Discovering Symbolic Partial Differential Equation by Abductive Learning

En-Hao Gao, Cunjing Ge, Yuan Jiang, Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
gaoeh@lamda.nju.edu.cn, {gecunjing, jiangyuan, zhouzh}@nju.edu.cn

Abstract

Discovering symbolic Partial Differential Equation (PDE) from data is one of the most promising directions of modern scientific discovery. However, effectively constructing an expressive yet concise hypothesis space and accurately evaluating expression values remain challenging due to the exponential explosion with the spatial dimension and the noise in the measurements. To address these challenges, we propose the ABL-PDE approach that employs the Abductive Learning (ABL) framework to discover symbolic PDEs. By introducing a First-Order Logic (FOL) knowledge base, ABL-PDE can represent various PDEs, significantly constraining the hypothesis space without sacrificing expressive power, while also facilitating the incorporation of problem-specific knowledge. Furthermore, the proposed consistency optimization process establishes a synergistic interaction between the knowledge base and the neural network, achieving robust structure identification, accurate coefficient estimation, and enhanced stability against hyperparameter variation. Experimental results on three benchmarks across different noise levels demonstrate the state-of-the-art performance of our approach in PDE discovery.

1 Introduction

Partial Differential Equations (PDEs) serve as fundamental mathematical tools for describing a vast array of physical phenomena in science and engineering, effectively capturing the intricate relationships between how quantities change over space and evolve over time. Due to their versatility, PDEs are widely employed across a range of applications, including airfoil design in aerodynamics [14], weather prediction in meteorology [9], and pricing analysis in quantitative finance [6].

Recently, physics-informed machine learning has demonstrated remarkable success in approximating the behavior of complex physical systems [3, 22], which significantly changes the way we represent and leverage PDEs. Despite these advancements, explicit form PDEs remain essential in applications where high reliability is paramount. Furthermore, their inherent structure allows a single PDE to be adapted to various scenarios by simply modifying the boundary conditions and domain geometries. Due to such superior interpretability and generalizability, discovering symbolic governing equations from data remains one of the most promising avenues in scientific discovery [4].

Significant efforts [7, 10, 12, 16, 18] have been made to the PDE discovery task. Figure 1 provides an overview of the entire framework. Most of these methods consist of two primary components: a derivative-integral calculator and a candidate term library. The calculator evaluates symbolic expressions based on either direct derivative estimation or weak formulation-based integral computation. Meanwhile, the library constructs the hypothesis space for the target equation, with each term representing a function of the variables of interest. The underlying PDE is typically assumed to be a linear combination of these terms and is identified through symbolic regression techniques.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

The first challenge of PDE discovery lies in constructing a comprehensive candidate term library. On one hand, this library must be sufficiently expressive to capture patterns within data; On the other hand, it should remain compact to enable an efficient PDE structure identification. Existing studies attempt to balance this trade-off by either constraining expressiveness to create a more concise library [11, 16] or maintaining flexibility at the expense of introducing numerous redundancies [8, 26, 27]. Moreover, accurately evaluating the expression values is also challenging. Since we can only access discrete measurements, derivatives or integrals must be estimated using approximate methods, the accuracy of which is significantly affected by the amount of data and the level of noise. Despite many studies attempting to address this issue using techniques such as ensemble learning [12] and weak formulation [11], most of them ignore the potential of using discovered physical information to in turn enhance the calculator's capability, leading to inferior performance in both structure identification and coefficient estimation.

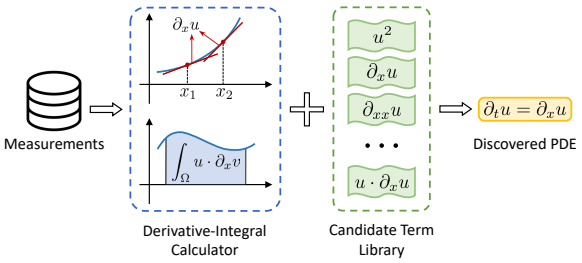


Figure 1: An illustration of PDE discovery framework.

In this paper, we propose the ABductive Learning for PDE discovery (ABL-PDE) approach to tackle the aforementioned challenges under the ABductive Learning (ABL) framework [28, 29]. ABL is a powerful paradigm that integrates data-driven machine learning with knowledge-driven logical reasoning in a balanced mutual promotion loop while maintaining the expressive power of both. Building on this framework, we design two components for ABL-PDE: a neural network-based learning module for derivative estimation and a First-order Logic (FOL) based reasoning module for candidate term generation, along with a consistency maximization process to bridge the two parts. Our approach balances the expressiveness and compactness of the hypothesis space and enables reciprocal enhancement between the calculator and the library.

Our contributions are summarized as follows:

- We propose a novel PDE discovery approach based on the ABL framework, transforming the previous unidirectional discovery pipeline into a bidirectional enhancement loop.
- We introduce a FOL knowledge base capable of representing a wide range of PDEs, substantially reducing redundancy and facilitating the incorporation of problem-specific knowledge.
- We design a consistency optimization process to bridge the two parts, which significantly enhances the robustness of structure identification, the accuracy of coefficient estimation, and the stability against hyperparameter variations.

We conduct extensive experiments on various PDE discovery tasks across different noise levels. The experimental results demonstrate the state-of-the-art performance of our approach.

2 Related Work

Based on whether the equation structure and the coefficients are determined synchronously, PDE discovery methods can be broadly classified into two categories.

The first category comprises synchronous methods, most of which rely on sparse regression techniques. A pioneering approach is SINDy [5], which was extended by PDE-FIND [23] from ODEs to PDEs. Weak SINDy [11] further enhances noise robustness by employing weak formulation. Apart from explicit sparse regression, several synchronous methods such as PDE-Net [18, 19] and Bayesian hidden physics models [1] optimize derivatives and coefficients end-to-end. Although generally exhibiting superior performance in fitting physical fields, they struggle to distinguish noise from small-coefficient terms. D-CIPHER [16] extends the boundary of PDE discovery to learn parameters within nonlinear terms. However, this comes with a more challenging optimization process, making it struggle to learn coupled PDE systems. A limitation of synchronous methods is their reliance on predefined, typically polynomial libraries. Although this compact library ensures term uniqueness, its inherent lack of expressiveness prevents the capture of complex structures like fractions.

The second category comprises asynchronous methods, often based on evolutionary algorithms, proposed to overcome the limitations of predefined libraries [8, 10, 12, 15, 25]. These methods first

determine the equation's form by discrete optimization and then estimate the equation's coefficients. Although such approaches incur higher computational costs due to discrete optimization, they mitigate the detrimental impact of interference terms on coefficient estimation while facilitating the incorporation of more flexible equation representation methods, such as binary trees [8, 15, 25]. However, it's difficult for these methods to impose constraints on the equation form, resulting in an excessive number of redundant and invalid PDEs within the hypothesis space. In contrast, by introducing a FOL knowledge base, our method achieves a better balance between expressiveness and compactness in PDE representation, while also facilitating the incorporation of prior knowledge.

3 Preliminaries

In this section, we first present the problem formulation and then briefly introduce the abductive reasoning technique used to construct the candidate term library.

3.1 Problem Setting

In this paper, we consider general dynamical systems governed by PDEs of the following form:

$$\partial_t u = \left[f^{(1)}(u, \boldsymbol{x}), f^{(2)}(u, \boldsymbol{x}), \ldots, f^{(i)}(u, \boldsymbol{x}), \ldots \right] \cdot \boldsymbol{\xi}, \tag{1}$$

where $u : [0, T] \times \Omega \to \mathbb{R}$ is the solution function with time horizon T and bounded spatial domain $\Omega \subset \mathbb{R}^d$, t and $\boldsymbol{x}$ denote the temporal and spatial coordinate, each $f^{(i)}$ is a function of u and $\boldsymbol{x}$, and $\boldsymbol{\xi}$ is their linear combination coefficient which is always sparse in practice. The data available is a set of triplets $\{(t_i, \boldsymbol{x}_i, \hat{u}(t_i, \boldsymbol{x}_i))\}_{i=1}^m$ representing m temporal-spatial coordinates, with measurements that may contain noise. Each $f^{(i)}(u, \boldsymbol{x})$ collectively forms the candidate term library that will be gradually expanded during the execution of our approach. We call $f^{(i)}(u, \boldsymbol{x})$ an *expression term*.

Definition 1 (Expression Term)**.** An *expression term* is recursively formed by the following rule:

$$\begin{aligned} \langle \texttt{expr} \rangle &::= \langle \texttt{idpv} \rangle \mid \langle \texttt{dpv} \rangle \mid \partial_{\langle \texttt{idpv} \rangle} \langle \texttt{expr} \rangle \mid \langle \texttt{expr} \rangle \ \langle \texttt{op} \rangle \ \langle \texttt{expr} \rangle \\ \langle \texttt{op} \rangle &::= + \mid - \mid \times \mid / \end{aligned}$$

Here, $\langle \cdot \rangle$ denotes a nonterminal symbol, $::=$ indicates definition, and $\mid$ signifies alternation. The nonterminal symbols $\langle \texttt{dpv} \rangle$ and $\langle \texttt{idpv} \rangle$ represent dependent and independent variables, respectively. Combining Equation (1) with expression terms, it suffices to represent a wide range of fundamental PDEs, such as the Diffusion equation, the Schrödinger equation, the Navier-Stokes equation, etc.

3.2 Abductive Reasoning

Abductive reasoning, also known as abduction, is a fundamental mode of logical reasoning that seeks explanations for observed facts. Abductive Logic Programming (ALP) formalizes this process within the context of logic programming by defining an abductive logic program as follows:

Definition 2 (Abductive Logic Program [17])**.** An abductive logic program is a triplet (P, A, IC), where P is the background knowledge encoded as a logic program, A is a set of abducible predicates representing hypotheses, and IC is a set of first-order formulae called integrity constraints. Abductive explanation for an observation O is a set Δ of ground atoms on abducible predicates A, such that:

(1) $P \cup \Delta \models O, IC$

(2) $P \cup \Delta$ is consistent

Here, $\models$ denotes logical entailment, meaning that each model of the left-hand side is also a model of the right-hand side. Abductive reasoning is essentially a search problem to find a set Δ on A which satisfies the three rules above. In our task, P is the logic program encoding the syntactic rules for expression terms, A is a subset of predicates used to represent expression terms, IC consists of various constraints on the term structure, and O is the number of operators in the term.

4 Our Approach

In this section, we first present an overview of our approach, ABductive Learning for PDE discovery (ABL-PDE), and then detail its components and the proposed consistency optimization process.

