Scientific Research Publishing

Journal of Computer and Communications, 2026, 14(5), 32-53
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Leaning Sparse Nonlinear Dynamical Systems via Frank-Wolfe Algorithm with Integral Strategy

Peiyun Yin, Xiaoge Guo

College of Informatics, Huazhong Agricultural University, Wuhan, China
Email: ypy1752023@163.com

How to cite this paper: Yin, P.Y. and Guo, X.G. (2026) Leaning Sparse Nonlinear Dynamical Systems via Frank-Wolfe Algorithm with Integral Strategy. *Journal of Computer and Communications*, **14**, 32-53.
https://doi.org/10.4236/jcc.2026.145003

Received: April 19, 2026
Accepted: May 22, 2026
Published: May 25, 2026

Abstract

Sparse identification of nonlinear dynamical systems is an important project, directly addressing the physics community's long-standing goal of data-driven discovery. Although many effective methods have been developed to enhance the ability for data-driven discovery of governing equations, critical challenges, including derivative computation and optimization methods, demanding more robust theoretical foundations and innovative computational strategies. This paper proposes a novel sparse optimization strategy for learning nonlinear dynamics from noisy data. The strategy is based on the Frank-Wolfe algorithm (also known as the conditional gradient method), an iterative first-order optimization technique for constrained convex optimization. Unlike traditional methods that rely on projection, the algorithm minimizes a linear approximation of the objective function over the feasible set to determine the search direction, naturally yielding sparse solutions, which is consistent with prior work in sparse identification of nonlinear dynamics (SINDy). However, our approach differs fundamentally: while SINDy enforces sparsity through user-defined thresholding, the Frank-Wolfe algorithm inherently produces sparse solutions by minimizing a convex objective over a compact set. This distinction endows our method with greater scalability to large-scale dynamics and noisy data. To further mitigate instabilities in numerical differentiation, particularly under noisy conditions, we incorporate an integral form to replace derivative estimation. The proposed framework thus offers a robust, scalable, and interpretable alternative for data-driven modeling of complex systems. Numerical experiments demonstrate the effectiveness and superiority of our proposed method, which combines subsampling and data normalization, in identifying nonlinear dynamical systems.

Keywords

Nonlinear Dynamic System, Sparse Identification, Subsampling, Data Normalization, The Frank-Wolfe Algorithm

1. Introduction

With the exponential growth of data and computing power, data-driven discovery of governing equations has emerged as a transformative technique in science and engineering, enabling the discovery of dynamical system behaviors from observational data [1]-[7]. These rediscovered equations, such as the Navier-Stokes equations [8] [9] and the Lorentz equations [5], provide fundamental mathematical tools for analyzing and predicting system evolution. Recently, this technique has been extended to complex systems, including multiscale systems [10], hybrid systems [11], and complex networks [12]. Despite these advancements, critical challenges remain, including the generation of candidate functions, the computation of derivatives and the optimization method, which call for more robust theoretical foundations and innovative computational strategies.

Currently, many research efforts focused on the deep learning framework for the discovery of governing equations [3] [13] [14]. A notable example is the Physics-Informed Neural Networks (PINNs) proposed in [3]. PINNs learn the solutions to nonlinear partial differential equations (PDEs) by leveraging deep neural networks (DNNs) to capture hidden physical laws from data. Since both the solution function and the system dynamics are approximated by deep neural networks, the selection of basis functions can be avoided, and the derivatives can be evaluated at machine precision using symbolic or automatic differentiation. Authors of [15] generalized PINN to the fractional PINNs (fPINNs) where the residual in the loss function is introduced by using both automatic differentiation for the integer-order operators and numerical discretization for the fractional operators. Building on the PINN, some other versions also are developed, including physics-informed DeepONets [14], Kolmogorov-Arnold networks [16] and Fourier neural operator [17]. However, as with other neural networks, the trained PINN can act as a "black box". It is often difficult to interpret the model's internal workings or extract clear physical insights.

Instead of using deep neural network to represent the system dynamics, pioneering studies of data-driven discovery, proposed by [1] and [2], introduced genetic programming-based symbolic regression as a paradigm-shifting method. This evolutionary algorithm searches the space of mathematical expressions to identify nonlinear differential equations that optimally balance model complexity (quantified by term count) and predictive accuracy. By doing so, it fundamentally circumvents the rigid equation form constraints imposed by conventional regression techniques [18], which require predefined symbolic structures. Symbolic regression further addresses a long-standing challenge in physics: the simultaneous op-

timization of both structural and parametric representations. This capability directly aligns with the community's goal of data-driven discovery, as demonstrated in studies on predictive modeling [19] and robust parameter estimation [20]. However, symbolic regression methods suffer from a key limitation, high computational costs and overfitting, especially for high-dimensional nonlinear dynamics. For this, some attempts have been made to improve the weakness [21]-[24]. In [24], authors combine dimensional analysis with the AI Feynman symbolic regression algorithm to show that dimensional analysis significantly improves the accuracy of the recovered equation. Therein, PINNs also are used to uncover hidden terms in differential equations.

In contrast, SINDy (Sparse Identification of Nonlinear Dynamics) is another inspiring data-driven method that leverages sparse regression to extract governing equations from observational data [5]. SINDy has demonstrated remarkable versatility in applications ranging from fluid dynamics [25] [26] to biological systems [27] [28]. The core of SINDy lies in its ability to achieve model interpretability through sparse constraints. To do this, this method constructs a large library of candidate functions (e.g., polynomials, trigonometric functions) and combines them with sequential threshold optimization techniques to automatically identify the nonlinear terms that significantly contribute to system dynamics. Unlike conventional least-squares regression [18] and the aforementioned symbolic regression that may overfit noisy observations, SINDy's sparse regularization intrinsically suppresses irrelevant terms, yielding parsimonious models that generalize better.

Nevertheless, since the sparse regularization works by empirical pruning of the user-defined thresholds [29], this method encounters challenges in scenarios where the true underlying dynamics are unknown or highly complex. Additionally, the method's performance can be sensitive to the choice of basis functions and the estimation of derivatives, which may introduce biases or errors in the identification process. Therefore, innovative algorithms are needed to enhance the performance of the SINDy method [30]-[33].

In this paper, we proposed a different sparse optimization strategy to learn the nonlinear dynamics from noisy data. The optimization strategy uses the classical Frank-Wolfe algorithm, also known as the conditional gradient method [34] [35]. It is an iterative first-order optimization algorithm used for constrained convex optimization. Instead of performing a projection, the algorithm minimizes a linear approximation of the objective function over the feasible set to determine the next search direction. Notably, this method produces sparse solutions which is consistent with the developed method [5] [7]. The difference is that the Frank-Wolfe algorithm is a mathematical optimization method, which aims at minimizing a convex, differentiable objective function over a compact convex set and then naturally obtains a sparse solution, while the SINDy method ensures the sparsity via the user-customized thresholding. Thus, our proposed method has the larger scalability to large-scale dynamics and noisy data. Additionally, to ease the insta-

bility of numerical differentiation especially in the noise case, the integral form, initially developed by [36], also is used to replace the estimation of derivatives. Thus, the proposed framework offers a robust, scalable, and interpretable alternative for data-driven modeling of complex systems.

This paper is organized as follows: in the second section, the problem studied in this paper is described as a sparse leaning problem; in the third section, the sub-sampling and data normalization are, and the used Frank-Wolfe algorithm to solve the sparse regression problem is introduced. In the fourth section, several numerical experiments are presented to verify the performance of the proposed method. The fifth section summarizes the text.

2. Problem Statement

Consider a nonlinear dynamic system with the following form:

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}(t)\right), \mathbf{x}(0) = \mathbf{x}_0, \tag{1}$$

where the vector $\mathbf{x}(t) = \left[x_1(t), x_2(t), \cdots, x_n(t)\right]^{\mathrm{T}} \in \mathbb{R}^n$ is the state of the system at time t, the function $\mathbf{f}(\cdot): \mathbb{R}^n \to \mathbb{R}^n$ represents the dynamic constraints that mathematically defines the equations of motion governing the system, $\mathbf{x}_0$ represents the initial state of the system, and $\dot{\mathbf{x}}$ represents the derivative of the state variable of the system with respect to time. In general, when the function $\mathbf{f}$ is known and satisfies Lipschitz continuity, the system (1) has a unique solution [36]. However, it is a challenge for the system modeling community to specify the $\mathbf{f}$ in terms of the extensive physical knowledge. This work focuses on discovering the dynamic function $\mathbf{f}$ from the measurement data containing noise.

