$ the control input. We presume that $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ in (1) is unknown, meaning we have no prior knowledge of the system dynamics. Instead, we assume that we have access to a dataset \mathbf{X} consisting of a sequence of $r \in \mathbb{N}$ state measurements corrupted by noise and control inputs \mathbf{U} taken at discrete times t_1, t_2, \dots, t_r , given by

$$\begin{aligned} \mathbf{X} &= \{\mathbf{x}(t_1) + \epsilon_1, \mathbf{x}(t_2) + \epsilon_2, \dots, \mathbf{x}(t_r) + \epsilon_r\} \\ \mathbf{U} &= \{\mathbf{u}(t_1), \mathbf{u}(t_2), \dots, \mathbf{u}(t_r)\}, \end{aligned} \quad (2)$$

where $\epsilon_i \sim \mathcal{N}(0, \delta^2 I)$. We assume that the derivatives of the state with respect to time $\dot{\mathbf{x}}(t) \in \mathbb{R}^n$ corresponding to $\mathbf{x}(t)$ are not directly measurable and must be approximated using only the available data, e.g. using (central) finite differencing. Let $\dot{\mathbf{X}}$ be the approximate state derivatives with respect to time of the points in the dataset \mathbf{X} in (2) after applying the corresponding control input in \mathbf{U} , such that $\dot{\mathbf{X}}_i$ is the derivative of \mathbf{X}_i using control input \mathbf{U}_i .

Intuitively, we can view \mathbf{X} and $\dot{\mathbf{X}}$ as matrices in $\mathbb{R}^{r \times n}$ where the i^{th} row \mathbf{X}_i corresponds to the state at time t_i

$$\mathbf{X} = \begin{bmatrix} - & \mathbf{X}_1 & - \\ & \vdots & \\ - & \mathbf{X}_r & - \end{bmatrix}, \quad (3)$$

and the i^{th} row $\dot{\mathbf{X}}_i$ is the time derivative of \mathbf{X}_i . The state measurements \mathbf{X} , affected by noise, lead to rough and inaccurate estimates of the time derivatives in $\dot{\mathbf{X}}$.

We assume that the dynamics can be described by a linear combination of relatively few elementary function terms such as polynomials of varying degrees, sinusoidal terms, or exponential functions. For instance,

$$\dot{\mathbf{x}}(t) = \Theta(\mathbf{x}(t), \mathbf{u}(t))^\top \Xi, \quad (4)$$

where $\Theta(\mathbf{x}(t), \mathbf{u}(t)) \in \mathbb{R}^p$ is the candidate function library of elementary basis functions evaluated at the current state $\mathbf{x}(t)$ and control input $\mathbf{u}(t)$ and $\Xi \in \mathbb{R}^{p \times n}$ is a matrix of real-valued coefficients that weight the candidate function terms. For simplicity, using the dataset \mathbf{X} , the applied control inputs \mathbf{U} , and the state derivatives $\dot{\mathbf{X}}$, we can write the relationship between the datasets via

$$\dot{\mathbf{X}} = \Theta(\mathbf{X}, \mathbf{U})^\top \Xi. \quad (5)$$

In practice, (5) does not exactly hold as the dataset \mathbf{X} is corrupted by noise, and the approximation of $\dot{\mathbf{X}}$ introduces additional error as given by

$$\dot{\mathbf{X}} = \Theta(\mathbf{X}, \mathbf{U})^\top \Xi + \eta \mathbf{Z}, \quad (6)$$

where \mathbf{Z} is a matrix of independent, identically distributed zero-mean Gaussian entries and η is the magnitude of the standard deviation of the noise.

To find Ξ , one can use ordinary least-squares with \mathbf{X} and $\dot{\mathbf{X}}$ to find the model f from (1). However, this approach

does not lead to a sparse representation, instead overfitting the model to the data and finding a solution with nonzero elements in every element of Ξ . Sparsity is desirable to prevent overfitting, particularly with noisy data. Fortunately, LASSO [17] has been shown to work well with noise, using L_1 regularization to promote sparsity in the solution.

Problem 1 (LASSO for Symbolic Regression). *We seek to solve the LASSO problem*

$$\xi_j = \underset{\xi \in \mathbb{R}^p}{\operatorname{argmin}} \|\Theta(\mathbf{X}, \mathbf{U})^\top \xi - \dot{\mathbf{X}}_j\|_2 + \lambda \|\xi\|_1, \quad (7)$$

where the optimization variable $\xi_j \in \mathbb{R}^p$ is the j^{th} column of Ξ from (6), $\dot{\mathbf{X}}_j$ is the j^{th} column of $\dot{\mathbf{X}}$ from (6), and $\lambda > 0$ is the L_1 regularization parameter.

