DISCOVER: Deep identification of symbolically concise open-form partial differential equations via enhanced reinforcement learning

Mengge Du,[1] Yuntian Chen,[2,*] and Dongxiao Zhang[2,3,†]
[1]*College of Engineering, Peking University, Beijing 100871, People's Republic of China*
[2]*Ningbo Institute of Digital Twin, Eastern Institute of Technology, Ningbo, Zhejiang 315200, People's Republic of China*
[3]*National Center for Applied Mathematics Shenzhen (NCAMS), Southern University of Science and Technology, Shenzhen, Guangdong 518000, People's Republic of China*

(Received 24 July 2023; accepted 12 January 2024; published 20 February 2024)

The working mechanisms of complex natural systems tend to abide by concise partial differential equations (PDEs). Methods that directly mine equations from data are called PDE discovery, which reveals consistent physical laws and facilitates our interactions with the natural world. In this paper, an enhanced deep reinforcement-learning framework is proposed to uncover symbolically concise open-form PDEs with little prior knowledge. Particularly, based on a symbol library of basic operators and operands, a PDE can be represented by a tree structure. A structure-aware recurrent neural network agent is designed to capture structured information, and is seamlessly combined with the sparse regression method to generate open-form PDE expressions. All of the generated PDEs are evaluated by a meticulously designed reward function by balancing fitness to data and parsimony, and updated by the model-based reinforcement learning. Customized constraints and regulations are formulated to guarantee the rationality of PDEs in terms of physics and mathematics. Numerical experiments demonstrate that our framework is capable of mining open-form governing equations of several dynamic systems, even with compound equation terms, fractional structure, and high-order derivatives. This method is also applied to a real-world problem of the oceanographic system and demonstrates great potential for knowledge discovery in more complicated circumstances with exceptional efficiency and scalability.

DOI: 10.1103/PhysRevResearch.6.013182

I. INTRODUCTION

Many phenomena in dynamic natural systems can be described by concise and elegant governing equations. There are two primary methods for exploring these equations: (1) induction, which involves deriving equations directly from data (e.g., Kepler's laws were built upon Brahe's observations [1]); and (2) deduction, which involves deriving equations with rigorous mathematics from general rules and principles. However, as the system's complexity and nonlinearity increase and the amount of data multiplies, it becomes increasingly challenging to derive governing equations for such a system. Rapid advancements in computer science have enabled artificial intelligence (AI) to assist scientists in uncovering physical laws. AI aids and stimulates scientific research by using its powerful processing capacity to detect patterns and prove conjectures [2], although human scientists are still needed for this process. A salient question is whether AI can autonomously mine the governing equations from data without the need for prior knowledge.

The essence of mining natural laws in dynamic systems is to identify the relationship between the state variables and their derivatives in space and time through observations, so as to extract the governing equations that can satisfy the laws of physics (e.g., conservation laws) [3]. Sparse regression is an essential and commonly used method to accomplish the differential equation discovery task. Sparse identification of nonlinear dynamical systems (SINDy) [1] first utilized the sparsity-promoting technique to identify the most important function terms in the preset library that conform to the data, in order to obtain an accurate and concise equation representation of dynamic systems of ordinary differential equations (ODEs). After that, SINDy's variants have extended this method to more challenging scenes. PDE-FIND [4] further explored other more complex and high-dimensional dynamic systems described by PDEs, such as the Navier-Stokes equation, using the method of sequential threshold ridge regression (STRidge). Indeed, state-of-the-art (SOTA) performance was achieved on various problems, such as boundary value problems [5], and low-data and high-noise problems [6–9]. Despite the remarkable success achieved, this series of methods is limited to a closed and overcomplete candidate library. On the one hand, the selection of candidate functions requires strong prior knowledge; otherwise, the computational burden will be dramatically increased. On the other hand, while sparse regression can determine the

*ychen@eitech.edu.cn
†zhangdx@sustech.edu.cn

potential function terms and their coefficients simultaneously, it is limited to generating linear combinations of these candidates, and the expressive ability is highly restricted.

To address this issue, methods involving an expandable library and symbolic representation were proposed. PDE-Net series [10,11] generated new interaction terms based on the topology of the proposed symbolic neural network. Genetic algorithms (GA) expanded the original candidate set through the recombination of gene fragments (i.e., basis function terms) [12]. Compared to SINDy, expandable-library methods were capable of generating interactive function terms with multiplication and addition operations incorporated. However, it remained deficient in producing the division operations and compound derivatives, much less discovering open-form equations. Symbolic genetic algorithm (SGA) [13] further adopted symbolic representations and represented each function term with a tree structure. PDEs can be formulated by the interaction and combination of function terms, and optimized by GA. However, the poor iterative stability caused by crossover and mutation operations may lead to a significant increase in computation time.

Uncovering physical laws through the free combinations of operators and symbols has achieved great progress in symbolic regression due to its great flexibility and few requirements for prior knowledge. The core of this method primarily comprises two parts: representation and optimization [14]. First, equations are reasonably represented by the predefined symbols. Second, the generated equations are iteratively updated by the optimization algorithms until a desired level of accuracy is reached. Due to the advantages of the evolutionary methods in solving optimization problems, many symbolic regression algorithms based on GA and tree-like representation of structures were put forward [15–19]. GA greatly expand the flexibility of representation; however, they are simultaneously hindered by inefficient optimization and noise interference. With the progress and development of deep learning, neural networks have gradually been leveraged to reveal common laws in nonlinear dynamic systems. Their application can be divided into two categories. One approach is to embed the operations and state variables into the construction process of the neural network (e.g., replacing activation functions with basic operators), and optimize the topology by constructing a supervised learning task in an end-to-end manner [11,20,21]. One representative work, named the equation learner (EQL) [22], is capable of discovering interpretable mathematical expressions from unknown dynamic systems with good extrapolation ability. The other is based on the idea of reinforcement learning (RL). The initial mathematical expressions are generated by the agent, and only expressions with larger rewards are retained to update the agent, which in turn promotes better expressions [23,24]. Compared with GA, gradient-based methods in deep learning show great superiority in searching efficiency, but are also insufficient in the diversity of representation. Recently, some attempts to combine GA and RL also indicate excellent potential in solving real-world regression problems [25,26]. Few studies, however, focus on the PDE discovery task. Based on the structure of EQL, Lu *et al.* [27] extended it to systems governed by ODEs or PDEs when only partial state observations are available. Zhang *et al.* [28] further proposed two variants of stacked EQL (SEQL) and hyper EQL (HEQL) to discover PDEs with varying coefficients. However, these methods are gradient-based methods with backpropagation and are only applicable to find differential symbolic models. Methods designed for the symbolic regression task, such as deep symbolic regression (DSR) [24], focus on identifying a single regression target, and cannot be used to discover governing equations containing abundant physical information (including complex partial differential terms). Due to the focus on fitting the expressions to the measurements, relevant algorithms are liable to suffer from overfitting problems caused by noisy data and generate redundant terms.

To ameliorate the limitations of fixed candidate libraries and accelerate the search process, we propose a framework, Deep Identification of Symbolically Concise Open-form PDEs Via Enhanced Reinforcement-learning (DISCOVER), which enables the efficient discovery of quantitatively accurate governing equations from complex nonlinear systems, while including the minimal possible number of function terms. Specifically, we design a structure-aware long short-term memory (LSTM) agent to generate symbolic PDE expressions. It combines structured input and monotonic attention to make full use of historical and structural information. Compared to random generation, equations generated in an autoregressive manner are more consistent with physical laws and easier to optimize with higher efficiency. STRidge is also well combined to determine the coefficients of function terms. To avoid generating unreasonable expressions, customized constraints and regulations are designed as priors based on the domain knowledge of physical and mathematical laws. During the evaluation stage, a new reward function is introduced to simultaneously ensure the parsimony and fitness to observations of discovered equations. Enhanced by structural information and physical priors, the proposed framework improves the quality of the samples generated by the agent. Simultaneously, it considerably improves the optimization efficiency of reinforcement learning in discrete symbol spaces.

Conceptually, this framework seamlessly combines the flexibility of symbolic mathematical representation with the efficiency of reinforcement-learning optimization. Experiments on multiple canonical dynamic systems and nonlinear systems in the phase separation field demonstrate that the proposed framework is capable of discovering concise PDEs in open form. Its flexibility and computational efficiency have also been significantly improved compared to sparse regression methods and GA-based methods, respectively. PDEs even with fractional structure and compound high-order function terms can be accurately identified. Finally, our framework is tested on real-world oceanographic data for the discovery of subgrid forcing, which is an unsolved problem with practical applications.

II. METHODOLOGY

A nonlinear dynamic system can usually be represented by a parameterized PDE given by

$$u_t = F(u, u_x, u_{xx}, \dots, x, \xi) \tag{1}$$

where u is the observations of interest collected from experiments or nature; u_t is the first-order time-derivative term; and

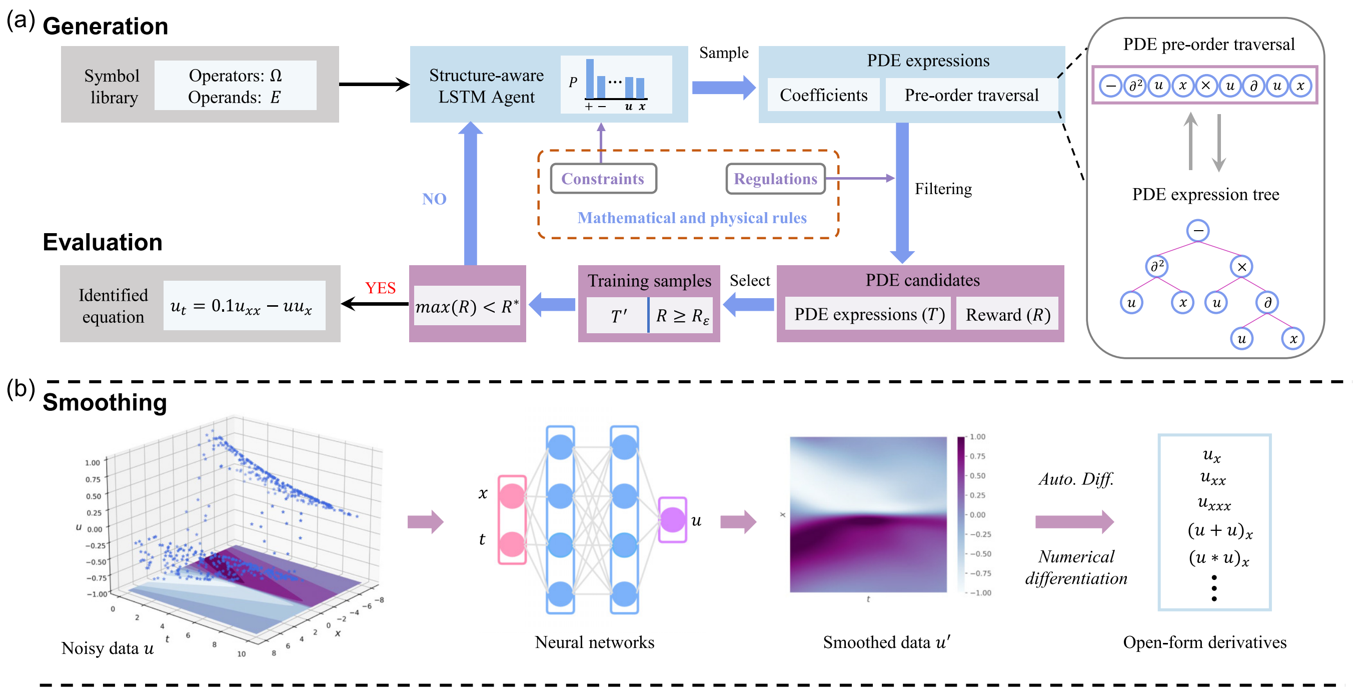


FIG. 1. Overview of the DISCOVER framework. (a) The main working process includes generation and evaluation. The symbol library is composed of operators ($\Omega = \{+, -, \times, \div, \wedge^2, \wedge^3, \partial, \partial^2, \partial^3$, sin, cos, log$\}$ and operands ($E = \{u, x, t, \text{const}\}$). The generated PDE traversal sequence can be represented by a unique corresponding PDE tree. R_ε is the $(1-\varepsilon)$ quantile of the reward in a batch, and R^* is the predefined reward threshold to terminate the search process. T refers to the generated PDE expression set that conforms to the laws of physics and mathematics, while T' represents the PDE expressions whose rewards are greater than the threshold R_ε, which are also the final training samples utilized to update the agent. (b) Smoothing procedures (the preprocessing part), in which the fully connected neural network is utilized to fit the noisy and sparse observations, so as to smooth data and generate metadata on the rest of the spatial-temporal domain. Numerical differentiation or automatic differentiation is incorporated to evaluate the open-form derivatives.

F is a nonlinear function on the right-hand side, composed of u and its spatial derivatives with different orders (e.g., u_x and u_{xx}). The coefficients of those candidate terms in F can be represented by ξ.

Uncovering PDEs from data by DISCOVER is composed of two parts: generation and evaluation [Fig. 1(a)]. The main purpose of the generation process is to first establish a reasonable symbol library and a set of rules for representing PDEs, and then generate diverse PDE expressions through the agent. A complete PDE expression includes two components, which are the PDE traversals with a tree structure and coefficients of function terms. The PDE traversals are produced autoregressively by sampling the probability distribution of the agent's output, which is constrained to ensure compliance with physical and mathematical laws. Function terms are isolated from the PDE tree and, subsequently, their coefficients are determined through the sparsity-promoting method. During the evaluation stage, the meticulously designed reward function is utilized to evaluate the generated PDE candidates that comply with the customized regulations. Then, the PDE candidates with higher rewards are selected as the training samples. The agent is iteratively updated with the risk-seeking policy gradient method to generate better-fitting expressions until the termination condition is met.

Note that observations collected from experiments may be sparse and noisy in real application scenarios. The fully connected neural networks are built to fit and smooth the available data, and the metadata are generated to assist the evaluation of derivatives [29] and noise resistance [Fig. 1(c)]. In essence, the neural network (NN) in DISCOVER is built as a surrogate model to learn the mapping relationship between spatial-temporal inputs and states of interest. The implicit relationship is then analyzed for interpretability and converted into a human-readable format (PDE).

In aggregate, DISCOVER can identify open-form governing equations directly from the data, while also possessing high efficiency and scalability. The high efficiency is due to three innovations of our framework: (1) DISCOVER utilizes a structure-aware agent to produce PDE expressions with a tree structure; (2) the coefficients of PDEs are determined by deconstructing the equation tree and combining the sparsity-promoting method to avoid multiple constant optimization processes; and (3) the whole optimization process of generating PDE expressions is neural-guided and parallelizable. In terms of scalability, the symbol library can be easily expanded, and customized constraints and regulations can be incorporated to accelerate the search process according to prior knowledge. Additional details are discussed below.