For this, we first assume that the data for the system (1) is available, which is denoted by $\left\{\mathbf{x}(t_k)\right\}_{k=1}^{K}$ at discrete time points $t_1, t_2, \cdots, t_K$ and $\mathbf{x}(t_0) = \mathbf{x}_0$. To discover the dynamics function $\mathbf{f}$, a widely adopted method is to represent the function $\mathbf{f}(\mathbf{x})$ as a linear combination of a set of trial functions. More specifically, the i-th ($i = 1, 2, \cdots, n$) component $f_i(\mathbf{x})$ of the function $\mathbf{f}(\mathbf{x})$ can be expressed as a linear combination of N trial functions $\theta_j(\mathbf{x})$, where the trial function can be the polynomial, trigonometric function, to name a few. Based on this, $f_i(\mathbf{x})$ can be expressed as follows:

$$f_i(\mathbf{x}) = \sum_{j=1}^{N} c_{i,j}\theta_j(\mathbf{x}), \tag{2}$$

and Equation (1) combined with Equation (2) can be rewritten into

$$\dot{x}_i(t) = \sum_{j=1}^{N} c_{i,j}\theta_j\left(\mathbf{x}(t)\right), \tag{3}$$

where $c_{i,j}$ is the to-be-identified coefficient. Since Equation (3) holds at any time, one can further write Equation (3) as a linear system in the matrix form:

$$\dot{\mathbf{x}}_i = \Theta(\mathbf{x})\mathbf{c}_i, \tag{4}$$

where

$$\mathbf{x}_i = \left[x_i(t_1), x_i(t_2), \cdots, x_i(t_K)\right]^{\mathrm{T}},$$

$$\dot{\mathbf{x}}_i = \left[\dot{x}_i(t_1), \dot{x}_i(t_2), \cdots, \dot{x}_i(t_K)\right]^{\mathrm{T}},$$

$$\Theta(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x}(t_1)) & \theta_2(\mathbf{x}(t_1)) & \cdots & \theta_N(\mathbf{x}(t_1)) \\ \theta_1(\mathbf{x}(t_2)) & \theta_2(\mathbf{x}(t_2)) & \cdots & \theta_N(\mathbf{x}(t_2)) \\ \vdots & \vdots & & \vdots \\ \theta_1(\mathbf{x}(t_K)) & \theta_2(\mathbf{x}(t_K)) & \cdots & \theta_N(\mathbf{x}(t_K)) \end{bmatrix},$$

and

$$\mathbf{c}_i = \left[c_{i,1}, c_{i,2}, \cdots, c_{i,N}\right]^{\mathrm{T}}.$$

Throughout this work, the candidate functions $\theta_j(x)$ are chosen as polynomial monomials of the state variables. For a system with state dimension n, a library of total degree d includes all monomials of the form $x_1^{p_1}, x_2^{p_2}, \cdots, x_n^{p_n}$ with $0 \le \sum_{i=1}^{n} p_i \le d$. The maximum degree d and the specific set of monomials may vary from one example to another, depending on the expected dominant nonlinearities of the target system. This adaptive choice improves computational efficiency and identification accuracy. For each numerical experiment in Section 4, we explicitly state the selected library in the corresponding subsection. The same library is used for all state variables within that experiment. As a consequence, one can identify the unknown coefficients $\mathbf{c}_i$ by solving Equation (4). However, it is difficult to estimate the derivatives from the noisy data due to the sensitivity of numerical computing.

To accurately infer $\mathbf{f}$ from data, the integral form can be adopted to avoid direct differentiation of noisy data [36]. Specifically, the integral form of Equation (4) is as follows:

$$\mathbf{x}_i(t) - \mathbf{x}_i(0) = \int_0^t \Theta(\mathbf{x}(\tau))\mathrm{d}\tau \mathbf{c}_i. \tag{5}$$

For each element $\theta_j(\mathbf{x}(t_k))$ of the matrix $\Theta(\mathbf{x}(t))$, we can approximate it using piecewise constant quadrature and represent it by

$$\varphi_j(\mathbf{x}(t_k)) = \int_0^{t_k} \theta_j(\mathbf{x}(\tau))\mathrm{d}\tau \approx \Delta t \sum_{l=1}^{K} \theta_j(\mathbf{x}(t_l)). \tag{6}$$

From this, we construct the feature matrix

$$\begin{aligned} \Phi(\mathbf{x}) &= \left[\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}), \cdots, \Phi_N(\mathbf{x})\right] \\ &= \begin{bmatrix} \varphi_1(\mathbf{x}(t_1)) & \varphi_2(\mathbf{x}(t_1)) & \cdots & \varphi_N(\mathbf{x}(t_1)) \\ \varphi_1(\mathbf{x}(t_2)) & \varphi_2(\mathbf{x}(t_2)) & \cdots & \varphi_N(\mathbf{x}(t_2)) \\ \vdots & \vdots & & \vdots \\ \varphi_1(\mathbf{x}(t_K)) & \varphi_2(\mathbf{x}(t_K)) & \cdots & \varphi_N(\mathbf{x}(t_K)) \end{bmatrix}. \end{aligned} \tag{7}$$

and collect all the displacements of the i-th state of the system from the initial time $t = 0$ to the time $t_k\ (k = 1, 2, \cdots, K)$ into a column vector:

$$\mathbf{X}_i = \mathbf{x}_i(t) - \mathbf{x}_i(0) = \begin{bmatrix} \mathbf{x}_i(t_1) - \mathbf{x}_i(0) \\ \mathbf{x}_i(t_2) - \mathbf{x}_i(0) \\ \vdots \\ \mathbf{x}_i(t_K) - \mathbf{x}_i(0) \end{bmatrix}. \tag{8}$$

By combining Equation (5)-Equation (8), we get the following discrete linear system:

$$\mathbf{X}_i = \Phi(\mathbf{x})\mathbf{c}_i, \tag{9}$$

where $\mathbf{X}_i \in \mathbb{R}^K$ is the displacement vector, $\Phi \in \mathbb{R}^{K\times N}$ is the feature matrix, and $\mathbf{c}_i \in \mathbb{R}^N$ is the unknown sparse vector. Therefore, the identification problem of nonlinear dynamic systems can be transformed into a sparse regression problem with a solution coefficient vector $\mathbf{c}_i$.

3. Sparse Recognition of Nonlinear Dynamical Systems Based on Frank-Wolfe Algorithm

This section mainly introduces a data-driven method for identifying nonlinear dynamical systems. In this method, the vector $\boldsymbol{X}_i$ and the feature matrix Φ in Equation (9) are subsampled [37], and further are normalized to obtain a new linear system. For the new linear system obtained after subsampling, the Frank-Wolfe algorithm [38] is used to solve the sparse coefficient vector $\boldsymbol{c}_i$.

3.1. Subsampling and Data Normalization

This subsection introduces the used subsampling and the random subsampling. For system subsampling, we first determine the sampling interval m_1 and the total sample M_1, and select position "1" as the sampling starting point to generate the sampling row index. For random subsampling, the total sample M_2 is determined, and the row index of unduplicated samples is randomly generated. The method of subsampling is to combine the row index generated by system subsampling and random subsampling and apply it to the vector $\mathbf{X}_i$ and the matrix Φ at the same time [8]. We mark the process of subsampling as $\mathcal{S}$, then the linear system obtained after subsampling is

$$\mathcal{S}\mathbf{X}_i = \mathcal{S}\Phi\mathbf{c}_i, \tag{10}$$

where $\mathcal{S}\mathbf{X}_i \in \mathbb{R}^{M_1+M_2}$, $\mathcal{S}\Phi \in \mathbb{R}^{(M_1+M_2)\times N}$ and the subsampling process is shown in Figure 1.

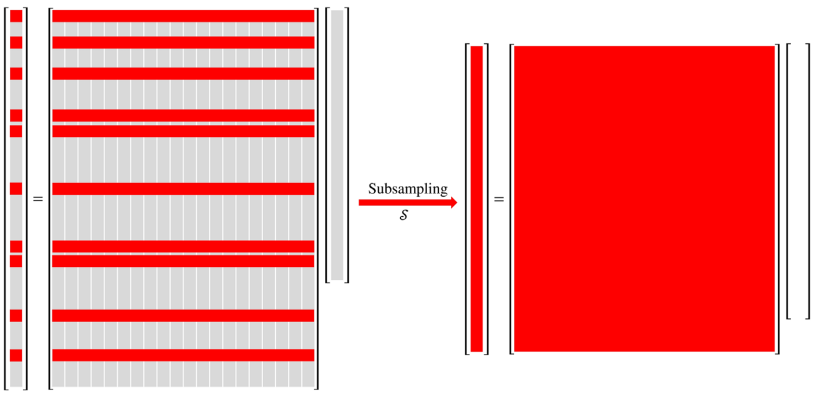


Figure 1. Take row-identical subsamples of vector $\mathbf{X}_i$ and matrix Φ in the linear system (9).

