Discover governing differential equations from evolving systems

Yuanyuan Li
Guangzhou Institute of Technology,
Xidian University, Guangzhou 510555, China

Kai Wu*
School of Artificial Intelligence,
Xidian University, Xi'an 710071, China

Jing Liu
Guangzhou Institute of Technology,
Xidian University, Guangzhou 510555

(Dated: January 20, 2023)

Discovering the governing equations of evolving systems from available observations is essential and challenging. However, current methods does not capture the situation that underlying system dynamics can be changed. Evolving systems are changing over time, which invariably changes with system status. Thus, finding the exact change points is critical. We propose an online modeling method capable of handling samples one by one sequentially by modeling streaming data instead of processing the entire dataset. The proposed method performs well in discovering ordinary differential equations, partial differential equations (PDEs), and high-dimensional PDEs from streaming data. The measurement generated from a changed system is distributed dissimilarly to before; hence, the difference can be identified by the proposed method. Our proposal performs well in identifying the change points and discovering governing differential equations in two evolving systems.

I. INTRODUCTION

Research on the information hidden in high-dimensional measurements, which has a transformative impact on the discovery of complex dynamic systems, has attracted the attention of scientific researchers [1–4]. Enabled by the increasing maturity of sensor technology, data storage tools, and the plummeting cost of computational resources, data-driven methods have promoted various innovations in describing high-dimensional measurements generated from experiments [4]. Many canonical dynamic models used to derive the fundamental governing equations of systems from observations are rooted in conservation laws and phenomenological behaviors in physical engineering and biological science [5]. Due to the complexity of dynamic systems and the uncertainty of variables, revealing the underlying governing equations representing the physical laws of the system from time-series data that gives a general description of the spatiotemporal activities is a tremendous challenge [3, 6–12].

In a series of developments, modeling methods for complex systems include empirical dynamic modeling [13], equation-free modeling [14, 15], and modeling emergent behavior [16]. Discovery patterns contain normal form identification [17], nonlinear Laplacian spectral analysis [18], dynamic automatic inference [19], and nonlinear regression [20]. Overall, popular methods for data-driven discovery of complex systems are mainly based on sparse identification [21, 22] and deep neural networks [23, 24], such as sparse identification of nonlinear dynamics (SINDy) [25] and PDE-Net [26]. These existing methods have provided an encouraging performance on static systems motivated by unchanging rules and architectures, where novelty and variability are seen as disruptive [27].

However, given the prevalence of nonstationarity and stochasticity, most complex systems evolve and change over time [28, 29]. The implicit governing equation changes when occurs external attack or mutation of the internal parameter to the system occurs. Moreover, the presence of evolving systems is promising in mathematical physics and engineering sciences [30]. Specifically, apart from applying an adaptive control strategy, inferring the evolution of governing equations over time from observed data is a significant part of real-time dynamic optimization, process monitoring, and nonlinear model predictive control [31]. In terms of many applications in different fields, such as financial markets, social network analysis, and neurological connectivity patterns, they model problems as a complex system of interconnected entities [32–35].

The above algorithms are batch learning methodologies. A common problem when applying these approaches to complex systems is that of ephemeral fitting [13, 36]. They ignore the changes in the system and strike a compromise solution, assuming that the underlying mechanisms are unchanged and deterministic in a

* kwu@xidian.edu.cn

particular sampling period [37]. It can be seen that exploiting existing data-driven approaches to respond to severe changes in evolving systems can be considerably misleading.

We are the first to regard the measurements reflecting the system's state as streaming data, which invariably changes with system status. Facilitated by different governing differential equations before and after system changes, data generated from a changed system is distributed dissimilarly to before, and hence the difference can be identified accordingly. In this article, we present an online algorithm that is capable of selecting a parsimonious model that most accurately represents the massive-scale real-time streaming data, called Online-GED. Our framework handles samples sequentially by modeling subsampled spatiotemporal sequences for high-dimensional real-time streaming measurements instead of processing the entire dataset directly.

Moreover, Online-GED can bypass batch storage processing and intractable cases of combinatorial brute-force search across all possible candidate terms. Mathematically, we conduct experiments with diverse spatiotemporal measurements generated from canonical models under the circumstance of static and time-varying dynamic systems to verify the excellent performance of Online-GED. By rediscovering extensively representative physical system expressions solely from streaming data, the method is demonstrated to work on various canonical instances, including the nonlinear Lorenz system, Burgers' equation, reaction-diffusion equation, etc. Experimental results show that our proposal provides an online technique for real-time online analysis of complex systems. We suggest that Online-GED, which overcomes the limitations mentioned above, is competent for deducting the governing differential equations if sequential measurements of complex dynamic systems are available. With the ability to handle streaming data, Online-GED can ideally cope with the parameter estimation of time-varying nonlinear dynamics and is general enough to detect multiple types of evolutionary patterns.

II. METHODS

With large amounts of data arriving as streams, any offline machine learning algorithm that attempts to store the complete dataset for analysis will fail due to running out of memory. The rise of streaming data raises the technical challenge of a real-time system, which must do any processing or learning that is encouraged by each data record observed in arriving order. Unlike batch processing, in real-time systems, the algorithm must adapt to different stream parts and produce an immediate output before seeing the next input. Let u denote the state vector of a real-time system at time t, and u_t represents its partial differentiation in the time domain. The system receives streaming inputs:

$$\cdots, u_{t-3}, u_{t-2}, u_{t-1}, u_t, u_{t+1}, u_{t+2}, u_{t+3}, \cdots$$

For example, vector u represents the position information of the particle shown in Fig. 1(a). Every step, the particle moves along the trajectory; the data will be captured by sensors and sent to the system that needs to analyze or respond in time. Instead of storing the entire stream, continuous online learning faces one input u_t at a time.

The rigorous model of a complex system is always a set of differential equations with unknown physics parameters [38]. Without losing generality, we consider a general parameterized physical system of the following form:

$$u_t = N\left(1, u, u^2, \ldots, u_x, uu_x, \ldots, u_{xx}, \ldots\right), \tag{1}$$

where the terms with subscript x represent partial differentiation of u in the space domain, "1" denotes the constant, and $N(\bullet)$ is an unknown combination of the nonlinear functions, partial derivatives, constant, and additional terms of $u(x,t)$. Our aim is to select the correct terms that are most relevant to dynamic information in the streaming data environment. In view of measurements of all considered state variables U, the right-hand side of Eq. (1) can be expressed by multiplying the library function matrix $\Theta(U)$ with the coefficient matrix Ξ as follows:

$$U_t = \Theta(U)\,\Xi, \tag{2}$$

$$\Theta(U) = \left[\theta_1(U),\ \theta_2(U),\ \ \ldots,\ \theta_p(U)\right]. \tag{3}$$

Columns of $\Theta(U)$, $\theta_s(U)\,(s = 1, 2, \ldots, p)$, correspond to p specific candidate terms for the governing equation, as shown in Fig. 1(b).

Online-GED is divided into two steps: (i) build complete libraries of candidate terms; (ii) update the structure of governing equations via the FTRL-Proximal style methodology. Each procedure is described as follows.

