D-CIPHER: Discovery of Closed-form PDEs

Krzysztof Kacprzyk  
University of Cambridge  
kk751@cam.ac.uk

Zhaozhi Qian  
University of Cambridge  
zq224@cam.ac.uk

Mihaela van der Schaar  
University of Cambridge, UCLA,  
The Alan Turing Institute  
mv472@cam.ac.uk

Abstract

Closed-form differential equations, including partial differential equations and higher-order ordinary differential equations, are one of the most important tools used by scientists to model and better understand natural phenomena. Discovering these equations directly from data is challenging because it requires modeling relationships between various derivatives that are not observed in the data (equation-data mismatch) and it involves searching across a huge space of possible equations. Current approaches make strong assumptions about the form of the equation and thus fail to discover many well-known systems. Moreover, many of them resolve the equation-data mismatch by estimating the derivatives, which makes them inadequate for noisy and infrequently sampled systems. To this end, we propose D-CIPHER, which is robust to measurement artifacts and can uncover a new and very general class of differential equations. We further design a novel optimization procedure, CoLLie, to help D-CIPHER search through this class efficiently. Finally, we demonstrate empirically that it can discover many well-known equations that are beyond the capabilities of current methods.

1 Introduction

Scientists have been using mathematical equations to describe the world for centuries. In particular, closed-form differential equations turned out to be one of the best tools to model physical phenomena. A differential equation describes a relationship between a quantity and its derivatives (rates of change); it is called closed-form if this relationship is described by a mathematical expression consisting of a finite number of variables, constants, arithmetic operations, and some well-known functions (e.g., exponent, logarithm, trigonometric functions).

Closed-form differential equations provide a general description of reality in a concise representation that is amenable to closer inspection by scientists. This renders them transparent and interpretable to human experts.

Discoveries of these equations required a thorough knowledge of the theory, strong mathematical skills, substantial creativity, and good intuition.

The goal of this work is to discover closed-form differential equations directly from data thus accelerating the process of scientific discovery.

Challenges in discovering differential equations from data

- Partial and higher-order derivatives. Many algorithms can only identify Ordinary Differential Equations (ODEs) which evolve only with respect to one variable (usually time). In contrast, many natural phenomena are described by equations involving many variables (e.g., spatial co-

[1] Detailed discussion in Appendix A.2

Preprint. Under review.
ordinates) called Partial Differential Equations (PDEs). Many systems also involve higher-order derivatives.

- **Derivatives not observed.** Discovering differential equations from data is challenging because the derivatives are usually not observed in the dataset (equation-data mismatch [30]). This makes verifying a candidate equation a non-trivial task. Most of the methods proposed in the literature try to resolve this issue by estimating the derivatives [8, 34]. However, estimating the derivative is difficult, especially when the data is sampled infrequently or with high noise [30, 23].

- **Strong assumptions and constrained search space.** The majority of algorithms for identifying differential equations make many assumptions about the form of the equation. In particular, they make the evolution assumption (defined and explained later) and assume that the equation can be represented as a linear combination of some prespecified functions and differential operators [8, 23]. However, many well-known systems, such as a forced harmonic oscillator or an inhomogeneous wave equation, cannot be represented in that way.

Currently, a few algorithms tackle only some of these challenges. In particular, Weak SINDy [23] is able to discover PDEs without estimating the derivative by utilizing a variational approach. However, the form of the equation is constrained to be in a form amenable for a sparse regression algorithm. D-CODE [30], on the other hand, uses a variational approach in conjunction with a symbolic regression algorithm to discover closed-form ODEs. However, it cannot handle higher-order derivatives or multiple independent variables, so it cannot be used to discover closed-form PDEs. The algorithms that do not require the evolution assumption appeared in [22] and [15] but they require derivative estimation and only consider equations represented as linear combinations of prespecified functions.

**Contributions**

In this work, we develop the **D**iscovery of **C**losed-form **P**artial and **H**igher-order **D**ifferential **E**quations in a **R**obust Way framework (D-CIPHER) that does not estimate the derivatives, requires fewer assumptions, has a broader search space than previous techniques, and works for both higher-order ODEs and PDEs. Our contributions are as follows:

- We examine the landscape of different types of PDEs from the discovery perspective. In particular, we introduce new notions such as evolution form, evolution assumption, derivative-bound part, and derivative-free part. We use them to describe what kinds of PDEs can be discovered with current methods and to motivate our new class of differential equations. (Section 2)

- We propose a new general class of PDEs (Variational-Ready PDEs) that admit the variational formulation (and thus allows to circumvent the derivative estimation). We also prove a theorem that motivates a novel objective function. (Section 3)

- We use the novel objective function to develop D-CIPHER, a new algorithm that searches over the Variational-Ready PDEs. (Section 5)

- We develop a new optimization procedure (CoLLie) to efficiently solve a constrained least-squares problem and thus help D-CIPHER search through this space efficiently. (Section 6)

## 2 Partial Differential Equations

**Notation and definitions.** We denote the set \{1, 2, \ldots, n\} as \([n]\) and the set of non-negative integers as \(\mathbb{N}_0\). Throughout this paper we let \(M, N, K \in \mathbb{N}\) be some natural numbers and let \(\Omega \subset \mathbb{R}^M\) be an open set inside \(\mathbb{R}^M\). A comprehensive table with all symbols used in this work can be found in Appendix A together with some definitions restated more formally.

### 2.1 Going beyond ODEs

The simplest differential equations are ordinary differential equations that describe quantities that evolve with respect to only one independent variable, usually time. Most methods assume that the ODE is explicit and as such can be represented as a system of first-order ODEs:

\[
\dot{u}_j(t) = f_j(t, u(t))
\]

where \(\dot{u}_j\) represents the derivative of \(u_j\). Then the discovery problem is reduced to deciding the order of the derivative (usually first or second) and the discovery of \(f_j\).

For PDEs it is not enough to talk about the derivative, as we can take derivatives with respect to different variables. We denote the mixed derivative as \(\partial^\alpha\), where \(\alpha \in \mathbb{N}_0^M\) is called a multi-index,
and define it as \( \partial^{\alpha} = \partial_{x_1}^{\alpha_1} \partial_{x_2}^{\alpha_2} \ldots \partial_{x_M}^{\alpha_M} \). Each \( \partial_{x_i}^{\alpha_i} = \partial^{\alpha_i} / \partial x_i^{\alpha_i} \) is a \( \alpha_i \)-order partial derivative with respect to \( x_i \) (the \( i \)-th independent variable)\(^2\). We define the order of \( \alpha \) as \( |\alpha| = \sum_{i=1}^M \alpha_i \). We call \( \partial^{\alpha} \) non-trivial if \( |\alpha| > 0 \).

A PDE of order \( K \) is any equation of the form

\[
f(x, u(x), \partial^{[K]} u(x)) = 0 \quad \forall x \in \Omega
\]

(2)

where \( \partial^{[K]} \) are all non-trivial mixed derivatives of all \( u_j \) \((j \in [N])\) up to the \( K \)-th order. We call a PDE closed-form if \( f \) is closed-form.

As PDEs might include many different combinations of derivatives, there is no generally accepted counterpart of explicit ODEs in the space of PDEs.

### 2.2 Evolution Assumption

Although there is no generally accepted notion of an explicit PDE, we define an evolution form of a PDE to be an equation of the form

\[
\partial^{\alpha} u_j(x) = f(x, u(x), \partial^{[K]/\alpha} u(x)) \quad \forall x \in \Omega
\]

(3)

where \( \partial^{[K]/\alpha} \) is \( \partial^{[K]} \) with \( \partial^{\alpha} \) omitted, \( \alpha \) is a known multi-index and \( j \in [N] \). Note that if \( u \) depends on only one variable and \( |\alpha| = K \) then equation (3) becomes exactly the definition of an explicit ODE.

In fact, many algorithms for PDE discovery assume a particular evolution form. We call it an evolution assumption (EA). However, this assumption requires the knowledge of \( \alpha \) and \( j \) which might not be trivial. Usually, \( \partial^{\alpha} \) is assumed to be the first derivative with respect to time (\( \partial_t \)) but it is not the case for many well-known PDEs such as the wave equation \((\partial_t^2 u - k \partial_x^2 u = 0)\) or Gauss’s law \((\nabla \cdot u = 0)\).

D-CIPHER does not need the evolution assumption. Moreover, it can discover some PDEs that cannot be put into the evolution form.

### 2.3 Derivative-bound part and derivative-free part

Any PDE can be expressed in the form

\[
f(x, u(x), \partial^{[K]} u(x)) - g(x, u(x)) = 0 \quad \forall x \in \Omega
\]

(4)

where we collect all the terms with the derivatives into \( f(x, u(x), \partial^{[K]} u(x)) \) and all terms without the derivatives into \( g(x, u(x)) \). We call \( f \) the derivative-bound part and \( g \) the derivative-free part. We also denote them as \( \partial \)-bound and \( \partial \)-free. The significance of the \( \partial \)-free part is that it can be evaluated directly given \( u \), whereas the \( \partial \)-bound part requires access to the derivatives which are not observed. It is important to note that for first-order ODEs, \( f \) is trivial and is just equal to \( \dot{u}_j \).