In the process of integrating the linear system (4), there will be a large difference of orders of magnitude, which will lead to poor identification effect. In order to solve this problem, we need to normalize the data to maintain data stability and improve the generalization ability of the model. For the vector $\mathcal{S}\mathbf{X}_i$ in the linear system (10), the maximum absolute is used to scale to it. Then, we have

$$\mathbf{Y}_i = \frac{\mathcal{S}\mathbf{X}_i}{\max\left(\left|\mathcal{S}\mathbf{X}_i\right|\right)}. \tag{11}$$

For matrix $\mathcal{S}\Phi$, applying the L_2 norm normalization to $\mathcal{S}\Phi$ column-wise:

$$\mathbf{D}_j = \frac{\mathcal{S}\Phi_j}{\left\|\mathcal{S}\Phi_j\right\|_2}, \tag{12}$$

then all normalized column vectors $\mathbf{D}_j$ are represented as columns of the new matrix $\mathbf{D}$ by:

$$\mathbf{D} = \left[\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_N\right]. \tag{13}$$

Therefore, the linear system after subsampling and normalization is as follows:

$$\mathbf{Y}_i = \mathbf{D}\mathbf{c}_i, \tag{14}$$

where $\mathbf{Y}_i \in \mathbb{R}^{M_1+M_2}$, $\mathbf{D} \in \mathbb{R}^{(M_1+M_2)\times N}$, $\mathbf{c}_i \in \mathbb{R}^N$. Using this method to preprocess data can help us to identify the governing equations of nonlinear dynamical systems more effectively in terms of reducing calculation cost, processing unbalanced data and improving data stability.

However, it is also worth noting that in practical applications, due to the fact that the data contains noise, Equation (14) does not strictly hold so that

$$\mathbf{Y}_i \approx \mathbf{D}\mathbf{c}_i. \tag{15}$$

This approximation makes it very difficult for us to solve the inverse problem. In order to deal with this challenge, we need to find an optimal $\hat{\mathbf{c}}_i$, which makes the error between $\mathbf{Y}_i$ and $\mathbf{D}\mathbf{c}_i$ extremely small, and promotes the sparsity of the coefficient vector $\mathbf{c}_i$. For this, we use the Lasso Regression [39]-[41] to solve this problem:

$$\hat{c}_i = \arg\min_{c_i \in \mathbb{R}^N} \frac{1}{2}\left\|Dc_i - Y_i\right\|_2^2 + \lambda\left\|c_i\right\|_1, \tag{16}$$

where λ is the regularization parameter.

3.2. Frank-Wolfe Algorithm

In this subsection, we propose to use the Frank-Wolfe algorithm [34] [35] to solve the Lasso problem corresponding to Equation (16). Frank-Wolfe algorithm is an iterative method for solving convex optimization problems, especially when the objective function is a differentiable convex function and the constraint is convex set. Therefore, it is especially effective when dealing with convex optimization problems with L_1 constraints.

According to [39], Equation (16) can be equitably expressed as:

$$\min_{c_i \in \mathbb{R}^N} \frac{1}{2}\left\|Dc_i - Y_i\right\|_2^2 \ \text{s.t.}\ \left\|c_i\right\|_1 \le \alpha, \tag{17}$$

where α is the regularization parameter. Problem (17) is equivalent to (16) via the KKT conditions: the Lagrange multiplier for the ℓ_1-ball constraint acts as the regularization parameter λ. Thus, a larger λ (stronger sparsity) corresponds to a smaller α. In practice, we treat α as the main hyperparameter and select it empirically as described in Section IV. Then, we solve the convex optimization problem (17) to find the optimal sparse coefficient vector c_i by using the Frank-Wolfe algorithm.

We define the objective function as

$$g\left(c_i\right) = \frac{1}{2}\left\|Dc_i - Y_i\right\|_2^2. \tag{18}$$

Since the Frank-Wolfe algorithm is also called the conditional gradient algorithm, which determines the optimization direction by calculating the gradient, we need to compute the gradient of the objective function $g\left(c_i^{(p)}\right)$ at the p-th iteration:

$$\nabla g\left(c_i^{(p)}\right) = D^{\mathrm{T}}\left(Dc_i^{(p)} - Y_i\right), \tag{19}$$

where $p = 1, 2, \cdots, p_{\max}$, $p_{\max}$ is the maximum number of iterations. During each iteration, for a given gradient $\nabla g\left(c_i^{(p)}\right)$, we need to find a direction $s^{(p)}$ that minimizes the following linear subproblem:

$$s^{(p)} = \arg\min_{\|s\|_1 \le \alpha} \left\langle \nabla g\left(c_i^{(p)}\right), s \right\rangle. \tag{20}$$

This problem can be simplified by using the dual norm. For the L_1 norm constraint problem, the dual norm is the L_∞ norm [42], *i.e.*

$$\left\|\nabla g\left(c_i^{(p)}\right)\right\|_\infty = \max_j \left|\left(\nabla g\left(c_i^{(p)}\right)\right)_j\right|. \tag{21}$$

To find the optimal direction $s^{(p)}$, we need to minimize the inner product $\left\langle \nabla g\left(c_i^{(p)}\right), s \right\rangle$. This is equivalent to moving in the direction of the maximum absolute component of the gradient under the constraints of the L_1 norm, where the maximum absolute component is given by [38]

$$j^{(p)} = \arg\max_j \left|\left(\nabla g\left(c_i^{(p)}\right)\right)_j\right|. \tag{22}$$

Since we need the inner product $\left\langle \nabla g\left(c_i^{(p)}\right), s \right\rangle$ to be minimum and $\|s\|_1 \le \alpha$, we need to choose the direction of s to be opposite to the direction of the gradient and take the largest possible step size α in the direction of the gradient

component with the largest absolute value [38], *i.e.*

$$s^{(p)} = -\alpha sign\left(\left(\nabla g\left(c_i^{(p)}\right)\right)_{j^{(p)}}\right) e_{j^{(p)}}, \tag{23}$$

where $e_{j^{(p)}}$ is a unit vector that takes the value "1" in the direction of the maximum absolute component of the gradient.

Next, we perform an exact linear search between the current solution $c_i^{(p)}$ and the direction $s^{(p)}$ to find the optimal step size $\hat{\gamma}$ to minimize the objective function $g(c_i)$:

$$\hat{\gamma} = \arg\min_{\gamma \in [0,1]} g\left(c_i^{(p)} + \gamma d^{(p)}\right), \tag{24}$$

where $d^{(p)} = s^{(p)} - c_i^{(p)}$ indicates the search direction, and

$$g\left(c_i^{(p)} + \gamma d^{(p)}\right) = \frac{1}{2}\left\|\gamma D\left(s^{(p)} - c_i^{(p)}\right) + Dc_i^{(p)} - Y_i\right\|_2^2 \tag{25}$$

To find the optimal step size $\hat{\gamma}$, we take the derivative of Equation (25) and make the derivative zero, then we have

$$\hat{\gamma}^{(p)} = \frac{\left(q^{(p)}\right)^{\mathrm{T}}\left(Dc_i^{(p)} - Y_i\right)}{\left(q^{(p)}\right)^{\mathrm{T}} q^{(p)}}, \tag{26}$$

where $q^{(p)} = D\left(s^{(p)} - c_i^{(p)}\right)$.

Once the optimal step size $\hat{\gamma}^{(p)}$ in each iteration is calculated, we can update the current solution $c_i^{(p)}$ by the optimal step size to get the next solution $c_i^{(p+1)}$, *i.e.*

$$c_i^{(p+1)} = c_i^{(p)} + \hat{\gamma}^{(p)}\left(s^{(p)} - c_i^{(p)}\right) \tag{27}$$

We define the dual gap $h^{(p)}$:

$$h^{(p)} = -\left(d^{(p)}\right)^{\mathrm{T}} \nabla g\left(c_i^{(p)}\right). \tag{28}$$

If the dual gap $h^{(p)}$ is less than the preset threshold, the algorithm is considered to have converged and the iteration is stopped. Additionally, the used algorithm also will stop the iterations when the maximum number of iterations is reached.

Finally, the normalized solution vector c_i is restored to the original scale:

$$c_i = \frac{c_i^{(p)} \max\left(\left|\mathcal{S}X_i\right|\right)}{\left(\mathcal{S}\Phi_{norm}\right)^{\mathrm{T}}}, \tag{29}$$

where we define

$$\mathcal{S}\Phi_{norm} = \left[\left\|\mathcal{S}\Phi_1\right\|_2, \left\|\mathcal{S}\Phi_2\right\|_2, \cdots, \left\|\mathcal{S}\Phi_N\right\|_2\right]. \tag{30}$$

for nonlinear dynamical system identification based on the Frank-Wolfe algorithm is summarized in **Algorithm 1**.