A. Build a candidate library

As a pre-processing step, we start by collecting the spatiotemporal series data at m time points and n spatial locations of the state variables. For each state variable, the captured state measurements are represented as a single column state vector $u \in C^{nm\times 1}$. Based on all the observables, e.g., variables x, y, and z in Lorenz systems, a series of functional terms associated with these quantities can be calculated and then reshaped into a single column as well, such as x^2, xy, xyz, x^2y, etc. Likewise, partial differential terms should be considered in the candidate library if PDEs govern the dynamics. Furthermore, "1" denotes the constant term that possibly appears in equations. The compositive function library $\Theta(U) \in C^{nm\times p}$ is a matrix that contains p designed functional terms. In terms of derivation, second-order finite differences [39] are devoted to the clean data from numerical simulations, while the easiest to implement and most reliable method for the noisy data is a polynomial interpolation [40]. Note that the computed time derivative $u_t \in C^{nm\times 1}$ is also a single-column vector presented on the left-hand side of

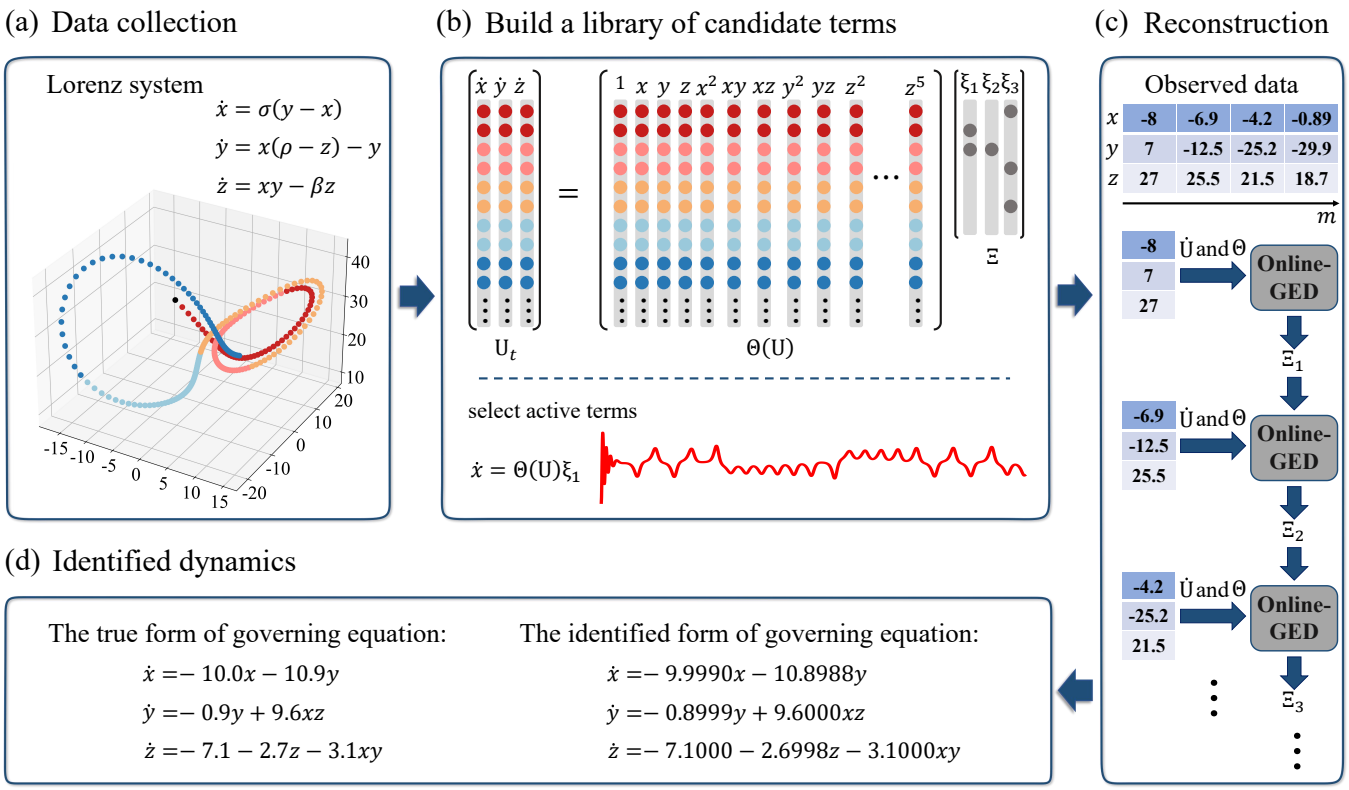

FIG. 1. Steps in Online-GED to address the evolving system discovery problem applied to infer the governing equation of the chaotic Lorenz system from streaming data. (a) The particle's trajectory of motion. Starting from the black point, the particle moves in one position after each time step. The points of different colors represent the trajectories in varying periods that are used as a whole by batch methods. However, the data record is collected as a stream from the chaotic Lorentz system in real time. (b) As the observations arrive one by one, take numerical derivatives of the current state vector and construct a library matrix Θ incorporating the candidate terms for governing differential equations. (c) Update the structure of discovered governing differential equations leveraging Online-GED considering one instance each time. (d) The discovered dynamical expression of the Lorenz system.

Eq. (2). The constant term in the library Θ sufficiently considers the bias term in the governing differential equation so that the model can be regarded as an unbiased representation of dynamics. For Online-GED, an important prerequisite for revealing the correct governing differential equation is that the candidate function library we constructed contains all the members that constitute the concise dynamics expression so as to ensure that the exact sparse coefficient Ξ is computed under iterations.

B. Optimization

Formulating a hypothesis that the governing differential equations are evolving at any moment, we model each example as a dynamic process to simulate the arrival of streaming data. Given a set of time-series state measurements at a fixed number of spatial locations in x, the goal is to construct the form of $N(\bullet)$ online. To fix this issue, the core of our innovation is utilizing real-time streaming data to denote the loss function. Considering a coefficient vector in Ξ, $\xi_j (j = 1, 2, \ldots, d)$, which corresponds to the specific state variable u, the fitness function is designed as follows:

$$\min_{\xi_j} = \sum_{i=1}^{mn} L_i(\xi_j) + \lambda_1 ||\xi_j||_1, \tag{4}$$

where $\lambda_1 \geq 0$ denotes the regularization coefficient, n is the number of positions, m denotes the length of the time series, $\xi_j = [\xi_{j1}, \xi_{j2}, \ldots, \xi_{jp}]^T$, and $L_i(\xi_j)$ is defined as follows:

$$L_i(\xi_j) = \frac{(\Theta_i \xi_j - \dot{U}_i)^2}{2}, \tag{5}$$

where $\dot{U}_i$ is the time derivative of the ith state observation, Θ_i is the ith row of the common library Θ that represents all candidate function values for the ith data. The sparsity constraints mean that the coefficient vector ξ_j is sparse with only a few non-zero entries, each representing a subsistent item in the function library. Subsequently, we exploit the follow-the-regularized-leader

(FTRL)-Proximal [41–43] style methodology to optimize the outcome by considering solely one instance each time.

Leveraging the fact that each instance is considered individually, we rewrite the loss function and define a single loss term (see Eq. (5)). For example, if the first state measurement u_1 is available, the loss is defined as: $L_1(\xi_j) + \lambda_1\|\xi_j\|_1 = 0.5(\Theta_1\xi_j - \dot{U}_1)^2 + \lambda_1\|\xi_j\|_1$. Next, if the second instance of u_2 arrives, the loss is defined as: $L_1(\xi_j)+L_2(\xi_j)+\lambda_1\|\xi_j\|_1 = 0.5(\Theta_1\xi_j-\dot{U}_1)^2+0.5(\Theta_2\xi_j-\dot{U}_2)^2 + \lambda_1\|\xi_j\|_1$. In this way, the computer only needs to store information about a single example, thus relieving memory stress. On the ith sample, the gradient is calculated as follows:

$$\nabla L_i(\xi_j) = (\Theta_i\xi_j - \dot{U}_i)\Theta_i. \tag{6}$$

Normally, the online gradient descent algorithm can be used to update the coefficient vector ξ_j^i after the arrival of the ith data by using:

$$\xi_j^{i+1} = \xi_j^i + C\nabla L_{_i}(\xi_j). \tag{7}$$

However, this method has been proven to lack general applicability [42]. Correspondingly, the FTRL-proximal style approach is introduced to solve the online problem. We use the following equation to update the coefficient vector ξ_j^i:

$$\begin{aligned}\xi_j^{i+1} =& arg\min_{\xi_j}\left(g_{1:i} - \sum_{k=1}^{i}\sigma_k\xi_j^k\right)\xi_j + \lambda_1\|\xi_j\|_1 \\ &+0.5\left(\lambda_2 + \sum_{k=1}^{i}\sigma_k\right)\|\xi_j\|_2^2 + 0.5\sum_{k=1}^{i}\sigma_k\|\xi_j^k\|_2^2,\end{aligned} \tag{8}$$

where λ_1 and λ_2 both denote the regularization coefficient, which is the positive constant, while σ_k is on the ith learning rate η_i aspect, defined by:

$$\sum_{k=1}^{i}\sigma_k = \sqrt{t} = \eta_i^{-1}. \tag{9}$$

Given the gradient $\nabla L_i(\xi_j)$ a shorthand g_i, we set $g_{1:i} = \sum_{k=1}^{i} g_k$. Then the update in Eq. (8) can be rewritten as follows:

$$\begin{aligned}\xi_j^{i+1} =& arg\min_{\xi_j}\left(g_{1:i} - \sum_{k=1}^{i}\sigma_k\xi_j^k\right)\xi_j + \lambda_1\|\xi_j\|_1 \\ &+0.5\left(\lambda_2 + \sum_{k=1}^{i}\sigma_k\right)\|\xi_j\|_2^2 + 0.5\sum_{k=1}^{i}\sigma_k\|\xi_j^k\|_2^2,\end{aligned} \tag{10}$$

where the last term $0.5\sum_{k=1}^{i}\sigma_k\|\xi_j^k\|_2^2$ is a constant with respect to ξ_j and negligibly impacts the update process. Let $Z_i = g_{1:i} - \sum_{k=1}^{i}\sigma_k\xi_j^k = Z_{i-1} + g_i - \sigma_i\xi_j^i$, and we ignore the last term and rewrite Eq. (10) as:

$$arg\min_{\xi_j}\left(Z_i\xi_j + \lambda_1\|\xi_j\|_1 + 0.5\left(\lambda_2 + \sum_{k=1}^{i}\sigma_k\right)\|\xi_j\|_2^2\right). \tag{11}$$

Let ξ_{js} and $Z_{i,s}(s = 1, 2, \ldots, p)$ represent the sth element of the vector ξ_j and Z_i, respectively. Equation (9) is generalized as

$$\begin{aligned}\xi_{js}^{i+1} =& arg\min_{\xi_{js}} Z_{i,s}\xi_{js} + \lambda_1|\xi_{js}| \\ &+0.5\left(\lambda_2 + \sum_{k=1}^{i}\sigma_k\right)\xi_{js}^2.\end{aligned} \tag{12}$$

Aiming to simplify the expression, we suppose w^* to be the optimal solution of ξ_{js}^i and $\Phi \in \nabla w^*$ to be its gradient. Then, Eq. (13) is satisfied. Equation (14) shows the subdifferential of $w = sgn(w)$.

$$Z_{i,s} + \lambda_1\Phi + (\lambda_2 + \sum_{k=1}^{i}\sigma_k)w^* = 0. \tag{13}$$

$$sgn(w) = \begin{cases} \Phi \in R| -1 \le \Phi \le 1, if \quad w = 0, \\ 1 \quad if \quad w > 0 \\ -1, \quad if \quad w < 0 \end{cases} \tag{14}$$

The solution to w^* can be discussed in three cases [44]:

1. If $|Z_{i,s}| \le \lambda_1$,
   (a) If $w^* = 0$, then $sgn(0) \in (-1, 1)$ and Eq. (13) is satisfied.
   (b) If $w^* > 0$, then $Z_{i,s} + \lambda_1 sgn(w^*) = Z_{i,s} + lambda_1 \ge 0$ and $(\lambda_2 + \sum_{k=1}^{i}\sigma_k)w^* > 0$. Equation (13) does not hold.
   (c) If $w^* < 0$, then $Z_{i,s}+\lambda_1 sgn(w^*) = Z_{i,s}-\lambda_1 \le 0$ and $(\lambda_2 + \sum_{k=1}^{i}\sigma_k)w^* < 0$. Equation (13) does not hold.
2. If $|Z_{i,s}| < -\lambda_1$,
   (a) If $w^* = 0$, then $sgn(0) \in (-1, 1)$ and Eq. (13) does not hold.
   (b) If $w^* > 0$, then $Z_{i,s}+\lambda_1 sgn(w^*) = Z_{i,s}+\lambda_1 < 0$ and $(\lambda_2 + \sum_{k=1}^{i}\sigma_k)w^* > 0$. Equation (13) holds and the solution is:

$$w^* = -\left(\lambda_2 + \sum_{k=1}^{i}\sigma_k\right)^{-1}(Z_{i,s} + \lambda_1). \tag{15}$$

   (c) If $w^* < 0$, then $Z_{i,s}+\lambda_1 sgn(w^*) = Z_{i,s}-\lambda_1 < 0$ and $(\lambda_2 + \sum_{k=1}^{i}\sigma_k)w^* < 0$. Equation (13) does not hold.
3. If $|Z_{i,s}| > \lambda_1$,
   (a) If $w^* = 0$, then $sgn(0) \in (-1, 1)$ and Eq. (13) does not hold.
   (b) If $w^* > 0$, then $Z_{i,s}+\lambda_1 sgn(w^*) = Z_{i,s}+\lambda_1 > 0$ and $(\lambda_2 + \sum_{k=1}^{i}\sigma_k)w^* > 0$. Equation (13) does not hold.

Algorithm 1: Online-GED

Input: α, β, λ_1, λ_2: the parameters of Online-GED;
thresh: the threshold for truncation;
epo: the iterations for reusing the finite data;
Θ, u_t: the pre-processed training data;
Output: ξ_j;

Set N as the length of training data and p as the column number in Θ;
for *each term s* ***in*** *1: 1: p* **do**
  Set Z_s=0, η_s=0, and ξ_{js}=0;
end
for *each epoch* ***in*** *1: 1: epo* **do**
  for *each instance i* ***in*** *1: 1: N* **do**
    Receive instance pair $(\Theta_i, \dot{U}_i)$ and set $I = \{k \ |\Theta_{ik} \neq 0\}$;
    for *each element k* ***in*** *I* **do**
      Compute ξ^i_{jk} using Eq. (20);
      for *each element k* ***in*** *I* **do**
        $g = (\Theta_i \xi^i_j - \dot{U}_i)\Theta_{ik}$;
        $\sigma = (\sqrt{\eta_k + g^2} - \sqrt{\eta_k})/\alpha$;
        $Z_k = Z_k + g - \sigma\xi^i_{jk}$;
        $\eta_k = \eta_k + g^2$;
      end
    end
  end
end
for *each term s* ***in*** *1: 1: p* **do**
  Set $\xi_{js} = 0$ if $|\xi_{js}| < thresh$;
end

(c) If $w^* < 0$, then $Z_{i,s} + \lambda_1 sgn(w^*) = Z_{i,s} - \lambda_1 > 0$ and $(\lambda_2 + \sum_{k=1}^{i} \sigma_k) w^* < 0$. Equation (13) holds and the solution is:

$$w^* = -\left(\lambda_2 + \sum_{k=1}^{i} \sigma_k\right)^{-1} (Z_{i,s} - \lambda_1). \tag{16}$$

The above discussion in different estimation scenarios is summarized in Eq. (17), where $w^* = \xi^i_{js}$.

$$w^* = \begin{cases} 0, & |Z_{i,s}| \leq \lambda_1 \\ -\frac{(Z_{i,s} - \lambda_1 sgn(Z_{i,s}))}{\left(\lambda_2 + \sum_{k=1}^{i} \sigma_k\right)}, & otherwise \end{cases} \tag{17}$$