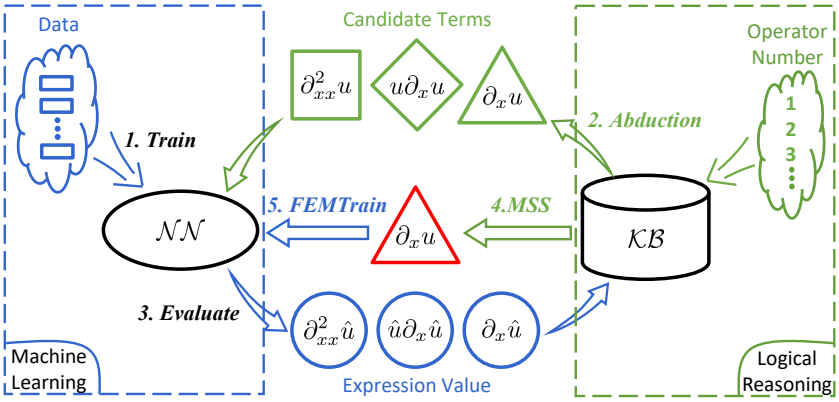


Figure 2: An illustration of ABL-PDE's framework.

4.1 Overview of ABL-PDE

As illustrated in Figure 2, ABL-PDE comprises two modules:

- **Machine Learning**. This module employs a neural network to fit the physical field and calculates derivatives through automatic differentiation.
- **Logical Reasoning**. This module incorporates an abductive logic program as the knowledge base and is responsible for generating candidate expression terms.

The discovery process commences with pretraining the neural network, during which no physical information is available (Step 1). Following this initial stage, an iterative deepening search is performed by progressively increasing the number of operators in the candidate terms. In each iteration, ABL-PDE first expands the candidate term library through abduction, utilizing the current number of operators as observations (Step 2). After this expansion, the abduced expression terms are transformed into numerical values (Step 3). Next, the proposed Monotone Subset Selection (MSS) algorithm is employed to identify terms with nonzero coefficients (Step 4). Subsequently, ABL-PDE estimates their coefficients and jointly optimizes the neural network through Finite Element Method-based Training (FEMTrain) (Step 5). Steps 4 and 5 together constitute the consistency optimization process, which is repeated until the discovered equation stabilizes and the coefficients converge.

4.2 Logic-based Knowledge Base

This part details our knowledge base KB, an abductive logic program (P, A, IC). Recall that abductive reasoning is a search problem to find a set Δ on A that satisfies the rules in Definition 2.

Background Knowledge P. A straightforward way is to adopt the syntax rule in Definition 1 as the background knowledge P. Since $P \cup \Delta$ is consistent (rule (2) in Definition 2), P actually gives the boundary of search space, i.e., Δ should be an expression term. Obviously, size of the search space grows exponentially with the operator number O, which is a significant challenge. We observe that numerous mathematically equivalent terms are different only in their representations. Therefore, we introduce *canonical form term* to reduce the search space by eliminating such equivalent cases.

Definition 3 (Canonical Form Term)**.** An expression term is called in *canonical form* if and only if it can be recursively generated by the following rules:

$$
\begin{aligned}
\langle \texttt{cfterm} \rangle &::= \langle \texttt{div} \rangle \mid \langle \texttt{mul} \rangle \mid \langle \texttt{add} \rangle \mid \langle \texttt{sub} \rangle \mid \langle \texttt{unit} \rangle \\
\langle \texttt{div} \rangle &::= \langle \texttt{mul} \rangle \,/\, \langle \texttt{add} \rangle \mid \langle \texttt{mul} \rangle \,/\, \langle \texttt{sub} \rangle \\
\langle \texttt{mul} \rangle &::= \langle \texttt{unit} \rangle \times \langle \texttt{mul} \rangle \mid \langle \texttt{unit} \rangle \\
\langle \texttt{add} \rangle &::= \langle \texttt{mul} \rangle + \langle \texttt{add} \rangle \mid \langle \texttt{mul} \rangle \\
\langle \texttt{sub} \rangle &::= \langle \texttt{add} \rangle - \langle \texttt{add} \rangle \\
\langle \texttt{unit} \rangle &::= \langle \texttt{dpv} \rangle \mid \langle \texttt{idpv} \rangle \mid \langle \texttt{par} \rangle \\
\langle \texttt{par} \rangle &::= \partial_{\langle \texttt{idpv} \rangle} \langle \texttt{dpv} \rangle \mid \partial_{\langle \texttt{idpv} \rangle} \langle \texttt{par} \rangle
\end{aligned}
$$

Table 1: General-purpose (C1, C2, C3) and domain-specific (C4, C5, C6) integrity constraints.

	Integrity Constraint	Description
C1	$\leftarrow \texttt{add}(X, L) \wedge L = \texttt{add}(Y, Z) \wedge X \succ Y.$	Unique addition order
C2	$\leftarrow \texttt{mul}(X, L) \wedge L = \texttt{mul}(Y, Z) \wedge X \succ Y.$	Unique multiplication order
C3	$\leftarrow \texttt{sub}(X, Y) \wedge \texttt{intersect}(X, Y).$	Disjoint subtraction operands
C4	$\leftarrow \texttt{getPower}(\texttt{mul}(X, Y), u, N) \wedge N > n.$	Power limit
C5	$\leftarrow \texttt{getOrder}(\texttt{par}(X, Y), x, M) \wedge M > m.$	Derivative order limit
C6	$\leftarrow \texttt{getDependent}(\texttt{par}(X, Y), Z) \wedge Z = v \wedge Y = y.$	Continuity equation constraint

The notation has the same meaning as in Definition 1. These rules impose structure constraints on expression terms derivable from each nonterminal symbol. We illustrate the constraint on partial derivatives ($\langle \texttt{par} \rangle$) in Example 1; for detailed explanations of other rules, please refer to Appendix A.

Example 1. Given two dependent variables u and v. The expression term $\partial_t(u \times v)$ is not in canonical form as it does not satisfy the rule of $\langle \texttt{par} \rangle$ nor any other rules. However, $\partial_t(u \times v) = \partial_t u \times v + u \times \partial_t v$, which is the linear combination of canonical form terms $\partial_t u \times v$ and $u \times \partial_t v$. In Section 5, we prove that every expression term can be represented by a linear combination of canonical form terms.

In our implementation, P is a logic program directly translated from the rules in Definition 3. Integrated with the operator number tracking capability in the deductive reasoning process, P can leverage abductive reasoning to generate canonical form terms. [1]

Abducible Predicates A**.** In this work, the design of abducible predicates is not a focus. We simply introduce an auxiliary predicate to only allow abducing terms derivable from $\langle \texttt{mul} \rangle$ and $\langle \texttt{div} \rangle$ Since we consider linear combinations of expression terms, this does not compromise the expressiveness.

Integrity Constraints IC**.** Integrity constraints enable us to declaratively state various constraints on the structure of expression terms. Table 1 presents several examples of such constraints: the upper section lists general-purpose constraints, while the lower section includes domain-specific ones. The logical symbols are used as follows: '$\leftarrow$' denotes implication (with the left-hand side `False` omitted), '$\wedge$' represents conjunction, and '$\succ$' indicates symbolic order, which is a built-in predicate in most logic programming languages. C1 and C2 in Table 1 state that an addition or multiplication sequence is invalid if the former of two adjacent subterms is symbolically greater than the latter. C3 ensures that the two operands of a subtraction do not intersect. C4 and C5 respectively constrain the power of u and the derivative order with respect to x in a term. C6 is pertinent to the continuity equation. Specifically, in our experiment on the Navier-Stokes equation, knowing that 2D incompressible flow satisfies the continuity equation ($-\partial_x u = \partial_y v$) allows us to eliminate terms involving partial derivatives of v with respect to y, since these terms can be systematically replaced by partial derivatives of u with respect to x (e.g., $v_y = -u_x$, $v_{yy} = -u_{xy}$). Please note that only the general-purpose constraints are included in `KB` to ensure a fair comparison with other methods.

4.3 Consistency Optimization

In this part, we first introduce the consistency measure and then present MSS and FEMTrain.

Consistency Measure. Given the operator number limit L and the dataset D, the consistency measure $\text{Con}_{L,D}(\texttt{NN}_\theta, \texttt{KB})$ of a neural network $\texttt{NN}_\theta$ and a knowledge base `KB` is defined as follows:

$$
\begin{aligned}
&-\big(\min_{\boldsymbol{\xi}, F} \text{Res}(\texttt{NN}_\theta, \boldsymbol{\xi}, F) + \lambda \cdot \text{MSE}(\texttt{NN}_\theta, D)\big) \\
&\qquad \text{s.t.} \quad F \subset \bigcup_{O=1}^{L} \Delta_{\texttt{KB},O},
\end{aligned}
\tag{2}
$$

where θ denotes the neural network parameters, $\boldsymbol{\xi}$ is the coefficient vector as in Eq. (1), F is a set of terms obtained from the abductive reasoning on `KB` for each operator number $O \leq L$, and λ is a hyperparameter utilized to balance Res and MSE.

[1] Please refer to Appendix B for a pedagogical example illustrating the program and the abduction process.

Let $F = \{f^{(i)}(u, \boldsymbol{x})\}_{i=1}^{N_F}$, the physics-informed residual is defined as: [2]

$$\text{Res}(\texttt{NN}_\theta, \boldsymbol{\xi}, F) = \left\| \partial_t \texttt{NN}_\theta(t, \boldsymbol{x}) - \sum_{i=1}^{N_F} \xi_i \cdot f^{(i)}(\texttt{NN}_\theta(t, \boldsymbol{x}), \boldsymbol{x}) \right\|_2. \tag{3}$$

Let $D = \{(t_i, \boldsymbol{x}_i, \hat{u}(t_i, \boldsymbol{x}_i))\}_{i=1}^m$ as in Section 3.1, the mean square error is defined as:

$$\text{MSE}(\texttt{NN}_\theta, D) = \frac{1}{m} \sum_{i=1}^{m} \left(\texttt{NN}_\theta(t_i, \boldsymbol{x}_i) - \hat{u}(t_i, \boldsymbol{x}_i)\right)^2. \tag{4}$$

The minimal residual in Eq. (2) decreases monotonically as the size of F increases. Thus, directly maximizing the consistency measure will lead to a trivial solution where $F = \bigcup \Delta_{\texttt{KB},O}$. Since the number of nonzero terms cannot be determined in advance, selecting an upper bound of $|F|$ is insufficient to balance sparsity and consistency. Thus we introduce *margin* and *k-level margin*.