A. Generating the preorder traversals of the PDE expression trees

PDE expression tree. A PDE can be represented by a binary tree via symbolic representation, and all of the tokens on the nodes are selected from a predefined library $\mathcal{L}$ [Fig. 2(a)]. The library consists of two categories of symbols: operators (the

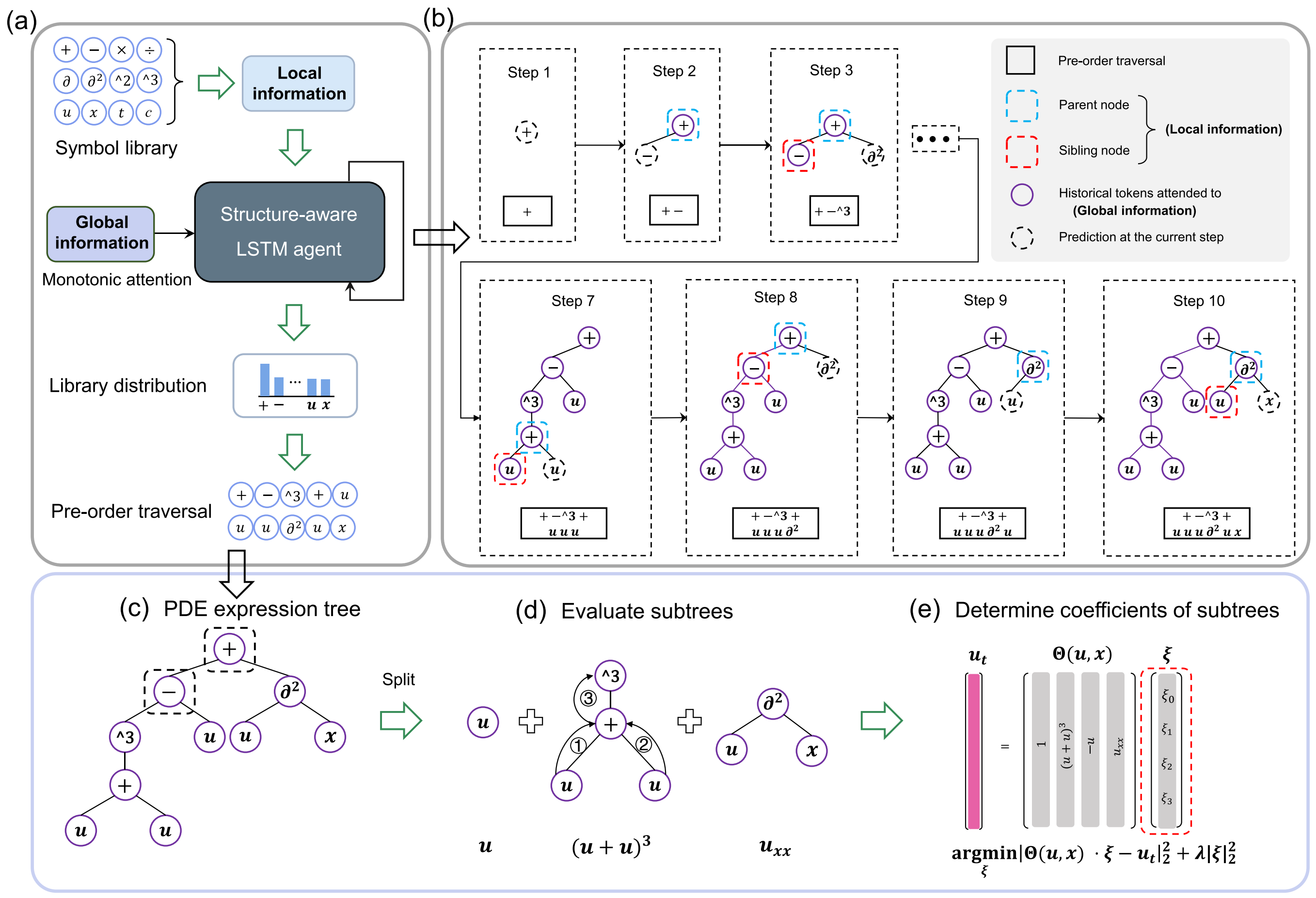


FIG. 2. Example of generating a candidate expression $u_t = u_{xx} - u + 0.1250(u + u)^3$ for the Chafee-Infante equation. (a) Procedures of generating preorder traversal of PDE expression trees by the structure-aware LSTM agent based on symbolic representation. The predefined symbol library consists of all of the operators and operands to generate the open-form PDEs. Local information and global information are both incorporated during the generation process. The former serves as inputs to the agent explicitly, and the latter is utilized by means of monotonic attention in the form of latent variables of aggregated historical information. The output of the LSTM at each time step is the probability distribution of all tokens in the library, and then one of the tokens is sampled based on it. The preorder traversal sequence is produced aggressively until the terminal condition is satisfied. (b) The evolution process of local information and global information is utilized during the generation process. Local information includes the parent node and sibling node of the current token in the corresponding expression tree. Global information takes historical information in the past into account. (c) The PDE expression tree is reconstructed from the corresponding preorder traversal. (d) Split the expression tree into subtrees according to $(+, -)$ operators at the top of the tree, and traverse to calculate each term's value. (e) Calculate the coefficients of the function terms based on STRidge.

first two rows) and operands (the bottom row). Compared with the symbolic regression problem, our library also introduces a differential operator with different orders to calculate the time and space derivatives of state variables. Note that since the LSTM agent is adopted in DISCOVER, which only produces sequential data step by step, preorder traversal sequences of PDE expression trees are generated in the generation part.

For a PDE expression tree, the interior nodes are all operators, the leaf nodes are all operands, and their arities are known. For example, the partial derivative ∂ is a binary operator with two children, and the space input x is an operand with zero children. This property ensures that each expression tree has a unique preorder traversal sequence corresponding to it. As a consequence, we can conveniently generate batches of preorder traversal sequences by means of the LSTM agent, instead of the expression trees. An expression tree and its preorder traversal can be represented as τ. The ith token in the traversal can be represented as τ_i and corresponds to an action under the current policy in reinforcement learning. When generating τ_i at the ith generation step, the output of the LSTM will be normalized to generate a probability distribution of all tokens in $\mathcal{L}$. The probability distribution is conditioned on the preceding generated tokens and can be represented as $p(\tau_i|\tau_{1:i-1};\theta)$, where θ refers to the parameters of the agent. τ_i is then sampled based on this distribution. A full binary tree is constructed when all leaf nodes in the expression tree are operands, and the generation process is terminated.

Structure-aware LSTM agent. The common LSTM treats the generation of PDE expressions as a sequence generation problem, which leads to two significant challenges. Firstly, the common LSTM fuses and couples historical information in a memory cell, with the weights of this cell shared throughout the sequential generation process. This mechanism induces a memory compression problem, which hinders the preservation of structured information. Secondly, the common LSTM uses the output of the previous step as the input for the subsequent prediction. However, the structured information that can be conveyed by this input is limited for the PDE expressions with a tree structure. Consequently, the information transfer and storage mechanism of the common LSTM restricts its ability to handle long sequences and structured information.

To address these issues and effectively generate PDE expression trees that have strong structural information, we propose a structure-aware architecture for the LSTM agent. Specifically:

(1) During the generation of the current node's token, we take its parent node and sibling node into the LSTM as inputs to convey the local information [24]. Figure 2(b) provides the entire evolution process of generating a candidate expression represented by a PDE tree. Taking the eighth time step as an example, the inputs are the parent node $(+)$ and sibling node $(-)$, instead of the previously sampled token u. It can be seen from the corresponding PDE tree that the token u is far away from the output of the current position, thereby not providing effective structural information. In contrast, the parent and sibling nodes indeed offer effective local information. Particularly, when the parent node or sibling node does not exist [as in step 1 in Fig. 2(b)], an empty token is assigned to the corresponding position.

(2) The monotonic attention mechanism is employed to enhance the agent's ability to capture global dependencies and avoid the loss of long-distance information during the step-by-step transmission process. Essentially, the structure-aware LSTM uses the attention mechanism to aggregate historical information (hidden states at each timestep) directly, rather than retrieving the context from the memory cell. The details of the monotonic attention layer are provided in Appendix A 2. As shown in Fig. 2(a), the output of the monotonic attention layer is further normalized to obtain the output probability of each element in the library, and the final output at the current time step is obtained after sampling.

Constraints and regulations. Generating PDE expression sequences in an autoregressive manner without restrictions tends to produce unreasonable samples. To reduce the search space and time consumption, we design a series of constraints and regulations based on mathematical rules and physical laws.

Constraints are imposed on the agent during the generation of preorder traversal sequences [24,26] and can be divided into two categories, including: (1) complexity limits of PDE expressions (e.g., the total length of the sequence shall not exceed the maximum length); and (2) relationship limits between operators and operands, e.g., the right child node of partial differential operators (e.g., ∂) must be space variables (e.g., x), and the left child node of ∂ cannot be space variables. In the specific implementation process, these constraints are applied by directly adjusting the categorical distribution over the symbol library prior to the sampling process. For example, the probability of tokens that violate the physical laws is directly set to zero. As the length of the generated sequence approaches the preassigned maximum length constraint, the probability of generating operands is increased to accelerate the construction of a complete PDE tree.

In addition to imposing restrictions during the generation phase, we also establish a series of regulations to double check the generated equations and withdraw the unreasonable expressions, e.g., the coefficient of the function term is too small or there are overflow errors. Note that regulations are aimed at removing unreasonable function terms, which is conducive to the rationality and simplicity of uncovered equations. This operation is similar to the replacement operations in genetic algorithms (i.e., the function term is set as 0), which to a certain extent increases diversity of generated equations. Applying these constraints and regulations is convenient and extensible, and can also be incorporated with other domain knowledge for new problems. More detailed descriptions can be found in Appendix A 3.

B. Determine coefficients of the PDE expression

Reconstruct and split the expression tree. After obtaining the preorder traversal sequence of the PDE, we first need to reconstruct the sequence into the corresponding tree structure according to the arities of operators and operands. Then, we can split the PDE tree into subtrees (i.e., function terms), based on the plus and minus operators at the top of the expression tree [Figs. 2(d) and 2(e)]. Subsequently, we solve for the value of each function term over the entire spatiotemporal domain. $(u+u)^3$ in Fig. 2(b) is taken as an instance. We traverse the subtree from bottom to top in a postorder traversal manner (①->②->③), and then perform operations at each parent node (operators). At this time, the values of its corresponding child nodes have already been calculated.

STRidge. STRidge is a widely used method in sparse regression, which can effectively determine nontrivial function terms and identify a concise equation by using the linear fit to observations. It can be utilized to solve for the coefficients of each function term based on the results in the previous step [Fig. 2(e)]. Note that the constant 1 is incorporated as a default constant term,

$$\xi = \arg\min_{\xi} \left|\Theta(u,x)\cdot\xi - u_t\right|_2^2 + \lambda\left|\xi\right|_2^2 \tag{2}$$

where λ measures the importance of the regularization term. In order to prevent overfitting, we also set a threshold tol, and function terms with coefficients less than tol will be directly ignored. Note that our method aims to solve the coefficients of function terms, and the possible constants in the function terms are not taken into consideration, which is also rare in PDEs. This setting is more efficient and conducive to solving the PDE discovery task. For extremely special situations, learnable constants can also be seamlessly incorporated into our framework.

C. Training the agent with the risk-seeking policy gradient method

Reward function. A complete representation of the generated PDEs, including function terms and their coefficients, has been obtained through the above procedures. In order to effectively evaluate the generated PDE candidates, we design a reward function for the PDE discovery problem that comprehensively considers the fitness of observations and the parsimony of the equation. Assuming that the generated PDE expression is g, it is formulated as follows:

$$R = \frac{1 - \zeta_1 \times n - \zeta_2 \times d}{1 + RMSE} \tag{3}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(u_{t_i} - g(u_i, x_i))^2} \tag{4}$$

where n is the number of function terms in the governing equation; d is the depth of the generated PDE expression tree; and ζ_1 and ζ_2 are penalty factors for parsimony, which are generally set to small numbers without fine tuning. In the default hyperparameter setting, the two penalty factors are fixed at 0.01 and 0.0001, respectively, and remain consistent across all of the numerical experiments. Even in special cases in which the impact of noise is significant, only a minor adjustment in magnitude is necessary; and N denotes the number of observations. It can be seen that the root mean squared error (RMSE) in the denominator evaluates the fitness of the PDE candidates to the data. $1/(1 + RMSE)$ enables normalization of the reward value to a range of 0 to 1. When the reward is equal to 1, the values on both sides of the equation exhibit precise equivalence, thereby resulting in minimized error. Such a design avoids the agent falling into exploring difficulties caused by sparse reward distribution and is widely used in symbolic regression tasks [23,24]. The numerator is an evaluation of the parsimony. The former is designed to avoid overfitting caused by redundant terms, especially when noisy observations are available. The latter is primarily to prevent unnecessary structures. The free combinations of symbols lead to many representations that, while numerically equivalent, exhibit different levels of complexity, such as u_x and $(u^2/u)_x$.

Risk-seeking policy gradient method. In the task of PDE discovery, the search space inherently comprises a discrete set of various mathematical expression combinations. Moreover, the nondifferentiable relationship between the rewards and the agent's parameters renders traditional optimization methods based on gradient descent inapplicable. Consequently, the deep reinforcement-learning training strategy is adopted. Specifically, the generated PDE expression sequences are equivalent to the episodes in RL, and the generation process of each token corresponds to the selection of actions. It is worth noting that the total reward is not the sum of each action with a discount factor, but is instead based on the evaluation of the final sequence. The policy π_θ refers to the distribution over the PDE expression sequences $p(\tau|\theta)$. The standard policy gradient is a risk-neutral method, with the goal of maximizing the expected return of the generated policy.

However, for PDE discovery, our intention is to ensure that the best-case sample is adequate for the problems, which is similar to symbolic regression [30,31] and neural architecture search (NAS) [32]. By referring to the risk-seeking policy gradient method in DSR [24], we train agents to improve the best-case performance rather than the average performance, to alleviate the mismatch between the objective and performance evaluation. The idea of this method comes from a well-known risk measure, conditional value at risk (CVaR), defined as $CVaR_\varepsilon(R) = \mathbb{E}[R \mid R \leqslant q_\varepsilon(R)]$, where q_ε is the ε quantile of the rewards. It is designed to improve the worst samples in the current policy and avoid risks, and is usually applied in vehicle driving or finance [33–35]. In contrast, the optimization objective of risk seeking is the expectation of the $(1-\varepsilon)$ quantile of the rewards $\tilde{q}_\varepsilon(R)$. Its return is given by

$$J_{\text{risk}}(\theta; \varepsilon) \doteq \mathbb{E}_{\tau\sim p(\tau|\theta)}[R(\tau) \mid R(\tau) \geqslant \tilde{q}_\varepsilon(R)]. \tag{5}$$

The gradient of the risk-seeking policy gradient can be estimated by

$$\nabla_\theta J_{\text{risk}}(\theta; \varepsilon) \approx \frac{\lambda_{pg}}{\varepsilon N}\sum_{i=1}^{N}[R(\tau^{(i)}) - \tilde{q}_\varepsilon(R)] \cdot \mathbf{1}_{R(\tau^{(i)})\geqslant \tilde{q}_\varepsilon(R)} \nabla_\theta \log p(\tau^{(i)} \mid \theta) \tag{6}$$

where N denotes the total number of samples in a mini batch; λ_{pg} measures the importance of rewards; and $\mathbf{1}_x$ represents a conditional judgment, which yields a return of 1 when the condition denoted by x is met, and 0 when it is not. Note that $\tilde{q}_\varepsilon$ is also chosen as the baseline reward and varies by the samples at each iteration. In addition, based on maximum entropy reinforcement learning [36], the entropy value of each output action under the current policy is also required to be maximized to prevent generating a certain action continuously. The gradient of entropy can be expressed by

$$\nabla_\theta J_{\text{entropy}}(\theta; \varepsilon) \approx \frac{1}{\varepsilon N}\sum_{i=1}^{N}(\lambda_{\mathcal{H}}\mathcal{H}(\tau^{(i)} \mid \theta)) \tag{7}$$

where $\lambda_{\mathcal{H}}$ is the temperature parameter that controls the relative importance of the entropy term against the reward. By combining these two parts, the agent can be optimized to iteratively improve the best-case performance, while increasing the exploration ability to avoid getting stuck in local optima.

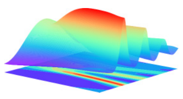
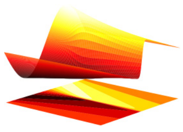
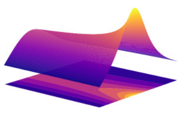
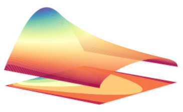
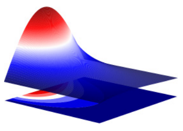
To sum up, the steps of DISCOVER mining PDEs mainly consist of three steps: (1) we first build a symbol library and define that a PDE can be represented as a tree structure. A structure-aware recurrent neural network agent is designed to generate the preorder traversal of PDE expression trees. It combines structured inputs and monotonic attention to capture the structural information; (2) the expression trees are then reconstructed and split into function terms, and their coefficients can be calculated by the sparse regression method; and (3) all of the generated PDE candidates are filtered by physical and mathematical constraints, and subsequently evaluated by a meticulously designed reward function considering the fitness to data and the parsimony of the equation. We adopt the risk-seeking policy gradient to iteratively update the agent to improve the better-fitting expressions until the best-case expression meets the accuracy and parsimony requirements that we set in advance. The first two steps correspond to the generation part, and the third step introduces the evaluation part (Fig. 1). In the next section, we will demonstrate the superiority of DISCOVER through some canonical problems.

III. EXPERIMENTS

A. Discovering open-form PDEs of canonical dynamic systems

We verified the accuracy and efficiency of DISCOVER in mining open-form PDEs containing strong nonlinearities with little prior knowledge. Specifically, we utilized the study cases

TABLE I. Summary of discovered results for different PDEs of mathematical physics. The subscripts m and n denote the number of discretizations. For noisy measurements, 10% noise is added for the KdV, Burgers', and Chafee-Infante equations, and 1% noise is added for PDE_divide and PDE_compound.