Moreover, we set a remarkable learning rate

$$\eta_{is} = \frac{\alpha}{(\beta + \sqrt{\sum_{k=1}^{i} (g_{k,s})^2})}, \tag{18}$$

where α and β are both positive constants, $g_{k,s}$ denotes the sth element of the gradient g_k and η_{is} is the learning rate of $g_{k,s}$. Thus, Eq. (9) can be rewritten as follows:

$$\sum_{k=1}^{i} \sigma_k = \frac{1}{\eta_{is}} = \frac{(\beta + \sqrt{\sum_{k=1}^{i} (g_{k,s})^2})}{\alpha}. \tag{19}$$

The final form of the closed solution is obtained and expressed as

$$\xi^i_{js} = \begin{cases} 0, & |Z_{i,s}| \leq \lambda_1 \\ \frac{(\lambda_1 sgn(Z_{i,s}) - Z_{i,s})}{(\lambda_2 + \frac{(\beta + \sqrt{\sum_{k=1}^{i} (g_{k,s})^2})}{\alpha})}, & otherwise, \end{cases} \tag{20}$$

where α and β are both positive constants, $g_{k,s}$ denotes the sth element of the gradient g_k.

Notably, ξ_{js} is always updated according to the current input instance, thereby guiding the detection of change points if there exists any variation in the evolving system. Additionally, the mechanism frees the computer from storing the whole large-scale data. It should be highlighted that the method suggested above is entirely different from the batch learning methods, which update gradients or individuals depending on all available measurements. Online-GED [45] updates gradients based on the current loss $L_i(\xi_j)(i = 1, 2, \ldots, mn)$ in each iteration. This online inference method takes advantage of a single available instance.

Online-GED can be easily extended to handle evolving systems. Specifically, the change point arises when the system is disturbed, and then the original distribution of generated data varies. Therefore, the corresponding loss value gradually decreases and stabilizes unless the change point appears along with a sudden increase in the loss curve. In this context, the coefficients in the former governing equation and the change location are simultaneously outputted according to the sudden increase of loss value. Meanwhile, we initialize the parameters of Online-GED, and then the changed governing equation is identified after a few iterations.

III. RESULTS

A. Discovering Single System from Streaming Data

1. *Example – Discovering Chaotic Lorenz System*

The online algorithmic procedure for identifying the correct governing differential equations of the chaotic Lorenz system from a given set of measurements is demonstrated in Fig. 1. The mathematical structure of the model discovery approach combines data collection, a library of potential candidate functions, and the online reconstruction method. The chaotic Lorenz system is highly nonlinear to be a canonical test case for model discovery. According to the following equations, we construct a high-dimensional dynamic instance.

$$\dot{x} = \sigma(y - x), \tag{21}$$

$$\dot{y} = x(\rho - z) - y, \tag{22}$$

$$\dot{z} = xy - \beta z. \tag{23}$$

To create a dataset that indicates the true nonlinear combination of Lorenz variables, the common set of parameters is $\sigma = 10$, β=8/3, and ρ=28, with the initial condition of $(x_0, y_0, z_0)^T = (-8, 7, 27)^T$. Ref. [3] has

TABLE I. Summary of online identification results for a wide range of canonical models. Online-GED is applied to reconstruct the correct model structure in each example. The discretization setting for the spatial and temporal sampling of the numerical simulation data is given, along with the relative L_2 error in recovering the parameters of these dynamical models for no noise.

Dynamic System	Governing Equation	Identified Equation	Error	Discretization
Damped harmonic oscillator 1	$\dot{x} = -0.1x + 2y$, $\dot{y} = -2x - 0.1y$	$\dot{x} = -0.1x + 2y$, $\dot{y} = -2x - 0.1y$	1.62e-05	$[x_0, y_0] = [2, 0]$, $m = 2500$
Damped harmonic oscillator 2	$\dot{x} = -0.1x^3 + 2y^3$, $\dot{y} = -2x^3 - 0.1y^3$	$\dot{x} = -0.0998x^3 + 2.0014y^3$, $\dot{y} = -2.0011x^3 - 0.1001y^3$	6.28e-04	$[x_0, y_0] = [2, 0]$, $m = 2500$
Lorenz system v1	$\dot{x} = \sigma(y - x)$, $\dot{y} = x(\rho - z) - y$, $\dot{z} = xy - \beta z$	$\dot{x} = -9.9990x - 10.8988y$, $\dot{y} = -0.8999y + 9.6xz$, $\dot{z} = -7.1 - 2.6998z - 3.1xy$	2.80e-04	$[x_0, y_0, z_0] = [-8, 7, 27]$ $m = 10000$
Lorenz system v2	$\dot{x} = 5.7z - 3.5xy + 2.1yz^2$, $\dot{y} = -10.3 - 2.7y + 2.6x^2 - z^2$, $\dot{z} = -10.9z - 5.6xy + 3.4yz$	$\dot{x} = 5.7004z - 3.5007xy + 2.1001yz$, $\dot{y} = -10.3001 - 2.6998y + 2.5999x^2 - z^2$ $\dot{z} = -10.8828z - 5.5957xy + 3.3992yz$	2.70e-04	$[x_0, y_0, z_0] = [-8, 7, 27]$ $m = 10000$
Hopf normal form	$\dot{x} = \mu x - \omega y - Ax\left(x^2 + y^2\right)$, $\dot{y} = \omega x + \mu y - Ay(x^2 + y^2)$	$\dot{x} = 0.2514x - 0.9995y - 1.0060x^3 - 1.0009xy^2$, $\dot{y} = 1.0006x + 0.2606y - 1.0422x^2y - 1.0392y^3$	1.85e-02	$[x_0, y_0] = [2, 0]$ $m = 3000, \mu = 0.25, \omega = 1, A = 1$
Diffusion from random walk	$u_t = 0.5u_{xx}$	$u_t = 0.500008u_{xx}$	1.69e-05	$t \in [0, 0.02]$, $m = 3, n = 8000$
Burgers'	$u_t = -uu_x + 0.1u_{xx}$	$u_t = -0.9993uu_x + 0.1002u_{xx}$	6.97e-04	$t \in [0, 10], m = 101$ $x \in [-8, 8], n = 256$
Korteweg-de Vries (KdV)	$u_t = -6uu_x - u_{xxx}$	$u_t = -6.1317uu_x - 1.0029u_{xxx}$	2.17e-02	$t \in [0.05, 0.175], m = 6$, $x \in [-10, 12], n = 256$
Kuramoto-Sivashinsky	$u_t = -uu_x - u_{xx} - u_{xxxx}$	$u_t = -0.9667uu_x - 0.9585u_{xx} - 0.9646u_{xxxx}$	2.55e-02	$t \in [0, 100]$, $m = 251$ $x \in [0, 100]$, $n = 1024$
Reaction-diffusion	Eq. (24)	$u_t = 0.1u_{xx} + 0.1001u_{yy} - 1.0001uv^2 - 1.0001u^3 + 0.9996v^3 + 0.9996u^2v + 1.0001u$	2.65e-04	$t \in [0, 10], m = 201$, *subsample* 0.285
	Eq. (B2)	$v_t = 0.1001v_{xx} + 0.1001v_{yy} + 1.0001v - 0.9996uv^2 - 0.9996u^3 - 1.0001v^3 - 1.0001u^2v$		$x, y \in [-10, 10], n = 512$

employed the SINDy autoencoder to discover a parsimonious model with only seven active terms that seems to mismatch the original Lorenz system. Interestingly, the dynamics of the resulting model exhibit an attractor with a 2-lobe structure, which is qualitatively consistent with the true Lorentz attractor. Then the sparsity pattern can be rewritten in the standard form by choosing an appropriate variable transformation. Remarkably, the ability to capture a concise model that describes dynamics on the attractor is most important in the Lorenz system.