4. Numerical Experiments and Results

In order to verify the effectiveness of our proposed identification method, several numerical experiments are presented in this section.

Algorithm 1. Frank-Wolfe algorithm for identification of nonlinear dynamical systems.

Input: Displacement vector $X_i \in \mathbb{R}^K$, feature matrix $\Phi \in \mathbb{R}^{K\times n}$, maximum number of iterations $p_{\max}$, parameter α and τ, threshold value tol.

Output: Vector c_i such that $X_i = \Phi c_i$.

According to Equation (11) and Equation (12), the displacement Vector X_i and the feature matrix Φ are normalized by using maximum absolute scaling and L_2 norm respectively to get Y_i and D, so that $Y_i = Dc_i$.

Initialise $c_i = 0$。

for $p = 1, 2, \cdots, p_{\max}$ **do:**

Computed search direction $d^{(p)}: d^{(p)} = s^{(p)} - c_i^{(p)}$,

$$s^{(p)} = \arg\min_{\|s\|_1 \le \alpha} \left\langle \nabla g\left(c_i^{(p)}\right), s \right\rangle.$$

Determine the optimal step size using precise line search:

$$\hat{\gamma}^{(p)} = \frac{\left(q^{(p)}\right)^{\mathrm{T}}\left(Dc_i^{(p)} - Y_i\right)}{\left(q^{(p)}\right)^{\mathrm{T}} q^{(p)}},$$

where $q^{(p)} = D\left(s^{(p)} - c_i^{(p)}\right)$.

Update solution: $c_i^{(p+1)} = c_i^{(p)} + \hat{\gamma}^{(p)}\left(s^{(p)} - c_i^{(p)}\right)$.

if dual gap $h^{(p)} = -\left(d^{(p)}\right)^{\mathrm{T}} \nabla g\left(c_i^{(p)}\right)$ **then**:

break.

end if

end for

Denormalization: $c_i = \dfrac{c_i^{(p)} \max\left(\left|\mathcal{S}X_i\right|\right)}{\left(\mathcal{S}\Phi_{norm}\right)^{\mathrm{T}}}$.

for $j = 1, 2, \cdots, N$ **do**

if $\dfrac{c_{i,j}^2}{\left\|c_i\right\|_2^2} < \tau$ **then**:

$c_{i,j} = 0.$

end if

end for

return Sparse vector c_i.

4.1. Logistic Equation

Consider the following Logistic equation

$$\dot{x}(t) = 1.6x(t) - \left(x(t)\right)^2. \tag{31}$$

This is a common model of population dynamics. We use the ode45 function in MATLAB to generate the simulation data $\mathbf{x}(t)$ of the Logistic equation under

the initial condition $x(0)=8\times10^{-8}$, the time interval $t\in[0,10]$, and the time-step $\Delta t=0.001$. The generated data trajectory is shown by the solid red line in Figure 2. However, to model the practical case, we add the noise to the data via $\tilde{\mathbf{x}}(t)=\mathbf{x}(t)+\eta(t)$, where $\eta(t)\sim N(0,\sigma^2)$, here we choose $\sigma=10^{-5}$.

After obtaining the noisy measurement data $\tilde{\mathbf{x}}(t)$ containing the noise, we can combine the initial condition $x(0)=8\times10^{-8}$ and Equation (8) to get the displacement vector $\mathbf{X}$, and then define the matrix Θ as follows

$$\Theta=\begin{bmatrix} | & | & | & \cdots & | \\ 1 & x & x^2 & \cdots & x^9 \\ | & | & | & \cdots & | \end{bmatrix}.$$

The high maximum degree (9) is intentionally chosen to test the method's ability to suppress irrelevant higher-order terms under noise. By combining Equation (6) and Equation (7), we can construct the feature matrix Φ.

Next, we use the subsampling method proposed in this paper, which combines systematic subsampling and random subsampling, to simultaneously conduct subsampling on the displacement vector $\mathbf{X}$ and the feature matrix Φ. When we perform systematic subsampling, the sampling interval $m_1=100$ and the total sample $M_1=12$ are selected, and the position "1" is selected as the sampling starting point to generate the sampling row index. When conducting random subsampling, the total sample $M_2=30$ is used, and the row index of unduplicated samples was randomly generated. The row index generated during the subsampling process is then applied to the displacement vector $\mathbf{X}$ and the feature matrix Φ to obtain $\mathcal{S}\mathbf{X}$ and $\mathcal{S}\Phi$. The subsampling process is shown in Figure 1.

Then, we apply the maximum absolute value scaling in Equation (11) and the L_2 norm normalization in Equation (12) to normalize the vector $\mathcal{S}\mathbf{X}$ and the matrix $\mathcal{S}\Phi$, respectively, and then get the linear system: $\mathbf{Y}=\mathbf{D}\mathbf{c}$. Finally, we can use the Frank-Wolfe algorithm proposed in this paper to solve the sparse vector $\mathbf{c}$. In the process of applying the Frank-Wolfe algorithm, we set the maximum number of iterations $p_{\max}=40000$, the regularization parameter $\alpha=3$, the threshold value $tol=10^{-8}$, and the parameter $\tau=10^{-4}$. The value $\alpha=3$ was chosen empirically after testing a small set of candidate values to achieve a desirable trade-off between sparse coefficient support and data fitting accuracy.

With the noise intensity $\sigma=10^{-5}$, the identification results for the Logistic equation are shown in Table 1. In addition, we also plotted the data trajectories of the learned governing equation, as shown by the dashed black line in Figure 2.

By comparing the true coefficients with the learned coefficients in Table 1, and analyzing the data trajectory of the exact system without noise and the identified system with noise intensity $\sigma=10^{-5}$ in Figure 2, it can be demonstrated that the Frank-Wolfe algorithm proposed in this paper, which combines subsampling and data normalization, accurately identifies the correct activity items. This also demonstrates the effectiveness of our approach. In addition, we can see from Fig. that the trajectory of the system identified by the method proposed in this paper almost coincides with the real trajectory, which also shows that the method is ro-

bust to the measurement noise. The experimental MSE is 1.4545×10^{-7} and the RMSE is 3.8137×10^{-4}.

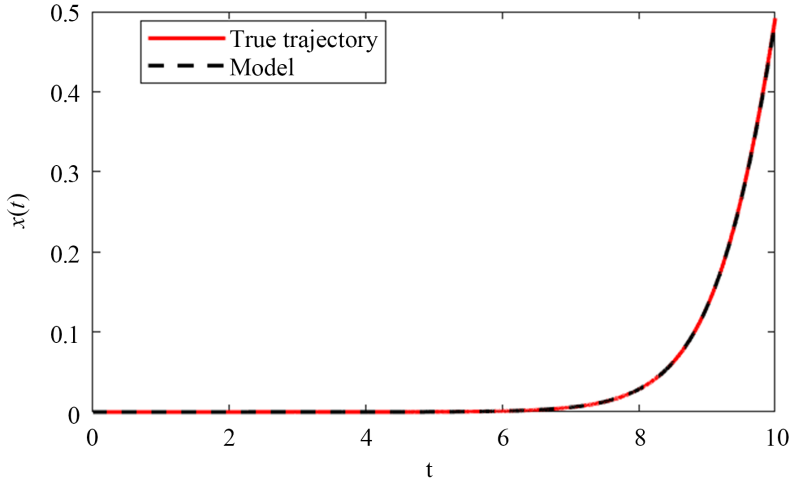


Figure 2. Data trajectories of logistic equation under given conditions. The solid red line represents the true trajectory of the exact system. The dashed black line represents the data trajectory of the sparse identified system under the condition of noise intensity $\sigma = 10^{-5}$.

Table 1. Identification results of Logistic Equation (31). The "Basis" represents the used basis functions. The "True" indicates the true coefficients. The "Learning" represents the parameters identified using the proposed method.

Basis	dx/dt	
	True	Learning
1	0	0
x	1.6	1.6002
x^2	−1	−1.0232
x^3	0	0
⋮	⋮	⋮
x^8	0	0
x^9	0	0

4.2. Lokta-Volterra System

Consider the Lokta-Volterra system

$$\begin{cases} \dot{x}_1(t) = 2/3 x_1(t) - 4/3 x_1(t) x_2(t), \\ \dot{x}_2(t) = -x_2(t) + x_1(t) x_2(t). \end{cases} \tag{32}$$

The Lotka-Volterra system, also known as the predatorprey model, is a mathematical model used to describe the dynamics of biological populations. Under the initial condition $\left[x_1(0), x_2(0)\right] = \left[10^{-5}, 5\times 10^{-4}\right]$, the time interval $t \in [0, 200]$, and the time-step $\Delta t = 0.01$, the simulation data of Lotka-Volterra system is gen-

erated. The generated data trajectory is shown by the solid red line in Figure 3(a). Similarly, in the presence of noise, we set the noise intensity $\sigma = 10^{-5}$.