PDE systems	PDE discovered (clean data)	Error (clean, noisy)	Data discretization
 KdV	$u_t = -0.5001(u \times u)_x - 0.0025u_{xxx}$	$0.01 \pm 0.01\%$, $1.0 \pm 1.41\%$	$x \in [-1, 1)_{m=512}, t \in [0, 1]_{n=201}$
 Burgers	$u_t = -1.0010uu_x + 0.1024u_{xx}$	$1.25 \pm 1.63\%$, $1.59 \pm 0.41\%$	$x \in [-8, 8)_{m=256}, t \in [0, 10]_{n=201}$
 Chafee-Infante	$u_t = 1.0002u_{xx} - 1.0008u + 1.0004u^3$	$0.05 \pm 0.03\%$, $1.88 \pm 0.54\%$	$x \in [0, 3]_{m=301}, t \in [0, 0.5]_{n=200}$
 PDE_compound	$u_t = 0.5002(u^2)_{xx}$	$0.04 \pm 0\%$, $0.05 \pm 10\%$	$x \in [1, 2)_{m=301}, t \in [0, 0.5]_{n=251}$
 PDE_divide	$u_t = -0.9979u_x/x + 0.2498u_{xx}$	$0.15 \pm 0.09\%$, $0.22 \pm 0.25\%$	$x \in [1, 2)_{m=100}, t \in [0, 1]_{n=251}$

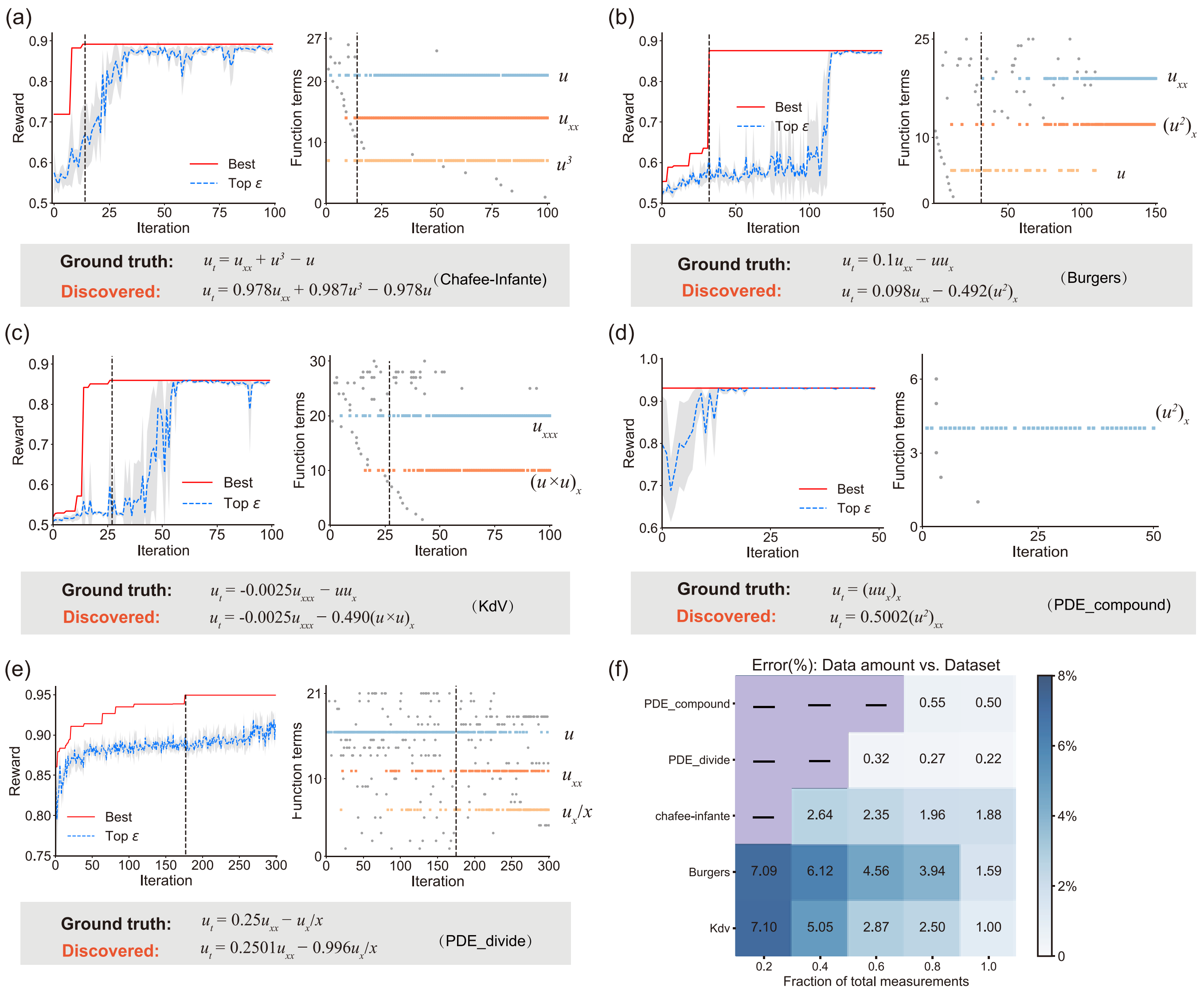
from previous studies [4,8,13,37], including Burgers' equation with nonlinear interaction terms, the KdV equation which has high-order derivatives, the Chafee-Infante equation with exponential terms, and the viscous gravity current equation (PDE_compound) with compound function terms and the equation including a fractional structure (PDE_divide). We use these equations to test the performance of DISCOVER and compare it against SGA [13], which is the latest PDE discovery model to handle complex open-form equations based on GA. The results show that our framework is not only accurate, but also computationally efficient and stable.

Study cases. Five canonical models of mathematical physics include: (1) The KdV equation: It was jointly discovered by Dutch mathematicians Korteweg and De Vries to describe the one-way wave on shallow water [38]. It takes the form of $u_t = auu_x + bu_{xxx}$, where a is set to -1, and b is set to -0.0025. (2) Burgers' equation: As a nonlinear partial differential equation that simulates the propagation and reflection of shock waves, Burgers' equation is widely used in numerous fields, such as fluid mechanics, nonlinear acoustics, gas dynamics, etc. [39]. We consider a 1D viscous Burgers' equation $u_t = -uu_x + vu_{xx}$, where v is equal to 0.1. (3) The Chafee-Infante equation: Another 1D nonlinear system is $u_t = u_{xx} + a(u - u^3)$ with $a = -1$, which was developed by Chafee and Infante [40]. It has been broadly employed in fluid mechanics [41], high-energy physical processes [42], electronics [43], and environmental research [44]. (4) PDE_compound and PDE_divide: These two equations are initially put forward in SGA and used to demonstrate that our framework is capable of mining open-form PDEs. PDE_compound with equation form $u_t = (uu_x)_x$ is proposed to describe viscous gravity currents with a compound function [45]. PDE_divide with equation form $u_t = -u_x/x + 0.25u_{xx}$ contains the fractional structure that conventional closed-library-based methods cannot handle.

The default hyperparameters used to mine the above PDEs are provided in Appendix C. Table I shows that under the premise of little prior knowledge, DISCOVER can uncover the analytic representation of the various physical dynamics mentioned above, and the discovered equation form is accurate and concise even with Gaussian noise added. Figure 3 presents the performance of DISCOVER in the mining equation under the condition of noisy and sparse data. For complex equations, there will be redundant or incorrect terms in the generated expressions during iteration [e.g., function term u in Burgers and PDE_divide; Figs. 3(b) and 3(e)]. However, since model-based optimization is directional, more accurate expressions will be generated as rewards of better cases increase, and fewer redundant terms appear [Figs. 3(a)–3(e)]. Furthermore, a risk-seeking policy gradient is utilized, and DISCOVER is able to find the optimal result with only the best-case performance considered. Meanwhile, more available data collected contribute to a more accurate surrogate model, and then it is more likely to uncover the correct equation [Fig. 3(f)]. More discussion on noise and data sparsity can be found in Appendix G 1. Note that the KdV equation, Burgers' equation, and the Chafee-Infante equation can be discovered correctly by conventional sparse regression methods and genetic algorithms, but not for the PDE_compound and PDE_divide equations. This is primarily because these methods rely on a limited set of candidates, and cannot handle equations that contain compound terms or fractional structures. The result demonstrates that our proposed method can directly mine open-form equations from data, which offers wider application scenarios and practicability.

B. Comparisons to the GA-based method

At the expense of flexibility and representation ability, common PDE discovery methods, such as SINDy, utilize



	0.2	0.4	0.6	0.8	1.0
PDE_compound	—	—	—	0.55	0.50
PDE_divide	—	—	0.32	0.27	0.22
chafee-infante	—	2.64	2.35	1.96	1.88
Burgers	7.09	6.12	4.56	3.94	1.59
Kdv	7.10	5.05	2.87	2.50	1.00

FIG. 3. Discovery process of some canonical physical problems with noisy and sparse measurements. [(a)–(e)] The evolution of rewards and function terms of the best-case expression of (a) the Chafee-Infante equation, (b) Burgers' equation, (c) the KdV equation, (d) PDE_compound, and (e) PDE_divide. Ten percent noise in the first three equations is used, and 1% noise is used in the last two equations. The figure on the left denotes the evolution of the best reward so far and the reward of the top ε expressions (training samples). The figure on the right denotes the evolution of function terms of the best case in the current iteration. Function terms that occur the most are individually marked with colored dots. The gray dashed-line marks the iteration where the expression with maximum reward appears. (f) Relative error of different physical systems with different volumes of data. Squares marked with a solid black line indicate that the correct form of the equation could not be found.

a closed library and sparse regression to uncover physical laws. Since the closed library cannot cover or generate all of the complex structures, methods based on a closed library are deficient in identifying complex equation forms, such as PDEs with fractional structures. For fairness of comparison, the latest symbolic PDE discovery method SGA is taken as an example, which is also capable of uncovering the equation representations of these five physical dynamics correctly. Specifically, we will compare the specific performance (time consumption and accuracy) of DISCOVER based on RL and SGA based on GA. Note that all of the experiments were replicated with five different random seeds for each PDE. As shown in Table II, the left column represents the error of coefficients when all of the function terms are correctly identified. Under the condition of noise-free data, both SGA and DISCOVER can mine the correct equation with a small error. In DISCOVER, the result of the equation term is truncated at the boundary. Consequently, the corresponding coefficients are relatively more accurate since the boundary error caused by numerical differentiation is further reduced. The right column of Table II presents the time consumed by the two methods to identify the optimal equation. It can be seen that, in addition to the Chafee-Infante equation, our method uses only approximately 40% of the time cost by SGA(fast) (a more efficient version) on average, which has higher computational efficiency. This is mainly because our method is model based, and the entire optimization process is directional with a positive gain. As the iteration proceeds, the generated

TABLE II. Comparison of the coefficient error of discovered results and running time for DISCOVER and SGA.

	Coefficient error		Running time (s)			
	DISCOVER	SGA	DISCOVER	SGA	SGA(fast)[a]	SGA(w/o u)[b]
KdV	$0.01 \pm 0.01\%$	$0.02 \pm 0.03\%$	243.66	1464.80	890.50	
Burgers	$1.25 \pm 1.63\%$	$1.26 \pm 1.62\%$	206.76	495.18	423.8	
Chafee-Infante	$0.05 \pm 0.03\%$	$0.05 \pm 0.03\%$	67.23	27.70	20.12	>1000
PDE_compound	$0.04 \pm 0\%$	$1.94 \pm 0\%$	13.31	604.10	557.20	
PDE_divide	$0.15 \pm 0.09\%$	$0.15 \pm 0.09\%$	1259.53	2046.24	1466.51	

[a]SGA(fast) is an optimized version of the original article code, with a faster computation speed.
[b]SGA(w/o u) represents the SGA(fast) version without prior knowledge u provided in advance.

equations become increasingly reminiscent of the authentic expression. Additional details of the optimization process can be found in Appendix F. In contrast, SGA expands the diversity of the generated equation representations primarily through crossover and mutation of gene fragments, which is more stochastic and uncontrollable. Although it facilitates the search for more complex equations, more computational time is also required.

In the process of uncovering the Chafee-Infante equation, SGA takes u as a default function term (this information itself is known and easily accessible). By introducing this prior knowledge, the optimal form of the equation can always be easily found in the first round of iterations. Without this knowledge, however, SGA is unable to identify the correct form of the equation within 300 iterations (>1000 s). The main reason for this problem is that SGA is modeled based on function terms that are randomly generated, and each function term is represented by a multilayer tree structure, while the term u is represented as a one-layer root node. In other words, the way that function terms are defined makes SGA prone to produce complex tree structures. However, DISCOVER models the equation as a whole with a preorder traversal sequence and then partitions it according to the operators ("+" or "–"). The symbol sequence is autoregressively generated and, consequently, both simple and complex function terms can be handled easily.

The above analysis reveals that the proposed method exhibits an obvious improvement in methodology compared to SGA, which equips DISCOVER with the capacity to address practical applications. This is principally manifested in the following aspects: (1) Our framework proposes a more reasonable way of generating PDE samples. The sequence generation with a structure-aware LSTM agent is more efficient than constructing PDE trees separately. Meanwhile, our framework can seamlessly incorporate potential physical priors as constraints in the generation process, thereby reducing the search space. (2) Neural-guided reinforcement learning is adapted for the PDE discovery task to effectively learn from high-quality expressions during the iterative process, promoting the generation of better-fitting samples. Conversely, the optimization process of SGA exhibits greater randomness and instability. (3) Owing to its efficiency and flexibility, DISCOVER is more adept at mining governing equations in complex real-world scenarios, such as high-dimensional and multi-variable nonlinear systems with large amounts of data. A real-world discovery example in an oceanographic system will be further demonstrated in subsequent experiments.

C. Discovering complex open-form PDEs of phase separation

To demonstrate that the proposed DISCOVER framework is capable of mining the high-dimensional PDEs with high-order derivatives, we introduce the Allen-Cahn and Cahn-Hilliard equations, which are firmly nonlinear gradient flow systems and frequently used to describe phase separation processes in fluid dynamics [46–48] and material sciences [49,50].

The Allen-Cahn equation. The 2D reaction-diffusion systems considered here can be represented by $u_t = \epsilon^2 \Delta u - f(u)$, where ϵ is a small constant value representing the interfacial thickness, and ϵ^2 is set to 0.001; Δu denotes the diffusion term; and $f(u) = u^3 - u$ represents the reaction term. For this example, Laplace operators appear in the equation in addition to the increase in spatial dimensionality. To ensure that the PDEs mined by DISCOVER conform to the physical laws, we set a constraint on the spatial symmetry of the generated equations. It requires that the order of the partial derivatives with respect to the spatial inputs x and y must be the same for the right-hand side (RHS) of the equation, i.e., the number of occurrences of x and y in the traversal sequence must be the same. According to the visualization of the trend of best-case reward and top ϵ rewards, the best-case reward soon reaches 0.95, and as the iteration proceeds, the reward makes four large jumps in total [Fig. 4(a)]. It can also be observed that better-fitting expressions increase as the training progresses, and their discovered equation gradually approaches the ground truth [Fig. 4(b)]. Specifically, the distribution of the RHS of the equations approaches the true distribution, and the gap between the left-hand side (LHS) term u_t and the RHS term (i.e., the residual) decreases and finally approaches zero [Fig. 4(d)]. Figure 4(c) illustrates the evolution of the equation terms of the best-case expression during the iterations. As the optimization process proceeds, the gradual increase in the number of equation terms included in the correct equation form proves that the expressions generated by the agent are becoming increasingly accurate, while other incorrect function terms gradually disappear.