Solving the LASSO problem produces a more sparse representation compared to least-squares. However, our experiments demonstrate that using noisy data in Equation (7) can result in identifying a model that inaccurately reflects the system dynamics and performs poorly in extrapolation for prediction. Furthermore, data with sparse temporal resolution, such as low frequency, might not provide enough information to learn models with rapidly evolving dynamics. Addressing the challenges posed by noise and data sparsity is crucial for accurately identifying system dynamics.

III. APPROACH

Several techniques exist for de-noising data including Fourier transforms, a range of filtering methods, and neural networks [18]. In this work, we propose using Gaussian process regression (kriging) for data smoothing to mitigate noise-related issues, interpolate data for symbolic regression, and enhance the analytical model's precision.

Like SINDy, Gaussian process regression yields a model for relating input and output data. Unlike SINDy, Gaussian process regression is *non*-parametric; it models a probability distribution of the data \mathbf{X} , which is a function of \mathbf{t} . This distribution is described by a mean function $m(\cdot)$ and covariance kernel function $k(\cdot, \cdot)$, and the negative log-likelihood of the data \mathbf{X} is given by

$$\begin{aligned} -\log p(\mathbf{X}) &= \frac{1}{2} \mathbf{X}^T (K + \theta_n^2 \mathbf{I})^{-1} \mathbf{X} \\ &\quad + \frac{1}{2} \log |K + \theta_n^2 \mathbf{I}| + \frac{n}{2} \log(2\pi), \end{aligned} \quad (8)$$

where $K = k(\mathbf{t}, \mathbf{t})$ and θ_n is a tunable hyperparameter for noise variance. Gaussian process regression performance is sensitive to the kernel choice; different kernels can lead to poor extrapolation, where the predicted mean reverts to the training dataset's mean function [5]. We use the standard squared-exponential kernel, characterized by its tunable hyperparameters: θ_f (signal variance) and θ_l (length scale)

$$k(t_i, t_j) = \theta_f^2 \exp\left(-\frac{1}{2\theta_l^2} \|t_i - t_j\|^2\right). \quad (9)$$

In practice, the hyperparameters are determined by minimizing (8) with respect to θ_f , θ_l , and θ_n . To use Gaussian process regression as a de-noising tool, we first assume that

\mathbf{X} was generated from a zero-mean Gaussian process at training times \mathbf{t} . Now, let \mathbf{X}_* be a random Gaussian vector generated from a Gaussian process at desired test times \mathbf{t}_*

$$\begin{bmatrix} \mathbf{X}_* \\ \mathbf{X} \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(\mathbf{t}_*, \mathbf{t}_*) & K(\mathbf{t}_*, \mathbf{t}) \\ K(\mathbf{t}, \mathbf{t}_*) & K(\mathbf{t}, \mathbf{t}) + \theta_n^2 I \end{bmatrix} \right), \quad (10)$$

where r is the number of training points, and r_* is the number of test points. $K(\mathbf{t}, \mathbf{t}_*)$ denotes the $r \times r_*$ matrix of the covariances evaluated at all pairs of training and test points, $K(\mathbf{t}, \mathbf{t})$ is a $r \times r$ matrix of covariances, and likewise for $K(\mathbf{t}_*, \mathbf{t})$ and $K(\mathbf{t}_*, \mathbf{t}_*)$. To obtain smoothed estimates of $\dot{\mathbf{X}}$ evaluated at the test points, we condition the distribution of the training data on the test data to compute the posterior

$$\mathbf{X}_{GP} = K(\mathbf{t}_*, \mathbf{t})[K(\mathbf{t}, \mathbf{t}) + \theta_n^2 I]^{-1} \mathbf{X}. \quad (11)$$