Given time intervals and initial conditions, we simulate the Lorenz system within a certain time horizon. Specifically, state measurements are collected under the constraints of the transformed expressions, including historical data of the state U and its time derivative U_t. The combination of spatial and temporal modes results in the trajectory of dynamics shown in Fig. 1(a). A library of potential candidate functions, $\Theta(U)$, is constructed to find the least terms required to satisfy $U_t = \Theta(U)\,\Xi$. Candidate functions can be polynomial functions, trigonometric functions, exponential functions, logarithmic functions, partial derivatives, constant items, and any additional terms about U. Using our proposed online technique, the coefficient $\Xi = [\xi_1; \xi_2; \ldots; \xi_d]$, can be identified to determine active terms of the dynamics. To elude the obstacle of large-scale data storage and batch gradient update, we design a cumulative loss function that solely considers the information of one instance each time so as to reflect the mode of arriving streaming data. The components and coefficients of the obtained governing differential equation are updated promptly after each data arrival. Consequently, we can subtly detect moderate or drastic variations in the system through changes in the resulting model. The experimental result in Fig. 1(d) implies that Online-GED can accurately reproduce the attractor dynamics from chaotic trajectory measurements. The pre-established sparse scenario guarantees that the resulting solution obtained by the online method can effectively balance model complexity with description ability to avoid overfitting, thereby promoting its interpretability and extensibility.

Due to the chaotic nature and the sensitivity to initial conditions, we test two more general Lorenz systems in our experiments , and the expressions are shown in Table I. In both cases, we conducted 10000 timesteps with time intervals dt=0.01 and the above initial condition.

Experimental results imply that Online-GED can accurately reproduce the dynamics of the Lorenz attractor from chaotic trajectory measurements.

2. Performance on Discovery for Canonical Models

We apply Online-GED to ten canonical models from mathematical physics and engineering sciences, and the results are shown in Table I. The selected physical systems contain dynamics along with periodic to chaotic behavior, ranging from linear to strongly nonlinear systems. The discretization settings for spatial and temporal sampling and the error in recovering the parameters in the dynamical model structure are detailed in Table I. Encouragingly, Online-GED can recover every physical system, even those with significant spatial subsampling. The notable results highlight the broad applicability of the method and the success of this online technique in discovering governing differential equations. Remarkably, the capacity for capturing nontrivial active terms, in particular, has essential explanatory implications for model discovery.

B. Discovering Evolving System

The first example considers an evolving two-dimensional damped harmonic oscillator that changes from cubic to linear.

$$\frac{d\begin{bmatrix}x\\y\end{bmatrix}}{dt} = \begin{bmatrix}-0.1 & 2\\ -2 & -0.1\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}, \tag{24}$$

$$\frac{d\begin{bmatrix}x\\y\end{bmatrix}}{dt} = \begin{bmatrix}-0.1 & 2\\ -2 & -0.1\end{bmatrix}\begin{bmatrix}x^3\\y^3\end{bmatrix}. \tag{25}$$

A particle in simple harmonic vibration is known as a harmonic oscillator, whose motion is the simplest ideal vibration model. Fig. 2 illustrates the dynamic data and the detection of change points. Firstly, we create the solution to Eq. (25), U_1, with 2,500 timesteps and the initial data $[2, 0]$. Then, we utilize the last instance in U_1 as the initial data and create the solution to Eq. (24) with 2,500 timesteps as well, recorded as U_2. The changing data is illustrated in Fig. 2. Respectively, we test the detection function in two scenarios: (i) U_1 and U_2 successively appear once (scenario 1). (ii) U_1 repeats 128 times to simulate the running system, and then U_2 appears and repeats 128 times as well (scenario 2). Specifically, the change point arises when the system changes, and then the original distribution of generated data varies. Therefore, the corresponding loss value gradually decreases and stabilizes unless the change point appears along with a sudden increase in the loss curve. In this context, the coefficients in the former governing equation and the change location are simultaneously outputted according to the sudden increase of loss value.

Meanwhile, we initialize the parameters of Online-GED, and then the changed governing equation is identified after a few iterations. With an augmented nonlinear library including polynomials up to the fifth order, the change point is spotted in both evolving circumstances, while the correct form of the nonlinearity can solely be obtained by reusing the finite data. It can be noticed that the loss curve tends to stabilize faster after resetting the training coefficient, thereby accelerating the convergence of Online-GED because the former result may mislead the training process. With an augmented nonlinear library including polynomials up to the fifth order, the correct form of the nonlinearity and the change point are obtained in evolving circumstances.

We also simulate the transition from the KdV to the Kuramoto-Sivashinsky equation as an illustrative example that exhibits qualitatively different dynamics as the system evolves. Fig. 3 illustrates the evolution plotted in space-time. The summary of Online-GED and the state-of-the-art methods for identifying the evolving systems in Table II demonstrates that our approach works successfully for a high-dimensional system with time-varying parameters. Most data-driven approaches failed to model for the discovery of governing differential equations from evolving systems with streaming data. They regard batch measurements as a whole and update gradients integrally to obtain an ephemeral fitting. Owing to the ignorance of changes in the system, existing methodologies are biased to strike a compromise solution, so that strenuous to identify system changes from consecutively generated data. Quite the contrary, Online-GED succeeds in simultaneously estimating forms and identifying changes in the governing equation.

IV. CONCLUSION AND DISCUSSION

In summary, we have proposed an online modeling method, Online-GED, capable of finding governing differential equations of evolving systems. However, existing algorithms treat measurements as a whole and ignore the system variations, thereby striking a compromise solution and failing to model for identifying system changes. To our knowledge, our proposal is the first method to find governing differential equations of evolving systems. Online-GED handles samples one by one sequentially by modeling streaming data instead of processing the entire dataset directly. Moreover, the change point can be identified by the dissimilar distribution of generated data from a changed system. We demonstrate that Online-GED works on various canonical instances. Additionally, we show the performance of simultaneously estimating forms and identifying changes in the governing equation via two time-varying dynamic systems. Results in Table I show that the inference of governing equation is accurate when utilizing Online-GED on clean data from numerical