Definition 4 (Margin)**.** Let H be a set of expression terms, and $F \subseteq H$. The *margin* of F with respect to H is defined as:

$$\text{Margin}_H(F) = \frac{\|\text{Res}^*(\texttt{NN}_\theta, F)\|_2 - \|\text{Res}^*(\texttt{NN}_\theta, H)\|_2}{\|\text{Res}^*(\texttt{NN}_\theta, H)\|_2}, \tag{5}$$

where $\text{Res}^*(\texttt{NN}_\theta, \cdot)$ is the minimal residual in Eq. (2) with respect to $\boldsymbol{\xi}$.

Definition 5 (k-Level Margin)**.** Let H be a set of expression terms, k a positive integer, and $|\cdot|$ denote set cardinality. The *k-level margin* of H is defined as

$$\text{LevelMargin}_H(k) = \min_{F' \subseteq H,\, |F'|=k} \text{Margin}_H(F'). \tag{6}$$

Intuitively, the margin of a subset F quantifies how self-sufficient it is in approximating the dynamics $\partial_t \texttt{NN}_\theta$. A smaller margin suggests that the terms within F are sufficient, while the terms outside it are negligible. Accordingly, the k-level margin measures the sufficiency of the best possible k-term model, indicating whether a parsimonious, k-term equation can accurately represent the system. As shown in Section 5, the level margin decreases monotonically as k increases, which enables us to adaptively control the expression term number by applying a margin threshold.

Monotone Subset Selection. Under a given margin threshold, evaluating the consistency measure $\text{Con}_{L,D}(\texttt{NN}_\theta, \texttt{KB})$ with respect to the neural network parameters θ is time-consuming, as the complexity of the residual minimization subproblem in Eq. (2) is no less than that of the sparse regression under a cardinality constraint, which is generally NP-hard [20]. To reduce computational costs, we propose Monotone Subset Selection (MSS), a two-stage algorithm featuring an initial stage of rapid, large-scale coarse selection succeeded by a precise, small-scale refined search.

Algorithm 1 MSS

Input: Neural network `NN`, Term library `termLib`, Initial sparsity `initSp`, Margin threshold `magThd`
Output: Minimal residual set under `magThd`-margin `minSet`

1: Initialization: $\texttt{minSet} \leftarrow \emptyset$
2: $\texttt{libVal} \leftarrow \text{Eval}(\texttt{NN}, \texttt{termLib})$
3: $\texttt{subLib} \leftarrow \text{POSS}(\texttt{libVal}, \texttt{termLib}, \texttt{initSp})$
4: $\texttt{left} \leftarrow 1, \texttt{right} \leftarrow |\texttt{subLib}|$
5: **while** $\texttt{left} \leq \texttt{right}$ **do**
6: $\quad \texttt{mid} \leftarrow \lfloor(\texttt{left} + \texttt{right})/2\rfloor$
7: $\quad \texttt{tmpMinSet}, \texttt{levMag} \leftarrow \text{Enum}(\texttt{subLib}, \texttt{mid})$
8: $\quad$ **if** $\texttt{levMag} < \texttt{magThd}$ **then**
9: $\quad\quad \texttt{minSet} \leftarrow \texttt{tmpMinSet}, \texttt{right} \leftarrow \texttt{mid} - 1$
10: $\quad$ **else**
11: $\quad\quad \texttt{left} \leftarrow \texttt{mid} + 1$

Algorithm 1 presents the details of MSS. In the first stage (cf. Line 2-3), MSS evaluates expression terms in the library and calls Pareto Optimization for Subset Selection (POSS) [21] to find a reduced term library `subLib` whose size should be no greater than the initial sparsity. POSS is a non-deterministic subset selection algorithm designed to minimize the residual in Eq. (2) under a cardinality constraint. In expectation, it is guaranteed to find a reasonably good solution within a polynomial number of queries to the objective function. This property makes it well-suited for the first stage, which focuses on rapidly identifying relevant terms rather than performing costly residual minimization. More details about

[2] Since the underlying PDE is expected to hold at every point within the domain, we are free to select the set of points for calculating the residual, which has been omitted for brevity.

POSS are provided in Appendix C. In the second stage (cf. Line 4-11), MSS employs binary search to identify the smallest subset size for which the level margin falls below the given threshold. Each time, it enumerates the subset of size `mid` to identify the one with the minimal residual and calculates the level margin, which is tractable within the reduced library size.

Finite Element Method-based Training. Once the set of active expression terms F in Eq. (2) is identified via MSS for a given neural network parameters θ, the next step of consistency optimization is to jointly optimize the network parameters θ and coefficient vector $\boldsymbol{\xi}$ to maximize the consistency measure. A standard approach for this sub-problem is to adopt the Physics-Informed Neural Networks (PINN) paradigm [22]. However, the efficacy of the PINN approach is often hindered by its sensitivity to hyperparameters, such as the loss weighting term λ in Eq. (2) [13]. To enhance the stability of model training and coefficient estimation, we move beyond pointwise collocation and instead enforce the physics-informed residual in a weighted-average sense across the domain. This is achieved through a *Galerkin projection* [24] of the residual from Eq. (3) onto the space spanned by Finite Element Method (FEM) basis functions, leading to the following objective function:

$$\|\boldsymbol{M}(\partial_t \texttt{NN}_\theta(t, \boldsymbol{x}) - \sum_{i=1}^{N_F} \xi_i \cdot f^{(i)}(\texttt{NN}_\theta(t, \boldsymbol{x}), \boldsymbol{x}))\|_2, \tag{7}$$

where $\boldsymbol{M}$ is the mass matrix constructed from the inner product of FEM basis functions. Please refer to Appendix D for detailed definitions and derivations. The overall consistency optimization process alternates between MSS and FEMTrain until the set F stabilizes and the coefficients $\boldsymbol{\xi}$ converge.

5 Theoretical Analysis

In this section, we first demonstrate that our logic-based knowledge base, `KB`, retains full expressive power, and then establish the soundness of MSS. Proofs are available in Appendix E.

We denote the abduction result of `KB` for a given operator number i as H_i. Thus, the set of all possible abduction results is given by $H = \bigcup_{i=0}^{\infty} H_i$. Besides, we refer to the set of all expression terms as S.

Definition 6 (Subsume)**.** A finite set of expression terms F is said to *subsume* an expression term s if there exists an expression term s' such that:

- s' can be derived from s through a finite sequence of algebraic operations and differentiation.
- s' can be symbolically expressed as a linear combination of expression terms in F, i.e., $s' = \sum_{f \in F} \alpha_f \cdot f$, where α_f are scalars.

As shown in Example 1, $\{\partial_x u \times v, u \times \partial_x v\}$ subsumes $\partial_x(u \times v)$.

Theorem 1. *For every $s \in S$, there exists a finite subset $H' \subset H$ such that H' subsumes s.*

According to the PDE formulation in Eq. (1), it suffices to consider the expressiveness of a candidate term library up to a linear combination. Theorem 1 states that every expression term in S has a mathematically equivalent form that is a linear combination of elements from H, which directly implies that the expressive power of `KB` is equivalent to that of the full expression term space S.

The soundness of MSS depends on the criterion utilized in its binary search, whose validity is ensured by the following monotonicity theorem.

Theorem 2. *For an expression term set F and integers $1 \leq k_1 < k_2 \leq |F|$, it holds that:*

$$\text{LevelMargin}_F(k_1) \geq \text{LevelMargin}_F(k_2).$$

6 Experiments

In this section, we present empirical studies to answer the following questions:

1. How does ABL-PDE perform against contenders in PDE discovery?
2. How robust is ABL-PDE against hyperparameter selection?
3. How does each component of ABL-PDE contribute to its performance?

Table 2: PDE discovery comparison results. The evaluation metric is the sum of L2RE (%) on all coefficients. Number with a dagger (†) indicates the discovered equation has redundant terms. Results of ABL-PDE (w/o FEM) and ABL-PDE are the mean and standard deviation of 5 experiments with different λ that used to balance the two sub-metrics in Eq. (2). We **bold** the best results in each task.

Method	Burgers' Equation			Schrödinger Equation			Naiver-Stokes Equation		
	0	5%	10%	0	5%	10%	0	5%	10%
PDERidge	22.68	23.86	24.54	10.34	10.26	9.99	**17.13**	16.58	40.92
DL-PDE++	22.68	23.86	24.54	10.34	10.26	9.99	57.25†	37.96†	98.52†
ABL-PDE (w/o CO)	22.68	23.86	24.54	42.16†	44.67†	48.76†	**17.13**	17.99	40.92
ABL-PDE (w/o FEM)	13.69 ± 2.81	14.65 ± 2.67	15.04 ± 3.87	3.05 ± 0.94	3.27 ± 1.09	2.87 ± 1.07	30.88 ± 5.24	26.47 ± 5.00	34.28 ± 7.50
ABL-PDE	**2.18 ± 1.06**	**2.79 ± 0.13**	**4.12 ± 0.34**	**1.03 ± 0.10**	**1.12 ± 0.06**	**1.08 ± 0.29**	18.02 ± 0.08	**15.22 ± 0.08**	**21.27 ± 1.72**

6.1 Experimental Setup

We conduct experiments on three types of discovery tasks: (i) one-dimensional PDE: Burgers' Equation, (ii) one-dimensional PDE system: Schrödinger Equation, (iii) two-dimensional PDE system: Navier-Stokes Equation. Tasks (ii) and (iii) each involve two coupled equations. We use data from public datasets [2], incorporating 5% and 10% Gaussian noise to simulate inaccurate measurements as in [27]. We use the L_2 Relative Error (L2RE, %) as in [11] to measure the quality of coefficient estimation and physical field approximation. More details are specified in Appendix F. Code is available at: `https://github.com/AbductiveLearning/ABL-PDE`.

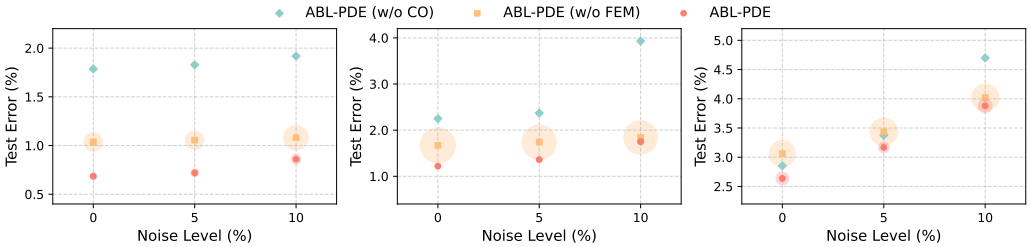
6.2 Compared Methods

We compare ABL-PDE with four methods: 1) **PDERidge**: We build this strong baseline by assuming a known PDE structure and applying ridge regression to estimate coefficients. This represents the best possible performance for most methods that focus on enhancing structure identification [8, 10, 16]. 2) **DL-PDE++**: DL-PDE [27] is a representative method among synchronous approaches that discussed in Section 2. We enhance it by applying a normalization trick and tuning the sparsity threshold to retain all correct terms while minimizing redundancy. 3) **ABL-PDE (w/o Con)**: It utilizes the pre-trained neural network and MSS algorithm to discover PDE, without the subsequent consistency optimization. 4) **ABL-PDE (w/o FEM)**: It employs the residual in Eq. (3) instead of the FEM residual in Eq. (7) during the consistency optimization. To ensure a fair comparison, we use the same pre-trained neural network as in ABL-PDE to compute derivatives for the first two methods.