We can evaluate \mathbf{X}_{GP} at the test points of the input data or interpolate \mathbf{t}_* to higher frequencies to increase the data for symbolic regression. The joint distribution of the training and test points for the state derivatives $\dot{\mathbf{X}}$ is given by

$$\begin{bmatrix} \dot{\mathbf{X}}_* \\ \dot{\mathbf{X}} \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(\mathbf{X}_*, \mathbf{X}_*) & K(\mathbf{X}_*, \mathbf{X}) \\ K(\mathbf{X}, \mathbf{X}_*) & K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I \end{bmatrix} \right), \quad (12)$$

where $K(\mathbf{X}, \mathbf{X}_*)$ denotes the $r \times r_*$ matrix of the covariances and similarly for $K(\mathbf{X}, \mathbf{X})$, $K(\mathbf{X}_*, \mathbf{X})$ and $K(\mathbf{X}_*, \mathbf{X}_*)$. σ_n is the noise variance hyperparameter for $\dot{\mathbf{X}}$. The kernel for $\dot{\mathbf{X}}$ is characterized by its tunable hyperparameters: σ_f (signal variance) and σ_l (length scale)

$$k(\mathbf{X}_i, \mathbf{X}_j) = \sigma_f^2 \exp \left(-\frac{1}{2\sigma_l^2} \|\mathbf{X}_i - \mathbf{X}_j\|^2 \right). \quad (13)$$

To calculate smoothed estimates of $\dot{\mathbf{X}}$ evaluated at the test points \mathbf{X}_* , we compute the posterior mean

$$\dot{\mathbf{X}}_{GP} = K(\mathbf{X}_*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1} \dot{\mathbf{X}}. \quad (14)$$

We note that $\dot{\mathbf{X}}_{GP}$ can also be computed using the smoothed or interpolated measurements \mathbf{X}_{GP} via

$$\dot{\mathbf{X}}_{GP} = K(\mathbf{X}_{GP*}, \mathbf{X}_{GP})[K(\mathbf{X}_{GP}, \mathbf{X}_{GP}) + \sigma_n^2 I]^{-1} \dot{\mathbf{X}}. \quad (15)$$

Now that we have shown how to obtain \mathbf{X}_{GP} and $\dot{\mathbf{X}}_{GP}$, we move forward to solve the problem in (7).

Remark 1. *The novelty of our method lies in how we treat \mathbf{X}_{GP} as the input to the kernel for smoothing $\dot{\mathbf{X}}$. This approach can produce smoothed and interpolated state derivatives $\dot{\mathbf{X}}_{GP}$ for symbolic regression.*

A. GPSINDy: Symbolic Regression with GP Denoising

First, we minimize the negative log-likelihood of the state \mathbf{X} with respect to the hyperparameters $\theta = [\theta_f, \theta_l, \theta_n]$, assuming its mean function $m(t) = 0$ via Equation (8). Then, we condition the distribution of the training data on the test data to obtain the posterior mean of the state evaluated at the test points \mathbf{t}_* according to (11).

Next, we minimize the negative log-likelihood of the state derivative $\dot{\mathbf{X}}$ with respect to the hyperparameters σ_f , σ_l , and σ_n as in (16), which is given by

$$-\log p(\dot{\mathbf{X}}) = \frac{1}{2} (\dot{\mathbf{X}} - m(\mathbf{X}))^\top (K + \sigma_n^2)^{-1} (\dot{\mathbf{X}} - m(\mathbf{X})) + \frac{1}{2} \log |K| + \frac{n}{2} \log(2\pi), \quad (16)$$

where $K = k(\mathbf{X}_{GP}, \mathbf{X}_{GP})$. Again, we assume a mean function $m(\mathbf{X}) = 0$. Then, we compute the posterior mean $\dot{\mathbf{X}}_{GP}$ using Equation (15). Finally, we update the LASSO problem from (7) to use \mathbf{X}_{GP} and $\dot{\mathbf{X}}_{GP}$ when solving for the coefficients of the system dynamics

$$\xi_j = \underset{\xi \in \mathbb{R}^p}{\operatorname{argmin}} \|\Theta(\mathbf{X}_{GP}, \mathbf{U})^\top \xi - \dot{\mathbf{X}}_{GP,j}\|_2 + \lambda \|\xi\|_1, \quad (17)$$

where $\dot{\mathbf{X}}_{GP,j}$ represents the j^{th} column of $\dot{\mathbf{X}}_{GP}$.

B. Optimization and Cross-Validation

LASSO can be computationally expensive for large data sets. Fortunately, the objective function in (17) is separable, making it suitable for optimization via splitting methods. One such method, the Alternating Direction Method of Multipliers (ADMM) [19], handles processing of large data sets by splitting its primary variable into two parts and then updating each part in an alternating fashion. We can solve the LASSO problem in (17) using ADMM by treating Ξ as the primary variable to be split as shown in Algorithm 1.