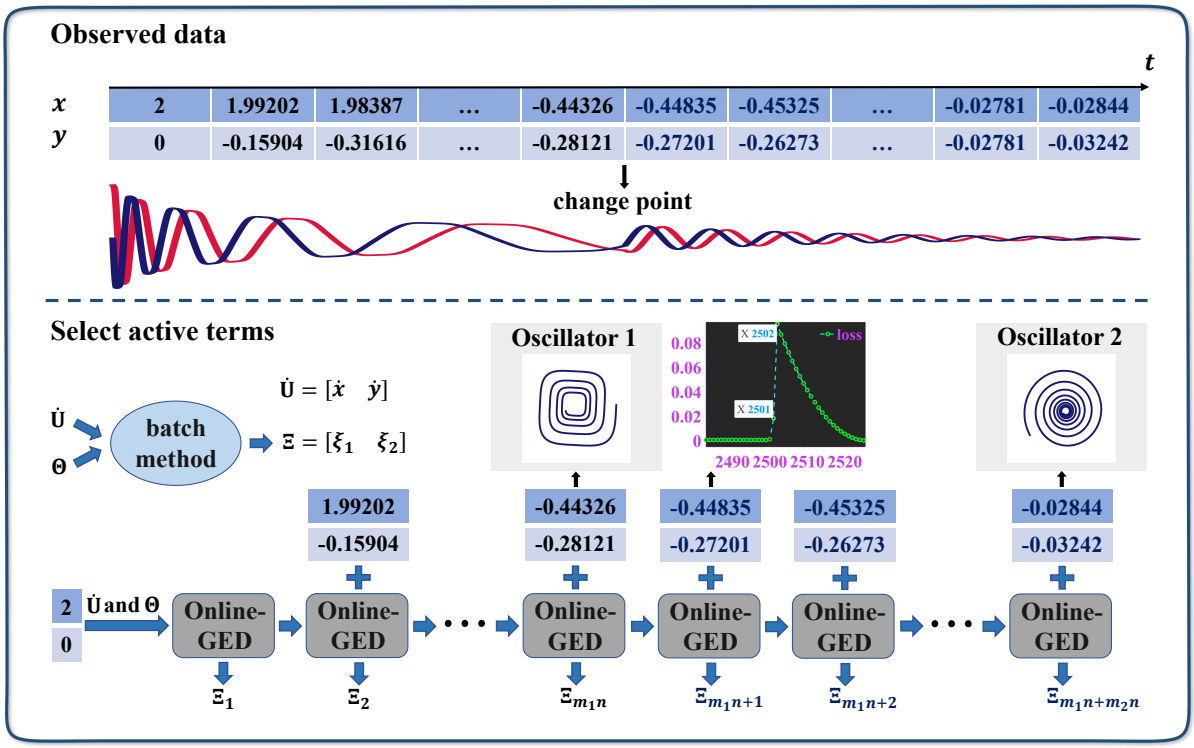

FIG. 2. Difference between batch learning methods and the proposed Online-GED in reconstructing governing equations from evolving systems. As instances successively arrive, the change point arises when the system is disturbed, and the original distribution of generated data varies. The batch method can only strike a compromise solution utilizing all data generated by the two completely different systems. By means of Online-GED, the loss value gradually decreases and stabilizes unless the change point appears along with a sudden increase in the loss curve. The former governing equation is discovered right before the change point, and the latter is identified after a few iterations.

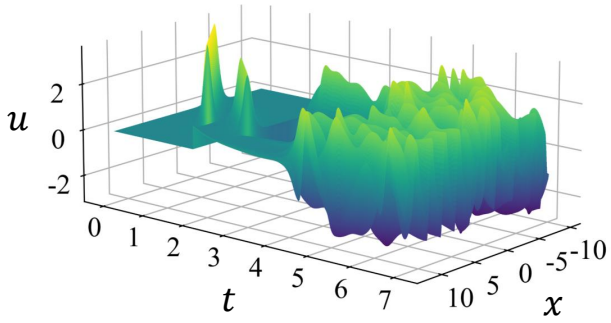

FIG. 3. The evolution from the KdV to the KS equation is plotted in space-time.

simulations.

More sampling points or longer time series correspond to the preferable identification of the internal control structure, while the KdV equation is an exception where the extending time series does not work (see Table III). One potential viewpoint is that approximating the soliton solution introduces great uncertainty. Nevertheless, it remains an open question to estimate the required length of the time series to distill the accurate underlying governing differential equations. However, implementing existing methodologies fundamentally depends on sufficiently large datasets, even though dynamics is only a parsimonious representation. Online-GED, on the contrary, is demonstrated to discover the correct structure of governing equations by iteratively reusing the finite available time series. Note that the computation of time derivatives results in the main error, which is magnified by the numerical roundoff. Thus, correct estimation of numerical derivatives is the most critical step and the most challenging task for Online-GED, especially in a noisy context. Perfect evaluation of the numerical derivatives can drastically improve the agreement of the innovation. Remarkably, the successful identification dramatically relies on a fortunate choice of variables and a nonlinear function basis that reflects the knowledge about the system of researchers' interest.

Online-GED is a viable tool capable of tackling streaming data from evolving systems for accomplishing the assignment of model discovery. The integration opens up a novel, interesting research insight for real-time modeling, online analysis, and control techniques of complex dynamic systems.

TABLE II. Summary of Online-GED and standard machine learning methods for identifying the evolving systems. Online-GED is successfully applied to reconstruct the correct governing PDEs before and after the system changes, while the standard machine learning methods strike a compromise solution utilizing all samples.

Methodology	Cubic_Linear	KdV_KS
Online-GED	**Phase 1** $\dot{x} = -0.1257x^3 + 2.0006y^3$ $\dot{y} = -2.0193x^3 - 0.1179y^3$ **Phase 2** $\dot{x} = -0.1006x + 1.9958y$ $\dot{y} = -1.9999x - 0.1000y$	**Phase 1** $u_t = -6.1036uu_x - 0.9460u_{xxx}$ **Phase 2** $u_t = -0.9894uu_x - 0.9713u_{xx} - 0.9158u_{xxxx}$
STRidge [4]	$\dot{x} = 0.5366y + 1.3167y^3 - 0.4861x^2y + 0.1735y^5 + 0.1678x^2y^3 + 0.1170x^4y$ $\dot{y} = -0.5751x + 0.5912xy^2 + 0.1335x^2y - 1.3422x^3 - 0.1340xy^4 - 0.2122x^3y^2 - 0.1503x^5$	$u_t = -1.6007u_x + 1.4328uu_{xx} - 0.5409u^2u_{xx} - 0.5814u_{xxx} + 0.8952uu_{xxx} + 0.5741uu_{xxxx}$
TrainSTRidge [4]	$\dot{x} = 0.5340y + 1.3186y^3 - 0.4700x^2y + 0.1734y^5 + 0.1610x^2y^3 + 0.1114x^4y$ $\dot{y} = -0.5751x + 0.5912xy^2 + 0.1335x^2y - 1.3422x^3 - 0.1340xy^4 - 0.2122x^3y^2 - 0.1503x^5$	$u_t = -1.6007u_x + 1.4328uu_{xx} - 0.5409u^2u_{xx} - 0.5814u_{xxx} + 0.8952uu_{xxx} + 0.5741uu_{xxxx}$
STLS [46]	$\dot{x} = 0.4920y + 1.4390y^3 - 0.4630x^2y + 0.1310y^5 + 0.1120x^2y^3 + 0.1320x^4y$ $\dot{y} = -0.5260x + 0.4480xy^2 - 0.1320x^2y - 1.4280x^3 - 0.0830xy^4 - 0.1510x^3y^2 - 0.1240x^5$	$u_t = -1.6007u_x + 1.4328uu_{xx} - 0.5409u^2u_{xx} - 0.5814u_{xxx} + 0.8952uu_{xxx} + 0.5741uu_{xxxx}$
Lasso [21]	$\dot{x} = 0.5367y + 1.3166y^3 - 0.4872x^2y + 0.1735y^5 + 0.1681x^2y^3 + 0.1179x^4y$ $\dot{y} = -0.5748x + 0.5925xy^2 + 0.1340x^2y - 1.3340x^3 - 0.1339xy^4 - 0.2136x^3y^2 - 0.1495x^5$	$u_t = -1.6007u_x + 1.4328uu_{xx} - 0.5409u^2u_{xx} - 0.5814u_{xxx} + 0.8952uu_{xxx} + 0.5741uu_{xxxx}$
ElasticNet [47]	$\dot{x} = 0.4364y + 1.3402y^3 + 0.1751y^5$, $\dot{y} = -0.4410x - 0.0945y^3 - 1.4026x^3 - 0.1417x^5$	$u_t = -1.6007u_x + 1.4328uu_{xx} - 0.5409u^2u_{xx} - 0.5814u_{xxx} + 0.8952uu_{xxx} + 0.5741uu_{xxxx}$
FoBaGreedy [48]	$\dot{x} = 0.2302y + 1.8805y^3 - 0.1003x^3$, $\dot{y} = -0.2487x - 0.0988y^3 - 1.8757x^3$	$u_t = -1.5771u_x + 1.5007uu_{xx} - 0.5332u^2u_{xx} - 0.6168u_{xxx} + 0.9653uu_{xxx}$

TABLE III. Summary of Online-GED for identifying the KdV equations of different spatial and temporal sampling of the numerical simulation data.