6.3 PDE Discovery Comparison

In this part, we seek to answer the first two questions. Table 2 presents the results of structure identification and coefficient estimation. Number with a dagger (†) indicates the discovered equation has redundant terms (See Appendix G.1 for the specific form). The performance of DL-PDE++ and ABL-PDE (w/o CO) is equivalent to PDERidge when the discovered structure matches the ground truth, due to their shared neural network for derivative computation. As shown in the table, ABL-PDE correctly identified the PDE structure across all three tasks under varying noise levels, achieving the best coefficient estimation in 8 out of 9 cases and ranking second only to the structure-known baseline, PDERidge, in the remaining one case. Notably, this performance was achieved utilizing the *same* initial sparsity of 10 and the *same* margin threshold of 0.05 for all three tasks. ABL-PDE has only one remaining hyperparameter, which will be discussed in Section 6.4. In our experiments, despite the simple structure of Burgers' equation, it remains challenging for coefficient estimation. The inferior performance of compared methods is primarily attributed to the formation of sharp shock waves in the solution, which can only be captured by an extremely dense mesh that is impractical in most real-world applications. ABL-PDE effectively mitigates this issue via consistency optimization, which utilizes the discovered physical information, even if it is initially incorrect, to improve the derivative estimation, leading to a 5x to 10x reduction in L2RE.

Beyond its robustness to noise, ABL-PDE is also resilient to hyperparameter selection. As shown in Table 2, the discovered equation of ABL-PDE (w/o CO) contains redundant terms in several tasks. This redundancy stems from the misalignment between the margin threshold and the data. Nevertheless, ABL-PDE finally eliminates these terms via consistency optimization. A by-product of our method is the improvement in the neural network's generalizability. Figure 3 presents the L2RE



(a) Burgers' Equation (b) Schrödinger Equation (c) Naiver-Stokes Equation

Figure 3: Test error comparison results. The evaluation metric is the sum of L2RE on each component of the predicted physical field. Results of ABL-PDE (w/o FEM) and ABL-PDE show the mean and standard deviation of 5 experiments under different λ used in Eq. (2). The area of the shadowed circle represents the magnitude of the standard deviation. Detailed results are provided in Appendix G.2.

of model prediction on the noise-free test data. Compared with the best performance achievable by supervised training alone (cf. ABL-PDE (w/o CO)), ABL-PDE consistently outperforms it across all tasks. Initially, the neural network performs suboptimally, and the candidate term library is too expansive. Consistency optimization progressively clarifies the ambiguity within the library, leading to a refined equation that, in turn, provides physical insights to improve neural network.

In addition to our main experiments, we benchmark our method against another three representative approaches: D-CIPHER [16], SPL [25], and SGA-PDE [8]. However, limitations in these methods prevent a direct comparison on our primary benchmarks. Neither D-CIPHER nor SGA-PDE are designed for coupled PDE systems, and SPL's methodology is tailored to discovering algebraic equations and ODEs. Therefore, to ensure fair comparisons, we evaluate our method on the original benchmarks used in these respective studies. Since the experimental setups for these comparisons differ from those of our primary benchmarks, we present the results in Appendix G.4-G.6. These experiments serve to further validate the effectiveness and versatility of our approach.

6.4 Ablation Study

Influence of FEMTrain. We investigate the effect of FEMTrain by comparing the performance of our method with and without the utilization of FEM residual. Selecting a hyperparameter, e.g., λ in Eq. (2), to balance the physical information and the supervision signal has long been an existing issue with physics-informed machine learning [13]. Therefore, we conduct five experiments for each task with hyperparameters: $[0.5\lambda, 0.75\lambda, \lambda, 1.25\lambda, 1.5\lambda]$. The mean and standard deviation of coefficient errors are presented in Table 2, and the results of test errors are visualized in Figure 3. In addition to the superior performance of ABL-PDE in coefficient estimation and test error, its stability to hyperparameter variations is particularly noteworthy. We use the area of the shadowed circle to reflect the magnitude of the standard deviation. This enhanced stability is crucial for scientific discovery, as finding an effective hyperparameter is much easier than finding the optimal one.

To further probe the stability of our method, we perform a rigorous, large-scale hyperparameter sensitivity study under 10% noise conditions. Specifically, we vary the loss weight λ by multiplicative factors of 0.1, 1, and 10, while testing margin thresholds of 0.01, 0.05, and 0.1. The comprehensive results and a discussion guiding hyperparameter selection are presented in Appendix G.7.

Influence of `KB`. We use the one-dimensional PDE with a single dependent variable as a showcase to demonstrate the effectiveness of `KB` in constraining the search space. Figure 4 illustrates the logarithmic growth curve of the candidate term number as the operator number limit increases. Compared to the full space of expression terms, the proposed canonical form reduces the problem size to a computationally feasible scale. Nevertheless, `KB` still demonstrates a reduction by a constant factor in the exponent, making it tractable for POSS. As terms with 10 operators can construct highly complex equations, our method can be applied to most scenarios.

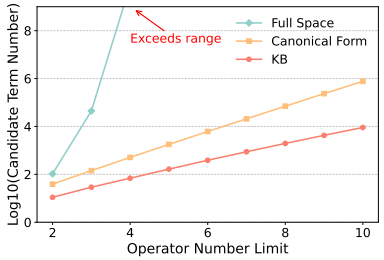


Figure 4: Logarithmic growth curve of the candidate term number as the operator number limit increases.

7 Conclusion

In this paper, we propose a novel PDE discovery method based on the abductive learning framework, aiming at leveraging the logic-based knowledge base and the neural network in a reciprocal way. Specifically, we introduce the ABductive Learning for PDE discovery (ABL-PDE) approach, which alternatively refines the knowledge base by utilizing a neural network to access data and enhances the neural network's performance by integrating the physical information provided by the knowledge base. Experimental results demonstrate the robustness of our approach in PDE structure identification, the accuracy in coefficient estimation, and the stability against hyperparameter variations. Furthermore, ABL-PDE is a general-purpose approach with sufficient flexibility in implementation, e.g., the neural network can adopt any advanced architecture, and the knowledge base can incorporate domain-specific information by declaring new integrity constraints. The limitation of ABL-PDE lies in its restriction to learning linear combination coefficients, which prevents it from handling unknown parameters within nonlinear terms. In future work, we will try to construct a parameterized candidate term library and incorporate advanced techniques for joint parameter optimization.

8 Acknowledgements

This research was supported by the Noncommunicable Chronic Diseases-National Science and Technology Major Project, No.2024ZD0531802. Cunjing Ge was supported by the National Natural Science Foundation of China (62202218). The authors would like to thank Reviewer #B51F of OpenReview for the thorough review and insightful suggestions.

References

[1] Steven Atkinson. Bayesian hidden physics models: uncertainty quantification for discovery of nonlinear partial differential operators from data. *arXiv:2006.04228*, 2020.

[2] Reza Akbarian Bafghi and Maziar Raissi. PINNs-Torch: Enhancing speed and usability of physics-informed neural networks with pytorch. In *The Symbiosis of Deep Learning and Differential Equations III*, 2023.

[3] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

[4] Steven L Brunton and J Nathan Kutz. Promising directions of machine learning for partial differential equations. *Nature Computational Science*, 4(7):483–494, 2024.

[5] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *National Academy of Sciences*, 113(15):3932–3937, 2016.

[6] Martin Burger, Luis Caffarelli, and Peter A Markowich. Partial differential equation models in the socio-economic sciences. *Royal Society A*, 372(2028):20130406, 2014.

[7] Alejandro Carderera, Sebastian Pokutta, Christof Schütte, and Martin Weiser. Cindy: Conditional gradient-based identification of non-linear dynamics–noise-robust recovery. *arXiv:2101.02630*, 2021.

[8] Yuntian Chen, Yingtao Luo, Qiang Liu, Hao Xu, and Dongxiao Zhang. Symbolic genetic algorithm for discovering open-form partial differential equations (SGA-PDE). *Physical Review Research*, 4(2):023174, 2022.

[9] Jean Coiffier. *Fundamentals of Numerical Weather Prediction*. Cambridge University Press, 2011.

[10] Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv:2305.01582*, 2023.

[11] A. Messenger Daniel and M. Bortz David. Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021.

[12] Urban Fasel, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Royal Society A*, 478(2260):20210904, 2022.

[13] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv:2211.08064*, 2022.

[14] Raymond M Hicks and Preston A Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[15] Zhongyi Jiang, Chunmei Wang, and Haizhao Yang. Finite expression methods for discovering physical laws from data. *arXiv preprint arXiv:2305.08342*, 2023.

[16] Krzysztof Kacprzyk, Zhaozhi Qian, and Mihaela van der Schaar. D-CIPHER: discovery of closed-form partial differential equations. *NeurIPS*, 2023.

[17] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.

[18] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning pdes from data. In *ICML*, 2018.

[19] Zichao Long, Yiping Lu, and Bin Dong. PDE-Net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.

[20] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[21] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. In *NeurIPS*, 2015.

[22] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[23] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[24] Gilbert Strang and George J. Fix. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, 2nd edition, 2008.

[25] Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search. In *ICLR*, 2023.

[26] Hao Xu, Haibin Chang, and Dongxiao Zhang. DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm. *Journal of Computational Physics*, 418:109584, 2020.

[27] Hao Xu, Dongxiao Zhang, and Nanzhe Wang. Deep-learning based discovery of partial differential equations in integral form from sparse and noisy data. *Journal of Computational Physics*, 445:110592, 2021.

[28] Zhi-Hua Zhou. Abductive learning: towards bridging machine learning and logical reasoning. *Science China Information Sciences*, 62(7):76101, 2019.