We calculate the displacement vector $\mathbf{X}_i\ (i = 1, 2)$ according to Equation (8). We construct a polynomial candidate library of total degree 4, including all monomials of the form $x_1^{p_1}, x_2^{p_2}$ with $p_1, p_2 \geq 0$ and $0 \leq p_1 + p_2 \leq 4$. We define the matrix Θ as follows:

$$\Theta = \begin{bmatrix} | & | & | & | & | & | & \cdots & | \\ 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & \cdots & x_2^4 \\ | & | & | & | & | & | & \cdots & | \end{bmatrix}$$

Although the true system (32) contains only linear and bilinear terms $\{x_1, x_2, x_1x_2\}$, the higher-degree monomials are included to test the ability of the Frank-Wolfe algorithm to suppress irrelevant terms and recover the correct sparse support under noisy measurements. Then, according to Equation (6) and Equation (7), the feature matrix Φ is obtained.

After obtaining the displacement vector $X_i\ (i = 1, 2)$ and the feature matrix Φ, we need to subsample for $X_i\ (i = 1, 2)$ and Φ. In this case, we need to subsample the displacement vectors X_1 and X_2 and their corresponding feature matrices Φ_1 and Φ_2 respectively. The feature matrices Φ_1 and Φ_2 are the same here, and we have added subscripts for the convenience of the subsampling process.

For the displacement vector X_1 and the feature matrix Φ_1, we select the sampling interval $m_{11} = 30$, the total sample $M_{11} = 28$, and the position "1" as the sampling starting point to generate the sampling row index. When conducting random subsampling, we select the total sample $M_{12} = 6$ and randomly generate the row index with no duplicate samples. Then the row index generated by the subsampling is applied to both the displacement vector X_1 and the feature matrix Φ_1 to obtain $\mathcal{S}X_1$ and $\mathcal{S}\Phi_1$. For the displacement vector X_2 and the feature matrix Φ_2, we select the sampling interval $m_{21} = 150$, the total sample $M_{21} = 110$, and the position "1" as the sampling starting point to generate the sampling row index. When conducting random subsampling, we select the total sample $M_{22} = 8$ and randomly generate the row index with no duplicate samples. Then the row index generated by the subsampling is applied to both the displacement vector X_2 and the feature matrix Φ_2 to obtain $\mathcal{S}X_2$ and $\mathcal{S}\Phi_2$.

Next, Equation (11) is used to perform maximum absolute value scaling on the vector $\mathcal{S}X_i\ (i = 1, 2)$, and Equation (12) is used to perform L_2 norm normalization on the matrix $Y_i = D_i c_i\ (i = 1, 2)$. Thus, the linear system: $Y_i = D_i c_i\ (i = 1, 2)$ is obtained. Finally, we use the Frank-Wolfe algorithm proposed in this paper to solve the convex optimization problem with L_1 constraints in Equation (16), and then obtain the sparse vectors c_1 and c_2.

In the process of applying the Frank-Wolfe algorithm to solve the sparse vector $c_i\ (i = 1, 2)$, for the sparse vector c_1, we set the maximum number of iterations $p_{1_{\max}} = 100000$ and the regularization parameter $\alpha_1 = 500$. For the sparse vector c_2, we set the maximum number of iterations $p_{2_{\max}} = 10200$ and the regulari-

zation parameter $\alpha_2 = 500$. They share a threshold value $tol = 10^{-8}$ and a parameter $\tau = 10^{-6}$. Both $\alpha_1 = 500$ and $\alpha_2 = 500$ were selected empirically based on several trials, where larger values of α (weaker sparsity) led to overfitting and smaller values (stronger sparsity) eliminated relevant terms. The value 500 gave the best compromise for both equations under the same subsampling setting. The identification results are shown in Table 2.

Additionally, we also plotted the true trajectories of each component of the Lotka-Volterra system and the identified trajectories of each component of the learned governing equations, as shown in Figure 3(b) and Figure 3(c).

By comparing and analyzing the true and estimated coefficients in Table 2, as well as the true and the identified trajectories in Figure 3, we verify the effectiveness of the proposed method for the identification of the Lokta-Volterra system. However, it can be seen from Figure 3(a) that there are still some errors in the identification of the entire system. In addition, we also calculated the MSE and RMSE between the true parameters and the estimated parameters of the Lotka-Volterra system, as shown in Table 3.

Table 2. Identification results of Lokta-Volterra system (32).

Basis	dx/d*t*		dy/d*t*	
	True	Learning	True	Learning
1	0	0	0	0
x	2/3	0.6666	0	0
y	0	0	-1	−1.0001
x^2	0	0	0	0
xy	−4/3	−1.3334	1	1.0009
y^2	0	0	0	0
⋮	⋮	⋮	⋮	⋮
y^4	0	0	0	0

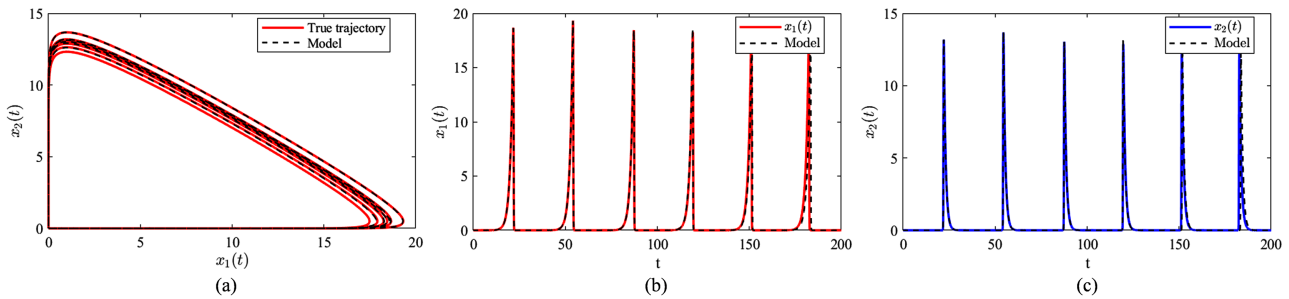


Figure 3. (a) Phase diagram of Lotka-Volterra system. The solid red line represents the true trajectory of the exact system. The dashed black line represents the data trajectory of the sparse identified system under the condition of noise intensity $\sigma = 10^{-5}$. (b) The true trajectory of the state component $x_1(t)$ is represented by the solid red line. The identified trajectory is represented by dashed black line. (c) The true trajectory of the state component $x_2(t)$ is represented by the solid blue line. The identified trajectory is represented by dashed black line.

Table 3. MSE and RMSE between the true parameters and the estimated parameters of the Lotka-Volterra system.

	State component x	State component y
MSE	1.7767	8.8032×10^{-1}
RMSE	1.3329	9.3826×10^{-1}

4.3. Two-Dimensional Damped Oscillator

Consider the two-dimensional damped harmonic oscillator with cubic dynamics

$$\begin{cases} \dot{x}_1(t) = -0.1x_1^3(t) + 2x_2^3(t), \\ \dot{x}_2(t) = -2x_1^3(t) - 0.1x_2^3(t). \end{cases} \tag{33}$$

The two-dimensional damped harmonic oscillator generates the simulation data under the initial condition $[x_1(0), x_2(0)] = [1, 1.8]$, the time interval $t \in [0, 10]$, and the time-step $\Delta t = 0.001$. The data trajectory is shown in Figure 4(a). In the presence of noise, we set the noise intensity $\sigma = 10^{-5}$.

After obtaining the noisy measurement data $\tilde{x}$, we can obtain the displacement vector $X_i (i = 1, 2)$ and the matrix Θ, where we construct a polynomial candidate library of total degree 4, including all monomials in x_1 and x_2 with $0 \le p_1 + p_2 \le 4$. We define the matrix Θ as follows:

$$\Theta = \begin{bmatrix} | & | & | & | & | & | & \cdots & | \\ 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & \cdots & x_2^4 \\ | & | & | & | & | & | & \cdots & | \end{bmatrix}$$

Although the true dynamics (33) contain only the cubic terms x_1^3 and x_2^3, we include all lower- and higher-degree monomials to evaluate whether the Frank-Wolfe algorithm can correctly identify the sparse support and reject the irrelevant terms in the presence of noise. Then we can obtain the feature matrix Φ by Equation (6) and Equation (7).