Discretization	m=3 n=256	m=6 n=256	m=10 n=256	m=10 n=128	m=6 n=128
Error	0.2807	0.0217	0.0090	0.0972	0.0068

ACKNOWLEDGMENTS

This work was supported in part by the Key Project of Science and Technology Innovation 2030 supported by the Ministry of Science and Technology of China under Grant 2018AAA0101302, in part by the National Natural Science Foundation of China under Grant 62206205, and in part by the Zhejiang Lab's International Talent Fund for Young Professionals.

Appendix A: Data Preprocessing

1. Subsampling Data

For massive-scale datasets, sparse sampling can be used to reduce the size of the data. Subsampling of the measurements is equivalent to obtaining a subset of rows, i.e., the samples occupy only a trivial part of the needlessly expensive full-state observations. Mathematically, this means ignoring information about some rows in a linear expression. For identification, it should be noted that we only require a small number of spatial points and their neighbors, whose responsibility is estimating the partial derivative terms in the candidate library. That is, local information around each measurement is necessarily wanted. Distinction allowing for application to subsampled spatiotemporal sequences, to an extent, is critically important due to experimentally and computationally prohibitive implementation of collecting full-state measurements [4].

2. Data Processing

As a pre-processing step, we first simulate the trajectories in the canonical dynamic systems and record the spatiotemporal measurements of the state variables, then design a combinational library of candidate functions. For derivation, second-order finite differences [39] are devoted to the clean data from numerical simulations, while the easiest to implement and most reliable method for the noisy data is a polynomial interpolation [40]. In principle,

we abandon those points close to the boundaries (∼20%) due to the absence of numerical derivatives. Specifically, our online method takes these streaming data arriving in time order as inputs and sequentially updates gradients as well as coefficient vectors, thereby being able to return an identified form on the right-hand side at an arbitrary time point. If we know any prior knowledge about the governing equation, for instance, if one of the potential terms is determined to be nonzero in advance, we can apply the additional information to the initialization phase. Moreover, truncation of the solution is a tool to maintain its sparsity, and different threshold values may provide distinguishing sparsity levels of the final output. As for the artificial noise added to the state measurements of governing equations, we use white noise. The noisy data is directly trained in experiments. In terms of the reaction-diffusion equation, exceptionally, we use the singular value decomposition (SVD) [49] technique to denoise some noisy spatiotemporal series, and the result is a low-dimensional approximation of datasets.

Appendix B: Canonical Models

1. Reaction Diffusion

The dynamic evolution of a reaction-diffusion system is represented by the following equations, which, although they are maybe deceptively simple, present nonlinear terms on the right-hand side that provide great expressiveness and freedom.

$$u_t = 0.1\nabla^2 u + \lambda(A)\, u - w(A)\, v, \tag{B1}$$

$$v_t = 0.1\nabla^2 v + w(A)\, u + \lambda(A)\, v. \tag{B2}$$

where $A = u^2+v^2$, $w(A) = -\beta A^2$, $\lambda(A) = 1-A^2$. Given the vast number of data points, we subsample 5000 discretized spatial points with 30 time points each, resulting in a high-dimensional input dataset.

2. Hopf Norm Form

In this example, the application of Online-GED is extended to a parameterized normal form of Poincaré-Andronov-Hopf bifurcation. Numerical analysis of the Hopf normal form is notably important to further the understanding of bifurcations.

$$\dot{x} = \mu x - wy - Ax\left(x^2+y^2\right), \tag{B3}$$

$$\dot{y} = wx + \mu y - Ay\left(x^2+y^2\right). \tag{B4}$$

We explored a normal form of Hopf bifurcation on two-dimensional ordinary differential equations systems. Conventionally, we consider it possible to identify Hopf normal forms associated with a certain bifurcation parameter μ. Given $w = 1$ and $A = 1$, we collected data from the noise-free system for eight various values of the parameter μ.

3. Fokker-Planck Equation

Fokker-Planck equation that reflects the connection between the diffusion equation and Brownian motion has been taken into account. It is equivalent to the convection-diffusion equation when applied to particle position distributions. Considering the simplest form of the diffusion equation where the diffusion coefficient is 0.5 and the drift term is zero, the corresponding Fokker-Planck equation is $u_t = 0.5u_{xx}$.

The above equation associated with a particle's position describes the time evolution of the probability distribution function, according to which a standard scalar Wiener process is generated. The Brownian motion model is usually simplified as a random walk in physics, such as the random movement of molecules in liquids and gases. Aiming to simulate the process, we added a distributed random variable with variance $dt = 0.01$ to the time series. We then sample the movement of the random walker and approximate the distribution function created by histograms (depending on $m = 3$). Additionally, we attempted to add an advection term (depend on $c = 2$) to the equation to bias the random walk.

4. Burgers' Equation

Here, we consider a fundamental partial differential equation in the fields of fluid mechanics, nonlinear acoustics, and aerodynamics. As a dissipative system in one-dimensional space, the general form of Burgers' equation is described given a diffusion coefficient c (also known as kinematic viscosity). For example, given the viscosity of fluid c, the speed of the fluid at the indicated spatial coordinate x and temporal coordinate t in a thin ideal pipe can be expressed as the following equation:

$$u_t = -uu_x + cu_{xx}, \tag{B5}$$

where the diffusion term u_{xx} contributes the advective form of the Burgers' equation. When c is equivalent to zero (implying the absence of the diffusion term), it becomes the inviscid Burgers' equation, a prototype for conservation equations that can develop shock waves.

5. Korteweg-de Vries Equation

The Korteweg-de Vries (KdV) equation is a nonlinear, dispersive partial differential equation for a function u of two dimensionless real variables, x and t, which refer to space and time, respectively. Numerically, its solutions seem to be decomposed into a collection of well-separated solitary waves that are almost unaffected in shape by passing through each other. The soliton solution is given by

$$u(x,t) = 0.5c \times sech^2[0.5\sqrt{c}(x - ct - a)], \tag{B6}$$

where c stands for the phase speed and a is an arbitrary constant. This equation describes a right-moving soliton that propagates with a speed proportional to the amplitude.

Moreover, it has been proven that the correct equation cannot be distinguished from a single propagating soliton solution, which is a result of the fact that some studied discretized expressions may be solutions to more than one PDE [4]. For example, both the one-way wave equation $u_t + cu_x = 0$ and the KdV equation admit the same traveling wave solution of the form $u = f(x - ct)$ if the initial data was a hyperbolic secant squared. Hence, we constructed time-series data for more than a single initial amplitude to rectify the ambiguity in selecting the governing PDE, thereby enabling the unique determination.

As the circumstance under a single propagating soliton solution, we respectively constructed two solutions without noise having the traveling speed c equal to 5 or 1 on grids with 6 timesteps and 256 spatial points. Ultimately, corresponding advection equations with different c were identified using a single traveling wave. We believe that two waves with different amplitudes and speeds may solve the KdV equation but rather the same advection equation.

6. Kuramoto-Sivashinsky Equation

The Kuramoto-Sivashinsky (KS) equation is a fourth-order nonlinear PDE derived by Yoshiki Kuramoto and Gregory Sivashinsky in the late 1970s. Specifically, it provides two dissipative terms u_{xxx} and u_{xxxx} based on Burgers' equation, where the fourth-order diffusion term accomplishes the stabilizing regularization rather than the second-order diffusion term u_{xx}, which leads to long-wavelength instabilities. By leveraging a spectral method, the numerical solution to the KS equation was created with 101 timesteps and 1024 spatial points

[1] G. Sugihara and R. M. May, Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series, Nature **344**, 734 (1990).

[2] J. A. Tropp and A. C. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, IEEE Transactions on Information Theory **53**, 4655 (2007).

[3] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, Data-driven discovery of coordinates and governing equations, Proceedings of the National Academy of Sciences **116**, 22445 (2019).