The challenge of unobserved derivatives might put constraints on the \( \partial \)-bound part of the PDE and thus in practice, it might not be possible to search over all closed-form PDEs. However, we observe that no such constraints need to be put on the \( \partial \)-free part (as it does not include the derivatives). We take full advantage of this observation and search separately for the two parts. We search over all closed-form functions \( g \) and for each candidate, we try to find the best counterpart \( f \) among the allowed expressions. This is very different from the previous approaches, which either do not need to find \( f \) as they work only for first-order ODEs \([30]\) or they constrain equally both the \( \partial \)-bound part and \( \partial \)-free part to be a linear combination of some pre-specified functions \([23]\).

### 3 Variational-Ready PDEs

In this section, we propose a new and very general class of PDEs, the Variational-Ready PDEs (VR-PDEs), which can be characterized without referring to the derivative. The VR-PDEs allow arbitrary \( \partial \)-free part but make some minor restrictions on the \( \partial \)-bound part. These restrictions allow one to use the variational formulation of PDEs to circumvent derivative estimation entirely.

\(^2\)Throughout this work we assume that the functions we use are smooth enough for the equality of mixed partials \([37]\) to hold. In that case any mixed derivative can be uniquely specified by a multi-index.
Despite the minor restriction, VR-PDEs contain many well-known PDEs, including all linear PDEs, Maxwell’s equations, and Navier-Stokes equations. Examples of equations from science that are VR are provided in Appendix B.

To define the new class of PDEs, we need the following definition.

**Definition 1 (Extended derivative and differential operator).** Let \( \alpha \in \mathbb{N}^M_0, |\alpha| \leq K \), be a multi-index. Let \( a : \mathbb{R}^M \rightarrow \mathbb{R} \) and \( h : \mathbb{R}^{M+N} \rightarrow \mathbb{R} \) be smooth functions. An extended derivative \( \mathbf{E} \), denoted \((\alpha, a, h)\), is an operator that maps a vector field \( \mathbf{u} : \mathbb{R}^M \rightarrow \mathbb{R}^N \) to a function \( \mathbf{E}[\mathbf{u}] : \mathbb{R}^M \rightarrow \mathbb{R} \) defined as:

\[
\mathbf{E}[\mathbf{u}](x) = a(x) \partial^\alpha[h(x, \mathbf{u}(x))]
\]

(5)

This theorem motivates the variational loss function as we expect the left-hand side of equation 8 to be closer to 0 the closer the candidate PDE is to the true one. To calculate how well a set of vector fields \( \mathcal{D} = \{ \mathbf{u}^{(d)} \}_{d=1}^D \) matches a VR-PDE \((\mathbf{E}[\mathbf{p}], g)\) we propose the following loss function.

\[
\mathcal{L}(\mathbf{E}[\mathbf{p}], g) = \sum_{d=1}^D \sum_{s=1}^S \left( \sum_{p=1}^P \mathcal{F}(\mathbf{E}_p, \mathbf{u}^{(d)}, \phi_s) - \int_\Omega g(x, \mathbf{u}^{(d)}(x)) \phi_s(x) dx \right)^2
\]

(9)

where \( \{ \phi_s \}_{s=1}^S \) is a set of predefined testing functions.

This novel loss function makes it possible to evaluate to what extent any VR-PDE matches the observed data. This loss can be used as an optimization objective in any algorithm that searches over some subspace of closed-form VR-PDEs. We propose D-CIPHER in section 5 as an example of such an algorithm.
### 4 Related Works

#### 4.1 Symbolic Regression

The goal of symbolic regression is to find a closed-form expression that best models the given dataset both in terms of accuracy and simplicity. In contrast with the conventional regression analysis which optimizes the parameters of a pre-specified model, symbolic regression aims at discovering both the general structure and the parameters of the model.

As the space of possible expressions is vast, most of the existing work focuses on developing optimization algorithms. Genetic Programming [17] has been widely used for that task [36]. A different strategy has been employed in AI Feynman [41, 42] that uses neural networks to reduce the search space by identifying simplifying properties like symmetry or separability. Optimization methods based on pre-trained neural networks [4], reinforcement learning [28], and Meijer-G functions [1] have also been proposed.

#### 4.2 Data-driven discovery of closed-form differential equations

Data-driven discovery of physical laws is an established area of machine learning [5, 36]. An important development in discovery of differential equations was to constrain the space of equations to a linear combination of predefined functions and use an algorithm that enforces sparsity to choose only the most important terms. The pioneering work in that area was SINDy [8] that was able to discover explicit ODEs of the form \( \dot{x}_i = \sum_j f_j(x) \). It was later extended to include implicit ODEs [22, 15] and PDEs [34, 35]. Various other extensions were proposed [21, 33]. Other developments include approaches based on Gaussian process regression [32, 31], or convolutional neural networks [19, 18]. Recently, the variational approach has been used as a viable alternative to derivative estimation [23, 24, 30].

### 5 D-CIPHER

#### 5.1 What problem are we solving?

We want to discover a system of PDEs that governs the functions we observe. However, in reality, we have access only to a finite number of noisy measurements. To make this distinction clear we differentiate between the **true fields** and the **observed fields**.

**Definition 5 (True Field).** We define a true field on \( \Omega \) as a vector valued function \( u : \Omega \to \mathbb{R}^N \) where each \( u_j : \Omega \to \mathbb{R} \) is a \( C^K \) function.

**Definition 6 (Observed field and sampling grid).** We define a sampling grid \( G \) to be a finite subset of \( \Omega \). Let \( u : \Omega \to \mathbb{R}^n \) be a true field on \( \Omega \).

An **observed field sampled from** \( u \) on a grid \( G \) is a function \( v : G \to \mathbb{R}^N \) of the form

\[
v_j(x) = u_j(x) + \epsilon_j(x) \quad \forall x \in G, \forall j \in [N]
\]

where \( \epsilon_j(x) \) corresponds to **noise**, a realisation of a zero-mean random variable.

The PDE discovery task is then defined as follows.
PDE Discovery. We are given a dataset of observed fields \( D = \{ v^{(d)} \}_{d=1}^{D} \) together with a grid \( G \). We assume that every trajectory \( v^{(d)} \) is sampled from a corresponding true field \( u^{(d)} \) on a grid \( G \). We assume that every true field \( u^{(d)} \) is governed by the same system of closed-form PDEs described by a set of functions \( \{ f_l \}_{l=0}^{L} \). Namely,
\[
 f_l(x, u^{(d)}(x), \partial^K u^{(d)}(x)) = 0 \quad \forall x \in \Omega \forall l \in [L]
\]  
(10)
The PDE discovery task is to infer the system of PDEs, \( \{ f_l \}_{l=0}^{L} \), from the dataset of observed fields \( D = \{ v^{(d)} \}_{d=1}^{D} \) and the sampling grid \( G \).

We assume that the underlying PDE is inside the class of VR-PDEs defined earlier. We also assume that its \( \partial \)-bound part is inside a subspace of extended differential operators spanned by a user-specified dictionary (see Step 1 in section 5.2).

5.2 D-CIPHER algorithm

We propose an algorithm that consists of three steps. In the first step, we define the subspace of closed-form VR-PDEs we want to search over to reflect our knowledge of the problem. In the second step, we reconstruct the fields from noisy measurements. In the last step, we solve an optimization problem using a modified symbolic regression algorithm. For more details, check Appendix C.

Step 1: Choose the form and incorporate prior knowledge. The space of closed-form extended differential operators is vast and searching through the whole space seems to be computationally infeasible. That is why a human expert should encode their prior knowledge of the system into a dictionary of non-degenerate extended derivatives \( Q = \{ \hat{E}_p \}_{p \in [P]} \). We use this dictionary to search over a finite-dimensional subspace of closed-form operators spanned by this set. In other words, we assume that the VR-PDE is of the form:
\[
 \sum_{p=1}^{P} \beta_p \hat{E}_p[u](x) - g(x, u(x)) = 0 \quad \forall x \in \Omega
\]  
(11)
where \( \beta \in \mathbb{R}^P \), \( g \) is any closed-form function of \( M + N \) variables, and \( \hat{E}_p = (\alpha_p, a_p, h_p) \).

For instance a default dictionary might include only the partial derivatives up to a certain order. For a 1+1 second-order system that means \( Q = \{ \partial_t, \partial_x, \partial_t \partial_x, \partial_t^2, \partial_x^2 \} \). That is already enough to discover heat and wave equations with any closed-form source. If, for instance, the user suspects the presence of the advection term \( uu_x \) (as in the Burger’s equation), the term \( \partial_x(u^2) \) can be included in the library.

It’s important to note that we do not assume any particular form of \( g \) apart from being closed-form.

Step 2: Estimate the fields. The dataset consists of noisy and infrequently sampled fields whereas the loss function is defined for the true fields. Thus we need to estimate the true fields \( \hat{u}^{(d)} \) from the observed fields \( v^{(d)} \). Any choice of reconstruction algorithm can be used and the user should choose it according to the problem setting and their domain knowledge.