Algorithm 1 GPSINDy with ADMM (LASSO)

Input: state measurements \mathbf{X} , control inputs \mathbf{U} , computed state derivatives $\dot{\mathbf{X}}$, L_1 parameter λ

Output: coefficients for active nonlinear terms Ξ

- 1: compute \mathbf{X}_{GP} and $\dot{\mathbf{X}}_{GP}$ using (11) and (15)
 - 2: construct data matrix Θ using \mathbf{X}_{GP} and \mathbf{U}
 - 3: **for** $k = 1, 2, \dots, n$ **do**
 - 4: $\xi_k = \operatorname{ADMM}(\Theta, \dot{\mathbf{X}}_k, \lambda)$
 - 5: **end for**
 - 6: $\Xi = [\xi_1, \xi_2, \dots, \xi_n]$
-

A suitable λ balances model complexity (determined by the number of nonzero coefficients in Ξ) and accuracy. Cross-validation determines the optimal sparsity parameter and hyperparameters by evaluating model performance across various training and test sets to achieve the best results [20]. In this work, we perform cross-validation with LASSO to achieve a sparse solution with the best model fit based on the dataset. We use ADMM to solve the LASSO problem in (17) to discover the dynamics for an unknown system using noisy measurements, and we call this method GPSINDy.

IV. EXPERIMENTS & RESULTS

We demonstrate the effectiveness of GPSINDy on the Lotka-Volterra model, a nonholonomic model with unicycle dynamics, and data from the NVIDIA JetRacer. This latter test underscores GPSINDy's robustness in handling noisy datasets from real-world hardware. We benchmark GPSINDy against the SINDy algorithm [1] and a neural network-based method, NNSINDy, which uses a neural network for data

Θ term	Ground Truth		SINDy		GPSINDy	
	\dot{x}_1	\dot{x}_2	\dot{x}_1	\dot{x}_2	\dot{x}_1	\dot{x}_2
x_1	1.1	0.0	1.108	0.0	1.097	0.0
x_2	0.0	-0.1	0.0	-0.997	0.0	-0.980
$x_1 x_2$	-0.4	0.4	-0.397	0.382	-0.358	0.396
x_2^2	0.0	0.0	0.0	0.0	0.0	-0.005
$\cos(x_1)$	0.0	0.0	0.0	0.016	-0.049	0.0
$x_1 \cos(x_1)$	0.0	0.0	0.0	-0.005	0.0	0.0
$x_1 x_1 \sin(x_2)$	0.0	0.0	-0.003	0.0	0.0	0.0
$x_1 x_2 \sin(x_1)$	0.0	0.0	0.0	-0.007	0.0	0.0
$x_1 x_2 \sin(x_2)$	0.0	0.0	0.003	0.0	0.0	0.0
$x_1 x_1 \cos(x_1)$	0.0	0.0	0.0	-0.009	0.0	-0.003
$x_1 x_1 \cos(x_2)$	0.0	0.0	-0.001	0.0	0.0	0.0

TABLE I: **GPSINDy learns better coefficients for the predator-prey model.** In this table we compare the coefficients learned by SINDy and GPSINDy with the ground truth coefficients. The bold values show the best learned coefficients compared to the ground-truth coefficients for both \dot{x}_1 and \dot{x}_2 .

refinement and LASSO for symbolic regression. The neural network used in NNSINDy has two layers with 32 hidden neurons each and is trained using the ADAM optimizer [21]. We refer to the ground-truth coefficients as Ξ_{GT} and the learned coefficients as $\Xi_{Learned}$ for all experiments.

Experimental Setup: In all of the experiments unless specified, we simulate the system for 30s at discrete time steps $t \in \{t_1, t_2, \dots, t_r\}$ with a sampling interval of 0.1s. We compute the derivatives $\dot{x}(t)$ using the truth dynamics and add noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ on top of $x(t)$ and $\dot{x}(t)$ to simulate measurement noise, thereby obtaining X and \dot{X} . We set aside the last 20% of the simulated data for validation purposes and use the rest for training. We smooth X and \dot{X} using Gaussian process regression as described in (11) and (14) to obtain X_{GP} and \dot{X}_{GP} from the training data and then compute $\Theta(X_{GP})$ using (18). We apply Gaussian process regression with a squared exponential kernel, optimizing the hyperparameters through maximum likelihood. Finally, using \dot{X}_{GP} and $\Theta(X_{GP})$, we solve the L_1 -regularized least-squares problem in (17) using $\lambda = 0.1$. For the baseline NNSINDy, we train a neural network to predict \dot{X} given the observations of X using the training data and apply LASSO regression to discover the coefficients.