The Cahn-Hilliard equation. With the fourth order of derivatives and more complicated compound terms, we consider a specific form of Cahn-Hilliard equation $u_t = \Delta(u^3 - u - \kappa \Delta u)$, where $\kappa = 0.5$ and denotes the surface tension at the interface. The internal term $u^3 - u - \kappa \Delta u$ within the Laplace operator denotes the chemical potential and manifests a complex form when fully expanded. For this reason, while retaining the symmetric constraint, we introduce a new Laplace operator in the symbol library and compare it with

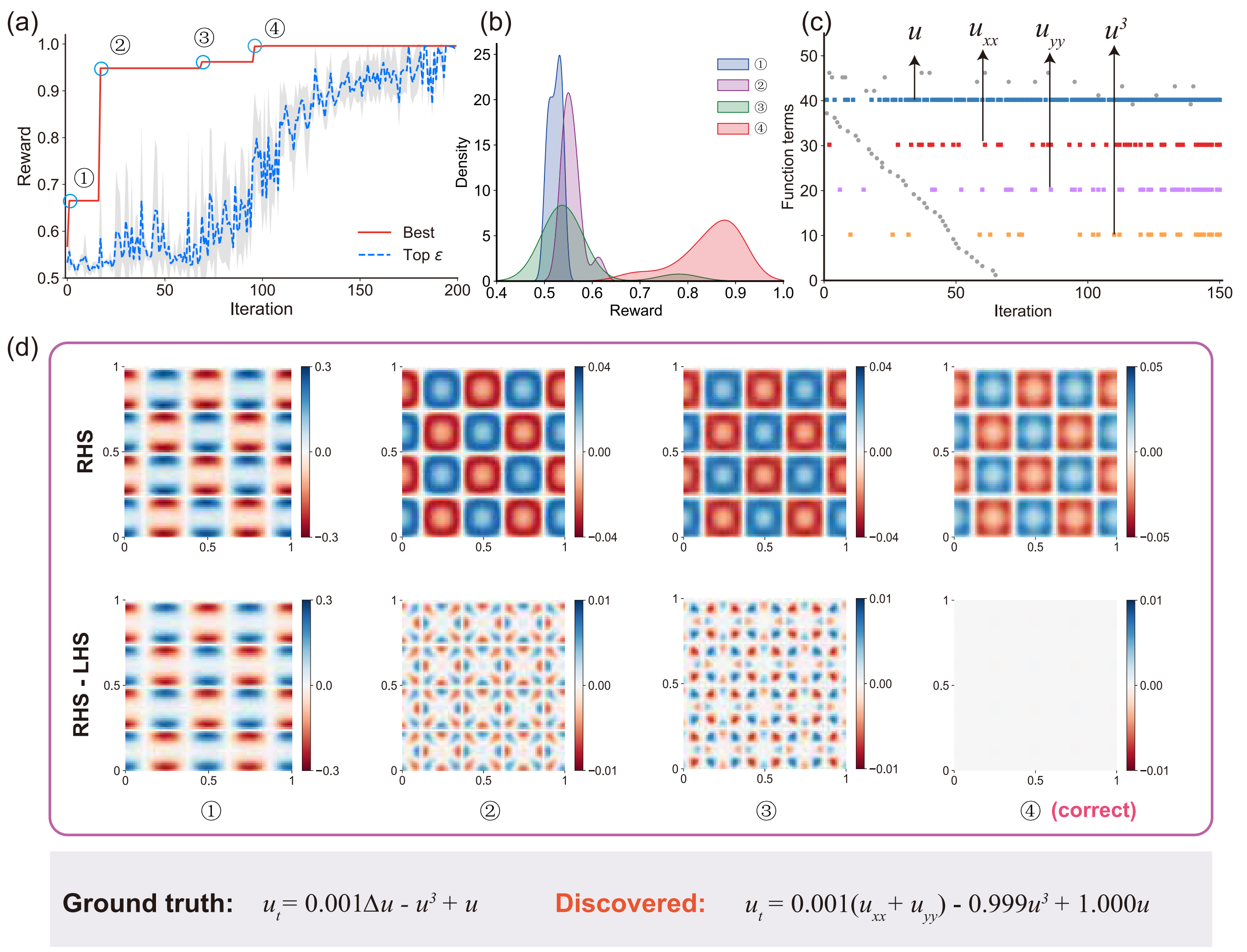


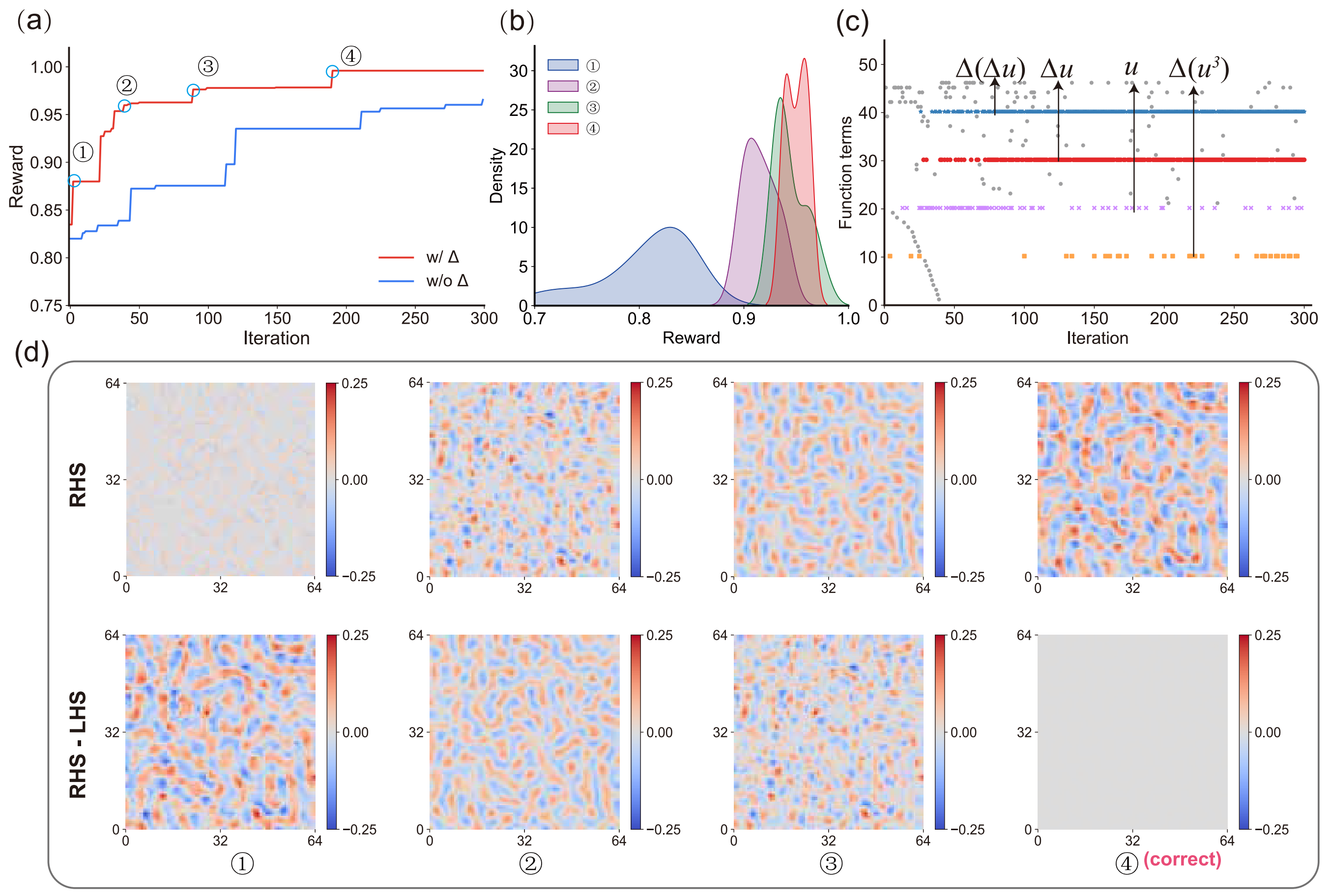
FIG. 4. The optimization process and intermediate results at the 40th time step of discovering the Allen-Cahn equation. (a) The distribution of the best-case reward and top ε fraction of rewards. ①–④ represent the positions where the reward goes through a sharp jump. (b) The Gaussian kernel density estimate of the top ε fraction of rewards at ①–④. (c) Evolution of the function terms of the best expression. The Y axis corresponds to the 90 most frequent equation terms with decreasing occurrences from the top to the bottom. Each point represents the corresponding function term appearing in the best-case expression produced by the agent at the current iteration step. The four most-frequently occurring equation terms are u, u_{xx}, u_{yy}, and u^3, respectively, which are taken out separately and correspond to function terms of the correct equation form. (d) The change of the RHS of the mined PDE and the difference between the RHS and LHS (u_t), which is the residual of the equation.

the default library configuration. Results show that the reward with the Laplace operator grows faster and can uncover the true equation form in 194th iterative steps [Fig. 5(a)]. However, by the default configuration, the correct equation form can be identified in 903rd iterative steps, which takes more than four times as many iterative steps. With the updating of the agent, the proportion of expressions with higher rewards gradually increases [Fig. 5(b)], and generated function terms gradually approach the true equation [Fig. 5(c)]. Note that equation term u mainly appears in the beginning stage of training, and its occurrence gradually decreases with better expressions learned. The transformations can be further observed through the evolution process of RHS terms and the residual [Fig. 5(d)]. Results demonstrate that our framework can seamlessly incorporate possible prior knowledge by expanding constraint and regulation systems or the symbol library to facilitate the searching process and uncover the underlying physical laws in dynamic systems with strong nonlinearity.

These two examples illustrate that DISCOVER can handle high-dimensional and high-order dynamical systems and conveniently introduce new constraints and operators to accelerate the discovery process with great scalability.

D. Comparisons with other PDE discovery methods

As shown in Table III, we have presented whether other methods can identify the equations from data mentioned in this article and summarized the boundaries of different methods. Closed library methods, including PDE-FIND [4] and PDE-NET [10], and expandable library methods, including PDE-NET 2.0 [11], EPDE [37], and DLGA [12], can find



Ground truth: $u_t = \Delta(u^3 - u - 0.5\Delta u)$ **Discovered:** $u_t = 1.000\Delta(u^3 - u) - 0.500(\Delta u)_{xx} + 0.499(\Delta u)_{yy}$

FIG. 5. The optimization process and intermediate results at the 250th time step of discovering the Cahn-Hilliard equation. (a) The distribution of the best-case reward and top ε fraction of rewards. ①–④ represent the positions where the reward goes through a sharp jump. (b) The Gaussian kernel density estimate of the top ε fraction of rewards with the Laplace operator at ①–④. (c) Evolution of the function terms of the best expression. The Y axis corresponds to the 100 most frequent equation terms with decreasing occurrences from the top to the bottom. The four most frequently occurring equation terms are $\Delta(\Delta u)$, Δu, u, and $\Delta(u^3)$, respectively. (d) The change of the RHS of the mined PDE and the difference between the RHS and LHS (u_t), which is the residual of the equation.

the linear combination of possible candidate function terms in the library and are capable of handling the KdV, Burgers', Chafee-Infante, and Allen-Cahn equations. The difference is that the closed library depends more on prior knowledge to ensure that all of the function terms that may appear in the equation are included in the library. Weak-form methods [7,51] typically employ space-time integration by parts to substitute the evaluation of pointwise derivatives. These methods possess inherent advantages in discovering equations that contain compound terms, especially when handling noisy data. Essentially, weak-form methods are still built on a complete candidate library and sparse regression, which leads to two problems when dealing with equations that may contain complex terms: (1) Prior knowledge is often required to preset the effective trial functions and the number of integrals in advance. Or function terms, such as $(u_x)^2$, cannot be covered. Meanwhile, a sufficiently large library could result in an unaffordable computational burden. (2) Weak-form methods are inadequate for handling equations with compound terms that have nested structures or multiple complex terms within them. Therefore, the ability of weak-form methods to mine complex governing equations is limited, especially in real-world applications without prior knowledge. Ensemble-SINDy (E-SINDy) employs statistical techniques, such as bootstrapping to achieve robust identification of both ODE and PDE systems and facilitates further uncertainty quantification [8]. While library bagging allows for the use of a larger candidate library, this method still cannot eliminate the dependence on prior knowledge. The integration of these methods with symbolic representation holds great potential and requires further investigation in future research.

The methods with symbolic representation, including SGA [13], DSR [24,26], and DISCOVER, cannot only uncover the conventional equations, such as the KdV equation, but also the complex PDEs with compound terms and fractional structures, and symbolic regression tasks. Among them, SGA utilizes GA to expand the search space, but is less efficient and currently confined to 1D dynamics and clean data. DSR and

TABLE III. Comparisons of DISCOVER and different methods on PDE discovery and symbolic regression tasks.

Equations	Correct expressions	PDE-FIND [4]	PDE-NET [10,11]	Weak form [7,51]	EPDE [37]	DLGA [12]	E-SINDy [8]	SGA [13]	DSR [24,26]	DISCOVER
KdV	$u_t = -uu_x - 0.0025u_{xxx}$	$\surd$[a]	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\not\surd$[b]	$\surd$
Burgers	$u_t = -uu_x + 0.1u_{xx}$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\not\surd$	$\surd$
Chafee-Infante	$u_t = u_{xx} + u - u^3$	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$	$\not\surd$	$\surd$	$\surd$
PDE_compound	$u_t = (uu_x)_x$			$\surd$			$\surd$	$\surd$	$\not\surd$	$\surd$
PDE_divide	$u_t = -u_x/x + 0.25u_{xx}$							$\surd$	$\not\surd$	$\surd$
Allen-Cahn	$u_t = 0.001\Delta u - u + u^3$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\not\surd$	$\surd$
Cahn-Hilliard	$u_t = \Delta(u^3 - u - 0.5\Delta u)$			$\not\surd$			$\not\surd$	$\surd$	$\not\surd$	$\surd$
Nguyen benchmarks [52]	Nguyen-(1-12)							$\not\surd$	$\surd$	$\not\surd$

[a] $\surd$ indicates that the corresponding equation can be found without requiring extra prior knowledge.
[b] $\not\surd$ indicates that the equation can only be found if specific conditions are satisfied or if necessary prior knowledge is available. When multiple datasets are considered, such as the Nguyen benchmarks [52], it implies that only a fraction of the true equations within the datasets can be discovered.

DISCOVER are both based on deep reinforcement-learning methods; whereas, DSR is designed for symbolic regression tasks whose optimization objective is not consistent with PDE discovery tasks. Specifically, the current DSR neither incorporates the computation of partial differential operators nor is it able to handle sparse and noisy data. The coefficients also cannot be determined efficiently, which may lead to prohibitive computational costs for high-dimensional nonlinear dynamic systems with large data volumes. A detailed description can be found in Appendix E. DISCOVER performs better for PDE discovery tasks, as it balances diversity and efficiency in the mining equation process with better scalability. Although not specifically for regression tasks, it can solve most of the symbolic tasks in Nguyen benchmarks [52] under the condition that the appropriate library is defined.

E. Real-world discovery in the oceanographic system

In ocean modeling, high-resolution simulations tend to encounter computational resource limitations, while coarse-resolution simulations usually fail to describe small-scale features. Subgrid parametrization is a common way of accounting for the impact of unresolved processes in large eddy simulation (LES). By incorporating a reasonable forcing term to parametrize the subgrid-scale process, it is expected to improve the performance of low-resolution models and increase computational efficiency. In this experiment, a benchmark problem in a two-layer oceanic model [53] is taken as an example, and we utilize DISCOVER to reveal interpretable and generalized equations for the subgrid forcing without excessive human intervention.

The optimization goal is to maximize the correlation between the discovered result and the output of filtering and coarse-grained high-resolution simulations. The basic operators include differential operators (∂_x, ∂_y, ∇^2, $\bar{\mathbf{u}} \cdot \nabla$) and arithmetic operator ($+$, $\times$). Basic operands include $\bar{q}$, $\bar{u}$, and $\bar{v}$. We adopt the iterative residual-fitting method to handle potentially lengthy equations, as proposed in [53], in which the difference between the original model output and the intermediate result is used as the target value for the subsequent iteration. The original benchmark employs a human-in-the-loop strategy to manually judge whether the terms found in the current iteration can be incorporated into the residual calculation. Unfortunately, this strategy interrupts the training process and hinders the reproducibility of the results. In DISCOVER, the searching process is totally automatic. First, extra constraints are set during the generation stage to limit the occurrence of unreasonable or uncommon combinations in turbulence, such as terms that include constants or instances where $(\nabla^2\bar{q})$ appears in $(\bar{\mathbf{u}} \cdot \nabla)$. Moreover, DISCOVER attempts to uncover the current optimal terms before computing the residual in each iteration. We further perform feature importance evaluation on each generated term based on the validation set's performance. Figure 6(a) illustrates the evolution of reward and discovered terms. The searching process terminated at iteration 6 when the newly discovered term resulted in a decrease in the overall reward of the validation set, especially for layer 2. The final discovered equation is shown below:

$$\hat{S}_q = (w_1\nabla^2 + w_2\nabla^4 + w_3\nabla^6)(\bar{\mathbf{u}} \cdot \nabla)\bar{q} + (w_4\nabla^4 + w_5\nabla^6)\bar{q} + w_6\nabla^6\bar{v}. \tag{8}$$

The first five terms uncovered can be explained by previous theoretic studies of parametrizing subgrid-scale processes [54–56], and the expression is more concise compared with results in [53]. Figures 6(b) and 6(c) illustrate the online and off-line metrics on held-out data. It can be seen that incorporating the discovered equation combination into the low-resolution model effectively improves the description of small-scale features. Further experimental details and analysis of the discovered equation can be found in Appendix G 4.

Note that subgrid parametrization is still a challenging and unresolved problem. This example demonstrates that DISCOVER is capable of uncovering a reasonable equation representation with minimal human intervention and shows great potential in solving practical applications.