When subsampling the displacement vector $X_i (i = 1, 2)$ and the feature matrix Φ, we use the same method as in the two examples above. We choose $m_{11} = 100$, $M_{11} = 30$, $M_{12} = 20$ and $m_{21} = 100$, $M_{21} = 30$, $M_{22} = 20$. And then we have the vector $\mathcal{S}X_i (i = 1, 2)$ and the matrix $\mathcal{S}\Phi_i (i = 1, 2)$. Then we use the method proposed in this paper to normalize the vector $\mathcal{S}X_i (i = 1, 2)$ and the matrix $\mathcal{S}\Phi_i (i = 1, 2)$ respectively, and we can get the linear system: $Y_i = D_i c_i (i = 1, 2)$.

Finally, we can use the Frank-Wolfe algorithm to solve the sparse coefficient vector $c_i (i = 1, 2)$. When solving the sparse coefficient vector c_1, we set the maximum number of iterations $p_{1_{\max}} = 3000$ and the regularization parameter $\alpha_1 = 3$. When solving the sparse coefficient vector c_2, we set the maximum number of iterations $p_{2_{\max}} = 26000$ and the regularization parameter $\alpha_2 = 10$. They share a threshold value $tol = 10^{-8}$ and a parameter $\tau = 2 \times 10^{-5}$.

These values were determined by empirical tuning: $\alpha_1 = 3$ and $\alpha_2 = 10$ each produced a sparse coefficient vector whose nonzero entries matched the true support while keeping the residual $\|Dc_i - Y_i\|_2$ low. Different α values for the two

equations reflect their different sensitivities to the ℓ_1 constraint. The identification results are shown in Table 4.

Table 4. Identification results of the two-dimensional damped oscillator (32).

Basis	dx/dt		dy/dt	
	True	Learning	True	Learning
1	0	0	0	0
x	0	0	0	0
y	0	0	0	0
x^2	0	0	0	0
xy	0	0	0	0
y^2	0	0	0	0
x^3	−0.1	−0.1002	−2	−2.0042
x^2y	0	0	0	0
xy^2	0	0	0	0
y^3	2	2.0001	−0.1	−0.1000
x^4	0	0	0	0
⋮	⋮	⋮	⋮	⋮
y^4	0	0	0	0

We also plotted the true trajectories of each component of the two-dimensional damped harmonic oscillator and the identified trajectories of each component of the learned governing equation, as shown in Figure 4(b).

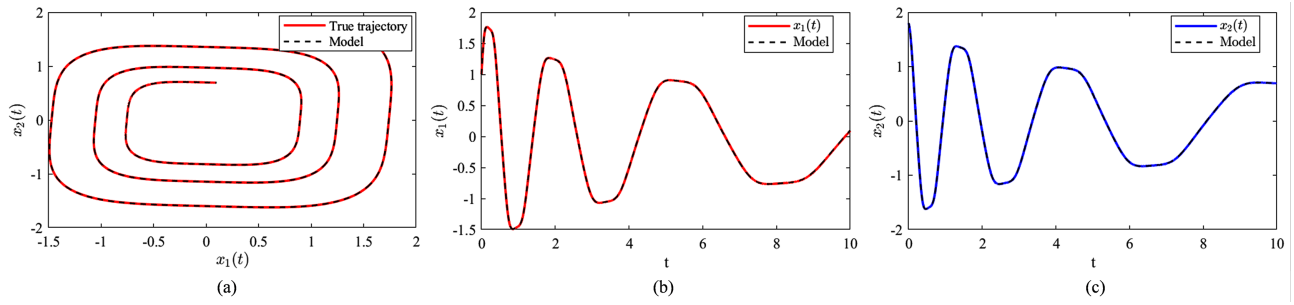


Figure 4. Phase diagram of the two-dimensional damped oscillator and the identified trajectories of state components x and y.

By comparing the true coefficients and learned coefficients in Table 3, and analyzing the true trajectories and identified trajectories in Figure 4(a) and Figure 4(b), it can be effectively demonstrated that the Frank-Wolfe algorithm proposed in this paper, which combines subsampling and data normalization, can correctly identify active items and approximate coefficients. The MSE and RMSE between the true parameters and the estimated parameters are shown in Table 5.

Table 5. MSE and RMSE between the true parameters and the estimated parameters of the two-dimensional damped oscillator.

	State component x	State component y
MSE	1.7299×10^{-5}	1.6199×10^{-5}
RMSE	4.1592×10^{-3}	4.0248×10^{-3}

4.4. Lorenz System

Consider the following Lorenz system

$$\begin{cases} \dot{x}_1(t) = -10x_1(t) + 10x_2(t), \\ \dot{x}_2(t) = 28x_1(t) - x_2(t) - x_1(t)x_3(t), \\ \dot{x}_3(t) = -8/3\,x_3(t) + x_1(t)x_2(t). \end{cases} \tag{34}$$

The Lorenz system is a three-dimensional nonlinear dynamical system that is one of the classic examples of chaos theory, demonstrating complex dynamical behavior. The Lorenz system generates the simulation data under the initial condition $\left[x_1(0), x_2(0), x_3(0)\right] = \left[-152, 150, 300\right]$, the time interval $t \in \left[0, 10\right]$, and the time-step $\Delta t = 0.001$. The data trajectory is shown in **Figure 5**. In the presence of noise, we set the noise intensity $\sigma = 10^{-5}$.

From the noisy measurement data $\tilde{x}$, we can get the displacement vector $X_i\left(i = 1, 2, 3\right)$ and the matrix Θ, where we construct a polynomial candidate library of total degree 3, including all monomials in $\left\{x_1, x_2, x_3\right\}$ with $0 \le p_1 + p_2 + p_3 \le 3$. The matrix Θ is defined as follows:

$$\Theta = \begin{bmatrix} | & | & | & | & | & | & | & \cdots & | \\ 1 & x_1 & x_2 & x_3 & x_1^2 & x_1x_2 & x_1x_3 & \cdots & x_3^3 \\ | & | & | & | & | & | & | & \cdots & | \end{bmatrix}$$

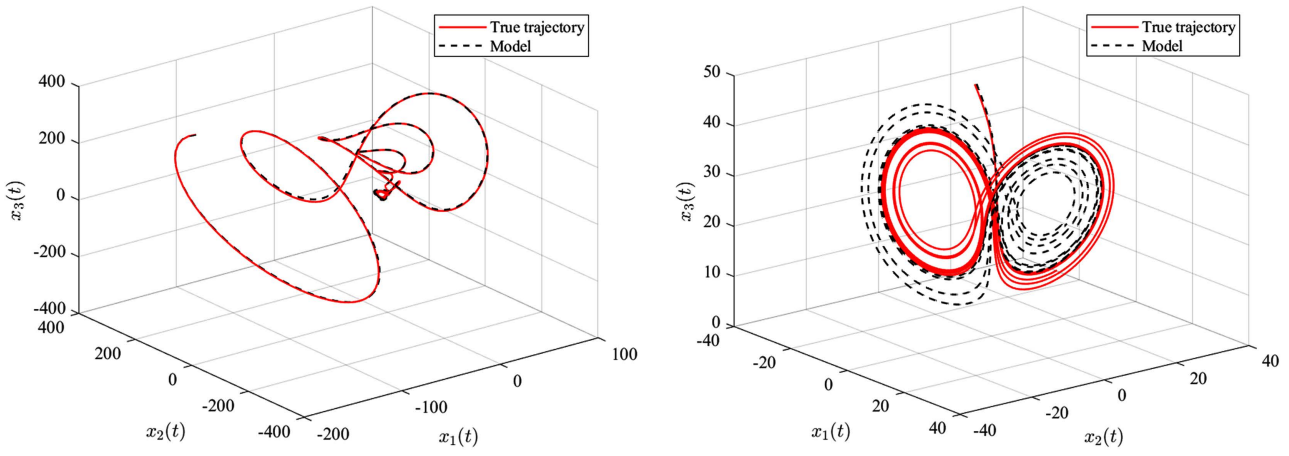
Although the true Lorenz system (34) contains only linear and bilinear terms $\left\{x_1, x_2, x_3, x_1x_2, x_1x_3\right\}$, the higher-degree monomials are included to rigorously test the ability of the Frank-Wolfe algorithm to suppress irrelevant terms and recover the exact sparse support in a chaotic system under noise. Then the feature matrix Φ can be obtained by Equation (6) and Equation (7).