We choose the candidate function library $\Theta(X, U)$ (unless specified) such that it consists of polynomial terms up to 3rd order, sinusoidal terms (sin and cos), and combinations of the polynomial and sinusoidal terms. For example,

$$\Theta(X, U) = \begin{bmatrix} 1 & X & X^{P_2} & \dots & \sin(U) & \dots \end{bmatrix}, \quad (18)$$

where X^{P_2} denotes the quadratic nonlinearities in the state variable X . While we have chosen Θ for the dynamical systems in our experiments, in practice, the function library $\Theta(X, U)$ could be expanded to include a wider range of nonlinear basis functions tailored to the system in question.

A. Lotka-Volterra Model (Predator/Prey)

We first consider the problem of identifying the equations of motion for the Lotka-Volterra model [22], which can be

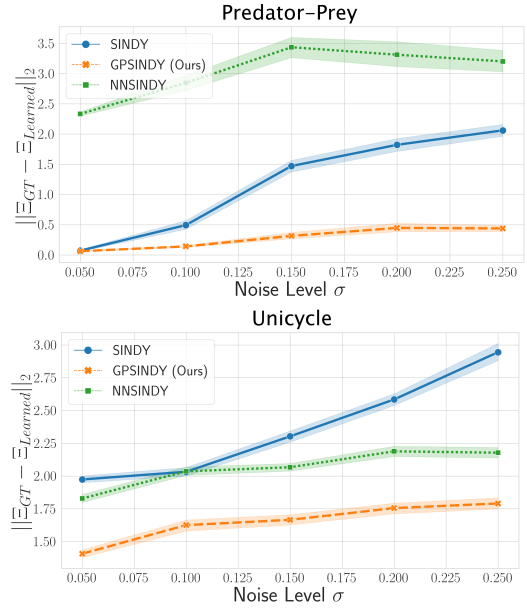


Fig. 1: **GPSINDy outperforms baselines in learning model coefficients for the Predator-Prey (top) and the unicycle model (bottom) under noisy measurements.** Contrast the dynamics learned by SINDy (blue), GPSINDy (orange), and NNSINDy (green) across varying levels for added noise standard deviation σ . The horizontal axis represents σ varying from 0.050 to 0.25 and the vertical axis the mean-squared error between the ground-truth coefficients (Ξ_{GT}) and the learned coefficients ($\Xi_{Learned}$). The ribbon indicates the standard deviation around the mean line. Lower overall error is better. Each experiment evaluates coefficients learned from noisy measurements with trials repeated over 40 seeds for each σ .

used to model the population of predator and prey species over time. The dynamics of the system are given by

$$\dot{x}_1 = ax_1 - bx_1x_2, \quad \dot{x}_2 = -cx_2 + dx_1x_2, \quad (19)$$

where x_1 represents the size of the prey population and x_2 represents the size of the predator population, $a = 1.1$ and $b = 0.4$ describe the prey growth rate and the effect of predation upon the prey population, and $c = 1.0$ and $d = 0.4$ describe the predator's death rate and the growth of predators based on the prey population. For this experiment, we standardize the data, i.e. normalize to have zero mean and unit variance, to improve parameter estimation of covariance functions and mitigate numerical issues associated with inverting ill-conditioned covariance matrices [23].

We first compare the learned coefficients Ξ between SINDy and our proposed approach in Table I. We observe that the estimates for the parameters a, b, c , and d obtained by GPSINDy are generally a closer approximation of the true underlying dynamics and that the coefficients matrix Ξ learned by GPSINDy is also more sparse than the one learned by SINDy. This is because our approach uses Gaussian processes to estimate \dot{X} , which is a smoother approximation of the true derivatives of X than the one corrupted by noise.