[29] Zhi-Hua Zhou and Yu-Xuan Huang. Abductive learning. In Pascal Hitzler and Md. Kamruzzaman Sarker, editors, *Neuro-Symbolic Artificial Intelligence: The State of the Art*, pages 353–369. IOS Press, Amsterdam, 2022.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction accurately represent the paper's contributions and scope. They are supported by the theoretical analysis and experimental results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discusses the limitations of the work in the conclusion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide the full set of assumptions and a complete (and correct) proof for each theoretical result in the paper.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed experimental settings in Appendix F and a detailed README file to reproduce the results in the supplementary material.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We use public datasets and provide code with a detailed README file to reproduce the results in the supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide detailed experimental settings in Appendix F and a detailed README file to reproduce the results in the supplementary material. We also analyze the influence of hyperparameters on the performance of the proposed method in the main text.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars for the proposed method in the main text.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the detailed experimental settings in Appendix F which contains the discussion of the compute resources used.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that the research conducted in the paper conforms to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly credit the creators of the assets and mention the license and terms of use explicitly.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We will release the code upon acceptance.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not involve research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The paper does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Detailed Explanation of the Canonical Form

The canonical form of expression terms is defined by the BNF grammar presented in Definition 3. This grammar uses standard notation: $\langle\cdot\rangle$ for nonterminal symbols, $::=$ for production rules (indicating that the nonterminal on the left can be replaced by the expression on the right), and | signifies alternation between possible forms. The notation used here is consistent with that in Definition 1. In this grammar, nonterminal symbols define the valid structure for terms corresponding to specific operators or types: $\langle\texttt{cfterm}\rangle$ for any canonical form term, $\langle\texttt{div}\rangle$ for division ($/$), $\langle\texttt{mul}\rangle$ for multiplication ($\times$), $\langle\texttt{add}\rangle$ for addition ($+$), $\langle\texttt{sub}\rangle$ for subtraction ($-$), $\langle\texttt{par}\rangle$ for partial differentiation (∂), $\langle\texttt{unit}\rangle$ for base multiplicative/derivative units, and $\langle\texttt{var}\rangle$, $\langle\texttt{dpv}\rangle$, $\langle\texttt{idpv}\rangle$ represent variables.

The production rules in Definition 3 define the set of expression terms that are in canonical form by imposing syntactic constraints on their structure. We explain the meaning of these rules as follows:

- The top-level rule, $\langle\texttt{cfterm}\rangle ::= \langle\texttt{div}\rangle \mid \langle\texttt{mul}\rangle \mid \langle\texttt{add}\rangle \mid \langle\texttt{sub}\rangle \mid \langle\texttt{unit}\rangle$, specifies that any expression in canonical form must be derivable from one of these five main nonterminals or be a base unit.
- The rule for division, $\langle\texttt{div}\rangle ::= \langle\texttt{mul}\rangle \ / \ \langle\texttt{add}\rangle \mid \langle\texttt{mul}\rangle \ / \ \langle\texttt{sub}\rangle$, defines the valid structure for division terms ($/$). It requires the numerator to be a term derivable from $\langle\texttt{mul}\rangle$ and the denominator to be a term derivable from either $\langle\texttt{add}\rangle$ or $\langle\texttt{sub}\rangle$. For example, a term corresponding to $(u+v)/(u-v)$ would be considered invalid under this rule because its numerator, $(u+v)$, is derivable from $\langle\texttt{add}\rangle$, not $\langle\texttt{mul}\rangle$.
- The recursive rule for multiplication, $\langle\texttt{mul}\rangle ::= \langle\texttt{unit}\rangle \ \times \ \langle\texttt{mul}\rangle \mid \langle\texttt{unit}\rangle$, defines the structure of multiplication terms ($\times$). The base case is any term derivable from $\langle\texttt{unit}\rangle$. The recursive step requires a term derivable from $\langle\texttt{unit}\rangle$ on the left of "$\times$" and a term derivable from $\langle\texttt{mul}\rangle$ on the right. This imposes a specific structure on multiplication chains; for instance, a product of three terms like $u \times v \times w$ can only be derived in the form $\langle\texttt{unit}\rangle \times \langle\texttt{mul}\rangle$, where the second term derivable from $\langle\texttt{mul}\rangle$ is $v \times w$. This enforces the structure $u \times (v \times w)$, preventing syntactically distinct but equivalent forms like $(u \times v) \times w$.
- The recursive rule for addition, $\langle\texttt{add}\rangle ::= \langle\texttt{mul}\rangle \ + \ \langle\texttt{add}\rangle \mid \langle\texttt{mul}\rangle$, similarly defines the structure of addition terms ($+$). It allows a base case of a term derivable from $\langle\texttt{mul}\rangle$, and a recursive step $\langle\texttt{mul}\rangle \ + \ \langle\texttt{add}\rangle$. This ensures a specific structure for addition chains; for example, an expression like $t + u \times v + w$ (where $u \times v$ is derivable from $\langle\texttt{mul}\rangle$) can only be derived as $\langle\texttt{mul}\rangle \ + \ \langle\texttt{add}\rangle$, where the first term derivable from $\langle\texttt{mul}\rangle$ is t and the term derivable from $\langle\texttt{add}\rangle$ is $u \times v + w$. This enforces the structure $t + (u \times v + w)$, preventing forms like $(t + u \times v) + w$.
- The rule for subtraction, $\langle\texttt{sub}\rangle ::= \langle\texttt{add}\rangle \ - \ \langle\texttt{add}\rangle$, defines subtraction terms ($-$), requiring both the left and right operands to be derivable from $\langle\texttt{add}\rangle$. This enforces a specific structure; for example, $u + v - w$ (where $u + v$ and w are derivable from $\langle\texttt{add}\rangle$) can only be derived as $(u + v) - w$, not $u - (w - v)$.
- The rule $\langle\texttt{unit}\rangle ::= \langle\texttt{dpv}\rangle \mid \langle\texttt{idpv}\rangle \mid \langle\texttt{par}\rangle$ defines terms derivable from $\langle\texttt{unit}\rangle$. These are referred to as unit expressions and serve as the base elements for multiplication and differentiation arguments.
- The rule for partial differentiation, $\langle\texttt{par}\rangle ::= \partial_{\langle\texttt{idpv}\rangle}\langle\texttt{dpv}\rangle \mid \partial_{\langle\texttt{idpv}\rangle}\langle\texttt{par}\rangle$, defines partial derivative terms (∂). It requires the derivative operator to be applied to a term derivable from either $\langle\texttt{dpv}\rangle$ or $\langle\texttt{par}\rangle$, and differentiation is always with respect to a term derivable from $\langle\texttt{idpv}\rangle$ (an independent variable). This imposes strong constraints, for instance, forbidding derivatives of products or sums as single terms, such as $\partial_x(u \times v)$, which would be invalid because $u \times v$ is neither derivable from $\langle\texttt{dpv}\rangle$ nor $\langle\texttt{par}\rangle$.

B A Pedagogical Example of Logic-based Knowledge Base

We provide a pedagogical example to elucidate the logic-based knowledge base and the abduction process discussed in the main text. As a pedagogical example, this knowledge base considers three basic operators ($+$, $-$, $/$), and the full implementation (`reasoning/expr.pl`) is provided in the code of the supplementary material. By querying `?- write_to_file.` in `SWI-Prolog`, we can generate all expression terms with up to a given number of operators (2 in this code). Abduced expressions are

all in the form similar to `mul(u,mul(v,v))`. A single integrity constraint is employed to ensure that multiplication expressions are in a canonical order, illustrating how various human knowledge can be declaratively stated through such constraints. The declarative nature simplifies the task by allowing the user to specify desired properties directly, with the generation of constraint-satisfying expressions entirely managed by the Prolog prover. The implementation of this simplified knowledge base and abduction process is detailed in the following Prolog code.

```
%%%%%%%%%%%%%%%%%%%%%%%%
% Background Knowledge
%%%%%%%%%%%%%%%%%%%%%%%%

% Declare the dependent variables
dep_var([u, v]).

% Rule for fraction expressions: (X/Y)
% Cur_num: Counter for the number of operations used so far
% Op_limit: Maximum operations allowed
fraction_expr(frac(X, Y), Cur_num, Op_limit) -->
    ['('],
    multiplication_expr(X, Cur_num1, Op_limit - 1), % Parse
        numerator (X)
    ['/'],
    addition_expr(Y, Cur_num2, Op_limit - Cur_num1 - 1), %
        Parse denominator (Y)
    {Cur_num is Cur_num1 + Cur_num2 + 1}, % Calculate total
        operations used (adding 1 for fraction)
    [')'].

% Rule for multiplication expressions: (X*Y)
% First clause handles actual multiplication
multiplication_expr(mul(X, Y), Cur_num, Op_limit) -->
    ['('],
    unit_expr(X, Cur_num1, Op_limit - 1), % Parse first operand
        (X)
    [*],
    multiplication_expr(Y, Cur_num2, Op_limit - Cur_num1 - 1),
        % Parse second operand (Y)
    {Cur_num is Cur_num1 + Cur_num2 + 1},
    {\+constraint(mul(X, Y))}, % Check that this multiplication
        doesn't violate integrity constraints
    [')'].

% Second clause for multiplication_expr
% This is how the recursion terminates
multiplication_expr(X, Cur_num, Op_limit) -->
    {0 =< Op_limit},
    unit_expr(X, Cur_num, Op_limit).

% Rule for addition expressions: (X+Y)
addition_expr(add(X, Y), Cur_num, Op_limit) -->
    ['('],
    multiplication_expr(X, Cur_num1, Op_limit - 1), % Parse
        first operand (X)
    [+],
    addition_expr(Y, Cur_num2, Op_limit - Cur_num1 - 1), %
        Parse second operand (Y)
    [')'],
    {Cur_num is Cur_num1 + Cur_num2 + 1}.

% Second clause for addition_expr
% This is how the recursion terminates
addition_expr(X, Cur_num, Op_limit) -->
```

```
    multiplication_expr(X, Cur_num, Op_limit).

% Rule for basic units
% This is a terminal rule that matches a single variable from
    the dependent variable list
unit_expr(X, Cur_num, Op_limit) -->
    {
        Cur_num is 0,
        Cur_num =< Op_limit,
        dep_var(DepVarList), member(X, DepVarList) % X must be
            one of the dependent variables
    },
    [X].

%%%%%%%%%%%%%%%%%%%%%%%%
% Abducible Predicates
%%%%%%%%%%%%%%%%%%%%%%%%
% Decide which terms can be utilized to explain the
    observation, i.e., which values of X can satisfy
    expression_term(X, Op_limit)
expression_term(X, Op_limit) --> fraction_expr(X, _, Op_limit).
expression_term(X, Op_limit) --> multiplication_expr(X, _,
    Op_limit).

%%%%%%%%%%%%%%%%%%%%%%%%
% Integrity Constraints
%%%%%%%%%%%%%%%%%%%%%%%%
% Constraint to enforce a canonical ordering of multiplication
    terms
% This prevents generating both (a*b) and (b*a) which are
    mathematically equivalent
constraint(mul(X, Y)) :- extract_first_mul_term(Y, FirstTerm),
    X @> FirstTerm.

% Helper predicate to extract the first term in a
    multiplication chain
% For mul(a, mul(b, c)), it returns 'a'
% For a simple term like 'u', it returns 'u'
extract_first_mul_term(Term, FirstTerm) :-
    (   functor(Term, mul, 2) % Check if Term is a
        multiplication
    ->  arg(1, Term, FirstTerm) % If yes, extract first argument
    ;   FirstTerm = Term % If not, the term itself is the first
        term
    ).

% Utility predicate to write all generated expressions to a file
write_to_file :-
    open('out.txt', write, Stream),
    findall([X, Y], phrase(expression_term(X, 2), Y), L), %
        Generate all expressions with max 2 operators
    forall(member([X, Y], L),
        (write(Stream, X), nl(Stream),
            write(Stream, Y), nl(Stream))),
    close(Stream).
```