Step 3: Optimize. We want to minimize the loss function in equation 9 for the reconstructed fields \( \{ \hat{u}^{(d)} \}_{d=1}^{D} \) among all PDEs of the form in equation 11. Thus we want to solve the following optimization problem:
\[
 \min_{g} \min_{||\beta||_1 = 1} \sum_{d=1}^{D} \sum_{s=1}^{S} \left( \sum_{p=1}^{P} F(\beta_p \hat{E}_p, \hat{u}^{(d)}, \phi_s) - \int_{\Omega} g(x, \hat{u}^{(d)}(x)) \phi_s(x) dx \right)^2
\]  
(12)
We want to highlight that this problem is substantially different from the ones encountered in previous techniques because we want to discover both \( g \) and \( \beta \). Thus the standard penalty on \( \beta \) such as the \( \lambda ||\beta||_2 \) or \( \lambda ||\beta||_1 \) would not work, as the loss would be minimized by \( g = 0 \) and \( \beta = 0 \). Therefore we put the constraint \( ||\beta||_1 = 1 \). We choose the L1 norm to encourage sparsity in the coordinates of the vector \( \beta \).

The inner minimization in equation 12 can be rewritten as a constrained least-squares problem.
\[
 \min_{||\beta||_1 = 1} \sum_{(d,s) \in [D] \times [S]} \left( \beta \cdot z^{(d,s)} - w^{(d,s)} \right)^2
\]  
(13)
where \( \hat{\mathcal{E}}_p = (\alpha_p, a_p, h_p) \) and \( z^{(d,s)} \in \mathbb{R}^p, w^{(d,s)} \in \mathbb{R} \) are defined as
\[
\begin{align*}
\hat{z}^{(d,s)}_p &= \int_\Omega h_p(x, \hat{u}^{(d)}(x)) (-1)^{|\alpha_s|} a_p(x) \phi_s(x) dx \\
\hat{w}^{(d,s)} &= \int_\Omega g(x, \hat{u}^{(d)}(x)) \phi_s(x) dx
\end{align*}
\]

We show the full derivation in Appendix C. \( z^{(d,s)} \) can be precomputed at the beginning of the algorithm without estimating the derivatives of the reconstructed fields. They can be easily calculated if the derivatives of the testing functions \( \phi_s \) and the derivatives of \( a_p \) can be easily and accurately computed.

As the optimization problem in equation 13 has to be solved many times for different closed-form expressions \( g \), it poses some unique challenges. As standard approaches are not sufficiently fast, we design a new heuristic algorithm to solve this problem. We describe it in the next section.

6 CoLLie

The problem from the previous section can be formulated as follows. Given matrix \( A \in \mathbb{R}^{m \times n} \) and vector \( b \in \mathbb{R}^n \), find a vector \( z \in \mathbb{R}^n \) that minimizes \( \|Az - b\|_2^2 \) such that \( \|z\|_1 = 1 \). Additionally, it has to be fast for any \( b \) but any time spent on pre-processing \( A \) is irrelevant (as in our problem \( A \) is fixed). The task is challenging as the the unit L1 sphere is not convex. A method that guarantees an optimal solution is based on an observation that the \( (n-1) \)-dimensional L1 sphere \( \{ z \in \mathbb{R}^n \mid \|z\|_1 = 1 \} \) consists of \( 2^n \) \( (n-1) \)-simplices, where a standard \( (n-1) \)-simplex is a convex set defined as: \( \{ z \in \mathbb{R}^n \mid \sum_{i=1}^n z_i = 1 \wedge z_i \geq 0 \forall i \in [n] \} \). Minimizing \( \|Az - b\|_2^2 \) on a simplex is a quadratic program \[6\] and there are many solvers available for this kind of problem \[2, 38, 3\]. However, that means that the computation time scales exponentially with the number of dimensions. This is prohibitively long for the inner optimization of our algorithm. Searching through the combinatorial space of closed-form expression is already challenging and it is of utmost importance to make sure that each evaluation is performed as quickly as possible. Therefore we design a heuristic algorithm CoLLie (Constrained L1 norm Least squares) that finds an approximate solution but is significantly faster (Figure 1). We provide a detailed description of CoLLie in Appendix D.

![Figure 1](image)

Figure 1: We compare CoLLie with an algorithm that uses CVXOPT \[2\] to solve each of the convex subproblems. We report the relative error between the loss obtained by CoLLie and the minimum loss achieved by CVXOPT. Panels B and C show the averages and the distributions of relative errors. The average relative error is below 0.005 and the bulk of the distribution is below \( 10^{-7} \). At the same time CoLLie is orders of magnitude faster (Panel A). The shaded region shows the standard deviation.

7 Experiments

We perform a series of experiments to show how well D-CIPHER is able to discover some well-known differential equations. Additional information about the experiments can be found in Appendix E.

Choice of dynamical systems. D-CIPHER was developed to discover equations that are beyond the capabilities of current methods. Thus for experiments, we choose the dynamical systems such that the \( \partial \)-free part of the target PDE is a closed-form function that cannot be expressed as a linear combination of functions from a finite dictionary if the parameters are not known a priori.

Evaluation metrics. To establish how well a discovered PDE matches the ground truth, we evaluate its \( \partial \)-free and \( \partial \)-bound parts separately. For the \( \partial \)-free part, we assign a binary variable indicating
without the source term, most of the current methods cannot be applied directly to discover this

Table 2: Simulation results for the inhomogeneous heat equation. We report the success probability of discovering the $\partial$-free part and the Average RMSE of the $\partial$-bound part. Standard deviations are shown in brackets.

| Method          | $\sigma_R = 0.05$ | 0.1 | 0.2 | $\sigma_R = 0.05$ | 0.1 | 0.2 |
|-----------------|-------------------|-----|-----|-------------------|-----|-----|
| D-CIPHER        | 0.64 (.07)        | 0.42 (.07) | 0.12 (.05) | 0.15 (.009) | 0.21 (.007) | 0.24 (.005) |
| Abl. D-CIPHER   | 0.46 (.07)        | 0.20 (.06) | 0.04 (.03) | 0.18 (.009) | 0.24 (.008) | 0.27 (.007) |

whether the correct functional form of the equation was recovered (please check Appendix for details). For the $\partial$-bound part, we measure the RMSE between the found coefficients of $\beta$ and the target ones. We report the averages and standard deviations for both parts. We call the averages respectively **Success Probability** and **Average RMSE**.

**Comparison.** We show results for D-CIPHER together with an ablated version of D-CIPHER where the derivatives are estimated and the standard MSE loss is used instead of the variational loss. We do not compare with any other methods as they would all fail to discover these equations.

**Implementation.** We use B-Splines as the testing functions and we estimate the fields in Step 2 of D-CIPHER with a Gaussian Process. The outer optimization in Step 3 is performed using a modified genetic programming algorithm and the inner optimization by CoLLie (Section 6).

**Forced and damped harmonic oscillator** is described by $\partial_t^2 u(t) + 2\theta_2 \omega_3 \sigma^2 u(t) + \omega_4 u(t) = g(t)$ for $t \in [0,T]$. We choose the driving force $g(t) = \sigma_3 \sin(\theta_4 t) + \sigma_5 \cos(\theta_6 t)$. We note that as it is a second order ODE, it cannot be recovered by D-CODE. We set the constants as $\theta_1 = 0.5, \theta_2 = 4.0, \theta_3 = 5.0, \theta_4 = 3.0$ and present the simulation results in Figure 2. D-CIPHER can discover the correct functional form of the $\partial$-free part and achieves a low RMSE for the coefficients of $\beta$ in most of the experimental settings. Moreover the performance is higher than or comparable to the ablated version of D-CIPHER, thus demonstrating gain from utilizing the variational approach.

**Inhomogeneous heat equation** is described by $\partial_t u(t, x) - \theta_1 \partial_x^2 u(t, x) = g(t, x)$. We choose the source to be $g(t, x) = \theta_3 e^{\theta_4 x}$. We show results for parameters $\theta_1 = 0.25, \theta_2 = 1.25, \theta_3 = 1.8$ in Table 2. D-CIPHER is able to discover the correct equation even in settings with a very high noise. It performs better than the ablated version, thus showing the importance of the variational objective.

**Inhomogeneous wave equation** is described by $\partial_t^2 u(t, x) - \theta_1 \partial_x^2 u(t, x) = g(t, x)$. This equation does not have the standard evolution form, as it does not involve the $\partial_t$ term. Thus, even without the source term, most of the current methods cannot be applied directly to discover this equation. We choose the source term to be $g(t, x) = \theta_3 e^{\theta_4 t} \sin(\theta_5 t)$ and set the parameters as $\theta_1 = 1.0, \theta_2 = 2.0, \theta_3 = 3.0$. In Figure 3, we show the absolute difference between the true field and the fields computed from the sources discovered by D-CIPHER and its ablated version across different measurement settings. D-CIPHER finds the correct functional form with coefficients not far from the ground truth. The ablated version fails to discover the correct functional form and the found $\partial$-free part does not reproduce the correct behavior of the system.