DISCUSSION

We propose a framework, named DISCOVER, for exploring open-form PDEs based on enhanced deep reinforcement-learning and symbolic representations. It reduces the demand for prior knowledge and is capable of dealing with compound terms and fractional structures that conventional library-based methods cannot handle. The structure-aware architecture

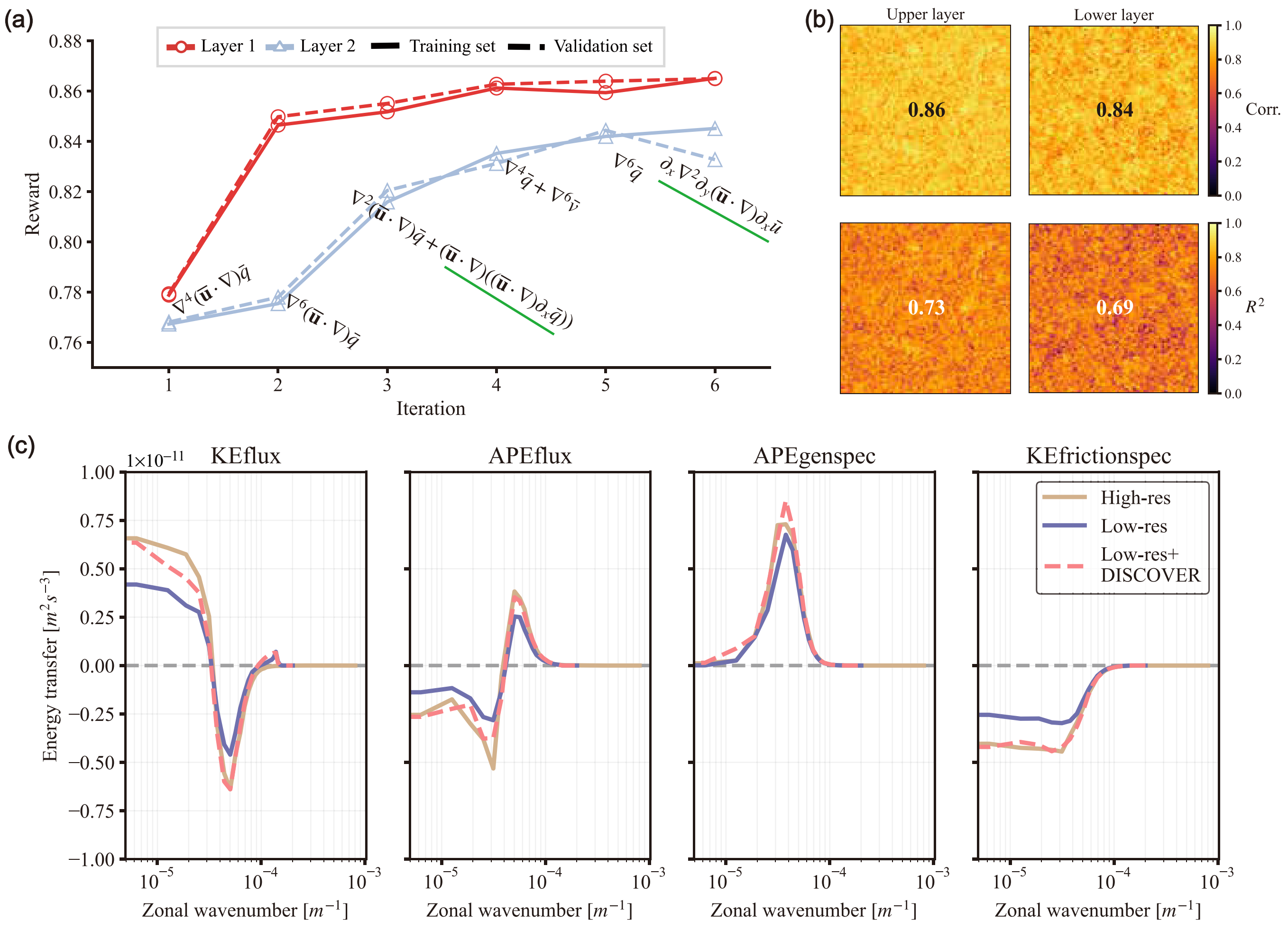


FIG. 6. Discovery of equations for unresolved subgrid-scale processes in ocean modeling. (a) Evolution of reward and terms discovered during the searching process. Terms underlined in green are removed after feature importance selection. (b) Off-line performance is measured by Pearson correlation and the coefficient of determination (R^2) between discovered expressions and target subgrid forcing. (c) On-line performance of energy budgets between high-resolution simulations, low-resolution simulations, and low-resolution simulations with the discovered forcing expression incorporated.

proposed is capable of learning the PDE expressions more effectively and can be applied to other problems with structured inputs. Furthermore, this framework achieves more efficient and stable performance by means of the neural-guided approach and sparsity-promoting methods compared to GA-based methods (e.g., SGA). In addition to some nonlinear canonical 1D problems (Burgers' equation with interaction terms, the KdV equation with high-order derivatives, the Chafee-Infante equation with derivative and exponential terms, and PDE_compound and PDE_divide with compound terms and fractional structure, respectively), we also demonstrate this framework on high-dimensional systems with high-order derivatives in the governing equations. The equation discovery for subgrid parametrization further corroborates its significant utility in real-world applications. Experiments show that the proposed framework is capable of uncovering the true equation form from noisy and sparse data, and can be applied to solving new tasks efficiently without extra physical knowledge incorporated.

Results demonstrate that the proposed framework can assist researchers in different fields to comprehend data and further uncover the underlying physical laws effectively. To extend the boundary of knowledge and serve a broader range of applications, three potential improvements should be considered in future work. Firstly, despite the greatly improved optimization efficiency of model-based methods, DISCOVER still requires considerably more time, often taking orders of magnitude longer to identify the correct equation compared to closed-library methods [1,4]. Efforts should be made to more effectively embed potential prior knowledge into the generation process of the model, thereby further reducing the search space. Secondly, if the model consistently produces bad expressions at the beginning, these inadequate PDE candidates will in turn cause the model to be updated in a direction further away from generating the true PDE expression. We still need to preserve the diversity of the generated equation forms in a reasonable manner to address the potential dilemma of exploration and exploitation. More importantly, this framework is data driven and is not applicable to scenarios with high noise and sparse data. To address this issue, the incorporation of weak-form methods [7,51] can be considered to mitigate the impact of noise on the evaluation of function terms, especially for higher-order derivative terms. The robustness may be further strengthened with discovered physical knowledge incorporated. Combining the methods of automatic machine learning [57,58] may have the potential to enhance DISCOVER to further identify open-form PDEs from high-noise and sparse data.

The implementation details of the whole process and relevant data are available on GitHub [59].

ACKNOWLEDGMENTS

This work was supported and partially funded by the National Center for Applied Mathematics Shenzhen (NCAMS), the Shenzhen Key Laboratory of Natural Gas Hydrates (Grant No. ZDSYS20200421111201738), the SUSTech – Qingdao New Energy Technology Research Institute, the China Meteorological Administration Climate Change Special Program (CMA-CCSP) (Grant No. QBZ202316), and the National Natural Science Foundation of China (Grant No. 62106116).

APPENDIX A: METHOD

In this part, we provide a detailed description of the proposed framework in terms of relevant methods, which include following four aspects: (1) the introduction of algorithms in DISCOVER; (2) the mechanism of the monotonic attention; (3) details of customized constraints and regulations; and (4) the risk-seeking gradient method.

1. Algorithm

Herein, we provide the algorithm details of DISCOVER. In the implementation process, a priority queue Q is introduced to store the fixed number of K PDE expressions with the best reward. It can be dynamically updated at each iteration. Whenever there are rewards higher than the internal expressions, the new expressions enter the queue, and expressions with lower rewards are omitted. Users can choose the best expressions by balancing accuracy and parsimony.

As shown in Algorithm 2, when PDE expression traversals are generated by the structure-aware agent, it is necessary to first reconstruct it into a tree and return the root node of it. After that, we parse the tree further and decompose it into subtrees according to the predefined addition and subtraction operators, as shown in Algorithm 3. Based on these two algorithms, we can obtain the representation of all of the function terms and then remove the function terms and PDE candidates that violate the regulations. Finally, coefficients of legal terms can be identified by STRidge [4].

2. Monotonic attention

It is well known that the LSTM recurrent neural network is effective at dealing with sequence problems, such as machine translation [60] and text generation [61]. The history information is recursive-compressed and stored in a memory cell. Three gates, including an input gate, an output gate, and a forget gate, control the flow of the information and decision-making. However, the constraint of this architecture is obvious. Prediction of the current time step largely depends

Algorithm 1. Identifying the open-form PDE from data with DISCOVER.

```
Input: symbol library: $\mathcal{L}$; time derivatives: $\mathbf{U}_t$; total
        generated expressions at each iteration: $N$;
        reward threshold: $\mathcal{R}^*$; learning rate: $\alpha$;
        $\alpha$; number of retained expressions in priority
        queue: $K$.
Output: The best PDE expression $\tau^*$.
Initialize: The structure-aware LSTM agent with
  parameters $\theta$; priority queue $Q$.
repeat
1. Generate $N$ PDE expression traversals $\mathcal{T} = \{\tau_i\}_N^{i=1}$.
2. Restore $\mathcal{T}$ into the tree structure and split PDE
   traversals into function terms $\Theta$.
3. Calculate the coefficients of the function terms $\hat{\xi}$.
        $\hat{\xi} = \text{argmin}_\xi \|\Theta\xi - \mathbf{U}_t\|_2^2 + \lambda\|\xi\|_2^2$
4. $\mathcal{T}_{val} \leftarrow \mathcal{T}$.    ▷Filtering the PDE expression traversals
   with predefined regulations.
5. Compute rewards $\mathcal{R}$ and the $1-\varepsilon$ quantile of rewards $\mathcal{R}_\varepsilon$.
6. $\mathcal{T}_\varepsilon \leftarrow \{\tau_i \in \mathcal{T}_{val} : \mathcal{R}(\tau_i) \geqslant \mathcal{R}_\varepsilon\}$.    ▷Select the
   traversals whose rewards are larger than $\mathcal{R}_\varepsilon$.
7. $\theta \leftarrow \theta + \alpha(\nabla_\theta J_{\text{risk}} + \nabla_\theta J_{\text{entropy}})$.    ▷Update the
   agent with the gradient ascent method.
8. Update $Q$ and keep $K$ PDE expressions with maximum rewards.
until The maximum reward of $Q$ is larger than $\mathcal{R}^*$.
```

Algorithm 2. Rebuilding the tree structure from the pre-order traversal.

```
Input: $\tau$                ▷ A PDE expression traversal.
Output: Root      ▷ Root node of the generated tree.
Function Rebuild_tree($\tau$):
    $S = []$           ▷ Store non-leaf nodes;
    full_node = None         ▷ The node whose arity is
      currently 0;
    while $len(N) \neq 0$ or $len(S) \neq 0$ do
        if full_node is None then
            node = $\tau$.pop(0);
            arity=node.arity
        else
            node=full_node;
            arity=0
        if arity$\geq 0$ then
            $S \leftarrow (node, arity)$;
        else
            full_node=node;
            if $len(S) \neq 0$ then
                last_op, last_arity = $S.pop(-1)$;
                last_op.children$\leftarrow$ node;
                last_arity-=1;
                if last_arity$\geq 0$ then
                    $S \leftarrow (last\_op, last\_arity)$ ;
                    full_node=None
                else
                    full_node=last_op
    return full_node
```

Algorithm 3. Splitting the PDE expression tree into function terms.

```
Input: root.  ▷ Root node of the PDE expression tree.
Output: List of root node of subtrees.
Function SplitTree(root):
    value = root.val    ▷ Fetch the symbol of the root node.;
    if value not in ['+', '−'] then
        return [root]   ▷ Terminate the split operation if the current node does not have a plus or minus symbol.
    return [SplitTree(root.children[0]), SplitTree(root.children[1])]
```

on the adjacent units, and recursively updating the information destroys the structural information of the input. Furthermore, the limited storage capacity of memory cells can result in escalating information loss as the length of the sequence being processed increases. Some attempts have been made to incorporate soft attention to comprehend internal structures and directly select useful information from the previous tokens [62–64].

In order to capture the long-distance dependencies and structural information, we wrapped the original LSTM with a monotonic attention layer (MAL), which mainly draws on the architecture and configuration of the "long short-term memory-network" (LSTMN) proposed for machine reading [64]. Attention memory is utilized here to simulate the human brain to read historical information and discern the relationships within it. The architecture of MAL is illustrated in Fig. 7. Compared with the standard LSTM, an extra hidden vector $\tilde{h}_t$ is introduced to store the relations between tokens. At time step t, the attention function can be calculated by

$$[h_t, c_t] = \text{LSTM}\,(x_t, [h_{t-1}, c_{t-1}]), \tag{A1}$$

$$a_i^t = W_v^T \tanh(W_q[h_t, c_t] + W_k \text{O}_i), \tag{A2}$$

$$s_i^t = \text{softmax}\left(a_i^t\right), \tag{A3}$$

where W_q, W_k, and W_v are the parameter matrices used to linearly project queries, keys, and their output, respectively; and the state vector s_t denotes a probability distribution over the previous inputs to measure the degree of attention to historical information. s_t is used to calculate our new hidden vector. By combining $\tilde{h}_t$ containing the relation information and the original hidden vector h_t, the final output can be represented as follows:

$$\tilde{h}_t = \sum_i^{t-1} s_i^t \text{O}_i, \tag{A4}$$

$$\text{O}_t = W_o[h_t, \tilde{h}_t]. \tag{A5}$$

The specific implementation refers to the source code in TensorFlow [65], with two modifications to more effectively integrate structured inputs. First, the structured inputs, namely, the parent and sibling nodes of the current timestep, are merged and fed into the LSTM cell. Second, to preserve the completeness of the input, the aggregated historical information $\tilde{h}_t$ is solely employed in computing the final output and is not recycled as input for the prediction of the next step.

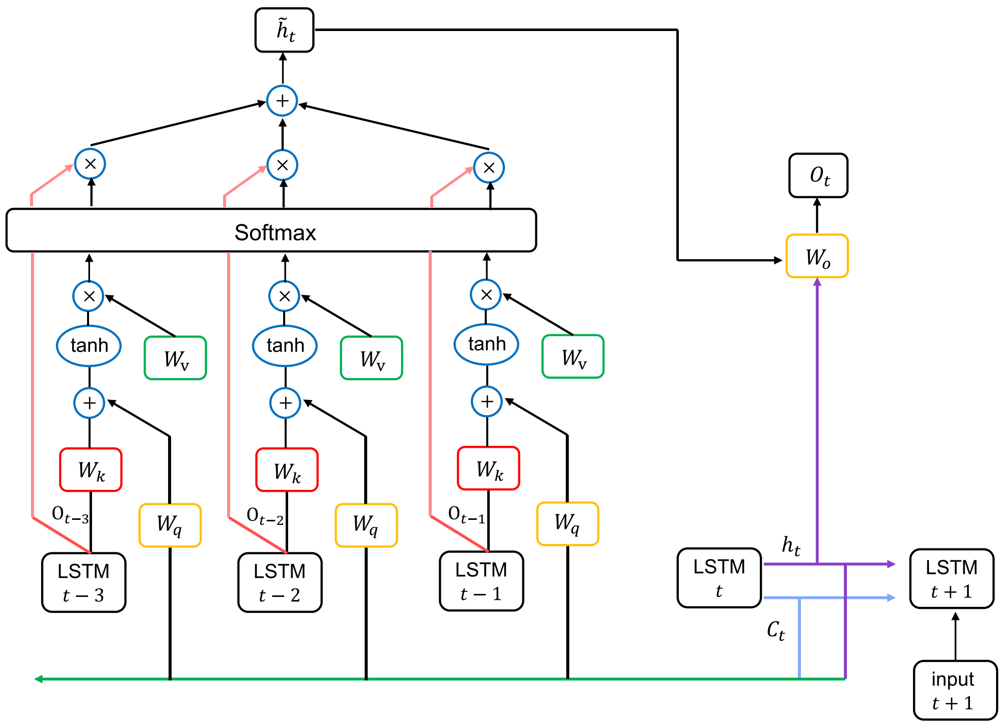


FIG. 7. The architecture of monotonic attention. Only the information of the past three time steps is integrated here. In fact, the time-span can be set longer, so that it can focus on further historical information.

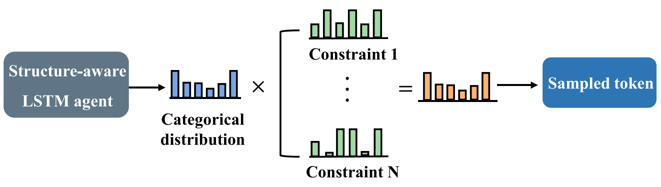


FIG. 8. Illustration of imposing constraints during the generation process.

Specific implementation details can be found in the provided code.

3. Constraints and regulations

The main purpose of constraints and regulations is to prevent the framework from generating equations that violate the laws of mathematics and physical laws, and reduce the search space. Among them, constraints are directly applied to the probability distribution of the symbol library at the regressive generation process, which is shown in Fig. 8. Specifically, we use hard constraints to prevent the generation of unreasonable symbols. For any unreasonable symbol, its logit (unnormalized probability) is set to negative infinity, resulting in a final output probability of zero. Moreover, soft constraints are applied to regulate the length of the generated traversal [66]. This is accomplished by making the probability of nonterminal symbols adjustable during the generation process. The logits of nonterminal symbols are determined using the formula below:

$$logit = L_{\text{out}} - \frac{(t-L)^2}{2s} \cdot \mathbf{1}_{t>L} \tag{A6}$$

where L_{out} denotes the output from LSTM, and it is added by a penalized term when the current generation step t exceeds the preset length L (whose default value is 10); and s is a hyperparameter to control the penalty's intensity with a default value of 5. As the length of the generated traversal increases, the generation probability of non-terminal nodes decreases, which speeds up the generation of a complete PDE tree. When the maximum length limit is reached, the hard constraint is applied, and the output probability of non-terminal nodes is set to 0.