C A Brief Introduction to POSS

POSS leverages Pareto Optimization to solve the following subset selection problem:

Algorithm 2 POSS

Input: all variables $V = \{X_1, \ldots, X_n\}$, a given criterion f and an integer parameter $k \in [1, n]$
Parameter: the number of iterations T and an isolation function $I : \{0,1\}^n \to \mathbb{R}$
Output: a subset of V with at most k variables

Let $s = \{0\}^n$ and $P = \{s\}$.
Let $t = 0$.
while $t < T$ **do**
  Select s from P uniformly at random.
  Generate s' from s by flipping each bit of s with probability $1/n$.
  if $\nexists z \in P$ such that $I(z) = I(s')$ and

$$(z.o_1 < s'.o_1 \wedge z.o_2 \leq s'.o_2) \text{ or } (z.o_1 \leq s'.o_1 \wedge z.o_2 < s'.o_2)$$

  then
    $Q = \{z \in P \mid I(z) = I(s') \wedge s'.o_1 \leq z.o_1 \wedge s'.o_2 \leq z.o_2\}$.
    $P = (P \setminus Q) \cup \{s'\}$.
  $t = t + 1$.
Return $\arg\min_{s \in P, |s| \leq k} f(s)$

Definition 7 (Sparse Regression)**.** Given all observation variables $V = \{X_1, \ldots, X_n\}$, a predictor variable Z and a positive integer k, define the mean squared error of a subset $F \subseteq V$ as

$$MSE_{Z,F} = \min_{\alpha \in \mathbb{R}^{|F|}} \mathbb{E}\left[\left(Z - \sum_{i \in F} \alpha_i X_i\right)^2\right].$$

Sparse regression is to find a set of at most k variables minimizing the mean squared error, i.e.,

$$\arg\min_{F \subseteq V} MSE_{Z,F} \quad \text{s.t.} \quad |F| \leq k.$$

We use it to roughly select the subset that contains possible relevant terms. The following is the pseudo-code of POSS:

In Algorithm 2, o_1 represents residual and o_2 represents the subset size. $I : \{0,1\}^n \to \mathbb{R}$ determines if two solutions are allowed to be compared: they are comparable only if they have the same isolation function value.

D More Details on FEMTrain

D.1 A Brief Introduction to FEM

Finite Element Method (FEM) is a well-established numerical method that provides a systematic framework for approximating solutions to PDEs.

We use a scalar field u with one spatial dimension, $[0, L]$, as a showcase to explain the process of using FEM to solve the following type PDE:

$$\partial_t u = f\left(u, x, \partial_x u, \partial_x^2 u, \ldots\right). \tag{8}$$

The first step to apply FEM is to discretize the spatial domain into smaller, finite subdomains called elements. Given an ordered set of points $X = \{x_1, x_2, \ldots, x_n\}$ with $x_1 = 0$ and $x_n = L$, they split the domain into $n - 1$ elements: $[x_1, x_2], \ldots, [x_{n-1}, x_n]$. The second step is to construct a set of basis functions $\{\phi_1, \phi_2, \ldots, \phi_n\}$ to approximate the solution u. ϕ_i is defined as follows:

$$\phi_i(x) = \begin{cases} 0, & \text{if } x < x_{i-1} \text{ or } x > x_{i+1}, \\ \frac{x - x_{i-1}}{x_i - x_{i-1}}, & \text{if } x_{i-1} \leq x < x_i, \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & \text{if } x_i \leq x \leq x_{i+1}. \end{cases}$$

The solution u is then approximated by $\tilde{u}$ defined as:

$$\tilde{u}\left(t,x\right)=\sum_{i=1}^{n}\alpha_i(t)\phi_i\left(x\right), \tag{9}$$

where $\alpha_i(t)=u(t,x_i)$. The third step is to transform Eq. (8) into *weak formulation* by multiplying both sides with each basis function and integrating over the domain:

$$\int_0^L \partial_t u\cdot\phi_i\,\mathrm{d}x=\int_0^L f\left(u,x,\ldots\right)\cdot\phi_i\,\mathrm{d}x. \tag{10}$$

Substitute Eq. (9) into Eq. (10), we get a linear system:

$$\boldsymbol{M}\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{\alpha}(t)=\boldsymbol{F}(t), \tag{11}$$

where $\boldsymbol{M}_{ij}=\int_0^L\phi_i\cdot\phi_j\,\mathrm{d}x$ is the so-called mass matrix in finite element analysis, $\boldsymbol{\alpha}(t)$ is the vector of coefficients in Eq. (9), and $F_i(t)=\int_0^L f\left(\tilde{u},x,\ldots\right)\cdot\phi_i\,\mathrm{d}x$. Eq. (11) is an Ordinary Differential Equation (ODE) system of $\boldsymbol{\alpha}$. Therefore, given the initial state, we can calculate $\boldsymbol{F}$ and forward $\boldsymbol{\alpha}$ iteratively, thus forecasting $\tilde{u}$ at any time.

D.2 FEM-based Residual

Firstly, we discretize the spatial domain Ω into a set of cubic elements $\mathcal{T}=\{\Omega_i\subset\mathbb{R}^d\}_{i=1}^{N_\Omega}$. Let $X=\{\boldsymbol{x}_i\in\mathbb{R}^d\}_{i=1}^{N}$ denote their vertices, and $\Phi=\{\phi_i:\Omega\to\mathbb{R}\}_{i=1}^{N}$ for the corresponding basis functions. Secondly, we use Φ to approximate $\partial_t u(t,\boldsymbol{x})\triangleq f^{(0)}(u(t,\boldsymbol{x}),\boldsymbol{x})$ and expression terms $\{f^{(i)}(u(t,\boldsymbol{x}),\boldsymbol{x})\}_{i=1}^{N_F}$ identified by MSS as follows:

$$\tilde{f}^{(i)}\left(u(t,\boldsymbol{x}),\boldsymbol{x}\right)=\sum_{j=1}^{N}\alpha_j^{(i)}(t)\phi_j\left(\boldsymbol{x}\right),i=0,1,\ldots,N_F, \tag{12}$$

where $\alpha_j^{(i)}(t)=f^{(i)}(u(t,\boldsymbol{x}_j),\boldsymbol{x}_j)$, which can be evaluated through the pre-trained neural network. Thirdly, we transform $f^{(0)}=\sum_{i=1}^{N_F}\xi_i\cdot f^{(i)}$ into the weak formulation:

$$\langle f^{(0)},\phi_j\rangle=\sum_{i=1}^{N_F}\xi_i\cdot\langle f^{(i)},\phi_j\rangle,j=1,2,\ldots,N, \tag{13}$$

where ξ_i is the coefficient as introduced in Eq. (1), and $\langle u,v\rangle=\int_\Omega u\cdot v\,\mathrm{d}\boldsymbol{x}$. Substituting (12) into (13) at a given time point t (omitted for brevity) yields the linear system:

$$\boldsymbol{M}\boldsymbol{\alpha}^{(0)}=\boldsymbol{M}\boldsymbol{A}\boldsymbol{\xi}, \tag{14}$$

where $\boldsymbol{M}_{ij}=\langle\phi_i,\phi_j\rangle$, $\boldsymbol{\alpha}^{(i)}=(\alpha_0^{(i)},\alpha_1^{(i)},\ldots,\alpha_N^{(i)})^T$, $\boldsymbol{A}=\left[\boldsymbol{\alpha}^{(1)}\,\boldsymbol{\alpha}^{(2)}\,\ldots\,\boldsymbol{\alpha}^{(N_F)}\right]$, and $\boldsymbol{\xi}=(\xi_0,\xi_1,\ldots,\xi_{N_F})^T$. Since $\{\boldsymbol{\alpha}^{(i)}\}_{i=0}^{N_F}$ are outputs of the neural networks, the following FEM-based residual can be utilized to optimize neural network parameters θ and PDE coefficients $\boldsymbol{\xi}$:

$$\|\boldsymbol{M}(\boldsymbol{\alpha}^{(0)}-\boldsymbol{A}\boldsymbol{\xi})\|_2 \tag{15}$$

Replacing $\mathrm{Res}(\mathtt{NN}_\theta,\boldsymbol{\xi},F)$ in Eq. (2) with this residual gives the loss function utilized in FEMTrain.

E Proofs

E.1 Theorem 1

Lemma 1. *Let e_1 and e_2 be two expressions generated by* `<sub>`*. Then, $e_1\times e_2$ can be rewritten as an expression generated by* `<sub>`*.*

Proof. According to Definition 3, e_1 can be expressed as $a - b$, e_2 can be expressed as $c - d$, where a, b, c, d are expressions generated by <add>, <mul>, <par>, <dep>, or <indep>. Without loss of generality, we only need to consider the <add> case, as all other cases are simply degenerate forms of this one. Since $(a - b) \times (c - d) = (ac + bd) - (bc + ad)$, we only need to prove $ac + bd$ and $bc + ad$ can be rewritten as expressions generated by <add>. Since a and c are summations of multiplication or unit expressions, it's obvious that we can rewrite ac as an expression generated by <add> according to basic algebraic laws, so as bd, bc, and ad. Therefore, $(ac + bd) - (bc + ad)$ is a subtraction between two expressions generated by <add>, which can be rewritten as an expression generated by <sub>.

□

Lemma 2. *Let e_1 and e_2 be two expressions generated by* *. Then, $e_1 \times e_2$ can be rewritten as an expression generated by* *.*

Proof. Without loss of generality, we assume $e_1 = \frac{a}{b-c}$, $e_2 = \frac{d}{e-f}$, where a and d are expressions generated by <mul>; b, c, e, and f are expressions generated by <add>. According to Lemma 1, the denominator of $e_1 \times e_2$ can be written as an expression generated by <sub>. The numerator of $e_1 \times e_2$ can be rewritten as an expression generated by <mul>, thus $e_1 \times e_2$ can be rewritten as an expression generated by .