Figure 2: Success probability of discovering the correct $\partial$-free part of the equation and the average RMSE between the recovered $\partial$-bound part and the target one across different experimental settings.
8 Discussion

Applications. As D-CIPHER can potentially discover any closed-form $\partial$-free part, it is especially useful when this part of the PDE captures an essential component of the system. We demonstrate it by finding the heat and vibration sources as well as the driving force of an oscillator. Beyond the spatio-temporal physical systems, D-CIPHER might prove useful in discovering population models structured by age, size, and spatial position [44, 43], age-dependent epidemiological models [14], and predator-prey models with age-structure [29]. All these systems are VR-PDEs where the $\partial$-free parts are crucial elements of the equations signifying the rates of mortality, infection, recovery, or growth.

Limitations and open challenges. D-CIPHER may fail in some scenarios, either due to challenging experimental settings or a challenging underlying PDE. Challenging experimental settings might include unobserved variables, high measurement noise, infrequent sampling, and inadequate domain (e.g., small time horizon). Challenging PDE form might include a PDE outside of the VR-PDE class or a $\partial$-free part that contains many variables and thus is difficult to find. We note that we address some of these challenges by utilizing a variational approach, defining VR-PDEs to be a very general class of equations, and designing CoLLie enabling a thorough search across closed-form expressions.

Ethics statement. We want to emphasize that D-CIPHER was designed to facilitate the process of scientific discovery by extracting closed-form PDEs from data. It is not intended to or capable of replacing human experts in the modeling process. No human-derived data was used.
Acknowledgments and Disclosure of Funding

This work is supported by the US Office of Naval Research (ONR), the National Science Foundation (NSF Grant number:1722516), and Roche. We want to thank Samuel Holt and Jonathan Crabbé for their useful comments and feedback on earlier versions of this work. We are grateful to Professor Yuanzhang Xiao for insightful discussions about the optimization algorithms.

References

[1] Ahmed M Alaa and Mihaela van der Schaar. Demystifying black-box models with symbolic metamodels. Advances in Neural Information Processing Systems, 32:11304–11314, 2019.
[2] MS Andersen, J Dahl, and L Vandenberghe. CVXOPT: A Python package for convex optimization, Version 1.1. 6, 2013, 2013.
[3] MOSEK ApS. The MOSEK optimization toolbox for MATLAB manual. Version 9.0., 2019.
[4] L. Biggio, T. Bendinelli*, A. Neitz, A. Lucchi, and G. Parascandolo. Neural Symbolic Regression that Scales. In 38th International Conference on Machine Learning, July 2021.
[5] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 104(24):9943–9948, June 2007.
[6] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.
[7] Richard P Brent. Algorithms for minimization without derivatives. Courier Corporation, 2013.
[8] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 113(15):3932–3937, April 2016.
[9] Carl De Boor and Carl De Boor. A practical guide to splines, volume 27. springer-verlag New York, 1978.
[10] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least Angle Regression. The Annals of Statistics, 32(2):93, 2004.
[11] Lev D Elsgolc. Calculus of variations. Courier Corporation, 2012.
[12] F. G. (Friedrich Gerard) Friedlander. Introduction to the theory of distributions / F.G. Friedlander: Cambridge University Press, Cambridge, 1982.
[13] Herbert Goldstein. Classical mechanics / Herbert Goldstein, John Safko, Charles Poole. Pearson, pearson new international edition.; third edition. edition, 2014. Publication Title: Classical mechanics.
[14] Frank Hoppensteadt. An Age Dependent Epidemic Model. Journal of the Franklin Institute, 297(5):325–333, May 1974.
[15] Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. SINDy-PF: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 476(2242):20200279, October 2020. Publisher: Royal Society.
[16] Doyo Kereyu and Genanew Gofe. Convergence Rates of Finite Difference Schemes for the Diffusion Equation with Neumann Boundary Conditions. American Journal of Computational and Applied Mathematics, page 11, 2016.
[17] John R. Koza. Genetic programming: on the programming of computers by means of natural selection. Complex adaptive systems. MIT Press, Cambridge, Mass, 1992.
[18] 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, December 2019.
[19] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from Data. In Proceedings of the 35th International Conference on Machine Learning, pages 3208–3216. PMLR, July 2018. ISSN: 2640-3498.
[20] Lynn H. Loomis. *Advanced calculus / Lynn H. Loomis and Shlomo Sternberg*. Addison-Wesley series in mathematics. Addison-Wesley Pub. Co., Reading, Mass.; London, 1968. Publication Title: Advanced calculus.

[21] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2204):20170009, August 2017.

[22] Niall M. Mangan, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):52–63, June 2016. Conference Name: IEEE Transactions on Molecular, Biological and Multi-Scale Communications.

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

[24] Daniel A. Messenger and David M. Bortz. Weak SINDy: Galerkin-Based Data-Driven Model Selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, January 2021.

[25] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, and others. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017. Publisher: PeerJ Inc.

[26] A. R. Mitchell and D. F. Griffiths. *The finite difference method in partial differential equations*. Wiley, Chichester [Eng.]; New York, 1980.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Brenden K Petersen, Mikel Landajuela Larra, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep Symbolic Regression: Recovering Mathematical Expressions From Data via Risk-seeking Policy Gradients. *ICLR 2021*, 2021.

[29] J. Promrak, G. C. Wake, and C. Rattanakul. Predator-prey Model with Age Structure. *The ANZIAM Journal*, 59(2):155–166, October 2017. Publisher: Cambridge University Press.

[30] Zhaozhi Qian, Krzysztof Kacprzyk, and Mihaela van der Schaar. D-CODE: Discovering Closed-form ODEs from Observed Trajectories. *The Tenth International Conference on Learning Representations*, 2022.

[31] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, March 2018.

[32] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198, January 2018.

[33] Chengping Rao, Pu Ren, Yang Liu, and Hao Sun. Discovering Nonlinear PDEs from Scarce Data with Physics-Encoded Learning. *ICLR 2022*, 2022.

[34] 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, April 2017.

[35] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, January 2017.

[36] Michael Schmidt and Hod Lipson. Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923):81–85, April 2009.

[37] Michael Spivak. *Calculus On Manifolds: a Modern Approach To Classical Theorems Of Advanced Calculus*. CRC Press, 2018. OCLC: 1029237047.

[38] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[39] Trevor Stephens. gplearn: Genetic programming in python, with a scikit-learn inspired and compatible api, 2022.
[40] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. Publisher: [Royal Statistical Society, Wiley].

[41] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[42] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, April 2020.

[43] G. F. Webb. Population Models Structured by Age, Size, and Spatial Position. In J. M. Morel, F. Takens, B. Teissier, Pierre Magal, and Shigui Ruan, editors, *Structured Population Models in Biology and Epidemiology*, volume 1936, pages 1–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. Series Title: Lecture Notes in Mathematics.

[44] Glenn F. Webb. *Theory of Nonlinear Age-Dependent Population Dynamics*. CRC Press, January 1985.

[45] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.
Table of supplementary materials

1. Appendix [A] notation and definitions
2. Appendix [B] variational formulation for linear PDEs and the proof of Theorem [I]
3. Appendix [C] details of the D-CIPHER framework, including pseudocode
4. Appendix [D] details of the CoLLie algorithm
5. Appendix [E] details of experiments and the implementation

A Notation and definitions

A.1 Notation

Table 3: Symbols used in this work

| Symbol | Meaning |
|--------|---------|
| $[n]$  | a set of numbers $\{1, \ldots, n\}$ |
| $\mathbb{N}$ | a set of natural numbers, i.e., $\{1, 2, 3, \ldots\}$ |
| $\mathbb{N}_0$ | a set of non-negative integers, i.e., $\{0, 1, 2, 3, \ldots\}$ |
| $M$ | the dimension of the domain of a vector field |
| $N$ | the dimension of the codomain of a vector field |
| $K$ | denotes the smoothness of functions or the maximum order of derivatives |
| $D$ | the size of the dataset of observed fields |
| $S$ | the number of testing functions |
| $\Omega$ | an open set in $\mathbb{R}^M$ |
| $\dot{u}(t)$ | the derivative of $u$ at $t$ |
| $\alpha$ | a multi-index, an element of $\mathbb{N}_0^M$ |
| $|\alpha|$ | the order of $\alpha$, $|\alpha| = \sum_i \alpha_i$ |
| $\partial_i^{\alpha_i}$ | $i$th-order partial derivative with respect to the $i$th variable |
| $\partial^{\alpha}$ | $\partial_1^{\alpha_1} \partial_2^{\alpha_2} \ldots \partial_M^{\alpha_M}$ |
| $\mathcal{C}^K$ | a set of functions with continuous partial derivatives $\partial^{\alpha}$ for all $|\alpha| \leq K$ |
| $\mathcal{E}$ | an extended derivative, Definition [I] |
| $\mathcal{E}_{[P]}$ | an extended differential operator, Definition [I] |
| $\mathcal{F}$ | the functional used in the variational loss, Definition [I] |
| $\mathcal{L}(\mathcal{E}_{[P]}, g)$ | the variational loss, equation [9] |
| $\mathcal{G}$ | a sampling grid, Definition [6] |
| $u$ | a true field, Definition [5] |
| $v$ | an observed field, Definition [6] |
| $D$ | a dataset of observed trajectories |
| $\epsilon$ | the noise |
| $Q$ | a dictionary of non-degenerate extended derivatives |
| $\beta$ | a vector describing the $\partial$-bound part of the VR-PDE |
| $\sigma_R$ | a noise ratio |
| $\Delta t$ | a sampling interval |

A.2 Definitions

**Definition 7** (Closed-form expressions and functions). A *closed-form* expression is a mathematical expression that consists of finite number of variables, constants, arithmetic operations and certain well-known function (e.g., logarithm, trigonometric functions). A function $f$ is called *closed-form* if it can be represented by a closed-form expression. E.g., $f(x, y) = x^2 \log(y) + \sin(3z)$.