We also quantitatively compare the performance of SINDy, NNSINDy, and GPSINDy on data corrupted by different levels of noise as shown in Figure 1. The results show GPSINDy consistently outperforms the baselines across all noise magnitudes, highlighting its robustness in dealing with

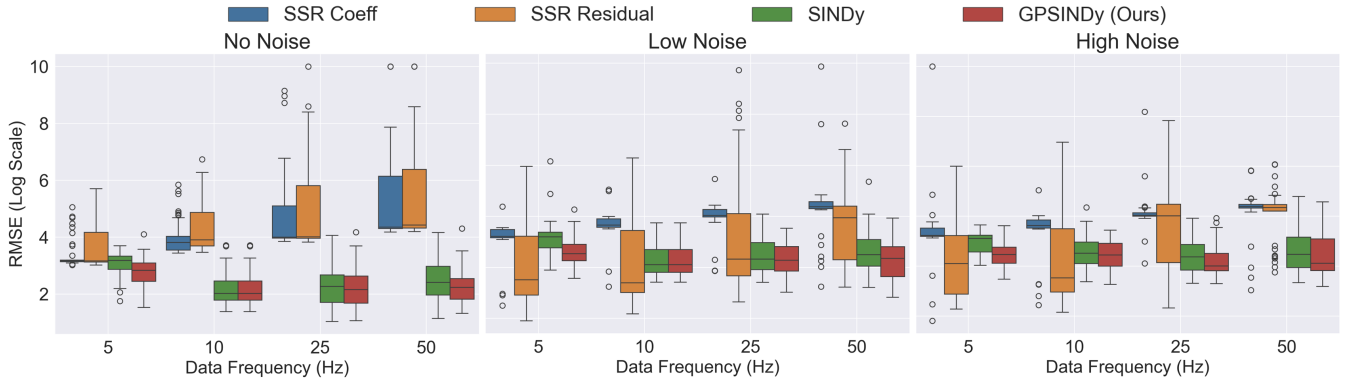


Fig. 2: **GPSINDy outperforms baselines on NVIDIA JetRacer trajectories under noisy measurements.** Each column represents the NVIDIA JetRacer dataset at different noise levels. The horizontal axis represents the data frequency (Hz) and the vertical axis quantifies the log root mean-squared error (RMSE) between the predicted states of a trajectory from learned system dynamics and ground-truth states. Lower overall RMSE is better. Each baseline is shown in a different color, with GPSINDy in red. This figure shows that over 45 rollouts, GPSINDy (red) achieves the lowest RMSE for most frequencies and noise levels. The baseline SSR Residual (yellow) sometimes beats GPSINDy; however, its significantly higher variance makes it unreliable for real-world settings.

noisy data. While SINDy is effective at low noise levels, it struggles at higher levels. NNSINDy, constrained by limited data, fails to effectively learn model coefficients.

B. Unicycle Dynamics (Simulation)

We now consider the nonholonomic unicycle system

$$\dot{x}_1 = x_3 \cos(x_4), \dot{x}_2 = x_3 \sin(x_4), \dot{x}_3 = u_1, \dot{x}_4 = u_2. \quad (20)$$

We set the control inputs as $u_1(t) = \sin(t)$ and $u_2(t) = \frac{1}{2} \cos(t)$ to perturb the dynamics from an initial condition $x_0 = [0, 0, 0.5, 0.5]^\top$. In practice, any function can be chosen which perturb the dynamics. Unlike the prior experiment, we opt to not standardize the states $x(t)$ since the data spread was already apt for Gaussian process regression. We note that we use polynomial terms up to 1st order in $\Theta(X_{GP}, U)$ as each method failed to identify the truth coefficients when 3rd order terms were included.

We compare the performance of SINDy, NNSINDy, and GPSINDy on data with varying noise levels in Figure 1. GPSINDy consistently outperforms other methods at higher noise levels. While NNSINDy and GPSINDy perform similarly for both the predator-prey and unicycle systems, SINDy’s performance notably degrades for the unicycle system, particularly with increased noise levels. All methods show significant errors in learned coefficients compared to true system dynamics, suggesting model mismatch. Nevertheless, GPSINDy learns more accurate coefficients even with complex dynamics like the nonholonomic unicycle system and noise-corrupted measurements.

C. JetRacer Hardware Demonstration

We also test our method real hardware data collected from a NVIDIA JetRacer, a 1/10 scale high speed car. We actuated the car to drive in a figure-8 made up of two circles, 3m in diameter. The nominal time for each lap was 5.5s with nominal velocity of 3.4 m s^{-1} . VICON sensors captured 22.85s of the system’s motion at discrete timesteps of 0.2s (50 Hz), and the control inputs U were saved at the same sampling rate for 45 total runs. We define the state X_i at

time t_i to be the measured x_1 and x_2 position in m, forward velocity magnitude v of the car in m s^{-1} , and heading angle ϕ (with respect to a global frame) in rad s^{-1} . We stack each state measurement (3) to gather X , after which we approximate \dot{X} using central finite differencing.