Constraints in the framework include:

(1) Total length of the sequence is less than 64.

(2) Number of plus and minus operators is less than 10.

(3) Relationship limits between differential operators and spatial inputs: the right child node of partial differential operators (e.g., ∂) must be space variables (e.g., x); the left child node of ∂ cannot be space variables; and the plus and the minus operator cannot appear in the descendants of ∂ (optional).

Regulations are applied to both function terms and the whole PDE expression, including:

(1) Descendants of partial differential operators do not contain state variables (e.g., u).

(2) Arithmetic underflow and overflow.

(3) Numerical errors during conducting sparsity-promoting methods.

(4) Coefficient of the corresponding function term is less than a default number (e.g., 1×10^{-5}).

APPENDIX B: DATA DESCRIPTION

The data used in this paper are mainly divided into two categories: 1D and 2D systems. The data descriptions of the 1D canonical systems can be found in SGA [13]. To verify the effectiveness of our framework in mining high-dimensional and high-order nonlinear systems, we introduce the Allen-Cahn and Cahn-Hilliard equations in 2D dynamic systems. They are originally introduced to describe the nonconservative and conservative phase variables in the phase separation process, respectively. Both models are recognized as gradient flow systems. Assume that the Ginzburg–Landau free energy functional takes the following form:

$$F = \int_{\omega} \frac{\gamma_1}{2}|\nabla u|^2 + \frac{\gamma_2}{4}(u^2-1)d\mathbf{x}. \tag{B1}$$

The Allen-Cahn equation can be obtained as an L^2 gradient flow with the following form:

$$u_t = \gamma_1 \Delta u + \gamma_2(u-u^3) \tag{B2}$$

where $\gamma_1 = 0.001$; and $\gamma_2 = 1$. The Cahn-Hilliard equation can be obtained as a H^{-1} gradient flow, as shown below:

$$u_t = \Delta(-\gamma_1 \Delta u + \gamma_2(u-u^3)) \tag{B3}$$

where $\gamma_1 = 0.5$; and $\gamma_2 = -1$. The specific discretization details of the two equations are given below:

(1) Allen-Cahn equation: The spatial domain is taken as $x \in [0, 1]^2$ with 64 points along each axis. The temporal domain is taken as $t \in (0, 5]$ with 100 points for discretization in time-scale. The initial condition is set as $u(0, x_1, x_2) = \sin(4\pi x_1)\cos(4\pi x_2)$, where (x_1, x_2) are discrete points in the spatial domain. The periodic boundary conditions are imposed with $u^d(t, -1) = u^d(t, 1)$, for $d = 0, 1$.

(2) Cahn-Hilliard equation: The spatial domain is taken as $x \in [0, 64]^2$ with 64 points along each axis. The temporal domain is taken as $t \in (0, 10]$ with 500 points for discretization in the time-scale. The random noisy initial condition is set as $u(0, x_1, x_2) = \text{rand}() - 0.5$, where (x_1, x_2) are discrete points in the spatial domain. The periodic boundary conditions applied are identical to those used in the Allen-Cahn equation.

APPENDIX C: HYPERPARAMETERS

The default hyperparameters used to mine the above PDEs are shown in Table IV. As mentioned above, we use the plus or minus operator that appears at the top level of the tree as the identifier to split the equation into several function terms. We require that the number of their occurrences should not exceed five, which means that the generated expressions can only be spliced into at most six function terms. The parsimony penalty factor which guarantees the simplicity of the equation is set to 0.01. In each iteration, the agent will generate a total of $N = 500$ PDE expressions. Finally, after filtering the illegal expressions and low rewards, only 2% ($\varepsilon = 0.02$) of the total expressions, i.e., 10 equations with the highest reward, are selected for the update of the agent. In the optimization process,

TABLE IV. Default hyperparameter settings for discovering open-form PDEs.

Hyperparameter	Default value	Definition
N_{subtree}	6	Maximum number of function terms
D_{subtree}	8	Maximum depth of subtrees
ζ_1	0.01	Parsimony penalty factor for redundant function terms
ζ_2	0.0001	Parsimony penalty factor for unnecessary structures
N	500	Total generated expressions at each iteration
ε	0.02	Threshold of reserved expressions
λ	0	Weight of the STRidge regularization term
tol	1×10^{-4}	Threshold of weights for reserved function terms
$\lambda_{\mathcal{H}}$	0.03	Coefficients of entropy loss
λ_{pg}	1	Coefficients of policy gradient loss
T	20	Time span of the monotonic attention

the coefficients of entropy loss and policy gradient loss are set to 0.03 and 1, respectively.

Note that the majority of the hyperparameters remain constant across all experiments, with the exception of two parameters: the total number of expressions generated at each iteration (N), and the threshold for reserved expressions (ε). These two parameters are adjusted in the two-dimensional experiments and the subsequent real-world application. During the process of uncovering the Allen-Cahn equation, the Cahn-Hilliard equation, and the subgrid force, we adjust the value of N to 2500, 4000, and 1000, respectively, and adjust ε correspondingly to 0.004, 0.003, and 0.005. This adjustment is primarily necessitated by the increased complexity and difficulty of the discovery tasks. These three experiments are associated with mining equations from high-dimensional nonlinear systems that involve multiple state variables. The potential combinations of different operators and operands become more diverse, thereby substantially expanding the search space. As a result, it becomes essential to augment the number of generated expressions in order to maximize the probability of yielding high-quality samples, whilst avoiding being stuck in the local optima. The impact of these two hyperparameters is further discussed and analyzed in the sensitivity analysis section.

APPENDIX D: RESULTS BY THE CLOSED-LIBRARY METHOD

In this section, we further present the computational efficiency of closed-library methods for the identification of canonical dynamical systems, excluding the PDE_divide. PySINDy [67], in which the STRidge algorithm [4] is implemented, is utilized for the equation identification. We set the upper bounds for both the power and derivative order of the state variable at five, creating a closed library with 30 basis function terms.

Table V demonstrates the coefficient error of discovered results and the running times by PySINDy. It can be seen that in noise-free scenarios, the closed-library method is capable of accurately identifying the corresponding equations, while exhibiting a substantial reduction in computational time. This efficiency gain is primarily attributable to the strong assumption, which posits that all of the components in the governing equation are monomials of state variables and their partial derivatives. Consequently, closed-library methods are more suitable for experienced experts to rapidly uncover governing equations within well-understood domains. The deficiency in expressive power makes closed-library methods unable to find complex equations, as shown in Table III.

In contrast, DISCOVER diminishes reliance on prior knowledge and enables the discovery of open-form PDEs at the expense of a larger search space and more execution time. Further improvements in computational efficiency are needed in future research for a broader range of applications.

APPENDIX E: COMPARISON TO DSR

DSR [24] is a powerful framework designed for symbolic regression tasks based on deep reinforcement-learning methods. However, compared with our framework, it is insufficient for identifying the governing equations of dynamic systems, especially when identifying PDEs. First, the current DSR neither incorporates the computation of partial differential operators, nor is it able to handle sparse and noisy data. Furthermore, even if DSR is modified to be compatible with PDE discovery tasks by including library expansion, and the introduction of regulations and constraints, it is still less efficient and effective to deal with relatively simple dynamic systems. As shown in Figs. 9(a) and 9(b), the optimization speed of the modified DSR (DSR_new) is significantly slower than DISCOVER. After 100 iterations, the maximum reward is still at a low level. This is mainly because the DSR represents coefficients through the constant operator, which significantly enlarges the search space, and thus proves to be inefficient for determining the structure of PDEs. In addition, DSR determines the constants through an iterative optimization process that aims to maximize the total reward. This method incurs prohibitive computational costs,

TABLE V. Discovery results and running time of PySINDy.

	Coefficient error	Running time (s)
KdV	$0.04 \pm 0.0\%$	6.03
Burgers	$1.25 \pm 1.61\%$	5.88
Chafee-Infante	$0.06 \pm 0.04\%$	7.02
PDE_compound	$0.04 \pm 0.0\%$	5.52

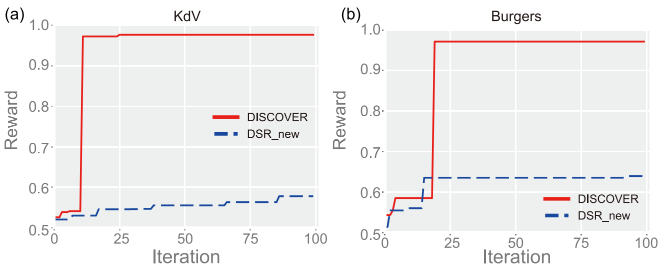


FIG. 9. Maximum reward distribution of DISCOVER and the modified DSR (DSR_new). (a) the KdV equation. (b) Burgers' equation.

particularly for high-dimensional nonlinear dynamic systems with large data volumes. Taking Burgers' equation as an example, DISCOVER completes 100 time-step iterations in less than one-tenth the time required by DSR. This contrast in efficiency highlights DISCOVER's superior efficiency and scalability, making it particularly well-suited for PDE discovery tasks.

APPENDIX F: THE OPTIMIZATION PROCESS OF DISCOVER

To demonstrate the concrete details of the optimization process, we provide the reward distribution of the best performing, top ε, and the mean of the top ε in each batch during the training process for different systems. It can be seen from Figs. 10 and 11 that as the iteration progresses, the best-case reward increases in a stepwise manner, and there is a sudden spike for the Burgers' equation and the KdV equation. The risk-seeking policy gradient method is aimed at improving the better-case performance, and thus it is not necessary for all of the generated equations to have high rewards. Note that the PDE_compound equation is relatively simple, and the correct equation form can be identified in the first few rounds of iterations. The PDE_divide is taken as an example to show the evolution process of the generated equation expressions during the iteration process, as shown in Fig. 11(b). It can be seen that from the 1st to the 99th iteration, the composition of the equation terms gradually approaches the correct one. The increase in reward gradually benefits from the increase in accuracy. From the 99th iteration to the 136th iteration, in addition to a further gain of accuracy, the parsimony of the expressions is also considered. Therefore, in the process of revealing equations from the data, when the accuracy of the generated terms is similar, the expressions with fewer terms have greater rewards. This also ensures that the final equation form is both accurate and parsimonious. Figure 12 illustrates the reward distribution of the 2D systems. For the Cahn-Hilliard equation, the inclusion of the Laplace operator in the library allows the agent to identify the correct equation within 200 iterations, which can be seen in Fig. 12(b). When only the first- and second-order differential operators are used, high-dimensional operations can only be represented by the nested structures from these operators. The numerically equivalent equation form is verbose, and it takes more than four times the number of iterations of the former one. The identified equation is shown below:

$$u_t = 0.999(u^3)_{xx} + 0.999(u^3)_{yy} - 0.999u_{xx} - 1.000u_{yy} - 0.499(u_{xx})_{xx} - 0.500(u_{xx})_{yy}. \quad \text{(F1)}$$

APPENDIX G: SUPPLEMENTARY EXPERIMENTS

In the experiment parts, we first discuss the structure-aware agent designed in DISCOVER, and further compare it with the standard LSTM agent. The results demonstrate that the structure-aware agent exhibits great superiority in capturing local information and long-distance information. We also discuss the effect of the number and utilization rate of generated samples on the efficiency of mining equations. Finally, a more detailed explanation of how our framework makes contributions to subgrid parametrization is provided.

1. Effect of noise levels and measurement points

Since observations obtained from real scenes are often sparse and noisy, it is necessary to test whether DISCOVER can mine the correct form of equations under different volumes of data and noise levels. During the evaluation stage, the finite difference method is utilized to calculate derivatives, which are sensitive to noise [29]. Therefore, we built a fully connected neural network to serve as a surrogate model, which not only smoothens available noisy data, but also facilitates the generation of metadata across different spatial-temporal locations. It is worth noting that PDE-FIND [4] utilizes polynomial interpolation to deal with noisy data, which is effective

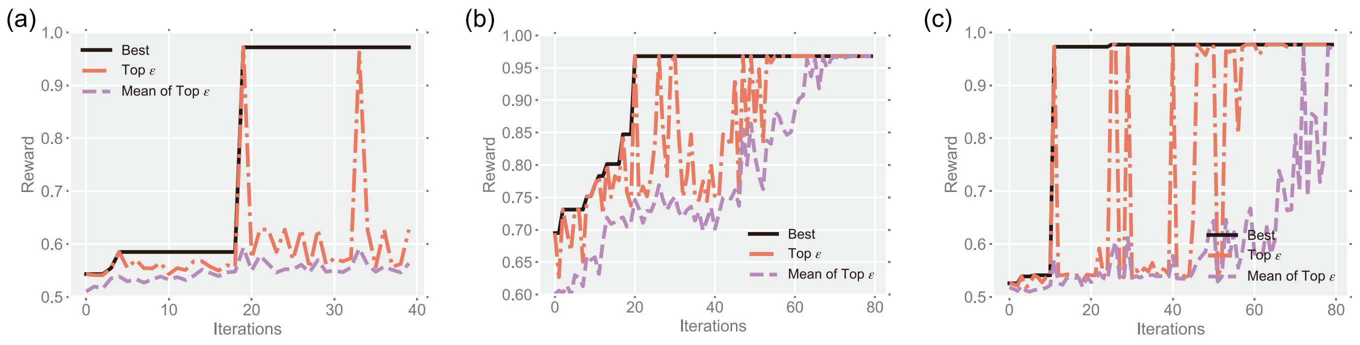


FIG. 10. Rewards distribution of the best case of all time, top ϵ, and mean of top ϵ of the samples. (a) Burgers' equation. (b) The Chafee-Infante equation. (c) The KdV equation.

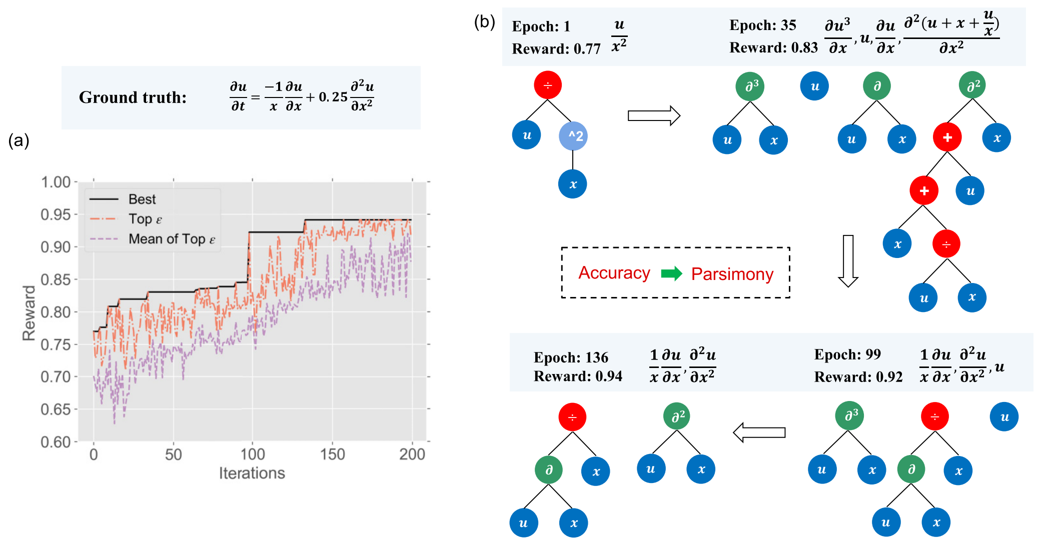


FIG. 11. The optimization process of PDE_divide. (a) Reward distribution. (b) Evolution of the function terms.

but limited to scenarios with a complete basis function library. All of the derivatives can be obtained by a one-time calculation. However, this approach is not feasible for our framework due to the excessive computational load associated with repeated derivatives evaluation. Additionally, polynomial interpolation struggles with processing boundary points. The metadata method is relatively simple and only involves the pre-processing part [29]. Once the noisy data are smoothed, we can seamlessly integrate automatic differentiation to evaluate derivatives.

To preprocess measurements, a fully connected feedforward neural network (FNN) is constructed to map the system inputs (i.e., spatial coordinate x and time t) to the state of interest u. The architecture of FNN is standardized across different systems, consisting of an input layer, three hidden layers with 64 neurons each, and an output layer. The Sine function is utilized as the activation function. We employ the mean square error (MSE) loss function to quantify the discrepancy between the measured data and the model's predictions. To ensure generalizability and prevent overfitting, the dataset is split into training and validation sets with a ratio of 8:2, and an early stopping mechanism is implemented. The predicted outputs are utilized for the final derivative evaluation.