□

Lemma 3. *Let e_1 and e_2 be two expressions generated by* *. Then, the numerator and denominator of e_1* op *e_2, where* op *$\in \{+, -, /\}$ can all be rewritten as expressions generated by* <sub>*.*

Proof. Without loss of generality, we assume $e_1 = \frac{a}{b-c}$, $e_2 = \frac{d}{e-f}$, where a and d are expressions generated by <mul>; b, c, e, and f are expressions generated by <add>. We prove the "+" case and others are similar. Since $e_1 + e_2 = \frac{a(e-f)+d(b-c)}{(b-c)(e-f)}$, according to Lemma 1, $a(e - f) + d(b - c)$ and $(b - c)(e - f)$ are all expressions generated by <sub>.

□

Lemma 4. *Let e be an expression generated by* *. Then, $\frac{\partial e}{\partial x}$ can be subsumed by a set $H' \subset H$, where x is an arbitrary independent variable.*

Proof. Without loss of generality, we assume $e = \frac{a}{b-c}$, where a is an expression generated by <mul>; b and c are expressions generated by <add>. According to differentiation rule, $\partial_x \frac{a}{b-c} = \frac{\partial_x a(b-c) - a\partial_x(b-c)}{(b-c)^2}$. According to Lemma 1, $(b-c)^2$ is an expression generated by <sub>. According to the differentiation rule and basic laws of multiplication, the numerator is an expression generated by <sub>. Thus, $\frac{\partial e}{\partial x}$ can be subsumed by a set $H' \subset H$.

□

Proof of Theorem 1. We use structural induction to prove Theorem 1.

1. **Base Case:** For each expression term s with zero operator, there exists a term $h \in H$ such that $s = h$.

   This case is obviously true.

2. **Inductive Hypothesis:** Assume that for each expression term s with no greater than n operators, there exists a subset of $H' \subset H$ such that H' subsumes s.

3. **Inductive Step:** Show that for each expression term s with $n + 1$ operators, there exists a subset of $H' \subset H$ such that H' subsumes s.

   If s is an addition or subtraction, according to the inductive hypothesis, the conclusion is obviously true.

   If s is a multiplication, assume $s = a \times b$, according to the inductive hypothesis, a and b can be subsumed by two sets $H_a \subset H$ and $H_b \subset H$ respectively. Without loss of generality, we assume each element of H_a and H_b are all expressions generated by . According to the distributive law of multiplication and Lemma 2, the conclusion holds.

If s is a division, let $s = a/b$, according to the inductive hypothesis, a and b can be subsumed by two sets $H_a \subset H$ and $H_b \subset H$ respectively. Without loss of generality, we assume each element of H_a and H_b are all expressions generated by `<div>`. Then, we only need to prove $\frac{c+d}{e+f}$ can be subsumed by a set $H' \subset H$, where c, d, e, and f are all expressions generated by `<div>`. Other cases are all degenerated cases. According to Lemma 3, $\frac{c+d}{e+f}$ can be rewrite as $\frac{g/h}{j/k}$, where g, h, j, and k are all expressions generated by `<sub>`. According to Lemma 1, it's obvious that there exists a set $H' \subset H$ subsumes $\frac{gk}{jh}$.

If s is a partial derivative, let $s = \partial e$. According to the inductive hypothesis, e can be subsumed by a set $H_e \subset H$. According to the linearity of differentiation, we can assume e is a single element of H. Without loss of generality, we assume e is an expression generated by `<div>`. According to Lemma 4, the conclusion holds.

4. **Conclusion:** By the principle of mathematical induction, Theorem 1 holds.

E.2 Proof of Theorem 2

Proof. Let F_1^* be a subset of F with cardinality k_1 that realizes the k_1-level margin, such that $\mathrm{LevelMargin}_F(k_1) = \mathrm{Margin}_F(F_1^*)$.

Since $k_1 < k_2 \leq |F|$, we can construct a new set F_2' by adding $k_2 - k_1$ arbitrary terms from $F \setminus F_1^*$ to F_1^*. By construction, $F_1^* \subseteq F_2'$ and $|F_2'| = k_2$.

The minimal residual is found by solving a linear least-squares problem. The solution space for the coefficients $\boldsymbol{\xi}$ corresponding to the terms in F_1^* is a subspace of that for the terms in F_2'. Therefore, the minimal residual over the larger set F_2' cannot be greater than the minimal residual over its subset F_1^*:

$$\left\|\mathrm{Res}^*(\mathtt{NN}_\theta, F_1^*)\right\|_2 \geq \left\|\mathrm{Res}^*(\mathtt{NN}_\theta, F_2')\right\|_2 .$$

From the definition of Margin (Definition 4), this directly implies that:

$$\mathrm{Margin}_F(F_1^*) \geq \mathrm{Margin}_F(F_2').$$

Furthermore, by the definition of k-Level Margin (Definition 5), the k_2-level margin is the minimum margin over all subsets of size k_2. The margin of our constructed subset F_2' must therefore be greater than or equal to this minimum:

$$\mathrm{Margin}_F(F_2') \geq \mathrm{LevelMargin}_F(k_2).$$

Combining these inequalities, we arrive at the desired result:

$$\mathrm{LevelMargin}_F(k_1) = \mathrm{Margin}_F(F_1^*) \geq \mathrm{Margin}_F(F_2') \geq \mathrm{LevelMargin}_F(k_2).$$

This completes the proof. □

F More Experimental Details

F.1 Equation Form

Table 3: Summary of Governing Equations

Equation	Form
Burgers' Equation	$\partial_t u = -u\partial_x u + \left(\frac{0.01}{\pi}\right)\partial_{xx} u$
Schrödinger Equation	$\begin{cases} \partial_t u = -0.5\partial_{xx} v - v(u^2 + v^2) \\ \partial_t v = 0.5\partial_{xx} u + u(u^2 + v^2) \end{cases}$
Navier-Stokes Equation	$\begin{cases} \partial_t u = -u\partial_x u - v\partial_y u - \partial_x p + 0.01(\partial_{xx} u + \partial_{yy} u) \\ \partial_t v = -u\partial_x v - v\partial_y v - \partial_y p + 0.01(\partial_{xx} v + \partial_{yy} v) \end{cases}$

F.2 L2 Relative Error

Definition 8 (L2 Relative Error)**.** Let $f = [f_1, f_2, \ldots, f_n]$ be a vector of true values and $\hat{f} = [\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n]$ be a vector of approximations. The L2 Relative Error (L2RE) is defined as:

$$\text{L2RE} = \frac{\|f - \hat{f}\|_2}{\|f\|_2} = \frac{\sqrt{\sum_{i=1}^{n} |f_i - \hat{f}_i|^2}}{\sqrt{\sum_{i=1}^{n} |f_i|^2}}.$$

Please note that for coefficient estimation, the L2RE is simplified as follows:

$$\sum_{i=1}^{n} \frac{|\xi_i - \hat{\xi}_i|}{|\xi_i|}.$$

The benefit of using this metric is that errors in large coefficients do not dominate and obscure errors in small coefficients.

F.3 Network Architecture and Training Details

Our model's architecture is a multilayer perceptron (MLP) composed of 8 hidden layers. For the Burgers' Equation, each layer has 20 neurons. This architecture is adapted for the Schrödinger Equation by modifying the output layer to predict two variables (the real and imaginary parts of the solution). For the more complex Navier-Stokes Equations, we increase the network's width to 40 neurons per hidden layer.

The training data consists of 1×10^4 (40%), 2×10^4 (40%), and 2×10^5 (20%) measurement points for the Burgers', Schrödinger, and Navier-Stokes experiments, respectively. We hold out 20% of the training data as a validation set, which is used to select the derivative computation weights and the hyperparameter λ. The networks are pre-trained for 5,000 epochs for the Burgers' and Schrödinger equations, and 100,000 epochs for the Navier-Stokes equations. Across all tasks, we use a consistent initial sparsity of 10 and a margin threshold of 0.05. During consistency optimization, MSS is applied every 1,000 epochs to refine the discovered equation. The values for λ are selected based on validation error and are summarized in Table 4. All experiments were conducted on a server equipped with four NVIDIA A6000 GPUs, each with 48 GB of memory.

Table 4: Hyperparameter λ settings.

Method	Burgers' Equation	Schrödinger Equation	Navier-Stokes Equations
ABL-PDE (w/o FEM)	0.05	0.5	1
ABL-PDE	5	5	200

G More Experimental Results

G.1 Details of Redundant Terms

Table 5 details the specific redundant terms for the dagger-marked (†) results presented in Table 2.

Table 5: Redundant terms of the dagger-marked results in Table 2.

Method	Equation	Noise Level	Redundant Terms
ABL-PDE (w/o CO)	Schrödinger Eq.	All Levels	u_t: u_{xx}, u, uv, u^3, v v_t: –
DL-PDE++	Navier-Stokes Eq.	0%	u_t: – v_t: v_x
		5%	u_t: p^2u, vv_x v_t: p_yu, pu_y
		10%	u_t: p^2u, vv_x, p^3, p^2 v_t: p_yu, pu_y, v_x

G.2 Test Error

Table 6 presents the detailed results of test errors on three benchmarks. The area of the shaded circles in Figure 3 is proportional to the square root of the standard deviation. We use the square root to improve the visibility of the standard deviation of ABL-PDE.

Table 6: Test error comparison results on three benchmarks. The evaluation metric is the sum of L2RE (%) on each dependent variable. Results of ABL-PDE (w/o FEM) and ABL-PDE are the mean and standard deviation of 5 experiments with different hyperparameters.

Method	Burgers' Equation			Schrödinger Equation			Naiver-Stokes Equation		
	0	5%	10%	0	5%	10%	0	5%	10%
ABL-PDE (w/o CO)	1.79	1.83	1.92	2.25	2.37	3.93	2.85	3.38	4.70
ABL-PDE (w/o FEM)	1.04 ± 0.11	1.06 ± 0.11	1.08 ± 0.19	1.67 ± 0.40	1.74 ± 0.38	1.84 ± 0.36	3.06 ± 0.22	3.44 ± 0.25	4.02 ± 0.24
ABL-PDE	$\mathbf{0.68 \pm 0.01}$	$\mathbf{0.72 \pm 0.02}$	$\mathbf{0.85 \pm 0.03}$	$\mathbf{1.22 \pm 0.01}$	$\mathbf{1.36 \pm 0.01}$	$\mathbf{1.75 \pm 0.01}$	$\mathbf{2.64 \pm 0.05}$	$\mathbf{3.17 \pm 0.04}$	$\mathbf{3.88 \pm 0.06}$

G.3 Redundancy Reduction

Table 7 presents the detailed number of candidate terms.