To evaluate performance for varying frequencies and noise levels, we downsample the 50 Hz state and control time histories to 25, 10, and 5 Hz and add noise $\epsilon_{low} \sim \mathcal{N}(0, 0.01^2)$ and $\epsilon_{high} \sim \mathcal{N}(0, 0.02^2)$ to the state measurements. We generate smoothed points X_{GP} and \dot{X}_{GP} at the same time points for each frequency and as well as twice the input frequency to provide more points for symbolic regression. Finally, we compute $\Theta(X_{GP}, U)$ using the smoothed points and then solve the L_1 -regularized least squares problem in (17) to obtain the GPSINDy dynamics model.

To achieve the best model fit, we tune λ individually for SINDy and GPSINDy via cross-validation, starting at $\lambda = 10^{-6}$ and increment logarithmically until reaching 1. Then, we increase λ by 10 until all of the coefficients regress to 0. We propagate the dynamics for each λ and, at the end, select the λ for each \dot{X} that best fit the data. Our baselines include Stepwise Sparse Regressor (SSR) Coefficient and Residual [24], algorithms designed mitigate measurement noise like GPSINDy and serve as suitable benchmarks. SSR Coeff truncates the smallest coefficient at each iteration while SSR Res computes multiple models chopping each coefficient, finally choosing the model with the lowest residual error.

In Figure 2, we show the results for all 45 rollouts for GPSINDy and the aforementioned baselines. For the datasets with added noise ϵ_{low} and ϵ_{high} , SSR Residual shows the lowest error at the lower frequencies, but GPSINDy gives the lowest error at the higher frequencies. However, the error bars for GPSINDy demonstrate lower variance than SSR Residual across all frequencies. GPSINDy beats all baselines for all frequencies with no added noise to the collected data, in particular the **real-world 50 Hz hardware data**.

Table II shows that GPSINDy achieves 20.78% lower RMSE error than SINDy, 83.16% lower error than SSR Coeff, and 82.23% lower error than SSR Res. GPSINDy has

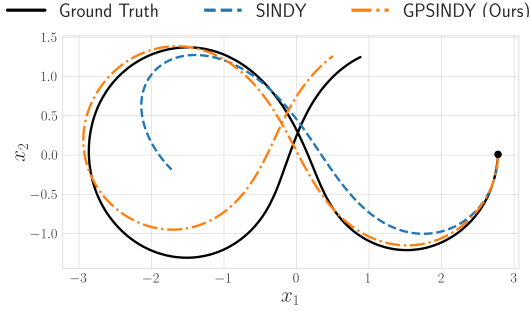


Fig. 3: **GPSINDy Trajectories Align Closely with Ground Truth for the Real JetRacer System.** The plot contrasts trajectories predicted from SINDy (blue) and GPSINDy (orange) with the ground truth (black) based on one rollout out of the total collected JetRacer data. The axes denote the JetRacer’s Cartesian coordinates. For this trajectory, the RMSE error norm between the x_1 and x_2 coordinates for SINDy on the testing data is 1.4m^2 , while for GPSINDy it is reduced to 0.23m^2 .

the lowest aggregate prediction error and variance among the chosen baselines.

	GPSINDy	SINDy	SSR Coeff	SSR Res
Mean RMSE	11.002	13.888	65.349	61.917
\pm variance	± 1.525	± 1.619	± 3.823	± 40.987

TABLE II: **GPSINDy Achieves Lowest Predicted Error Among Baselines on JetRacer Trajectories Under Noisy Measurements.** This table shows the mean RMSE and variance for all frequencies (5, 10, 25, and 50 Hz) and noise levels (no noise, low noise, and high noise). Although SSR Residual sometimes beats GPSINDy on lower frequencies as shown in Figure 2, GPSINDy has the lowest mean RMSE and variance across all frequencies and noise levels.

V. DISCUSSION

We have devised a method to learn models using data with high noise and sparsity in symbolic regression algorithms such as SINDy. We smooth and interpolate sparse, noisy measurements using Gaussian Process regression and then solve the LASSO problem with ADMM to learn more accurate dynamics over SINDy and other baselines. We demonstrate our approach on a Lotka-Volterra system, on an simulated unicycle system, and on noisy data taken from hardware experiments using an NVIDIA JetRacer system. Our results show that using Gaussian processes significantly improves system identification for SINDy.