**Remark.** In practice, we do not want to consider any finite expression. Any symbolic regression algorithm penalizes expressions that are too long putting a soft constraint on the number of elements used. That is why deep neural networks are not considered closed-form even if they satisfy the conditions in Definition [I].
\textbf{Definition 8} (Multi-index). An \textit{n-dimensional multi-index} $\alpha$ is an \textit{n}-tuple
\[ \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \]
where $\forall i \in [n]$ $\alpha_i \in \mathbb{N}_0$. Thus $\alpha \in \mathbb{N}_0^n$. We define the order of $\alpha$ as $|\alpha| = \sum_{i=1}^n \alpha_i$.

\textbf{Definition 9}. For any \textit{n-dimensional multi-index} $\alpha$ we define a mixed derivative
\[ \partial^\alpha = \partial_1^{\alpha_1} \partial_2^{\alpha_2} \cdots \partial_n^{\alpha_n} \]
where $\partial^{\alpha_i} = \partial^{\alpha_i} / \partial x_i$ is a $\alpha_i$th-order partial derivative with respect to $x_i$ (the $i$th independent variable). We call $\partial^\alpha$ non-trivial if $|\alpha| > 0$. We denote the list of all non-trivial partial derivatives of $u$ up to order $K$ as $\partial^{(K)} u$.

\textbf{Definition 10} (Closed-form Partial Differential Equation). Let $f$ be a closed-form real smooth function. We say that a vector field $u : \Omega \rightarrow \mathbb{R}^N$ is governed by a $K$th-order closed-form PDE described by $f$ if
\[ f(x, u(x), \partial^{[K]} u(x)) = 0 \quad \forall x \in \Omega \quad (15) \]
where $\partial^{[K]} u$ are all non-trivial mixed derivatives of all $u_j$ ($j \in [N]$) up to the $K$th order.

\section{B Variational-Ready PDEs}

\subsection{B.1 Variational Formulation of PDEs}

In this section, we provide the standard variational formulation of PDEs for linear PDEs [12].

\textbf{Definition 11} (Linear differential operator). Let $\mathcal{A}$ be a finite set of multi-indices. A linear differential operator $L$ is defined as
\[ L = \sum_{\alpha \in \mathcal{A}} a_{\alpha} \partial^\alpha \]
where $a_{\alpha} \in C^K$ is a non-zero sufficiently smooth function of dependent variables. If $\max_{\alpha \in \mathcal{A}} |\alpha| = n$ then we call $L$ an $n$th-order linear differential operator. If all $a_{\alpha}$ are constant we say that $L$ has constant coefficients.

The \textit{adjoint} of $L$, denoted $L^\dagger$, is a linear differential operator defined as
\[ L^\dagger u(x) = \sum_{\alpha \in \mathcal{A}} (-1)^{|\alpha|} \partial^\alpha (a_{\alpha}(x)u(x)) \quad (16) \]

\textbf{Proposition 1} (Variational Formulation of PDEs for linear PDEs). Let $K \in \mathbb{N}$. Consider a scalar field $u : \Omega \rightarrow \mathbb{R}$, such that $u \in C^K$, a $K$th-order linear differential operator $L$, and a continuous function $g : \Omega \rightarrow \mathbb{R}$. Let $\phi : \Omega \rightarrow \mathbb{R}$ be a testing function. Then $u$ satisfies a linear PDE
\[ L[u(x)] - g(x) = 0 \quad \forall x \in \Omega \quad (17) \]
if and only if
\[ \int_{\Omega} \left[u(x)L^\dagger \phi(x) - g(x)\phi(x)\right] \, dx = 0 \quad (18) \]
for all testing functions $\phi : \Omega \rightarrow \mathbb{R}$.

Note that the integrals are always well-defined as $\phi$ has a compact support.

\subsection{B.2 Theorem}

Before we prove the Theorem\textsuperscript{1} we need the following lemma, which is a particular formulation of the Fundamental lemma of calculus of variations [11]. We also need a generalized version of the divergence theorem [20].

\textbf{Lemma} (Fundamental lemma of calculus of variations). Let $K \in \mathbb{N}$, $\Omega$ be an open set in $\mathbb{R}^M$, and $u : \Omega \rightarrow \mathbb{R}$ be a continuous function. Then $u$ is equal to 0 on the whole $\Omega$ if and only if
\[ \int_{\Omega} u(x) \phi(x) \, dx = 0 \quad \text{for all } C^K \text{ functions } \phi : \Omega \rightarrow \mathbb{R} \text{ with compact support}. \]
We now prove the converse. We can now prove Theorem 1.

\[ \int \]

\[ \text{Theorem 2 (Divergence theorem). Let } \Omega \text{ be an open set in } \mathbb{R}^M \text{ and let } f, g \text{ be continuous on } \Omega = \Omega \cup \partial \Omega \text{ and continuously differentiable on } \Omega. \text{ Then} \]

\[ \int_{\Omega} \partial_i [f(x)] g(x) dx = -\int_{\Omega} f(x) \partial_i [g(x)] dx + \int_{\partial \Omega} \nu_i f(x) g(x) dx \]

where \( \nu \) is a normal unit vector to the boundary \( \partial \Omega \).

In a 1-dimensional setting, the statement of the theorem reduces to the integration by parts.

We can now prove Theorem 2.

Proof. Let us denote \( E_p = (\alpha_p, a_p, h_p) \). Then the PDE in equation \[ 7 \] can be written as:

\[ \sum_{p=1}^{P} a_p(x) \partial^{\alpha_p} [h_p(x, u(x))] - g(x, u(x)) = 0 \quad \forall x \in \Omega \]  

(22)

The LHS is continuous as all \( a_p \), and \( h_p \) are smooth, \( g \) is continuous, \( u \in C^K \), and \( |\alpha_p| \leq K \forall p \in |P| \). Thus we can use the fundamental lemma of calculus of variations to say that the equation \[ 22 \] is true if and only if

\[ \int_{\Omega} \left[ \sum_{p=1}^{P} a_p(x) \partial^{\alpha_p} [h_p(x, u(x))] - g(x, u(x)) \right] \phi(x) dx = 0 \]  

(23)

for all testing functions \( \phi \). We transform the LHS of equation \[ 23 \] using linearity to:

\[ \sum_{p=1}^{P} \int_{\Omega} a_p(x) \partial^{\alpha_p} [h_p(x, u(x))] \phi(x) - \int_{\Omega} g(x, u(x)) \phi(x) dx \]  

(24)

Let us now focus on

\[ \int_{\Omega} a_p(x) \partial^{\alpha_p} [h_p(x, u(x))] \phi(x) \]  

(25)

and let us denote \( \alpha_p = (\alpha_{p1}, \ldots, \alpha_{pM}) \). Then \( \partial^{\alpha_p} = \partial_1^{\alpha_{p1}} \ldots \partial_M^{\alpha_{pM}} \) and the expression can be written as

\[ \int_{\Omega} \partial_1^{\alpha_{p1}} \ldots \partial_M^{\alpha_{pM}} [h_p(x, u(x))] a_p(x) \phi(x) dx \]  