Next, the displacement vector $X_i\left(i = 1, 2, 3\right)$ and the feature matrix Φ are subsampled. In the subsampling process, $m_{11} = 190$, $M_{11} = 20$, $M_{12} = 5$ are selected for the displacement vector X_1 and the feature matrix Φ_1. For the displacement vector X_2 and the feature matrix Φ_2, we choose $m_{21} = 149$, $M_{21} = 52$, $M_{22} = 8$. For the displacement vector X_3 and the feature matrix Φ_3, we choose $m_{31} = 33$, $M_{31} = 6$, $M_{32} = 36$. Then we normalize the vector $\mathcal{S}X_i\left(i = 1, 2, 3\right)$ and the matrix $\mathcal{S}\Phi_i\left(i = 1, 2, 3\right)$ according to Equation (11) and Equation (12), and obtain the linear system: $Y = Dc$. Finally, the sparse coefficient vector $c_i\left(i = 1, 2, 3\right)$ is solved by the Frank-Wolfe algorithm proposed in this paper. When solving the sparse coefficient vector c_1, we set the maximum number of iterations $p_{1_{\max}} = 70000$ and the regularization parameter $\alpha_1 = 3.8$. When solving the sparse coefficient vector c_1, we set the maximum number of iterations $p_{2_{\max}} = 49600$ and the regularization parameter $\alpha_2 = 14$. When solv-

ing the sparse coefficient vector c_3, we set the maximum number of iterations $p_{3_{\max}} = 68000$ and the regularization parameter $\alpha_3 = 9.59$. They share a threshold value $tol = 10^{-8}$ and a parameter $\tau = 10^{-6}$. Each α_i was chosen empirically by trying a set of candidate values and selecting the one that produced the correct sparsity pattern with the smallest fitting error. The distinct values (3.8, 14, 9.59) are necessary because the three equations have different magnitudes and sensitivities to the ℓ_1 constraint. The identification results are shown in Table 6. In addition, we also plotted the true trajectories of each state component of the Lorenz system, as shown in Figure 6. The MSE and RMSE between the true parameters and the estimated parameters are shown in Table 7.

Table 6. Identification results of Lorenz system (34).

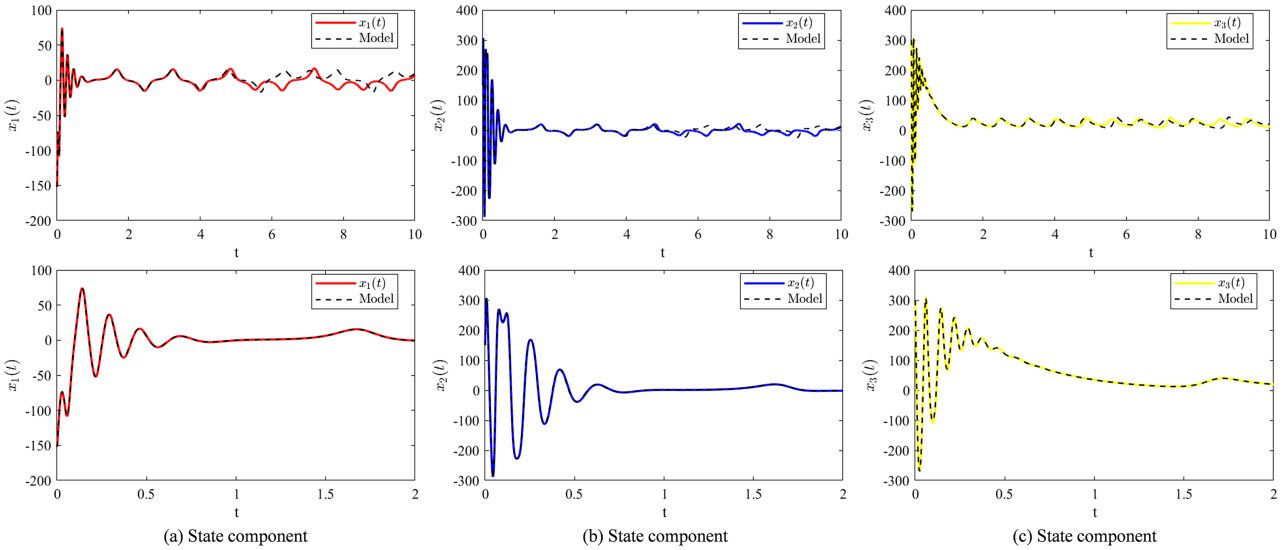
Basis	dx/dt		dy/dt		dz/dt	
	True	Learning	True	Learning	True	Learning
1	0	0	0	0	0	0
x	−10	−10.1408	28	28.0949	0	0
y	10	10.0281	−1	−1.0010	0	0
z	0	0	0	0	−8/3	−2.6615
x^2	0	0	0	0	0	0
xy	0	0	0	0	1	0.9965
xz	0	0	−1	−1.0050	0	0
y^2	0	0	0	0	0	0
yz	0	0	0	0	0	0
z^2	0	0	0	0	0	0
x^3	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
z^3	0	0	0	0	0	0



(a) The data trajectory of the Lorenz system under the condition of noise intensity $\sigma = 10^{-5}$.

(b) Partial trajectory of the Lorenz system within the range $x \in [-40, 40]$, $y \in [-40, 40]$.

Figure 5. Identified trajectories of the Lorenz system.



(a) State component (b) State component (c) State component

Figure 6. Trajectory of state component *x*, *y*, *z*.

Table 7. MSE and RMSE between the true parameters and the estimated parameters of the Lorenz system.

	State component x	State component y	State component z
MSE	6.4427×10^1	9.1607×10^1	6.3988×10^1
RMSE	8.0267	9.5712	7.9992

By comparing the true coefficients with the learned coefficients in **Table 5**, and analyzing the true trajectories and identified trajectories in **Figure 5** and **Figure 6**, we can see that the method proposed in this paper is very effective for the identification of complex Lorenz system. Additionally, it can be seen from **Figure 5(a)** and **Figure 5(b)** that the considered Lorentz system is sensitive to the system parameters. When the identified parameters contain a very small error, the true trajectory of the Lorenz system also exhibits slight deviations.

5. Conclusion

In this work, we proposed a different sparse optimization strategy to learn the parsimonious governing equation from noisy data. The proposed method combines the Frank-Wolfe algorithm with the integral strategy to enhance the robustness in the noisy condition. Additionally, the subsampling is also used to reduce the computational complexity. Several numerical experiments demonstrate that the proposed method can accurately capture the system equations from noisy data.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Bongard, J. and Lipson, H. (2007) Automated Reverse Engineering of Nonlinear Dy-

namical Systems. *Proceedings of the National Academy of Sciences*, **104**, 9943-9948. https://doi.org/10.1073/pnas.0609476104

[2] Schmidt, M. and Lipson, H. (2009) Distilling Free-Form Natural Laws from Experimental Data. *Science*, **324**, 81-85. https://doi.org/10.1126/science.1165893

[3] Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2019) Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *Journal of Computational Physics*, **378**, 686-707. https://doi.org/10.1016/j.jcp.2018.10.045

[4] Cai, S., Mao, Z., Wang, Z., Yin, M. and Karniadakis, G.E. (2021) Physics-Informed Neural Networks (PINNs) for Fluid Mechanics: A Review. *Acta Mechanica Sinica*, **37**, 1727-1738. https://doi.org/10.1007/s10409-021-01148-1

[5] Brunton, S.L., Proctor, J.L. and Kutz, J.N. (2016) Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems. *Proceedings of the National Academy of Sciences*, **113**, 3932-3937. https://doi.org/10.1073/pnas.1517384113

[6] Brunton, S.L., Kutz, J.N. and Proctor, J.L. (2017) Data Driven Discovery of Governing Physical Laws. *SIAM News*, **50**.