The relative l_2 error defined as $||\hat{\xi}-\xi_{\text{true}}||_1/||\xi_{\text{true}}||_1$ is utilized to assess the accuracy of identified coefficients of function terms. The identified equations and error rates across different systems under different noise levels are shown in Table VI. Only 80% of the total measurements were randomly sampled and utilized to train the surrogate model. It can be seen that DISCOVER is able to identify the correct equation form of the KdV and Burgers' equations with a relatively small error even when the data are subjected to noise levels as high as 5%. For the other three equations, satisfactory results can be obtained only when the noise level is restricted to 1%. Notably, in two-dimensional cases, although the correct equation structure of the Allen-Cahn equation can be identified at 1% noise, a significant increase is observed in the error of coefficients. For the Cahn-Hilliard equation, due to the existence of the fourth-order derivative and compound structure, the correct equation form cannot be found with current settings when the data are noisy.

It is worth noting that our method is far from perfect. The metadata method is still unable to handle high-noise cases. Automatic differentiation can alleviate this problem to a large extent; however, evaluating function terms of arbitrary form remains a challenge that necessitates the incorporation of automatic machine learning [57]. Meanwhile, available physical

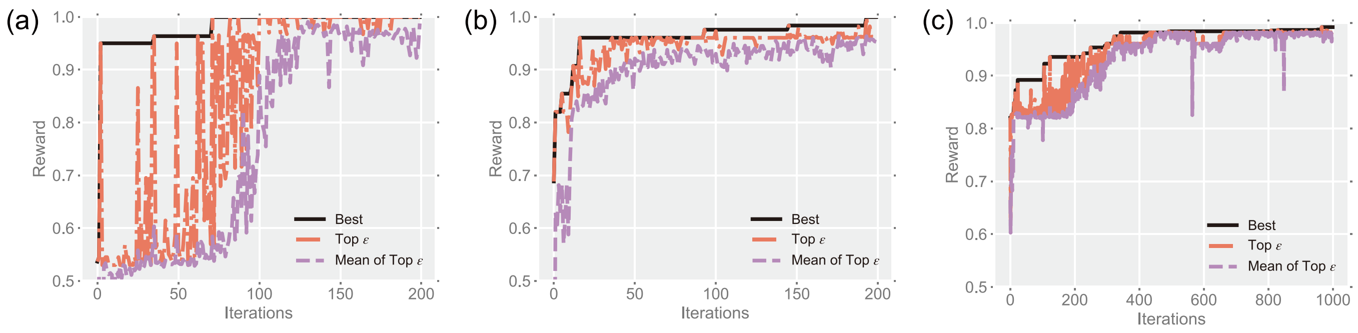


FIG. 12. Rewards distribution of the best case of all time, top ϵ, and mean of top ϵ of the samples. (a) The Allen-Cahn equation. (b) The Cahn-Hilliard equation (with Laplace operator). (c) The Cahn-Hilliard equation (without Laplace operator).

TABLE VI. Summary of discovered results for different PDEs of mathematical physics under different noise levels.

	Noise level	Identified PDE	Error (%)
KdV		Correct PDE: $u_t = -u \times u_x - 0.0025u_{xxx}$	
	Clean data	$u_t = -0.5001(u \times u)_x - 0.0025u_{xxx}$	0.09 ± 0.07
	1% noise	$u_t = -0.4983(u \times u)_x - 0.0025u_{xxx}$	0.31 ± 0.04
	10% noise	$u_t = -0.9748u \times u_x - 0.0024u_{xxx}$	2.50 ± 0.028
Burgers		Correct PDE: $u_t = -uu_x + 0.1u_{xx}$	
	Clean data	$u_t = -1.0010uu_x + 0.1024u_{xx}$	1.25 ± 1.61
	1% noise	$u_t = -0.4992(u \times u)_x + 0.0982u_{xx}$	0.95 ± 1.12
	10% noise	$u_t = -0.4886(u \times u)_x + 0.0943u_{xx}$	3.94 ± 2.35
Chafee-Infante		Correct PDE: $u_t = u_{xx} + u - u^3$	
	Clean data	$u_t = 1.0002u_{xx} - 1.0008u + 1.0004u^3$	0.04 ± 0.03
	1% noise	$u_t = 0.9866u_{xx} - 0.9862u + 0.9877u^3$	1.32 ± 0.08
	10% noise	$u_t = 0.9777u_{xx} - 0.9766u + 0.9869u^3$	1.96 ± 0.56
PDE_compound		Correct PDE: $u_t = (uu_x)_x$	
	Clean data	$u_t = 0.5002(u^2)_{xx}$	0.04
	1% noise	$u_t = 0.5003(u^2)_{xx}$	0.05
PDE_divide		Correct PDE: $u_t = -u_x/x + 0.25u_{xx}$	
	Clean data	$u_t = -0.9979u_x/x + 0.2498u_{xx}$	0.14 ± 0.10
	1% noise	$u_t = -0.9803u_x/x + 0.2478u_{xx}$	1.42 ± 0.78
Allen-Cahn		Correct PDE: $u_t = 0.001\Delta u - u^3 + u$	
	Clean data	$u_t = 0.001(u_{xx} + u_{yy}) - 0.999u^3 + 1.000u$	0.025 ± 0.05
	1% noise	$u_t = 0.0007(u_{xx} + u_{yy}) - 0.884u^3 + 0.870u$	21.72 ± 10.94

constraints need to be incorporated in FNN in order to enhance the robustness of the model to noise and minimize the reliance on extensive datasets.

2. Comparison between structure-aware LSTM and standard LSTM

Our model introduces structured information in the agent, which allows LSTM to attend to the previously generated tokens and equation structure when predicting the current output. However, the standard LSTM agent can only obtain the information from the last token and the composite history information stored in the memory cell. To demonstrate the rationality and superiority of our model, we compare the performance of the two different types of LSTM agents on four equations while maintaining consistent hyperparameters. The relatively simple PDE_compound equation is not considered here. Figures 13(a)–13(d) show the distribution of the rewards under each agent setting during the training process. Owing to the enhanced ability to capture structural and long-range information within equations, our model produces expressions that more closely approximate the true forms, thereby yielding greater rewards in each iteration. Figure 13(e) illustrates the iterations required to identify the optimal equation form. Notably, the structure-aware LSTM achieves this with fewer iterations and maintains almost the same computational load.

3. Sensitivity analysis

Based on the theory of risk-seeking policy gradient, only ε fraction of the best expressions is selected to update the agent during the training process to improve the best-case performance. The training effect and calculation time are directly impacted by the quality and number of the expressions at each iteration. Two relevant hyperparameters are considered, including the total number of generated expressions at each iteration N, and the quantile ε of the rewards used to filter

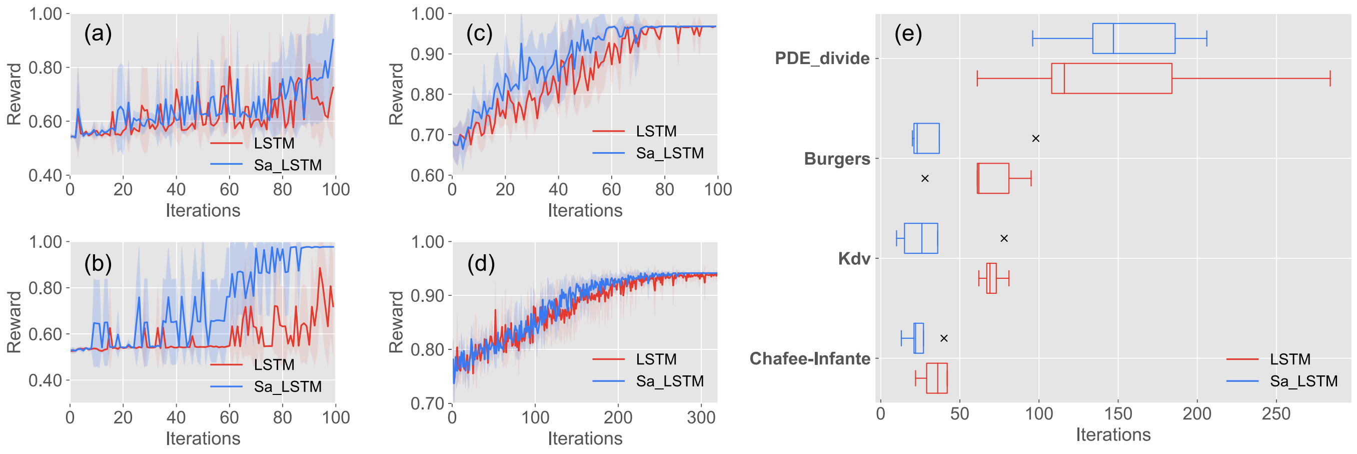


FIG. 13. Maximum reward distribution for four PDEs. (a) Burgers' equation. (b) The KdV equation. (c) The Chafee-Infante equation. (d) PDE_divide. (e) Iterations needed to discover the correct equation. Sa_LSTM refers to the proposed structure-aware LSTM agent.

expressions. Then, we focus on discussing their specific impact on the entire optimization process.

Figure 14 illustrates the maximum reward distribution of four PDEs with different N. It is obvious that the more expressions generated in each round, the better the ultimate selected expressions to update the agent, and the fewer iterations required to find the optimal equation. However, it is worth noting that generating more expressions also requires more computation resources. A trade-off must be established between the volume of expressions generated and the computational time invested to ensure an optimal and efficient discovery process. In the actual training process, generating as many expressions as possible, especially at the beginning, is crucial to avoid getting stuck in local optima.

The other parameter ε determines the proportion of the expressions generated in each round that can actually be used to update the agent. When the parameter is set to 0, all expressions will be used for the agent update, which is the standard policy gradient approach. As shown in Fig. 15, choosing a relatively small and reasonable value for this parameter is necessary for the agent to learn the optimal solution, which can accelerate the training process. However, with the policy gradient method, it is expected that the rewards of all expressions in each batch are maximized, which obviously slows down the update speed and may even fail to find the final correct expression.

4. Equation-discovery parametrization in oceanic modeling

In climate modeling, high-resolution simulations are usually time consuming and computationally expensive, while coarse-resolution simulations may fail to describe small-scale features. Subgrid parametrization is a common way of accounting for the impact of unresolved processes. By

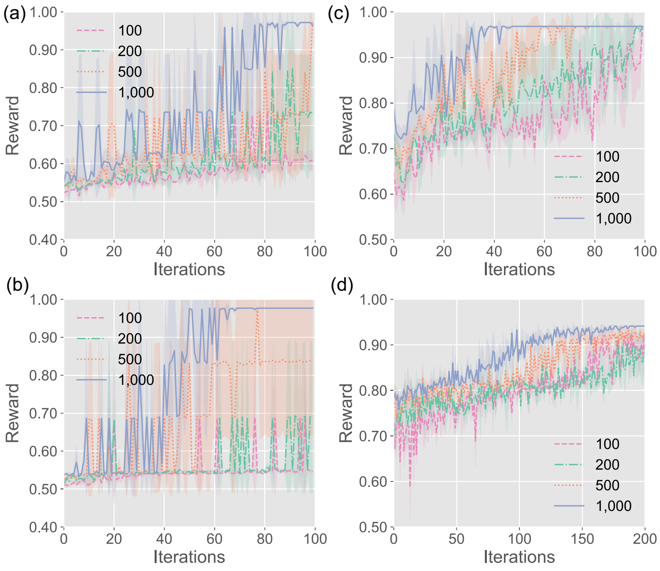


FIG. 14. Maximum reward distribution with different N for four PDEs. (a) Burgers' equation. (b) The KdV equation. (c) The Chafee-Infante equation. (d) PDE_divide.

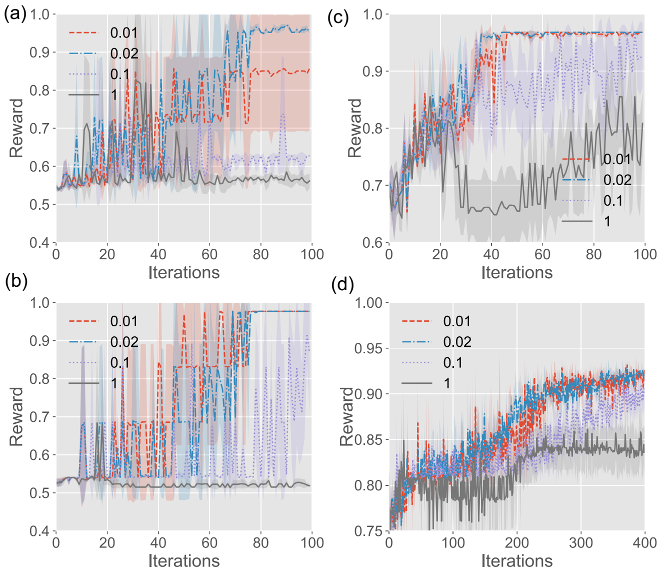


FIG. 15. Maximum reward distribution with different ε for four PDEs. (a) Burgers' equation. (b) The KdV equation. (c) The Chafee-Infante equation. (d) PDE_divide.

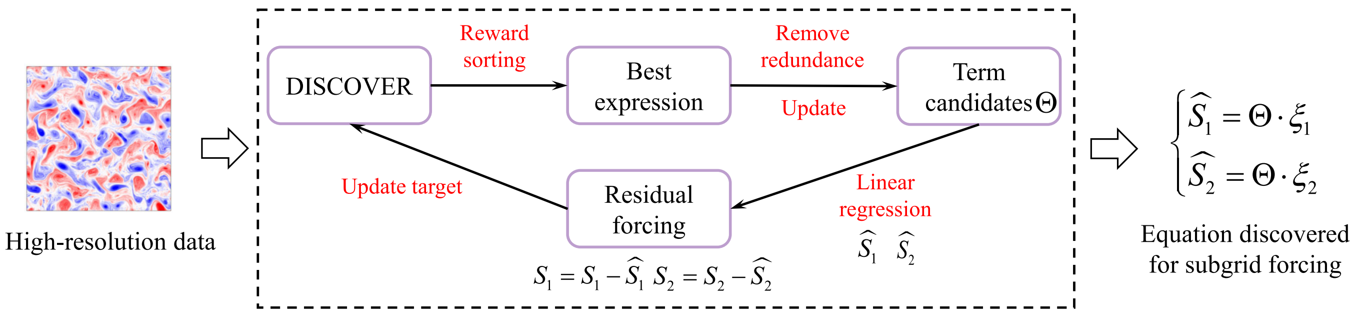

High-resolution data
DISCOVER
Reward sorting
Best expression
Remove redundance
Update
Term candidates Θ
Update target
Residual forcing
Linear regression
$\widehat{S}_1$ $\widehat{S}_2$
$S_1 = S_1 - \widehat{S}_1$ $S_2 = S_2 - \widehat{S}_2$
$\widehat{S}_1 = \Theta \cdot \xi_1$
$\widehat{S}_2 = \Theta \cdot \xi_2$
Equation discovered for subgrid forcing

FIG. 16. Workflow of the equation-discovery process by utilizing DISCOVER.

incorporating a reasonable forcing term to parametrize the subgrid-scale process, it is expected to improve the performance of low-resolution models and avoid the high computational intensity of high-resolution simulations [68]. In our experiment, we intend to utilize DISCOVER to uncover equations that can parametrize the behavior of subgrid-scale processes. We use oceanographic data from a benchmark problem [53] that comprises two flow regimes: the eddy and jet configurations. All of the training and validation data are taken from the eddy configuration. Note that high-resolution and low-resolution modeling refers to simulations with a grid size of 256 × 256 and 64 × 64, respectively. The target subgrid forcing is diagnosed by filtering and coarse-grained high-resolution simulations.

To accommodate the broad range of magnitudes present in the calculations, the correlation between the discovered result and the subgrid forcing is utilized as the training target. We adopt the iterative residual-fitting method to handle potentially lengthy equations proposed in [53], in which the difference between the original model output and the intermediate result is used as the target value for the next iteration. However, this method is vulnerable to unreasonable redundant terms, which can adversely affect the results of subsequent iteration steps. To mitigate this issue, the original benchmark employs a human-in-the-loop strategy to manually determine whether the terms found in the current iteration can be incorporated into the residual calculation. Unfortunately, this strategy interrupts the training process and poses challenges to the reproducibility of the results.

To reduce manual intervention and discover unified equations, we made adjustments in three aspects compared with the benchmark. First, we considered the correlation performance of the upper and lower layers comprehensively during the reward calculation process.

$$R = \frac{1}{2}\left(\frac{\mathrm{Cov}(S_1, \hat{S}_1)}{\sigma_{S_1}\sigma_{\hat{S}_1}} + \frac{\mathrm{Cov}(S_2, \hat{S}_2)}{\sigma_{S_2}\sigma_{\hat{S}_2}}\right) \tag{G1}$$

where S and $\hat{S}$ refer to the target subgrid forcing and predicted value, respectively; and the subscripts represent different layers. Second, extra constraints are imposed during the generation stage to prevent the emergence of unreasonable or uncommon combinations in turbulence, such as terms that comprise constants or instances where $(\nabla^2\bar{q})$ appears within $(\bar{\mathbf{u}} \cdot \nabla)$. Note that imposing constraints embodies the controllability of DISCOVER in generating candidates and is not supported in the genetic algorithm utilized in the benchmark [53]. This operation effectively eliminates the possible redundant terms. Finally, we conducted a feature importance evaluation to further remove unnecessary terms during each residual calculation. Terms that cause a decrease or trivial increase in reward are removed. The workflow is illustrated in Fig. 16.