Table 7: Number of candidate terms w.r.t. operator number limit.

Number	2	3	4	5	6	7	8	9	10
Full	105	44205	7.8E+9	2.4E+20	2.4E+41	2.3E+83	2.1E+167	1.7E+335	1.2E+671
Canonical	39	143	510	1784	6141	20849	69948	234789	764761
ABL-PDE	11	29	69	165	387	879	1955	4254	9053

G.4 Comparison with D-CIPHER

Discovery of Closed-Form Partial Differential Equations (D-CIPHER) [16] is a recent study aims at going beyond the linear combination assumption that exists in most PDE discovery methods. While allowing for more flexible forms of the derivative-free part, D-CIPHER comes at the cost of a harder optimization process. As stated by its authors, D-CIPHER exhibits difficulty in learning coupled PDE systems. Consequently, it was not suitable for our experiments on the Schrödinger and Navier-Stokes equations discussed in the main text.

Despite this limitation, we evaluated D-CIPHER on its original Burgers' and Heat Equation datasets. Adhering to their hyperparameter settings (10 datasets, 10 trials), we report the best results using their RMSE metric in Table 8.

It is crucial to note that the setup for their Burgers' Equation experiment differs significantly from our benchmark. The coefficient of the u_{xx} term in their study is 0.2, substantially larger than the $0.01/\pi$ used in our work. This larger coefficient corresponds to a higher-viscosity regime, which naturally dampens the formation of sharp shock waves. In this context, the strong performance of both methods is readily achievable, particularly in the absence of noise.

Table 8: Comparison Results with D-CIPHER.

Method	Burgers' Equation			Heat Equation		
	0	5%	10%	0	5%	10%
D-CIPHER	1.61×10^{-2}	8.13×10^{-2}	1.49×10^{-1}	4.37×10^{-4}	1.87×10^{-2}	2.62×10^{-2}
ABL-PDE	$\mathbf{1.77 \times 10^{-3}}$	$\mathbf{4.17 \times 10^{-3}}$	$\mathbf{7.11 \times 10^{-3}}$	$\mathbf{4.04 \times 10^{-4}}$	$\mathbf{5.03 \times 10^{-5}}$	$\mathbf{1.17 \times 10^{-4}}$

G.5 Comparison with SPL

Symbolic Physics Learner (SPL) [25] presents a novel approach that uses Monte Carlo Tree Search (MCTS) to discover governing equations. Its primary focus is on identifying fundamental physical laws that are often expressed as algebraic equations, and systems of Ordinary Differential Equations

(ODEs). This scope, which does not include the PDEs central to our work, makes a direct application to our benchmarks non-trivial. To establish a point of comparison, we therefore adapted our method to the ODE setting and evaluated it on the Lorenz system with 5% Gaussian noise, a benchmark used in SPL's experiments. For our method, we used the same hyperparameters as in our main experiments: an initial sparsity of 10 and a margin threshold of 0.05. For the loss weighting parameter λ, we simply used a trick to rescale the supervised loss and the physics loss to similar magnitudes for all experiments: $\text{loss} = \text{loss}_1 + \frac{\text{loss}_2}{\text{loss}_2/\text{loss}_1 + 1e-8}$. We present the discovered equations in Table 9.

Table 9: Comparison of discovered equations with SPL.

Method	**Discovered Equations**
Ground Truth	$\dot{x} = -10x + 10y$ $\dot{y} = 28x - y - xz$ $\dot{z} = -\frac{8}{3}z + xy$
SPL [25]	$\dot{x} = -9.966x + 9.964y$ $\dot{y} = 27.764x - 0.942y - 0.994xz$ $\dot{z} = -2.655z + 0.996xy$
ABL-PDE (w/o CO)	$\dot{x} = -9.807x + 9.956y + 0.0323xy - 0.0185yz$ $\dot{y} = 27.72x - 0.976y - 0.989xz$ $\dot{z} = -2.698z + 0.950xy + 0.0238yz$
ABL-PDE	$\dot{x} = -10.003x + 10.002y$ $\dot{y} = 27.869x - 0.976y - 0.995xz$ $\dot{z} = -2.667z + 0.999xy$

The results in Table 9 highlight two key advantages of ABL-PDE. First, compared to SPL, ABL-PDE achieves higher precision in coefficient estimation, with its discovered parameters being notably closer to the ground truth values. Second, the comparison with the ablation model, ABL-PDE (w/o CO), underscores the effectiveness of our consistency optimization process. The ablation model incorrectly includes several superfluous terms (e.g., xy and yz in the $\dot{x}$ equation), whereas ABL-PDE successfully prunes these, recovering the correct symbolic form for the whole system.

G.6 Comparison with SGA-PDE

Symbolic Genetic Algorithm for open-form PDE Discovery (SGA-PDE) [8] is a prominent asynchronous method that utilizes a genetic algorithm to discover PDEs. It represents candidate equations with a flexible binary tree structure, which allows for a wide range of possible equation forms. However, this flexibility makes it challenging to impose structural constraints. As the official implementation of SGA-PDE does not support the discovery of coupled PDE systems, and our primary benchmarks are systems, we conducted a comparison on their original, noise-free PDE datasets to ensure a fair evaluation. For our method, we used the same hyperparameters as in our main experiments: an initial sparsity of 10 and a margin threshold of 0.05. The loss weighting parameter λ adopts the same trick as in the SPL comparison. The discovered equations are presented in Table 10.

A key difference in the experimental setup is that SGA-PDE calculates derivatives via numerical differentiation on a full mesh grid. In contrast, our method operates on sparsely sampled data. Specifically, for the Burgers' equation, we used 25% points of the data; for the Chafee-Infante equation, we used 50% points of the data; and for the KdV equation, we used 20% points of the data. The results in Table 10 show that ABL-PDE achieves a comparable level of accuracy to SGA-PDE while using significantly less data. The comparison with the ablation model, ABL-PDE (w/o CO), underscores the effectiveness of our consistency optimization process in improving the robustness of the discovery process to hyperparameter selection.

G.7 Large Scale Hyperparameter Sensitivity Study

To further investigate the robustness of our method, we conduct a sensitivity analysis over a wider range of hyperparameters, focusing on the challenging scenario with 10% noise. We vary the loss

Table 10: Comparison of discovered equations with SGA-PDE.

Method	Discovered Equation
Burgers' Equation	
Ground Truth	$u_t = -uu_x + 0.1u_{xx}$
SGA-PDE [8]	$u_t = -1.0011uu_x + 0.1024u_{xx}$
ABL-PDE (w/o CO)	$u_t = -0.83uu_x + 0.096u_{xx} - 0.026u + 0.038u^3$
ABL-PDE	$u_t = -0.999uu_x + 0.10002u_{xx}$
KdV Equation	
Ground Truth	$u_t = -0.0025u_{xxx} - uu_x$
SGA-PDE [8]	$u_t = -0.0025u_{xxx} - 1.0004uu_x$
ABL-PDE (w/o CO)	$u_t = -0.00237u_{xxx} - 0.988uu_x$
ABL-PDE	$u_t = -0.00249u_{xxx} - 0.995uu_x$
Chafee-Infante Equation	
Ground Truth	$u_t = u_{xx} - u + u^3$
SGA-PDE [8]	$u_t = 1.0002u_{xx} - 1.0008u + 1.0004u^3$
ABL-PDE (w/o CO)	$u_t = 0.843u_{xx} - 0.852u + 0.916u^3 - 0.0619uu_x + 0.182u_x$
ABL-PDE	$u_t = 0.994u_{xx} - 0.987u + 0.996u^3$

weight λ by an order of magnitude (0.1λ, λ, 10λ) and the margin threshold (0.01, 0.05, 0.1). The results for ABL-PDE and its ablation variant, ABL-PDE (w/o CO), on all three PDE benchmarks are summarized in Tables 11 to 13.

Analysis The results demonstrate that ABL-PDE is highly robust to variations in λ. Specifically, with the margin threshold set to 0.05, our method consistently discovers the correct equations, or ones with minimal errors (e.g., 3(1) for Burgers' and 7(1) for Schrödinger's), across all tested λ values.

The choice of the margin threshold is crucial, and our experiments suggest that 0.05 is an effective value. A threshold that is too small (e.g., 0.01) tends to introduce numerous redundant terms. Although the consistency optimization process is designed to prune such terms, an excessive amount of incorrect physical information can trap the optimization in a local minimum. Restarting the consistency optimization or employing a solver-based verification could potentially address this issue, which we leave as future work. Conversely, a threshold that is too large (e.g., 0.1) requires the algorithm to identify and add missing terms to the initial discovery, a task that is empirically more difficult than pruning redundant ones. Therefore, we empirically favor a relatively small margin threshold.

Table 11: Hyperparameter sensitivity analysis for Burgers' Equation with 10% noise. Each cell shows the number of discovered terms (sum of redundant and missing terms in parentheses). Entries in red denote correctly identified equations.

Model	λ	Margin Threshold		
		0.01	0.05	0.1
ABL-PDE (w/o CO)	0.1λ	4(2)	2(0)	2(0)
	λ	4(2)	2(0)	2(0)
	10λ	4(2)	2(0)	2(0)
ABL-PDE	0.1λ	3(1)	3(1)	4(2)
	λ	5(3)	2(0)	2(0)
	10λ	4(2)	2(0)	2(0)

Table 12: Hyperparameter sensitivity analysis for Schrödinger Equation with 10% noise.

Model	$\boldsymbol{\lambda}$	**Margin Threshold**		
		0.01	0.05	0.1
ABL-PDE (w/o CO)	0.1λ	16(10)	11(5)	8(2)
	λ	16(10)	11(5)	8(2)
	10λ	16(10)	11(5)	8(2)
ABL-PDE	0.1λ	15(9)	7(1)	6(0)
	λ	14(8)	6(0)	6(0)
	10λ	12(6)	6(0)	6(0)

Table 13: Hyperparameter sensitivity analysis for Navier-Stokes Equations with 10% noise.

Model	$\boldsymbol{\lambda}$	**Margin Threshold**		
		0.01	0.05	0.1
ABL-PDE (w/o CO)	0.1λ	16(5)	10(0)	8(2)
	λ	16(5)	10(0)	8(2)
	10λ	16(5)	10(0)	8(2)
ABL-PDE	0.1λ	11(0)*	10(0)	8(2)
	λ	14(3)	10(0)	8(2)
	10λ	15(4)	10(0)	8(2)

*Note: Under the margin threshold of 0.01, the discovered equation for u_t includes both uu_x and uv_y. Due to the continuity equation, these terms are equivalent. Although uv_y is not in the standard form, we do not classify it as a redundant term, resulting in 11 discovered terms with 0 error.