(26)
Let us denote the support of $\phi$ as $B$. As $\phi$ is equal to zero outside of its support, we can write the expression as
\[
\int_B \partial_1^{\alpha_{p_1}} \ldots \partial_M^{\alpha_{p_M}} \left[ h_p(x, u(x)) \right] a_p(x) \phi(x) \, dx
\]  
(27)
Without loss of generality, let us assume that $\alpha_{p_1} > 0$. By the divergence theorem, this can be rewritten as
\[
- \int_B \partial_1^{\alpha_{p_1}-1} \ldots \partial_M^{\alpha_{p_M}} \left[ h_p(x, u(x)) \right] \partial_1^{\alpha_{p_1}} [a_p(x) \phi(x)] \, dx
\]  
(28)
because the integral over the boundary is equal to 0
\[
\int_{\partial B} \nu_1 \partial_1^{\alpha_{p_1}-1} \ldots \partial_M^{\alpha_{p_M}} \left[ h_p(x, u(x)) \right] a_p(x) \phi(x) \, dx = 0
\]  
(29)
as $\phi$ has a compact support (and thus vanishes on the boundary). We can perform this operation $\alpha_{p_1}$ times to shift the whole derivative $\partial_1^{\alpha_{p_1}}$ to the second part of the equation and obtain
\[
(-1)^{\alpha_{p_1}} \int_B \partial_2^{\alpha_{p_2}} \ldots \partial_M^{\alpha_{p_M}} \left[ h_p(x, u(x)) \right] \partial_1^{\alpha_{p_1}} [a_p(x) \phi(x)] \, dx
\]  
(30)
Then we repeat this for other derivatives and we end up with the following expression:
\[
(-1)^{\alpha_{p_1}} \cdots (-1)^{\alpha_{p_M}} \int_B h_p(x, u(x)) \partial_1^{\alpha_{p_1}} \cdots \partial_M^{\alpha_{p_M}} [a_p(x) \phi(x)] \, dx
\]  
(31)
As the integrand is zero outside of $B$, this can be rewritten as:
\[
(-1)^{|\alpha|} \int_{\Omega} h_p(x, u(x)) \partial^{\alpha} [a_p(x) \phi(x)] \, dx
\]  
(32)
or more compactly, using the functional defined in Definition 4 as:
\[
\mathcal{F}(E_p, u, \phi)
\]  
(33)
Therefore equation 24 can be written as:
\[
\sum_{p=1}^{P} \mathcal{F}(E_p, u, \phi) - \int_{\Omega} [g(x, u(x)) \phi(x)] \, dx
\]  
(34)
Thus, we proved that equation 22 is true if and only if
\[
\sum_{p=1}^{P} \mathcal{F}(E_p, u, \phi) - \int_{\Omega} [g(x, u(x)) \phi(x)] \, dx = 0
\]  
(35)
for all testing functions $\phi$.  

B.3 Examples
The examples of VR-PDEs can be found in Table 4.

C D-CIPHER

C.1 Rewrite the inner optimization as a constrained least squares
Let us rewrite the objective in equation 12
\[
\sum_{d=1}^{D} \sum_{s=1}^{S} \left( \sum_{p=1}^{P} \mathcal{F}(\beta_p \hat{E}_p, \hat{u}^{(d)}, \phi_s) - \int_{\Omega} g(x, \hat{u}^{(d)}(x)) \phi_s(x) \, dx \right)^2
\]  
(36)
First let us observe that
\[
\mathcal{F}(\beta_p \hat{E}_p, \hat{u}^{(d)}, \phi_s) = \int_{\Omega} h_p(x, \hat{u}^{(d)}(x)) (-1)^{|\alpha|} \partial^{\alpha} [\beta_p a_p(x) \phi_s(x)] \, dx = \beta_p z_p^{(d, s)}
\]  
(37)
Table 4: Examples of equations which are Variational-Ready

| Name                        | Equation                                                                 | Linear | VR |
|-----------------------------|--------------------------------------------------------------------------|--------|----|
| Damped wave eq. with a source | $u_{tt} + \rho u_t - \kappa \nabla^2 u = g(x)$                          | ✔      | ✔  |
| Gauss law                   | $\nabla \cdot E = \rho/\epsilon_0$                                      | ✔      | ✔  |
| Burger’s equation           | $u_t + uu_x - \nu u_{xx} = 0$                                            |        | ✔  |
| Navier-Stokes equations     | $u_t + (u \cdot \nabla) u - \nu \nabla^2 u = -1/\rho \nabla p + g$       |        | ✔  |
| Korteweg-De Vries equation | $u_t + u_{xxx} - 6uu_x = 0$                                             |        | ✔  |
| Kuramoto-Sivashinsky equation | $u_t - \nabla^2 (u^c) = 0$                                             |        | ✔  |
| Fisher’s equation           | $u_t - \nabla^2 (u^c) = 0$                                             |        | ✔  |
| Liouville’s equation        | $u_x + u_{yy} = \kappa e \rho u$                                       |        | ✔  |
| Porous medium equation      | $u_t - u_{xx} = -\sin(u)$                                              |        | ✔  |

if we let $z^{(d,s)}_p \in \mathbb{R}$ be defined as

$$z^{(d,s)}_p = \int_\Omega h_p(x, \hat{\theta}^{(d)}(x))(-1)^{|\alpha_p|} \partial^{\alpha_p} (a_p(x)\phi_s(x)) \, dx$$  \hspace{1cm} (38)

Moreover, if we define $w^{(d,s)} \in \mathbb{R}$ as

$$w^{(d,s)} = \int_\Omega g(x, \hat{\theta}^{(d)}(x))\phi_s(x) \, dx$$  \hspace{1cm} (39)

we can rewrite expression (36) as

$$\sum_{d=1}^{D} \sum_{s=1}^{S} \left( \sum_{p=1}^{P} \beta_p z^{(d,s)}_p - w^{(d,s)} \right)^2$$  \hspace{1cm} (40)

Now, the sum over $p$ can be written as a dot product between $z^{(d,s)} \in \mathbb{R}^P$ and $\beta \in \mathbb{R}^P$. We can also combine the sums over $d$ and $s$. We obtain

$$\sum_{(d,s) \in \{D\} \times \{S\}} \left( \beta \cdot z^{(d,s)} - w^{(d,s)} \right)^2$$  \hspace{1cm} (41)

which is exactly the same as the objective in equation 13.

C.2 Pseudocode

The pseudocode of D-CIPHER is presented in Algorithm 1.

C.3 Testing functions

Testing functions, by definition, need to be sufficiently smooth functions (in $C^K$ class) with compact support. Moreover, the result proved in [30] suggests that these functions should be a subset of a Hilbert basis in L2 space. In particular, that means they should be orthonormal.

We use B-Splines [9] as the testing functions in our experiments because we can control their smoothness and the derivatives are easy to compute. We scale and shift them appropriately so that they are orthonormal.

D CoLLie

D.1 Lagrangian

The problem that CoLLie is supposed to solve is a constrained least-squares optimization defined as:

$$\text{minimize } ||Az - b||_2^2$$
$$\text{subject to } ||z||_1 - 1 = 0$$  \hspace{1cm} (42)
Algorithm 1 D-CIPHER

Input: Observed fields $D = \{v^{(d)}\}_{d=1}^D$, grid $\mathcal{G}$
Input: Symbolic regression optimization algorithm $\mathcal{O}$
Input: Smoothing algorithm $\mathcal{S}$
Input: Testing functions $\{\phi_s\}_{s=1}^S$
Output: Target PDE

$\hat{u}^{(d)} = \mathcal{S}(v^{(d)}) \forall d \in [D]$ \hspace{1cm} \triangleright \text{Step 1}

$Z_{p}^{(d,s)} \leftarrow \int_{\Omega} h_{p}(x, \hat{u}^{(d)}(x))(-1)^{|\alpha_p|} \partial^{\alpha_p}(a_{p}(x)\phi_{s}(x))dx$

procedure LOSS($g$)
\begin{itemize}
  \item initialize vector $w \in \mathbb{R}^D \times \mathbb{R}^P$
  \item $w_{(d,s)} \leftarrow \int_{\Omega} g(x, \hat{u}^{(d)}(x))\phi_{s}(x)dx$
  \item $\beta \leftarrow \text{COLLIE}(Z, w)$
  \item $L = ||Z\beta - w||_2^2$ \hspace{1cm} \triangleright \text{Section 6}
\end{itemize}

end procedure

$g = \mathcal{O}(\text{LOSS})$

$w_{(d,s)} \leftarrow \int_{\Omega} g(x, \hat{u}^{(d)}(x))\phi_{s}(x)dx$

$\beta \leftarrow \text{COLLIE}(Z, w)$ \hspace{1cm} \triangleright \text{Section 6}

return $\sum_{p=1}^{D'} \beta_p \hat{e}_p(u)(x) - g(x, u(x)) = 0$

where $A \in \mathbb{R}^{m \times n}$ has a full column rank, $b \in \mathbb{R}^m$, and $z \in \mathbb{R}^n$ for some $m, n \in \mathbb{N}$.

We consider the Lagrangian $L : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$ associated with this problem defined as

$$L(z, \lambda) = ||Az - b||_2^2 + \lambda(||z||_1 - 1) \tag{43}$$

Now let us define $\hat{z} : \mathbb{R} \rightarrow \mathbb{R}^n$ as

$$\hat{z}(\lambda) = \arg \min_{z \in \mathbb{R}^n} L(z, \lambda) = \arg \min_{z \in \mathbb{R}^n} ||Az - b||_2^2 + \lambda||z||_1 \tag{44}$$

The goal of our algorithm is to find $\lambda^* \in \mathbb{R}$ such that $||\hat{z}(\lambda^*)||_1 = 1$. Let us define a function $q : \mathbb{R} \rightarrow \mathbb{R}$ as

$$q(\lambda) = ||\hat{z}(\lambda)||_1 \tag{45}$$

The goal can be phrased as finding $\lambda^* \in \mathbb{R}$ such that $q(\lambda^*) = 1$.

Let us note that $\hat{z}(0)$ is just a solution to the ordinary least squares (OLS) problem with no constraints and its norm is $q(0)$.

D.2 Extending LARS

Case 1. $q(0) \geq 1$.