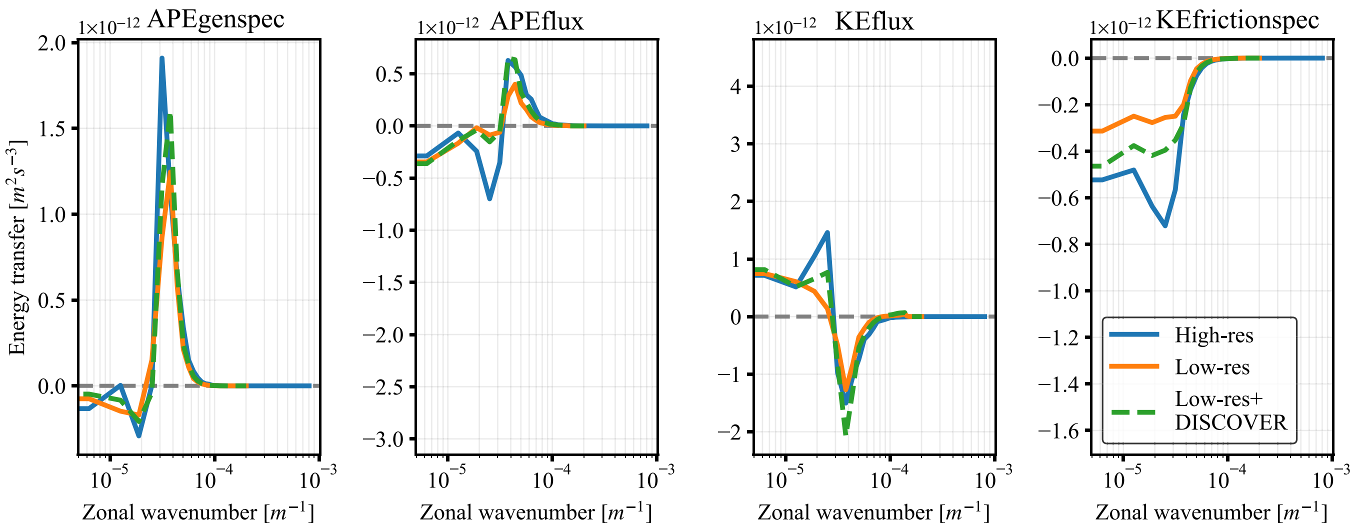


FIG. 17. Online metrics on jet configuration data.

As a comparison, we also present the benchmark results,

$$S_q^{\mathrm{GP}} = (w_1\nabla^2 + w_2\nabla^4 + w_3\nabla^6)(\bar{\mathbf{u}}\cdot\nabla)\bar{q} + (w_4\nabla^4 + w_5\nabla^6)\bar{q} + (\bar{\mathbf{u}}\cdot\nabla)^2\nabla^2(w_6\bar{v}_x + w_7\bar{u}_y). \qquad \text{(G2)}$$

The first five terms of our model align with the findings from previous theoretical studies [54–56]. For example, the last two terms refer to the dissipation and redistribution of energy in different scales. It proves that the discovered results are reasonable in theory. For evaluating these results, we employ both off-line and online metrics. Off-line metrics are utilized to assess the approximation accuracy to the target values, while online metrics are employed to evaluate the difference between the low-resolution simulations with the subgrid parametrization incorporated and high-resolution simulations. The performances in both online and off-line scenarios for the jet configuration are shown in Figs. 17 and 18, respectively. It is evident that the equation-based parametrizations are effective and perform well compared to low-resolution simulations. This not only affirms the interpretability of the discovered equations, but also their strong potential for robust generalization.

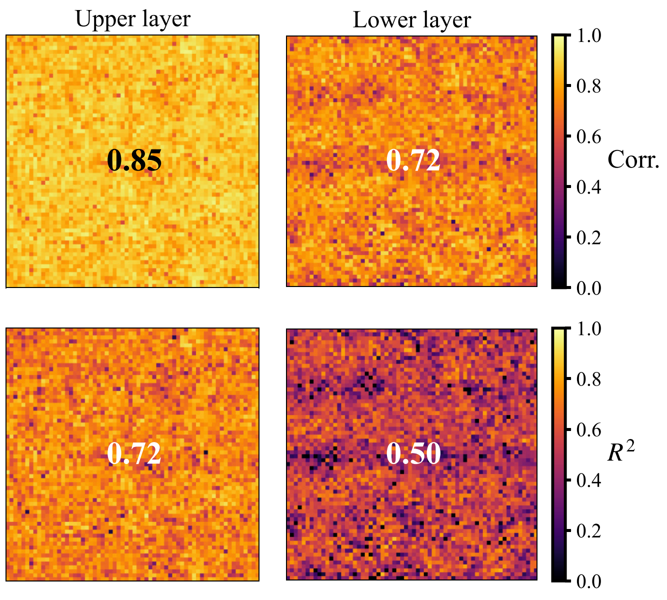


FIG. 18. Offline metrics on jet configuration data.

[1] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA **113**, 3932 (2016).

[2] A. Davies, P. Veličković, L. Buesing, S. Blackwell, D. Zheng, N. Tomašev, R. Tanburn, P. Battaglia, C. Blundell, A. Juhász *et al.*, Advancing mathematics by guiding human intuition with AI, Nature (London) **600**, 70 (2021).

[3] F. P. Kemeth, T. Bertalan, T. Thiem, F. Dietrich, S. J. Moon, C. R. Laing, and I. G. Kevrekidis, Learning emergent partial differential equations in a learned emergent space, Nat. Commun. **13**, 3318 (2022).

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

[5] D. E. Shea, S. L. Brunton, and J. N. Kutz, SINDy-BVP: Sparse identification of nonlinear dynamics for boundary value problems, Phys. Rev. Res. **3**, 023255 (2021).

[6] K. Kaheman, J. N. Kutz, and S. L. Brunton, SINDy-PI: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics, Proc. R. Soc. A. **476**, 20200279 (2020).

[7] D. A. Messenger and D. M. Bortz, Weak SINDy for partial differential equations, J. Comput. Phys. **443**, 110525 (2021).

[8] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton, Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control, Proc. R. Soc. A. **478**, 20210904 (2022).

[9] Z. Chen, Y. Liu, and H. Sun, Physics-informed learning of governing equations from scarce data, Nat. Commun. **12**, 6136 (2021).

[10] Z. Long, Y. Lu, X. Ma, and B. Dong, PDE-Net: Learning PDEs from data, in *Proceedings of the International Conference on Machine Learning* (PMLR, Stockholm, Sweden, 2018), Vol. 80, p. 3208.

[11] Z. Long, Y. Lu, and B. Dong, PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. **399**, 108925 (2019).

[12] H. Xu, H. Chang, and D. Zhang, DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm, J. Comput. Phys. **418**, 109584 (2020).

[13] Y. Chen, Y. Luo, Q. Liu, H. Xu, and D. Zhang, Symbolic genetic algorithm for discovering open-form partial differential equations (SGA-PDE), Phys. Rev. Res. **4**, 023174 (2022).

[14] Y. Chen and D. Zhang, Integration of knowledge and data in machine learning, arXiv:2202.10337.

[15] M. Schmidt and H. Lipson, Distilling free-form natural laws from experimental data, Science **324**, 81 (2009).

[16] D. A. Augusto and H. J. Barbosa, Symbolic regression via genetic programming, in *Proceedings of the Sixth Brazilian Symposium on Neural Networks* (IEEE, Rio de Janeiro, Brazil, 2000), Vol. 1, pp. 173–178.

[17] S. Sun, R. Ouyang, B. Zhang, and T.-Y. Zhang, Data-driven discovery of formulas by symbolic regression, MRS Bull. **44**, 559 (2019).

[18] S. S. M. Astarabadi and M. M. Ebadzadeh, Genetic programming performance prediction and its application for symbolic regression problems, Inf. Sci. **502**, 418 (2019).

[19] M. A. Haeri, M. M. Ebadzadeh, and G. Folino, Statistical genetic programming for symbolic regression, Appl. Soft Comput. **60**, 447 (2017).

[20] G. Martius and C. H. Lampert, Extrapolation and learning equations, arXiv:1610.02995.

[21] S. Sahoo, C. Lampert, and G. Martius, Learning equations for extrapolation and control, in *International Conference on*

Machine Learning (PMLR, Stockholm, Sweden, 2018), Vol. 80, pp. 4442–4450.

[22] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Čeperić, and M. Soljačić, Integration of neural network-based symbolic regression in deep learning for scientific discovery, IEEE Trans. Neural Netw. Learning Syst. **32**, 4166 (2020).

[23] F. Sun, Y. Liu, J.-X. Wang, and H. Sun, Symbolic physics learner: Discovering governing equations via Monte Carlo tree search, in *International Conference on Learning Representations* (ICLR, Kigali, Rwanda, 2023).

[24] B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients, in *Proceedings of the International Conference on Learning Representations* (ICLR, Virtual Event, Austria, 2021).

[25] H. Zhang and A. Zhou, Rl-GEP: Symbolic regression via gene expression programming and reinforcement learning, in *Proceedings of the International Joint Conference on Neural Networks* (IJCNN, Shenzhen, China, 2021), pp. 1–8.

[26] T. N. Mundhenk, M. Landajuela, R. Glatt, C. P. Santiago, D. M. Faissol, and B. K. Petersen, Symbolic regression via deep reinforcement learning enhanced genetic programming seeding, in *Proceedings of the Advances in Neural Information Processing Systems* (NeurIPS, virtual, 2021), Vol. 34, pp. 24912–24923.

[27] P. Y. Lu, J. Ariño Bernad, and M. Soljačić, Discovering sparse interpretable dynamics from partial observations, Commun. Phys. **5**, 206 (2022).

[28] M. Zhang, S. Kim, P. Y. Lu, and M. Soljačić, Deep learning and symbolic regression for discovering parametric equations, IEEE Trans. Neural Netw. Learn. Syst., 1 (2023).

[29] H. Xu, H. Chang, and D. Zhang, Dl-pde: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data, Comm. Comput. Phys. **29**, 698 (2021).

[30] S.-M. Udrescu and M. Tegmark, Ai feynman: A physics-inspired method for symbolic regression, Sci. Adv. **6**, eaay2631 (2020).

[31] L. Billard and E. Diday, Symbolic regression analysis, in *Classification, Clustering, and Data Analysis* (Springer, New York, 2002), pp. 281–288.

[32] T. Elsken, J. H. Metzen, and F. Hutter, Neural architecture search: A survey, J. Mach. Learn. Res. **20**, 1997 (2019).

[33] A. Tamar, Y. Glassner, and S. Mannor, Policy gradients beyond expectations: Conditional value-at-risk, arXiv:1404.3862.

[34] A. Tamar, Y. Glassner, and S. Mannor, Optimizing the CVaR via sampling, in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (AAAI, Austin, Texas, 2015), p. 2993.

[35] T. Hiraoka, T. Imagawa, T. Mori, T. Onishi, and Y. Tsuruoka, Learning robust options by conditional value at risk optimization, in *Proceedings of the Advances in Neural Information Processing Systems* (NeurIPS, Vancouver, Canada, 2019), Vol. 32, p. 2619.

[36] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in *Proceedings of the International Conference on Machine Learning* (PMLR, Stockholm, Sweden, 2018), pp. 1861–1870.

[37] M. Maslyaev, A. Hvatov, and A. V. Kalyuzhnaya, Partial differential equations discovery with epde framework: Application for real and synthetic data, J. Comput. Sci. **53**, 101345 (2021).

[38] D. J. Korteweg and G. de Vries, On the change of form of long waves advancing in a rectangular channel, and a new type of long stationary wave, Philos. Mag **39**, 422 (1895).

[39] J. M. Burgers, A mathematical model illustrating the theory of turbulence, Adv. Appl. Mech. **1**, 171 (1948).

[40] A. C. Newell and J. A. Whitehead, Finite bandwidth, finite amplitude convection, J. Fluid Mech. **38**, 279 (1969).

[41] B. Straughan, Jordan–cattaneo waves: Analogues of compressible flow, Wave Motion **98**, 102637 (2020).

[42] Y. Mao, Exact solutions to (2+1)-dimensional Chaffee–Infante equation, Pramana **91**, 9 (2018).

[43] A. Debussche, M. Högele, and P. Imkeller, Asymptotic first exit times of the Chafee-Infante equation with small heavy-tailed Lévy noise, Electron. Commun. Probab. **16**, 213 (2011).

[44] A. Korkmaz, Complex wave solutions to mathematical biology models I: Newell–Whitehead–Segel and Zeldovich equations, J. Comput. Nonlinear Dyn. **13**, 081004 (2018).

[45] G. Gardner, J. Downie, and H. Kendall, Gravity segregation of miscible fluids in linear models, Soc. Pet. Eng. J. **2**, 95 (1962).

[46] S. M. Allen and J. W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, Acta Metall. **27**, 1085 (1979).

[47] J. Shen and X. Yang, Numerical approximations of Allen-Cahn and Cahn-Hilliard equations, Discrete Contin. Dyn. Syst. **28**, 1669 (2010).

[48] J. W. Cahn and J. E. Hilliard, Free energy of a nonuniform system. I. interfacial free energy, J. Chem. Phys. **28**, 258 (1958).

[49] M. Z. Bazant, Thermodynamic stability of driven open systems and control of phase separation by electro-autocatalysis, Faraday Discuss. **199**, 423 (2017).

[50] M. Khater, C. Park, D. Lu, and R. A. Attia, Analytical, semi-analytical, and numerical solutions for the Cahn–Allen equation, Adv. Differ. Equ. **2020**, 9 (2020).

[51] P. A. Reinbold, L. M. Kageorge, M. F. Schatz, and R. O. Grigoriev, Robust learning from noisy, incomplete, high-dimensional experimental data via physically constrained symbolic regression, Nat. Commun. **12**, 3219 (2021).

[52] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and E. Galván-López, Semantically-based crossover in genetic programming: Application to real-valued symbolic regression, Genet. Program. Evolvable Mach. **12**, 91 (2011).

[53] A. Ross, Z. Li, P. Perezhogin, C. Fernandez-Granda, and L. Zanna, Benchmarking of machine learning ocean subgrid parameterizations in an idealized model, J. Adv. Model. Earth Syst. **15**, e2022MS003258 (2023).

[54] C. Meneveau and J. Katz, Scale-invariance and turbulence models for large-eddy simulation, Annu. Rev. Fluid Mech. **32**, 1 (2000).

[55] M. F. Jansen and I. M. Held, Parameterizing subgrid-scale eddy effects using energetically consistent backscatter, Ocean Modell. **80**, 36 (2014).

[56] J. A. Anstey and L. Zanna, A deformation-based parametrization of ocean mesoscale eddy Reynolds stresses, Ocean Modell. **112**, 99 (2017).

[57] M. Du, Y. Chen, and D. Zhang, Autoke: An automatic knowledge embedding framework for scientific machine learning, IEEE Trans. Artif. Intell. **4**, 1564 (2023).

[58] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res. **18**, 5595 (2017).

[59] GitHub, https://github.com/menggedu/DISCOVER.

[60] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics, Doha, Qatar, 2014), pp. 1724–1734.

[61] L. Li and T. Zhang, Research on text generation based on LSTM, Int. Core. J. Eng. **7**, 525 (2021).

[62] Y. Wang, M. Huang, X. Zhu, and L. Zhao, Attention-based LSTM for aspect-level sentiment classification, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (EMNLP, Austin, Texas, 2016), pp. 606–615.

[63] Y. Li, Z. Zhu, D. Kong, H. Han, and Y. Zhao, Ea-lstm: Evolutionary attention-based lstm for time series prediction, Knowl. Based. Syst. **181**, 104785 (2019).

[64] J. Cheng, L. Dong, and M. Lapata, Long short-term memory-networks for machine reading, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Austin, Texas, 2016), pp. 551–561.

[65] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, {TensorFlow}: A system for {Large-Scale} machine learning, in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (2016), pp. 265–283.

[66] M. Landajuela, B. K. Petersen, S. K. Kim, C. P. Santiago, R. Glatt, T. N. Mundhenk, J. F. Pettit, and D. M. Faissol, Improving exploration in policy gradient search: Application to symbolic optimization, arXiv:2107.09158.

[67] A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau *et al.*, PySINDy: A comprehensive python package for robust sparse system identification, J. Open Source Softw. **7**, 3994 (2022).

[68] S. Khani and F. Porté-Agel, Evaluation of non-eddy viscosity subgrid-scale models in stratified turbulence using direct numerical simulations, Eur. J. Mech. B Fluids **65**, 168 (2017).