[4] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Data-driven discovery of partial differential equations, Science advances **3**, e1602614 (2017).

[5] J. P. Crutchfield and B. McNamara, Equations of motion from a data series, Complex Systems **1**, 417 (1987).

[6] S. Li, E. Kaiser, S. Laima, H. Li, S. L. Brunton, and J. N. Kutz, Discovering time-varying aerodynamics of a prototype bridge by sparse identification of nonlinear dynamical systems, Physical Review E **100**, 022220 (2019).

[7] P. A. Reinbold and R. O. Grigoriev, Data-driven discovery of partial differential equation models with latent variables, Physical Review E **100**, 022219 (2019).

[8] S. Maddu, B. L. Cheeseman, C. L. Müller, and I. F. Sbalzarini, Learning physically consistent differential equation models from data using group sparsity, Physical Review E **103**, 042310 (2021).

[9] A. Somacal, Y. Barrera, L. Boechi, M. Jonckheere, V. Lefieux, D. Picard, and E. Smucler, Uncovering differential equations from data with hidden variables, Physical Review E **105**, 054209 (2022).

[10] D. E. Shea, S. L. Brunton, and J. N. Kutz, Sindy-bvp: Sparse identification of nonlinear dynamics for boundary value problems, Physical Review Research **3**, 023255 (2021).

[11] H. Xu and D. Zhang, Robust discovery of partial differential equations in complex situations, Physical Review Research **3**, 033270 (2021).

[12] Y. Chen, Y. Luo, Q. Liu, H. Xu, and D. Zhang, Symbolic genetic algorithm for discovering open-form partial differential equations (sga-pde), Physical Review Research **4**, 023174 (2022).

[13] H. Ye, R. J. Beamish, S. M. Glaser, S. C. Grant, C.-h. Hsieh, L. J. Richards, J. T. Schnute, and G. Sugihara, Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling, Proceedings of the National Academy of Sciences **112**, E1569 (2015).

[14] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, C. Theodoropoulos, *et al.*, Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis, Commun. Math. Sci **1**, 715 (2003).

[15] J. L. Proctor, S. L. Brunton, B. W. Brunton, and J. Kutz, Exploiting sparsity and equation-free architectures in complex systems, The European Physical Journal Special Topics **223**, 2665 (2014).

[16] A. J. Roberts, *Model emergent dynamics in complex systems*, Vol. 20 (SIAM, 2014).

[17] A. J. Majda, C. Franzke, and D. Crommelin, Normal forms for reduced stochastic climate models, Proceedings of the National Academy of Sciences **106**, 3649 (2009).

[18] D. Giannakis and A. J. Majda, Nonlinear laplacian spectral analysis for time series with intermittency and low-frequency variability, Proceedings of the National Academy of Sciences **109**, 2222 (2012).

[19] B. C. Daniels and I. Nemenman, Automated adaptive inference of phenomenological dynamical models, Nature Communications **6**, 1 (2015).

[20] H. U. Voss, P. Kolodner, M. Abel, and J. Kurths, Amplitude equations from spatiotemporal binary-fluid convection data, Physical Review Letters **83**, 3422 (1999).

[21] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) **58**, 267 (1996).

[22] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Inferring biological networks by sparse identifica-

tion of nonlinear dynamics, IEEE Transactions on Molecular, Biological and Multi-Scale Communications **2**, 52 (2016).

[23] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control* (Cambridge University Press, 2022).

[24] B. Lusch, J. N. Kutz, and S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature communications **9**, 1 (2018).

[25] 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 **113**, 3932 (2016).

[26] Z. Long, Y. Lu, X. Ma, and B. Dong, Pde-net: Learning pdes from data, in *International Conference on Machine Learning* (PMLR, 2018) pp. 3208–3216.

[27] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, Network inference via the time-varying graphical lasso, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017) pp. 205–213.

[28] C. J. Walters, Nonstationarity of production relationships in exploited populations, Canadian Journal of Fisheries and Aquatic Sciences **44**, s156 (1987).

[29] M. Kolar, L. Song, A. Ahmed, and E. P. Xing, Estimating time-varying networks, The Annals of Applied Statistics , 94 (2010).

[30] G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and S. Munch, Detecting causality in complex ecosystems, Science **338**, 496 (2012).

[31] X. Zhu and J. Xu, Estimation of time varying parameters in nonlinear systems by using dynamic optimization, in *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.* (2005) pp. 5 pp.–.

[32] A. Namaki, A. H. Shirazi, R. Raei, and G. Jafari, Network analysis of a financial market based on genuine correlation and threshold method, Physica A: Statistical Mechanics and its Applications **390**, 3835 (2011).

[33] A. Ahmed and E. P. Xing, Recovering time-varying networks of dependencies in social and biological studies, Proceedings of the National Academy of Sciences **106**, 11878 (2009).

[34] S. Myers and J. Leskovec, On the convexity of latent social network inference, Advances in Neural Information Processing systems **23** (2010).

[35] R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana, Estimating time-varying brain connectivity networks from functional mri time series, NeuroImage **103**, 427 (2014).

[36] R. Ranjan, Modeling and simulation in performance optimization of big data processing frameworks, IEEE Cloud Computing **1**, 14 (2014).

[37] E. R. Deyle, M. Fogarty, C.-h. Hsieh, L. Kaufman, A. D. MacCall, S. B. Munch, C. T. Perretti, H. Ye, and G. Sugihara, Predicting climate effects on pacific sardine, Proceedings of the National Academy of Sciences **110**, 6430 (2013).

[38] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher, Sparse dynamics for partial differential equations, Proceedings of the National Academy of Sciences **110**, 6634 (2013).

[39] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems* (SIAM, 2007).

[40] O. Bruno and D. Hoch, Numerical differentiation of approximated functions with limited order-of-accuracy deterioration, SIAM Journal on Numerical Analysis **50**, 1581 (2012).

[41] B. McMahan, Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (2011) pp. 525–533.

[42] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, *et al.*, Ad click prediction: a view from the trenches, in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2013) pp. 1222–1230.

[43] K. Wu, X. Hao, J. Liu, P. Liu, and F. Shen, Online reconstruction of complex networks from streaming data, IEEE Transactions on Cybernetics **52**, 5136 (2022).

[44] S. S. Chen, D. L. Donoho, and M. A. Saunders, Atomic decomposition by basis pursuit, SIAM Review **43**, 129 (2001).

[45] Available code: https://github.com/xiaoyuans/Online-GED.

[46] M. Budišić, R. Mohr, and I. Mezić, Applied koopmanism, Chaos: An Interdisciplinary Journal of Nonlinear Science **22**, 047510 (2012).

[47] H. Zou and T. Hastie, Regularization and variable selection via the elastic net, Journal of the royal statistical society: series B (statistical methodology) **67**, 301 (2005).

[48] T. Zhang, Adaptive forward-backward greedy algorithm for sparse learning with linear models, Advances in neural information processing systems **21**, 1921–1928 (2008).

[49] M. Gavish and D. L. Donoho, The optimal hard threshold for singular values is $4/\sqrt{3}$, IEEE Transactions on Information Theory **60**, 5040 (2014).

[50] E. J. Candès, J. Romberg, and T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, IEEE Transactions on Information Theory **52**, 489 (2006).

[51] D. Ruelle and F. Takens, On the nature of turbulence, Communications in Mathematical Physics **20**, 167 (1971).

[52] J. J. Bramburger, D. Dylewsky, and J. N. Kutz, Sparse identification of slow timescale dynamics, Physical Review E **102**, 022204 (2020).

[53] D. Dylewsky, E. Kaiser, S. L. Brunton, and J. N. Kutz, Principal component trajectories for modeling spectrally continuous dynamics as forced linear systems, Physical Review E **105**, 015312 (2022).