Extensive experimentation ultimately led to the simplicity of our method. We initially tried optimizing the marginal log-likelihood for the data \tilde{X} while simultaneously learning the dynamics coefficients Ξ using a novel ADMM-based method. We replaced the mean function $m(X)$ in Equation (16) with $\Theta(X, U)\Xi$ from Equation (5) and iteratively tuned the hyperparameters while solving for Ξ , but this approach did not lead to accurate results. We also experimented with different kernel functions including the periodic kernel and various Matern kernels, finding that the squared-exponential kernel worked best with our data.

Future work should benchmark our method against directly taking the derivative of Gaussian Processes, provide a more thorough comparison between existing approaches, and conduct more testing on different dynamic systems.

REFERENCES

- [1] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.
- [2] A. Cortiella, K.-C. Park, and A. Doostan, “Sparse identification of nonlinear dynamical systems via reweighted ℓ_1 -regularized least squares,” *Computer Methods in Applied Mechanics and Engineering*, vol. 376, p. 113620, 2021.
- [3] J. Wentz and A. Doostan, “Derivative-based SINDy (DSINDy): Addressing the challenge of discovering governing equations from noisy data,” *Computer Methods in Applied Mechanics and Engineering*, vol. 413, p. 116096, Aug. 2023. arXiv:2211.05918 [math].
- [4] E. Kaiser, J. N. Kutz, and S. L. Brunton, “Sparse identification of nonlinear dynamics for model predictive control in the low-data limit,” *Proceedings of the Royal Society A*, 2018.
- [5] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [6] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton, “Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control,” *Proceedings of the Royal Society A*, vol. 478, no. 2260, p. 20210904, 2022.
- [7] H. Schaeffer and S. G. McCalla, “Sparse model selection via integral terms,” *Physical Review E*, vol. 96, no. 2, p. 023302, 2017.
- [8] L. Boninsegna, F. Nüske, and C. Clementi, “Sparse learning of stochastic dynamical equations,” *The Journal of chemical physics*, vol. 148, no. 24, 2018.
- [9] K. Kaheman, J. N. Kutz, and S. L. Brunton, “Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics,” *Proceedings of the Royal Society A*, 2020.
- [10] G. L’Erario, L. Fiorio, G. Nava, F. Bergonti, H. A. O. Mohamed, E. Benenati, S. Traversaro, and D. Pucci, “Modeling, identification and control of model jet engines for jet powered robotics,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2070–2077, 2020.
- [11] M. Hoffmann, C. Fröhner, and F. Noé, “Reactive sindy: Discovering governing reactions from concentration data,” *The Journal of chemical physics*, vol. 150, no. 2, 2019.
- [12] Z. Chen, Y. Liu, and H. Sun, “Physics-informed learning of governing equations from scarce data,” *Nature communications*, 2021.
- [13] K. Kaheman, S. L. Brunton, and J. N. Kutz, “Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data,” *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015031, 2022.
- [14] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” *Advances in neural information processing systems*, vol. 31, 2018.
- [15] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Machine learning of linear differential equations using gaussian processes,” *Journal of Computational Physics*, vol. 348, pp. 683–693, 2017.
- [16] M. Heinonen, C. Yildiz, H. Mannerström, J. Intosalmi, and H. Lähdesmäki, “Learning unknown ode models with gaussian processes,” in *International conference on machine learning*, pp. 1959–1968, PMLR, 2018.
- [17] R. Tibshirani, *Regression Shrinkage and Selection via the Lasso*. Oxford University Press, 1996.
- [18] S. V. Vaseghi, *Advanced digital signal processing and noise reduction*. John Wiley & Sons, 2008.
- [19] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, pp. 1–122, 01 2011.
- [20] K. Ito and R. Nakano, “Optimizing support vector regression hyperparameters based on cross-validation,” in *Proceedings of the International Joint Conference on Neural Networks*, IEEE, 2003.
- [21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2015.
- [22] V. Křivan, “Prey–predator models,” in *Encyclopedia of Ecology* (S. E. Jørgensen and B. D. Fath, eds.), pp. 2929–2940, Oxford: Academic Press, 2008.
- [23] H. C. Lingmont, F. Alijani, and M. A. Bessa, “Data-driven techniques for finding governing equations of noisy nonlinear dynamical systems,” 2020.
- [24] L. Boninsegna, F. Nüske, and C. Clementi, “Sparse learning of stochastic dynamical equations,” *The Journal of chemical physics*, vol. 148, no. 24, 2018.