If we assume that $\hat{z}$ is continuous then $q$ is also continuous. From the continuity and the fact that $\lim_{\lambda \rightarrow +\infty} q(\lambda) = 0$ and $q(0) \geq 1$ we infer that there exists a $\lambda \geq 0$ such that $q(\lambda) = 1$. Moreover, for $\lambda \geq 0$ the problem in equation \[44\] is the same as in LASSO \[40\]. Therefore we just need to perform LASSO for different $\lambda$ and choose the one that gives the solution with L1 norm equal to 1.

To do it in practice we use Least Angle Regression (LARS) \[10\], a popular algorithm used to minimize the LASSO objective. It generates complete solution paths, i.e., a function $c : \mathbb{R}_+ \rightarrow \mathbb{R}^n$ defined as

$$c(\lambda) = \arg \min_{z \in \mathbb{R}^n} ||Az - b||_2^2 + \lambda||z||_1 \tag{46}$$

which is equivalent to $\hat{z}$ for $\lambda \geq 0$. An illustration of LARS solution paths can be seen in Figure 4. Each line corresponds to a function $c_i$ which describes the coefficient for the $i$th covariate. The paths are defined from some $\lambda_0$ where all $c_i(\lambda_0) = 0$ to $\lambda = 0$ where $c(0) = \hat{z}(0)$. In other words, the
solution paths cover the whole range of constraints from the strictest, effectively imposing the L1 norm of $z$ to be 0, up to no constraints, solving the OLS problem.

The solution paths from the LARS algorithm are piecewise linear and the output are the values of the coefficients for points $\lambda_0 > \ldots > \lambda_n = 0$ where the slopes change. We calculate the norm at each of these points, $||c(\lambda_j)||_1$, and find $j \in [n]$ such that $||c(\lambda_j)||_1 < 1 \leq ||c(\lambda_{j+1})||_1$. As each $c_i$ is a linear function on $[\lambda_j, \lambda_{j+1})$ and we know both $c(\lambda_{j-1})$ and $c(\lambda_j)$, we can effectively search for $\lambda \in [\lambda_j, \lambda_{j+1})$ such that $||c(\lambda)||_1 = 1$. The search can be performed by any root-finding algorithm. We use Brent’s method [7].

**Case 2.** $0 < q(0) < 1$.

This is much more difficult as it corresponds to solving the problem in equation 44 for $\lambda < 0$. The solutions given by LARS algorithm are too small. In fact, the solution with the biggest norm is $c(0) = \mathbb{1}(0)$, the OLS solution, with norm exactly $q(0) < 1$.

To address this challenge, we propose the following heuristic. We extend the solution paths generated by LARS beyond $\lambda = 0$ for $\lambda < 0$. We assume that the paths will continue to be piecewise linear and that they will keep the slope they have in the last interval $[\lambda_n = 0, \lambda_{n-1}]$. Let us denote this slope as

$$\Delta c_i = \frac{c_i(0) - c_i(\lambda_{n-1})}{0 - \lambda_{n-1}}$$

(47)

This is graphically represented in Figure 4. Formally, these extended paths, $\bar{c} : \mathbb{R} \to \mathbb{R}$ are defined as:

$$\bar{c}_i(\lambda) = \begin{cases} c_i(\lambda), & \lambda \geq 0 \\ c_i(0) + \lambda \Delta c_i, & \lambda < 0 \end{cases}$$

(48)

Now, we want to find $\lambda < 0$ such that $||\bar{c}(\lambda)||_1 = 1$. To achieve this in practice, we first make the following observations.

For any $\lambda < q$ we say that $\bar{c}_i(\lambda)$ is on the right side if $\bar{c}_i(\lambda) \Delta c_i \leq 0$ and we say that $\bar{c}_i(\lambda)$ is on the wrong side if $\bar{c}_i(\lambda) \Delta c_i > 0$. In other words, being on the wrong side just means that the path is yet to cross the x-axis if we keep decreasing $\lambda$. We can easily find $\lambda'$ such that for all $\lambda < \lambda'$ all $\bar{c}_i(\lambda)$ are on the right side (none of the paths will ever cross the x-axis).

$$\lambda' = \min \left\{ \frac{0 - c_i(0)}{\Delta c_i} \mid i \in [n] \wedge c_i(0) \Delta c_i > 0 \right\}$$

(49)

If $||\bar{c}(\lambda')||_1 \geq 1$ we just need to search the interval $[\lambda', 0]$ for $\lambda$ such that $||\bar{c}(\lambda')||_1 = 1$.

If $||\bar{c}(\lambda')||_1 < 1$ then we need to search $\lambda < \lambda'$. However, by definition, for all $\lambda < \lambda'$, all $c_i(\lambda)$ are on the right side. That means $||\bar{c}(\lambda)||_1$ as a function of $\lambda$ is just a linear function on the interval $(-\infty, \lambda')$. To see that, let us observe that

$$||\bar{c}(\lambda)||_1 = \sum_{i=1}^{n} |\bar{c}_i(\lambda)| = \sum_{i=1}^{n} \text{sign}(\bar{c}_i(\lambda))\bar{c}_i(\lambda)$$

(50)

Additionally, for $\lambda < \lambda'$ all $c_i(\lambda)$ are on the right side, so we have $\text{sign}(\bar{c}_i(\lambda)) = -\text{sign}(\Delta c_i)$. We can rewrite $||\bar{c}(\lambda)||_1$ as:

$$||\bar{c}(\lambda)||_1 = \sum_{i=1}^{n} (-\text{sign}(\Delta c_i) c_i(0) + \lambda \Delta c_i)$$

$$= -\sum_{i=1}^{n} \text{sign}(\Delta c_i) c_i(0) - \left( \sum_{i=1}^{n} \text{sign}(\Delta c_i) \Delta c_i \right) \lambda$$

(51)

$$= -\sum_{i=1}^{n} \text{sign}(\Delta c_i) c_i(0) - \left( \sum_{i=1}^{n} |\Delta c_i| \right) \lambda$$

Therefore the solution can be found using the following equation

$$\lambda^* = \lambda' + \frac{1 - ||\bar{c}(\lambda')||_1}{\sum_{i=1}^{n} |\Delta c_i|}$$

(52)
Case 3. $q(0) = 0$. In that case we just return a precomputed solution to the problem

$$\begin{align*}
\text{minimize} & \quad ||Az||^2_2 \\
\text{subject to} & \quad ||z||_1 - 1 = 0
\end{align*}$$

which we compute by subdividing the problem into $2^n$ quadratic programs and solving each of them separately using CVXOPT algorithm [2] as described in Section 6.

Figure 4: Panel A shows and example of solution paths calculated by the LARS algorithm. Panel B shows their extended versions as defined in Case 2 in D.2. The x-axis is reversed, so $\lambda$ decreases as it moves to the right.

D.3 Comparison

We perform a comparison between CoLLie and an algorithm based on CVXOPT [2] as described in Section 6. CoLLie uses different procedures depending on the problem, thus for a fair comparison we generate equal number of tests falling under Case 1 and Case 2 (two main cases) as described in D.2. To achieve that, we generate a random $m \times n$ matrix $A$ where each entry is sampled from the standard normal distribution. We use $m = 1000$ and $n$ ranging from 2 to 7. We then generate a vector $\hat{z} \in \mathbb{R}^n$ such that each entry is sampled from a uniform distribution on $[-0.5, 0.5]$. Then $\hat{z}$ is normalized to have L1 norm equal to 1. We sample a number $l$ from a uniform distribution on $[-1, 3]$ and multiply $\hat{z}$ by $l$ to obtain $z' = l\hat{z}$. By this procedure, we are guaranteed that cases $||z'||_1 < 1$ and $||z'||_1 \geq 1$ are equally likely. We then let $b = Az'$. The task is then to find $z$ with L1 norm equal to 1 that minimizes $||Az - b||^2_2$. As $q(0) = ||z'||_1 = l$, Case 1 and Case 2 are equally likely.

We perform 1000 experiments for each $n$. As losses for optimal solutions can be on widely different scales, we report the relative error between the loss obtained by CoLLie and the loss obtained by the algorithm based on CVXOPT, which seems to always find the optimal solution.

The time is measured on a single computer with an Intel Core i5-6500 CPU (4 cores) and 16GB of RAM.

E Experiments

E.1 Ablated D-CIPHER

The ablated version uses the standard MSE loss with estimated derivatives and thus solves the following optimization problem:

$$\min\limits_g \min_{||\beta||} \sum_{d=1}^{D} \sum_{x \in G} \left( \sum_{p=1}^{P} \hat{\beta}_p [v^{(d)}](x) - g(x, v^{(d)}(x)) \right)^2$$

where $v^{(d)}$ is the observed field, $G$ is the sampling grid, and $\hat{\beta}_p [v^{(d)}](x)$ requires derivative estimation.

The ablated version uses the same symbolic regression algorithm to search over closed-form $g$ and CoLLie for the inner optimization.
E.2 Implementation

**Step 1.** For all experiments, we use the default dictionaries. That means that for the second-order ODE (damped and forced harmonic oscillator) we use the dictionary \( \{ \partial_t, \partial_t^2 \} \), and for the 2-dimensional, second-order PDEs (wave and heat equations), we use \( \{ \partial_t, \partial_x, \partial_t \partial_x, \partial_t^2, \partial_x^2 \} \).