[7] Pan, W., Yuan, Y., Gonçalves, J. and Stan, G. (2016) A Sparse Bayesian Approach to the Identification of Nonlinear State-Space Systems. *IEEE Transactions on Automatic Control*, **61**, 182-187. https://doi.org/10.1109/tac.2015.2426291

[8] Rudy, S.H., Brunton, S.L., Proctor, J.L. and Kutz, J.N. (2017) Data-Driven Discovery of Partial Differential Equations. *Science Advances*, **3**, e1602614. https://doi.org/10.1126/sciadv.1602614

[9] Schaeffer, H. (2017) Learning Partial Differential Equations via Data Discovery and Sparse Optimization. *Proceedings of the Royal Society A*: *Mathematical*, *Physical and Engineering Sciences*, **473**, Article 20160446. https://doi.org/10.1098/rspa.2016.0446

[10] Brunton, S.L. and Kutz, J.N. (2019) Methods for Data-Driven Multiscale Model Discovery for Materials. *Journal of Physics*: *Materials*, **2**, Article 044002. https://doi.org/10.1088/2515-7639/ab291e

[11] Yuan, Y., Tang, X., Zhou, W., Pan, W., Li, X., Zhang, H., *et al.* (2019) Data Driven Discovery of Cyber Physical Systems. *Nature Communications*, **10**, Article No. 4894. https://doi.org/10.1038/s41467-019-12490-1

[12] Gao, T. and Yan, G. (2022) Autonomous Inference of Complex Network Dynamics from Incomplete and Noisy Data. *Nature Computational Science*, **2**, 160-168. https://doi.org/10.1038/s43588-022-00217-0

[13] Raissi, M., Yazdani, A. and Karniadakis, G.E. (2020) Hidden Fluid Mechanics: Learning Velocity and Pressure Fields from Flow Visualizations. *Science*, **367**, 1026-1030. https://doi.org/10.1126/science.aaw4741

[14] Chen, Z., Liu, Y. and Sun, H. (2021) Physics-Informed Learning of Governing Equations from Scarce Data. *Nature Communications*, **12**, Article No. 6136. https://doi.org/10.1038/s41467-021-26434-1

[15] Pang, G., Lu, L. and Karniadakis, G.E. (2019) fPINNs: Fractional Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, **41**, A2603-A2626. https://doi.org/10.1137/18m1229845

[16] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljacic, M., Hou, T.Y. and Tegmark, M. (2025) KAN: Kolmogorov-Arnold Networks. *The Thirteenth International Conference on Learning Representations* (*ICLR* 2025), Singapore, 21-23 April 2025, 1-47. https://openreview.net/forum?id=Ozo7qJ5vZi

[17] Li, Z., Kovachki, N.B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. and Anandkumar, A. (2021) Fourier Neural Operator for Parametric Partial Differential Equations. 9*th International Conference on Learning Representations* (*ICLR* 2021), Virtual Event, Austria, 3-7 May 2021, 1-16.

[18] Ljung, L. (1998) System Identification. In: Procházka, A., Uhlíř, J., Rayner, P.W.J. and Kingsbury, N.G., Eds., *Applied and Numerical Harmonic Analysis*, Birkhäuser Boston, 163-173. https://doi.org/10.1007/978-1-4612-1768-8_11

[19] Quade, M., Abel, M., Shafi, K., Niven, R.K. and Noack, B.R. (2016) Prediction of Dynamical Systems by Symbolic Regression. *Physical Review E*, **94**, Article 012214. https://doi.org/10.1103/physreve.94.012214

[20] Reinbold, P.A.K., Kageorge, L.M., Schatz, M.F. and Grigoriev, R.O. (2021) Robust Learning from Noisy, Incomplete, High-Dimensional Experimental Data via Physically Constrained Symbolic Regression. *Nature Communications*, **12**, Article 3219. https://doi.org/10.1038/s41467-021-23479-0

[21] Vaddireddy, H., Rasheed, A., Staples, A.E. and San, O. (2020) Feature Engineering and Symbolic Regression Methods for Detecting Hidden Physics from Sparse Sensor Observation Data. *Physics of Fluids*, **32**, Article 015113. https://doi.org/10.1063/1.5136351

[22] Petersen, B.K., Larma, M.L., Mundhenk, T.N., Santiago, C.P., Kim, S.K. and Kim, J.T. (2021) Deep Symbolic Regression: Recovering Mathematical Expressions from Data via Risk Seeking Policy Gradients. *International Conference on Learning Representations* (*ICLR* 2021), Virtual Event, Austria, 3-7 May 2021. https://www.iclr.cc/virtual/2021/poster/2578

[23] Cornforth, T. and Lipson, H. (2012) Symbolic Regression of Multiple-Time-Scale Dynamical Systems. *Proceedings of the* 14*th Annual Conference on Genetic and Evolutionary Computation*, Philadelphia, 7-11 July 2012, 735-742. https://doi.org/10.1145/2330163.2330266

[24] Podina, L., Darooneh, D., Grewal, J. and Kohandel, M. (2024) Enhancing Symbolic Regression and Universal Physics Informed Neural Networks with Dimensional Analysis. arXiv:2411.15919. https://arxiv.org/abs/2411.15919

[25] Fukami, K., Murata, T., Zhang, K. and Fukagata, K. (2021) Sparse Identification of Nonlinear Dynamics with Low-Dimensionalized Flow Representations. *Journal of Fluid Mechanics*, **926**, A10. https://doi.org/10.1017/jfm.2021.697

[26] Zhang, M., Zeng, F., Yousif, M.Z., Zhu, Z., Lee, J.S. and Lim, H. (2025) Convolutional Autoencoder-Augmented Sparse Identification of Noisy Nonlinear Fluid Dynamics from Unlabelled Data. *Physics of Fluids*, **37**, Article 095177. https://doi.org/10.1063/5.0280421

[27] Mangan, N.M., Brunton, S.L., Proctor, J.L. and Kutz, J.N. (2016) Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics. *IEEE Transactions on Molecular*, *Biological and Multi-Scale Communications*, **2**, 52-63. https://doi.org/10.1109/tmbmc.2016.2633265

[28] Wu, X., McDermott, M. and MacLean, A.L. (2025) Data-Driven Model Discovery and Model Selection for Noisy Biological Systems. *PLOS Computational Biology*, **21**, e1012762. https://doi.org/10.1371/journal.pcbi.1012762

[29] Zhang, L. and Schaeffer, H. (2019) On the Convergence of the Sindy Algorithm. *Multiscale Modeling & Simulation*, **17**, 948-972. https://doi.org/10.1137/18m1189828

[30] Fasel, U., Kutz, J.N., Brunton, B.W. and Brunton, S.L. (2022) Ensemble-SINDy: Robust Sparse Model Discovery in the Low-Data, High-Noise Limit, with Active Learning and Control. *Proceedings of the Royal Society A*: *Mathematical*, *Physical and En-*

gineering Sciences, **478**, Article 20210904. https://doi.org/10.1098/rspa.2021.0904

[31] Kaheman, K., Kutz, J.N. and Brunton, S.L. (2020) Sindy-PI: A Robust Algorithm for Parallel Implicit Sparse Identification of Nonlinear Dynamics. *Proceedings of the Royal Society A*: *Mathematical*, *Physical and Engineering Sciences*, **476**, Article 20200279. https://doi.org/10.1098/rspa.2020.0279

[32] Tran, G. and Ward, R. (2017) Exact Recovery of Chaotic Systems from Highly Corrupted Data. *Multiscale Modeling & Simulation*, **15**, 1108-1129. https://doi.org/10.1137/16m1086637

[33] Mangan, N.M., Kutz, J.N., Brunton, S.L. and Proctor, J.L. (2017) Model Selection for Dynamical Systems via Sparse Regression and Information Criteria. *Proceedings of the Royal Society A*: *Mathematical*, *Physical and Engineering Sciences*, **473**, Article 20170009. https://doi.org/10.1098/rspa.2017.0009

[34] Levitin, E.S. and Polyak, B.T. (1966) Constrained Minimization Methods. *USSR Computational Mathematics and Mathematical Physics*, **6**, 1-50. https://doi.org/10.1016/0041-5553(66)90114-5

[35] Frank, M. and Wolfe, P. (1956) An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, **3**, 95-110. https://doi.org/10.1002/nav.3800030109

[36] Schaeffer, H. and McCalla, S.G. (2017) Sparse Model Selection via Integral Terms. *Physical Review E*, **96**, Article 023302. https://doi.org/10.1103/physreve.96.023302

[37] Ma, C., Huang, C., Cheng, C. and Li, X. (2024) Deterministic-Like Data-Driven Discovery of Stochastic Differential Equations via the Feynman-Kac Formalism. *The European Physical Journal Special Topics*, **234**, 721-741. https://doi.org/10.1140/epjs/s11734-024-01270-8

[38] Braun, G., Carderera, A., Combettes, C.W., Hassani, H., Karbasi, A., Mokhtari, A., and Pokutta, S. (2022) Conditional Gradient Methods. arXiv:2211.14103

[39] Hastie, T., Tibshirani, R. and Friedman, J. (2009) The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd Edition, Springer.

[40] James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013) An Introduction to Statistical Learning: With Applications in R (Springer Texts in Statistics, Vol. 103). Springer.

[41] Tibshirani, R. (1996) Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B*: *Statistical Methodology*, **58**, 267-288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

[42] Balasubramanian, K. and Ghadimi, S. (2018) Zeroth Order (Non) Convex Stochastic Optimization via Conditional Gradient and Gradient Updates. *Advances in Neural Information Processing Systems*, **31**, 3459-3468.