**Step 2.** Field estimation is performed using the Gaussian Process Regression from the Python library scikit-learn [27]. The kernel is chosen to be the RBF kernel [45] with an added White kernel to account for noise. The observed field is initially standardized by subtracting the mean and dividing by the standard deviation. Then the GaussianProcessRegressor is fitted to the data. The estimated fields are generated by predicting the values of a trained Gaussian Process on a full integration grid and then scaling back to their original range (by multiplying by the standard deviation and adding the mean).

**Step 3.** The search over the closed-form expression is performed using the symbolic regression library gplearn [39]. We use a custom fitness function that solves the inner optimization problem in equation [12]. This inner optimization is performed by CoLLie (Section 6). The integration is performed using Riemann sums.

Ablated version of D-CIPHER. The derivative estimation is performed by first fitting a Gaussian process (in the same way as in Step 2) and then using the finite difference to estimate the derivative in one of the coordinates for all points in the sampling grid. To obtain higher-order derivatives, a Gaussian process is fitted again and the derivative is once again calculated using the finite difference (possibly in a different direction than the first time).

E.3 Hyperparameters

**Gaussian process regression.** The kernel parameters of the Gaussian Process are automatically adjusted during training. The default bounds of the length scale of the RBF kernel and the noise level of the White kernel are used, i.e., \((1e^{-5}, 1e5)\).

**GPlearn.** We do not perform parameter tuning for the gplearn library and use the same parameters as in D-CODE [30] except for the parsimony coefficient and the number of generations.

| Hyperparameter     | Value       |
|--------------------|-------------|
| population size    | 15000       |
| tournament size    | 20          |
| p crossover        | 0.6903      |
| p subtree mutation | 0.1330      |
| p hoist mutation   | 0.0361      |
| p point mutation   | 0.0905      |
| generations        | 20 and 30   |

The number of generations is chosen to be 30 for the damped and forced harmonic oscillator and 20 for the inhomogeneous heat and wave equations.

Please check [39] for the detailed description of these parameters.

We modify the implementation of the parsimony coefficient. The standard implementation adds to the loss the length of the equation multiplied by the parsimony coefficient. In our implementation, we increase the loss by the parsimony coefficient. This modification is performed because for different experiments we record the loss on widely different scales. To prevent tuning this parameter for every experimental setting we introduce a penalty that can work on different scales. The parsimony coefficient is chosen manually by performing experiments for a few values. The value used in the experiments is 0.05.

The set of allowed mathematical operations is: \( \{ +, -, \times, \div, \sin, \exp, \log \} \)

We want to emphasize that we use the same configuration of gplearn in D-CIPHER and its ablated version.
**Integration and number of testing functions.** For the damped and forced harmonic oscillator we use 10 testing functions and the integration step 0.01. For the inhomogeneous heat and wave equations we use 100 testing functions and integrate on a grid with steps $\delta t = 0.01$ and $\delta x = 0.01$.

**Derivative estimation in the ablated version of D-CIPHER.** The Gaussian process is configured the same way as described above. The interval used in the finite difference method to estimate the derivative was chosen to be: $10^{-3}$.

E.4 Data generation

**Damped and forced harmonic oscillator.** The true fields are created by analytically solving the equation $\partial_t^2 u(t) + 2\theta_1 \partial_x \partial_t u(t) + \theta_2^2 u(t) = g(t)$ with random initial conditions for $u(0)$ and $\partial_t u(0)$. The observed fields are then created by sampling $t \in [0, T]$, evaluating $u(t)$, and adding a Gaussian noise. $T = 2$ was used in the experiments.

**Inhomogeneous heat equation.** The true fields are computed by solving $\partial_t u(t, x) - \theta_1 \partial_x^2 u(t, x) = g(t, x)$ with Neumann boundary conditions $\partial_x u(t, 0) = \partial_x u(t, X) = 0$ and an initial condition $u(0, x) = u_0(x)$, where $u_0$ is randomly sampled from a Gaussian process. The equation is solved using the implicit BTCS scheme [15] with steps $\delta t = 0.001$ and $\delta x = 0.001$. The observed field is generated by sampling $(t, x) \in [0, T] \times [0, X]$, evaluating the true field $u(t, x)$ and adding a Gaussian noise. $T = 2$ and $X = 2$ are used in the experiments.

**Inhomogeneous wave equation.** The true fields are computed by solving $\partial_t^2 u(t, x) - \theta_1 \partial_x^2 u(t, x) = g(t, x)$ with Dirichlet boundary conditions $u(t, 0) = u_0(0)$, $u(t, X) = u_0(X)$, where $u_0$ is randomly sampled from a Gaussian process and specifies the initial condition $u(0, x) = u_0(x)$. The equation is solved using the Implicit Difference Method [20] with steps $\delta t = 0.001$ and $\delta x = 0.001$. The observed field was generated by sampling $(t, x) \in [0, T] \times [0, X]$, evaluating the true field $u(t, x)$ and adding a Gaussian noise. $T = 2$ and $X = 2$ were used in the experiments.

E.5 Experimental settings

**Damped and forced harmonic oscillator.**

Default values are in bold.

- Noise ratio ($\sigma_R$): 0.001, 0.005, **0.01**, 0.1, 0.2, 0.5
- Number of samples ($D$): 1, 2, 5, **10**, 15
- Domain ($\Omega$): $[0, 2]$
- Grid ($G$): $\{0, 0.08, \ldots, 2\}$, $\{0, 0.1, \ldots, 2\}$, $\{0, 0.13, \ldots, 2\}$, $\{0, 0.2, \ldots, 2\}$, $\{0, 0.4, \ldots, 2\}$

**Inhomogeneous heat equation**

- Noise ratio ($\sigma_R$): 0.05, 0.1, 0.2
- Number of samples ($D$): 10
- Domain ($\Omega$): $[0, 2] \times [0.2]$
- Grid ($G$): $\{0, 0.07, \ldots, 2\} \times \{0, 0.07, \ldots, 2\}$

**Inhomogeneous wave equation**

- Noise ratio ($\sigma_R$): 0.001, 0.01, 0.015
- Number of samples ($D$): 10
- Domain ($\Omega$): $[0, 2] \times [0.2]$
- Grid ($G$): $\{0, 0.07, \ldots, 2\} \times \{0, 0.07, \ldots, 2\}$

E.6 Correct functional form

To measure success probability we need to establish whether two closed-form functions match. The previous approach [30] considered their functional forms, i.e., expressions where all numeric
constants are replaced by placeholders. By this measure, functions \( \sin(3x) \) and \( \sin(3.5x) \) match as they have the same functional form \( \sin(Cx) \), where \( C \) is a placeholder.

However, this definition is quite restrictive because functions \( \sin(3x), \sin(3x) + 0.001, 1.001 \sin(3x), \) and \( \sin(3x + 0.001) \) all have different functional forms.

We consider it an open challenge to design a good metric that would meaningfully reflect whether the correct equation is discovered. We propose the following.

For a target function \( f \), we consider its augmented form \( \tilde{f} \), defined as \( \tilde{f}(x) = C_1 f(C_3 x + C_4) + C_2 \), where all \( C_i \) are placeholders. Then all numeric constants are turned into placeholders as well. At the end, we combine the constants. For instance, \( C_1 + C_2 \) becomes just \( C_3 \).

As an example, let us consider a function \( f(x) = 1.3e^{2x} \). The augmented functional form is created in the following way:

1. Augment: \( C_1 \times 1.3e^{2x} (C_3 x + C_4) + C_2 \)
2. Replace: \( C_1 \times C_5 e^{C_6 (C_3 x + C_4)} + C_2 \)
3. Combine: \( C_1 e^{C_3 x + C_4} + C_2 \)

We perform this procedure for the target function. We can now take the standard functional form of the candidate function and check whether it matches the augmented functional form of the target function, taking into account that some of the constants might not be present in the candidate expression.

To aid in this procedure, we use a Python library for symbolic mathematics, SymPy [25].

E.7 Computation time

The average computation time for a single experiment with the damped and forced harmonic oscillator is 281 seconds with a standard error of 4.5 seconds. The average computation time for a single experiment with an inhomogeneous heat equation is 68 minutes with a standard error of 38 seconds. This time is measured on a single computer with an Intel Core i5-6500 CPU (4 cores) and 16GB of RAM.

The experiments are run simultaneously on 5 computers like the one described above. The total time for all experiments (all seeds, all equations, all experimental settings, and both versions of D-CIPHER) is 62 hours.

E.8 Licenses

The licenses of the software used in this work are presented in Table 6.

| Software  | License                           |
|-----------|-----------------------------------|
| gplearn   | BSD 3-Clause "New" or "Revised" License |
| cvxopt    | GNU General Public License        |
| cvxpy     | Apache License                    |
| sympy     | New BSD License                   |
| scikit-learn | BSD 3-Clause "New" or "Revised" License |
| numpy     | liberal BSD license               |
| pandas    | BSD 3-Clause "New" or "Revised" License |
| scipy     | liberal BSD license               |
| python    | Zero-Clause BSD license           |